Derivation of Convolutional Neural Networks

Haoxiang Xu
Intelligent Media Computing(IMC) Lab
School of Software, Sun Yat-sen University
Guangzhou, China, 510006
hawsang0206@gmail.com

August 10, 2014

1 Introduction

This document is based on Dr. Jake Bouvries's *Notes on Convolutional Neural Networks,* and discusses the derivation of convolutional neural networks(CNNs). This document is free in format, contains abundant graphical representations as well as intuitive discussions, and you may regard it as a guidance for CNNs derivation rather than a formal paper.

Since it is a common phenomenon that newcomers will get lost in the mathematical theories behind CNNs, this document will analyze the key formulas of CNNs in detail, decompose the derivation procedure and provide supplementary graphical interpretations. This document is suitable for beginners rather than professionals.

Disclaimer: This rough note could contain errors, exaggerations, and false claims. Original version is in Chinese, and my translation may introduce inaccurate expressions.

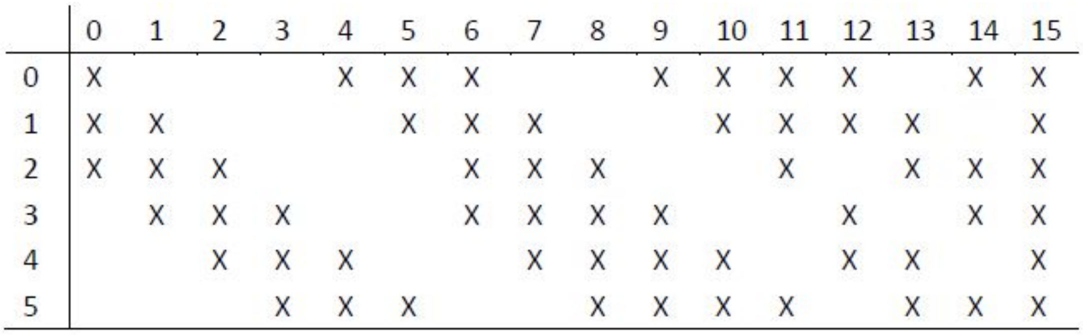
2 Feed-forward Pass

2.1 Convolutional Layer

Let $\mathrm{x}_i^l \in \mathbf{R}^{M_l \times M_l}$ represent the ith feature graph of the lth layer, and let $\mathrm{k}_{ij}^l \in \mathbf{R}^{K_l \times K_l}$ represent the convolutional template that corresponds to the feature graphs from the ith feature graph of the (l-1)th layer to the jth feature graph of the lth layer. Thus there will be at most $\mathrm{i} \times j$ convolutional templates. Because it is not fully connected between the (l-1)th layer and the lth layer, the number of convolutional templates is usually smaller than $\mathrm{i} \times j$.

For example, in [1], the connection between S2 and C3 is illustrated by table 1, and X represents the status of connected. S2 has i=6 feature graphs, and C3 has j=16 feature graphs. However, there are 60 Xs in the table 1, thus there are 60 convolutional templates. k_{ij}^{l} represents the convolutional template that corresponds to X in the ith row and the jth column in the table.

TABLE 1



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	X				X	X	X			X	X	X	X		X	X
1	X	X				X	X	X			X	X	X	X		X
2	X	X	X				X	X	X			X		X	X	X
3		X	X	X			X	X	X	X			X		X	X
4			X	X	X			X	X	X	X		X	X		X
5				X	X	X			X	X	X	X		X	X	X

If we define * as the convolution operator that applies to the connection between the (*l*-1)th layer and the *l*th layer, that is, the convolution representation for the connection between the ith feature graph of the (*l*-1)th layer and the jth feature graph of the *l*th layer is $\mathrm{x}_{i}^{l-1} * \mathrm{k}_{\mathrm{ij}}^{1}$. Using index set M_j = {i | i*th* *map connected to* j*th* *map*} represent the feature graphs of the (*l*-1)th layer that connected to the jth feature graph of the *l*th layer.

$$\mathbf{x}_{j}^{l} = f\left(\sum_{i \in M_j} \mathbf{x}_{i}^{l-1} * \mathbf{k}_{ij}^{l} + b_{j}^{l} \mathbf{I}_{(M_{l-1}-K_l+1)\times(M_{l-1}-K_l+1)} \right) \tag{1}$$

f(•) in (1) is the active function, b_{j}^{l} is the biasing scalar and $\mathrm{x}_{j}^{l} \in \mathbf{R}^{(M_{l-1}-K_l+1)\times(M_{l-1}-K_l+1)}$.

In summary, for the example above, the number of parameters between S2 and C3 is $60\times5\times5+16=1516$.

2.2 Polymerization Layer (Desampling layer)

2.2.1 Average Polymerization

If average polymerization is adopted, then

$$\mathbf{x}_{j}^{l} = f(\beta_{j}^{l} \mathit{down}(x_{j}^{l-1}) + b_{j}^{l} \mathbf{I}_{(\frac{M_l}{n})\times(\frac{M_l}{n})}) \tag{2}$$

β_{j}^{l} and $\mathrm{b}_{\mathrm{j}}^{\mathrm{l}}$ are the scalars in (2), and *down*(•) represents the desampling function which sums up every $\mathrm{n}\times\mathrm{n}$ areas(non-overlapping) in the input feature graph x_{j}^{l-1}. For this reason, the size of output feature graph x_{j}^{l} is *n* times smaller than the size of x_{j}^{l-1}. If we define $x_{j}^{l} \in \mathbf{R}^{M_l\times M_l}$, then $x_j^{l-1} \in \mathbf{R}^{(\frac{M_l}{n})\times(\frac{M_l}{n})}$.

2.2.2 Maximum Polymerization

If maximum polymerization is adopted, then

$$x_j^l = down(x_j^{l-1}) \tag{3}$$

Down（•） represents the desampling function in (3) which selects the maximum value among all the $n \times n$ areas in the input feature graph x_j^{l-1}. No active function is required here.

3 Obtain Gradient Using Back Propagation

Define *Error* as $\delta^l = \frac{\partial E}{\partial z^l}$.

In *Notes on Convolutional Neural Networks*, we know

$$z^l = W^l x^{l-1} + b^l,\ x^l = f(z^l)$$

$$\therefore \delta^l = ((W^{l+1})^T \delta^{l+1}) \circ f'(z^l)$$

3.1 Calculating the Error in the Convolutional Layer

If we want to obtain the error in the lth layer(Convolutional Layer), we must begin our calculation with the part before the active function of the $(l+1)$th layer(polymerization layer). Define the error in this layer as δ^{l+1}.

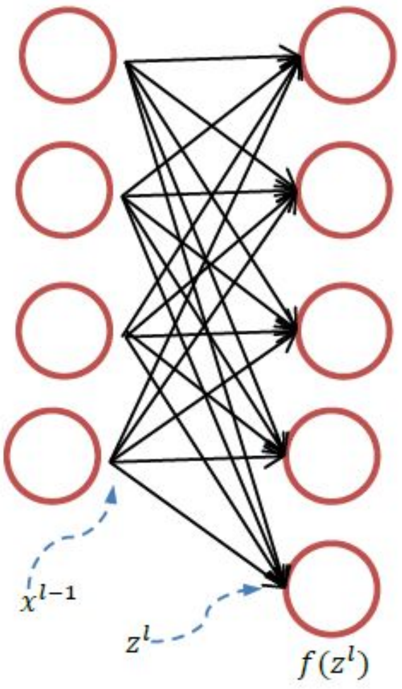


FIGURE 1

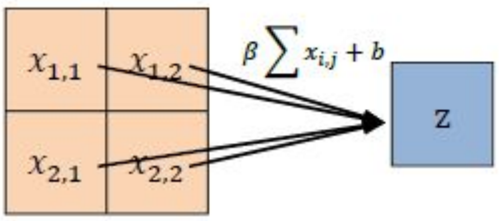
3.1.1 Average Polymerization

In average polymerization is adopted, we can obtain from equation (1) and (2) that

$$\begin{cases} \mathbf{z}_j^{l+1} = \beta_j^{l+1} down(\mathbf{x}_j^l) + b_j^{l+1} \mathbf{I}_{(\frac{M_{l+1}}{n})\times(\frac{M_{l+1}}{n})} \\ \mathbf{x}_j^l = f(\mathbf{z}_j^l) \end{cases}$$

For unit **z** in z_j^{l+1}, its corresponding *error* is the value δ in the same position in δ_j^{l+1}. In other words,

$$\delta^{l+1} = \frac{dE}{dz^{l+1}}$$



However, **z** derivation of **x** will generate a $n \times n$ matrix, that means $\frac{d\mathbf{z}^{l+1}}{d\mathbf{x}^l} = \beta \mathbf{I}_{n\times n}$.

Besides this, $\frac{d\mathbf{x}^l}{d\mathbf{z}^l} = f'(\mathbf{z}^l)$. In conclusion, we can get the error in the lth layer,

FIGURE 2

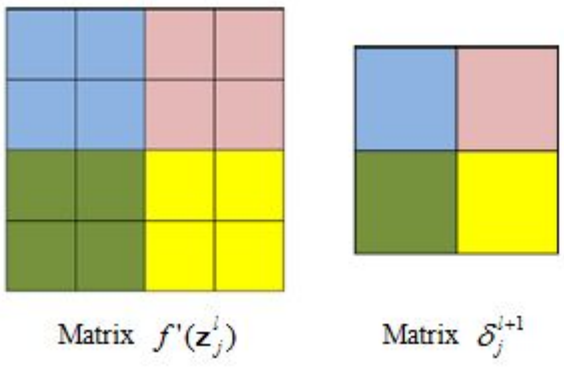
$$\delta^l = \frac{dE}{d\mathbf{z}^l} = \frac{dE}{d\mathbf{z}^{l+1}}\frac{d\mathbf{z}^{l+1}}{d\mathbf{z}^l} = \frac{dE}{d\mathbf{z}^{l+1}}\left(\frac{d\mathbf{z}^{l+1}}{d\mathbf{x}^l} \circ \frac{d\mathbf{x}^l}{d\mathbf{z}^l}\right) = \delta^{l+1}(\beta \mathbf{I}_{n\times n} \circ f'(\mathbf{z}^l)) = \beta\delta^{l+1} f'(\mathbf{z}^l) \qquad (4)$$

When taking the whole feature graph $\mathbf{z}_j^{l+1}$ into consideration, $\beta = \beta_j^{l+1}$, and δ^{l+1} corresponds to the z_j^l whose size is $\mathrm{n} \times \mathrm{n}$.

$$\boldsymbol{\delta}_j^l = \beta_j^{l+1}\begin{pmatrix} [\boldsymbol{\delta}_j^{l+1}]_{1,1} f'_{1,1}(\mathbf{z}_j^l) & \cdots & [\boldsymbol{\delta}_j^{l+1}]_{1,\frac{M_{l+1}}{n}} f'_{1,\frac{M_{l+1}}{n}}(\mathbf{z}_j^l) \\ \vdots & \ddots & \vdots \\ [\boldsymbol{\delta}_j^{l+1}]_{\frac{M_{l+1}}{n},1} f'_{\frac{M_{l+1}}{n},1}(\mathbf{z}_j^l) & \cdots & [\boldsymbol{\delta}_j^{l+1}]_{\frac{M_{l+1}}{n},\frac{M_{l+1}}{n}} f'_{\frac{M_{l+1}}{n},\frac{M_{l+1}}{n}}(\mathbf{z}_j^l) \end{pmatrix} \qquad (5)$$

The calculation of δ_j^{l+1} and $f'(\mathbf{z}_j^l)$ is shown in the following graph FIGURE 3.

Parts in matrix δ_j^{l+1} will multiply with parts in matrix $f'(\mathbf{z}_j^l)$ of the same colo The obtained matrix is in the same size as matrix $f'(\mathbf{z}_j^l)$. Brief notation for this is



$$\delta_j^l = \beta_j^{l+1}(up(\delta_j^{l+1}) \circ f'(\mathbf{z}_j^l)) \qquad (6)$$

The up(•) in (6) is upsampling operation, which copies every original input units *n* times in both horizontal and vertical directions. That's $up(\mathbf{x}) = \mathbf{x} \otimes \mathbf{I}_{n\times n}$, and $\otimes$ is Kronecker product.

Define $\mathbf{A} \otimes \mathbf{B} = \begin{pmatrix} a_{11}\mathbf{B} & \cdots & a_{1n}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\mathbf{B} & \cdots & a_{mn}\mathbf{B} \end{pmatrix}$

3.1.2 Maximum Polymerization

When maximum polymerization is adopted, we can obtain from (3) and (1) that

$$\begin{cases} z_j^{l+1} = down(x_j^l) \\ x_j^l = f(z_j^l) \end{cases}$$

Similar to the inference in 3.1.1, we can get $\delta^l = \frac{dE}{dz^l} = \frac{dE}{dz^{l+1}} \frac{dz^{l+1}}{dz^l} = \frac{dE}{dz^{l+1}} \left(\frac{dz^{l+1}}{dx^l} \circ \frac{dx^l}{dz^l} \right)$

That is $\frac{dz^{l+1}}{dx^l} = l\{x^l = \max x^l\}$, and the size of $l\{\ \bullet\ \}$ is similar as the size of x^l. The value of it is 1 when judged to be true, and 0 otherwise.

$$\delta^l = \frac{dE}{dz^{l+1}} \left(\frac{dz^{l+1}}{dx^l} \circ \frac{dx^l}{dz^l} \right) = \delta^{l+1} l\{x^l = \max x^l\} \circ f'(z^l) \tag{7}$$

When the whole feature graph z_j^{l+1} is considered, the error is

$$\boldsymbol{\delta}_j^l = \begin{pmatrix} [\boldsymbol{\delta}_j^{l+1}]_{1,1} \mathbf{1}\{\mathbf{x}_{1,1}^l = \max \mathbf{x}_{1,1}^l\} \circ f'(\mathbf{z}_{1,1}^l) & \cdots & [\boldsymbol{\delta}_j^{l+1}]_{1,\frac{M_{l+1}}{n}} \mathbf{1}\{\mathbf{x}_{1,\frac{M_{l+1}}{n}}^l = \max \mathbf{x}_{1,\frac{M_{l+1}}{n}}^l\} \circ f'\left(\mathbf{z}_{1,\frac{M_{l+1}}{n}}^l\right) \\ \vdots & \ddots & \vdots \\ [\boldsymbol{\delta}_j^{l+1}]_{\frac{M_{l+1}}{n},1} \mathbf{1}\{\mathbf{x}_{\frac{M_{l+1}}{n},1}^l = \max \mathbf{x}_{\frac{M_{l+1}}{n},1}^l\} \circ f'\left(\mathbf{z}_{\frac{M_{l+1}}{n},1}^l\right) & \cdots & [\boldsymbol{\delta}_j^{l+1}]_{\frac{M_{l+1}}{n},\frac{M_{l+1}}{n}} \mathbf{1}\{\mathbf{x}_{\frac{M_{l+1}}{n},\frac{M_{l+1}}{n}}^l = \max \mathbf{x}_{\frac{M_{l+1}}{n},\frac{M_{l+1}}{n}}^l\} \circ f'\left(\mathbf{z}_{\frac{M_{l+1}}{n},\frac{M_{l+1}}{n}}^l\right) \end{pmatrix} \tag{8}$$

That is, the error will only be propagated through the pixel that reached its maximum value, and there will be no error in other positions.

3.1.3 Calculating the Gradient of the Parameters in Convolutional Layer

We have already know the value of the error generated when it reaches the 1th layer(convolutional layer), thus we can calculate the gradient of the parameters in the 1th layer via this error.

$$\because z_j^l = \sum_{i \in M_i} x_i^{l-1} * k_{ij}^l + b_j^l l_{(M_{l-1}-K_l+1)\times(M_{l-1}-K_l+1)} \tag{9}$$

$$\therefore \frac{dE}{db_j^l} = \frac{dE}{dz_j^l} \frac{dz_j^l}{db_j^l} = \left(\frac{dE}{dvec(z_j^l)} \right)^T \frac{dvec(z_j^l)}{db_j^l} = (vec(\delta_j^l))^T vec(l_{(M_{l-1}-K_l+1)\times(M_{l-1}-K_l+1)}) = \sum_{p,q}^{M_{l-1}-K_l+1, M_{l-1}-K_l+1} [\delta_j^l]_{p,q}$$

As the inference shown above, the gradient for bias b_j^l is the sum of all the elements in error δ_j^l. Please reference Table 2 in Page 199 of reference [2] to understand the derivation in (9) . The $vec(\delta_j^l)$ in the equation represents unfolding the matrix δ_j^l by column to vectors which have $(M_{l-1} - K_l + 1) \times (M_{l-1} - K_l + 1)$ columns and 1 row. Thus (9) can be represented with convolutional symbol in the following format

$$\frac{dE}{db_j^l} = \delta_j^l * l_{(M_{l-1}-K_l+1)\times(M_{l-1}-K_l+1)} \tag{10}$$

We then begin calculating the gradient of k_{ij}^l. For a certain pixel value δ^l of δ_j^l, its convolutional value in corresponding position is z^l

$$\frac{dE}{dz^l}\frac{dz^l}{dk_{ij}^l} = \delta^l x^{l-1} \tag{11}$$

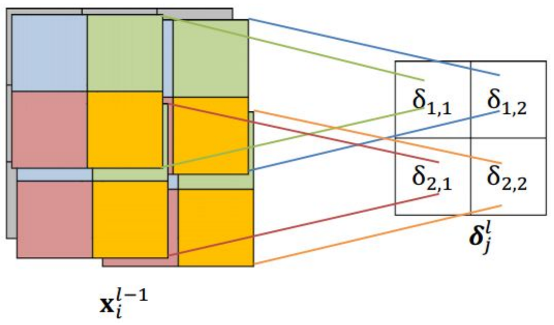
$$\frac{dE}{dk_{ij}^l} = \frac{dE}{dz_j^l}\frac{dz_j^l}{dk_{ij}^l} = \left(\frac{dE}{dvec(z_j^l)}\right)^T \frac{dvec(z_j^l)}{dvec(k_{ij}^l)^T}$$

$$= unvec_{K_l,K_l}\left(\left(vec(\delta_j^l)\right)^T\left(vec\left(\left[c_i^{l-1}\right]_{1,1}\right)\ldots vec\left(\left[c_i^{l-1}\right]_{K_l,1}\right)vec\left(\left[c_i^{l-1}\right]_{1,2}\right)\ldots vec\left(\left[c_i^{l-1}\right]_{K_l,K_l}\right)\right)\right) \tag{12}$$

The $unvec_{K_l,K_l}(\bullet)$ in the equation is to transform the vector by columns into a matrix which has K_l columns and K_l rows. The $\left[c_i^{l-1}\right]_{p,q}$ represents the matrix which is formed with all the elements in x_i^{l-1} which has convoluted with the element in the pth row and the qth column of k_{ij}^l, and this matrix has $(M_{l-1}-K_l+1)$ rows and $(M_{l-1}-K_l+1)$ columns. Therefore, there are $K_l \times K_l$ matrix $\left[c_i^{l-1}\right]_{p,q}$, and we will obtain a matrix which has $(M_{l-1}-K_l+1)\times(M_{l-1}-K_l+1)$ rows and $K_l \times K_l$ columns. Thus equation 12 can be represented with convolutional symbol in the following format

$$\frac{dE}{dk_{ij}^l} = x_i^{l-1} * \delta_j^l \tag{13}$$

FIGURE 4



As shown in FIGURE 4, we know that $x_i^{l-1} \in R^{3\times3}, k_{ij}^l \in R^{2\times2}, \delta_j^l \in R^{2\times2}$, and every area x^{l-1} which contains four color-blocks corresponds to a pixel δ^l in δ_j^l. Their correspondence is indicated by line segments in the graph. The color of those color-blocks indicated that this pixel has been multiplied with the same pixel in the k_{ij}^l when being convoluted. For example, the color blue corresponds to $\left[k_{ij}^l\right]_{1,1}$, which indicates all the pixels in this position in x_i^{l-1} has multiplied with $\left[k_{ij}^l\right]_{1,1}$ when being convoluted. Thus if we want to obtain the derivative of $\left[k_{ij}^{l}\right]_{1,1}$, only those blue-colored pixels in x_i^{l-1} will left(other positions' derivation of $\left[k_{ij}^{l}\right]_{1,1}$ will be zeros). Referencing equation 11, we know that $\frac{dE}{dk_{ij}^{l}} = \delta_{1,1}\left[x_i^{l-1}\right]_{1,1} + \delta_{1,2}\left[x_i^{l-1}\right]_{1,2} + \delta_{2,1}\left[x_i^{l-1}\right]_{2,1} + \delta_{2,2}\left[x_i^{l-1}\right]_{2,2}$, and we can also know that $\begin{pmatrix} [x_i^{l-1}]_{1,1} & [x_i^{l-1}]_{1,2} \\ [x_i^{l-1}]_{2,1} & [x_i^{l-1}]_{2,2} \end{pmatrix} = [c_i^{l-1}]_{1,1}$. In the similar way, we can obtain the derivatives of other colors, and the summarized result can be represented by equation 12 and equation 13.

However, in *Notes on Convolutional Neural Networks*, the author used the term *mutual correlation* to replace the term *convolution*. Equation 12 will change to the following format

$$\left(Jvec\left(\delta_j^l\right)\right)^T \left(vec\left(\left[c_i^{l-1}\right]_{1,1}\right)\cdots vec\left(\left[c_i^{l-1}\right]_{K_l,1}\right)vec\left(\left[c_i^{l-1}\right]_{1,2}\right)\cdots vec\left(\left[c_i^{l-1}\right]_{K_l,K_l}\right)\right)$$

The J in this equation is the permutation matrix, and $Jvec\left(\delta_j^l\right)$ will inverse the original column's order of $Jvec\left(\delta_j^l\right)$. Thus

$$\frac{dE}{dk_{ij}^{l}} = J\left(x_i^{l-1} * J\delta_j^l J\right)J \tag{14}$$

The *inverse* here is to match the **convn** function in Matlab. Cause in **convn(**A,B**)**, B will be rotated in all dimension, and B will be multiplied and summed up in corresponding positions afterwards. This operation will be **A*JBJ** in the Matlab code, which means B will be rotated two times, and it is the same in the forward calculation procedure.

3.2 Calculating the Error in the Polymerization Layer

If we want to obtain the error in the lth layer(polymerization layer), we must begin our calculation with the preceding part of the active function in the (l+1)th layer(convolutional layer). If the error in this layer is δ^{l+1} , we can get the following equation set from (1)

$$\begin{cases} z_j^{l+1} = \sum_{i \in M_j} x_i^l * k_{ij}^{l+1} + b_j^{l+1} I_{(M_l - K_{l+1}+1)\times(M_l - K_{l+1}+1)} \\ x_i^l = f\left(z_i^l\right) \end{cases}$$

The x_i^l here does not refer to any specific δ^{l+1} , but refers to all the δ^{l+1} that connected with x_i^l . For example, the 0th input graph in TABLE 1 is connected with the output graph 0,4,5,6,9,10,11,12,14 and 15. That is, x_0^l has connections with the elements in the set

$$\left\{\delta_j^{l+1} \mid j = 0,4,5,6,9,10,11,12,14,15\right\}.$$

Using index set M_i ={j|ith map connected of jth map} to represent the set of feature graphs in the (l+1)th layer which has connection with the ith feature graph in the lth layer, and taking the pixel set $\{\delta_j^{l+1} \mid j \in M_i\}$ in $\{\delta_j^{l+1}\}$ into consideration first. We know its corresponding part is x^l, thus

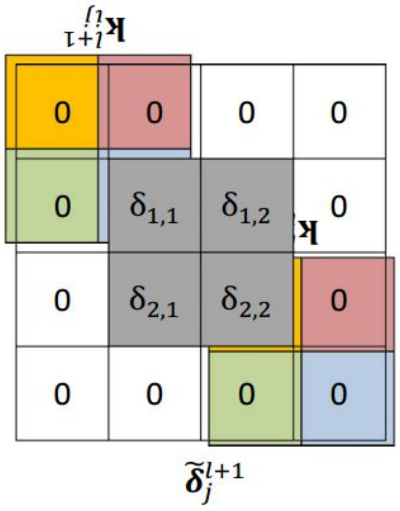
$$\frac{dE}{dx^l} = \frac{dE}{dz_j^{l+1}} \frac{dz_j^{l+1}}{dx^l} = \delta_j^{l+1} k_{ij}^{l+1} \tag{15}$$

When referencing FIGURE 4, we know $[x_i^l]_{1,1}$ is only associated with the blue-colored $[k_{ij}^l]_{1,1}$, that is, $\frac{dE}{d[x_i^l]_{1,1}} = [\delta_j^{l+1}]_{1,1}[k_{ij}^{l+1}]_{1,1}$. $[x_i^l]_{1,2}$ is only associated with the blue-colored and green-colored parts, that is, $\frac{dE}{d[x_i^l]_{1,2}} = [\delta_j^{l+1}]_{1,1}[k_{ij}^{l+1}]_{1,2} + [\delta_j^{l+1}]_{1,2}[k_{ij}^{l+1}]_{1,1}$. $[x_i^l]_{2,2}$ is associated with all the parts, that is,

$\frac{dE}{d[x_i^l]_{2,2}} = [\delta_j^{l+1}]_{1,1}[k_{ij}^{l+1}]_{1,2} + [\delta_j^{l+1}]_{1,2}[k_{ij}^{l+1}]_{2,1} + [\delta_j^{l+1}]_{2,1}[k_{ij}^{l+1}]_{1,2} + [\delta_j^{l+1}]_{2,2}[k_{ij}^{l+1}]_{1,1}$. The rest can be done in the same manner.

$$\left\{\left(\frac{dE}{dx_i^l}\right)_j\right\} = \tilde{\delta}_j^{l+1} * \left(Jk_{ij}^{l+1}J\right), \forall j \in M_i \tag{16}$$

FIGURE 5



$\tilde{\delta}_j^{l+1}$ means filling the margin of δ_j^{l+1} with 0. The equation 16 can be illustrated with FIGURE 5.

When considering $\forall j \in M_i$

$$\frac{dE}{dx_i^l} = \sum_j^{M_i} \tilde{\delta}_j^{l+1} * (Jk_{ij}^{l+1}J) \tag{17}$$

Thus the error in the lth layer(polymerization layer) is

$$\delta_i^l = \frac{dE}{dz_i^l} = \frac{dE}{dx_i^l} \circ \frac{dx_i^l}{dz_i^l} = f'\left(z_i^l\right) \circ \sum_j^{M_i} \tilde{\delta}_j^{l+1} * (Jk_{ij}^{l+1}J) \tag{18}$$

If maximum polymerization is adopted in the polymerization layer, we can get $f'\left(z_i^l\right) = 1$.

In *Notes on Convolutional Neural Networks,* only the impact of δ_j^{l+1} on δ_j^l is considered.

3.2.1 Calculating the Gradient of the Parameters in Polymerization Layer

We have already calculated the error generated when it reaches the lth layer(polymerization layer), thus we can calculate the gradient of the parameters in the lth layer using this calculated error.

3.2.1.1 Maximum Polymerization

From equation 3, we know there is no parameter.

3.2.1.2 Average Polymerization

From equation 2, we know that $\mathbf{z}_j^l = \beta_j^l down(x_j^{l-1}) + b_j^l \mathbf{I}_{(\frac{M_l}{n})\times(\frac{M_l}{n})})$

$$\therefore \frac{dE}{db_j^l} = \frac{dE}{dz_j^l}\frac{dz_j^l}{db_j^l} = \frac{dE}{dvec(z_j^l)}\frac{dvec(z_j^l)}{db_j^l} = \left(vec(\delta_j^l)\right)^T vec\left(\mathbf{I}_{(\frac{M_l}{n})\times(\frac{M_l}{n})}\right) = \sum_{p,q}^{\frac{M_l}{n},\frac{M_l}{n}}\left[\delta_j^l\right]_{p,q} \tag{19}$$

Equation 19 can be represented with convolution in the following appearance

$$\frac{dE}{db_j^l} = \delta_j^l * \mathbf{I}_{(\frac{M_l}{n})\times(\frac{M_l}{n})} \tag{20}$$

$$\frac{dE}{d\beta_j^l} = \left(vec(\delta_j^l)\right)^T vec\left(down(x_j^{l-1})\right) = \sum_{p,q}^{\frac{M_l}{n},\frac{M_l}{n}}\left[\delta_j^l\right]_{p,q} \cdot \left[down(x_j^{l-1})\right]_{p,q} \tag{21}$$

Equation 21 can be represented with convolution in the following appearance

$$\frac{dE}{d\beta_j^l} = \delta_j^l * down(x_j^{l-1}) \tag{22}$$

Reference

[1] LeCun, Yann, et al. Gradient-based learning applied to document recognition. Proceedings of the IEEE 86.11 (1998): 2278-2324.

[2] J.R. Magnus and H. Neudecker, Matrix differential calculus with applications in statistics and econometrics. 3rd Ed. Wiley 2007

[3] Scherer, Dominik, Andreas Müller, and Sven Behnke. Evaluation of pooling operations in convolutional architectures for object recognition. Artificial Neural Networks–ICANN 2010.Springer Berlin Heidelberg, 2010. 92-101.