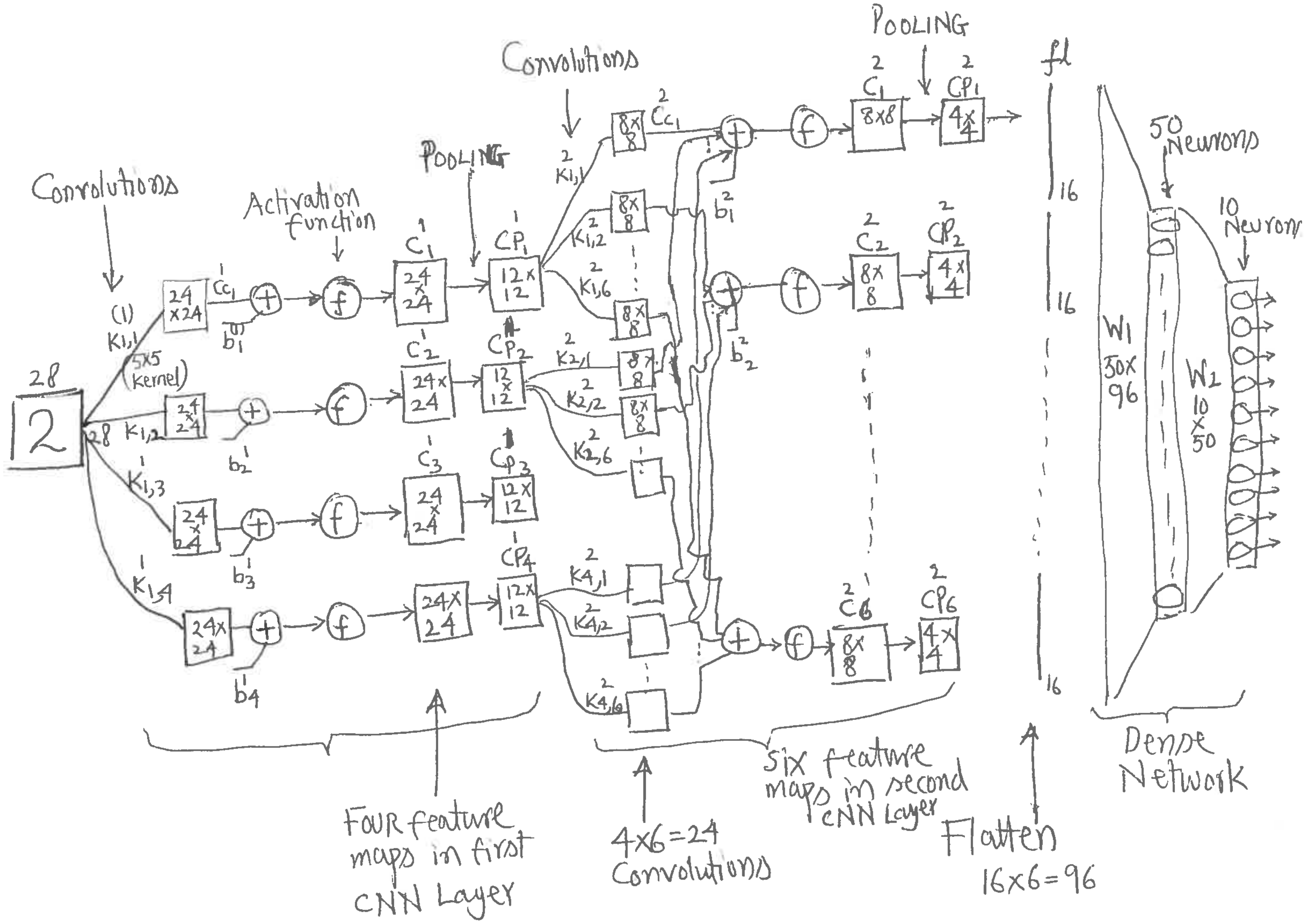
ARCHITECTURE OF A CNN WITH 4 Feature maps in First Layer and 6 Feature maps in Second Layer



Number of Parameters to be Learned:

$$4 \times \underbrace{5\times5}_{\text{kernel}} + \underbrace{4\times1}_{\text{bias}} + 24\times\underbrace{5\times5}_{\text{kernel}} + \underbrace{6\times1}_{\text{bias}} + \underbrace{50\times96}_{W1} + \underbrace{50\times1}_{\text{bias}} + \underbrace{10\times50}_{W2} + \underbrace{10\times1}_{\text{bias}}$$

$$= 6170$$

Kernel, Bias and Weights Initialization

$$K^1_{1,m} = \text{uniform}\left(\pm\sqrt{\frac{4}{(1+4)\times 5^2}}\right)$$

$m = 1..4$

$$K^2_{m,n} = \text{Uniform}\left(\pm\sqrt{\frac{6}{(4+6)\times 5^2}}\right)$$

$n = 1..6$

$$W_1 = \text{Uniform } \pm\left(\sqrt{\frac{6}{96+50}}\right)$$

$$W_2 = \text{Uniform } \pm\left(\sqrt{\frac{6}{10+50}}\right)$$

All biases initialized to zeros.

FORWARD PASS

Suppose input image is called I

then $C^1_m = f(I \otimes K^1_{1,m} + b^1_m)$

$$C^1_m(i,j) = f\left(\sum_{u=-2}^{2}\sum_{v=-2}^{2} I(i-u, j-v) \cdot K^1_{1,m}(u,v) + b^1_m\right)$$

$m = 1..4$

$\otimes$ = convolution

Assuming four feature maps in first CNN Layer and performing 5x5 convolutions.

If we are using average pooling, then

$$CP^1_m(i,j) = \frac{1}{4}\sum_{u=0}^{1}\sum_{v=0}^{1} C^1_m(2i-u, 2j-v)$$

$i,j = 1..12$

Similarly in the second layer,

$$C_n^2(i,j) = f\left(\sum_{m=1}^{4} \sum_{u=-2}^{2} \sum_{v=-2}^{2} CP_m^1(i-u, j-v) \cdot K_{m,n}^2(u,v) + b_n^2\right)$$

$n = 1..6$

size of C_n^2 is 8×8, so i,j will vary $1..8$

The average pooling in second CNN layer can be described as

$$CP_n^2(i,j) = \frac{1}{4} \sum_{u=0}^{1} \sum_{j=0}^{1} C_n^2(2i-u, 2j-v)$$

$i,j = 1..4$

The output of 6 pooling layers in the second CNN layer is flattened into $16 \times 6 = 96$ values. These 96 values are fed to a regular dense Neural Network with 50 hidden layer neurons and 10 output layer neurons.

$$a_1 = f(\text{flatten} \times W_1 + b_1)$$ — output of hidden layer

$$a_2 = f(a_1 \times W_2 + b_2)$$ — output

For the last layer, the activation function for a multi-class problem is usually Softmax, and the loss function as Cross Entropy.

Backpropagation for CNN :

If last layer is Softmax, and loss function is Cross Entropy. then δ in last layer is given by

$$\delta_i = s_i - y_i \qquad \text{(see derivation of Softmax handout)}$$

or $\delta = S - Y$

$$\Delta W_2 = \frac{\partial L}{\partial W_2} = \frac{\partial L}{\partial S_i} \times \frac{\partial S_i}{\partial W_2} = \delta \times a_1^T$$

$$\Delta b_2 = \delta$$

δ_1 = delta in hidden layer $= W_2^T \times \delta_{\text{last layer}} \times f'$

f' — derivative of activation function

$$\Delta W_1 = \frac{\partial L}{\partial a_2} \times \frac{\partial a_2}{\partial W_1} = \delta_1 \times \text{flatten}^T$$

flatten — 96x1 in our example

$\Delta b_1 = \delta_1$

$\delta_{flatten}$ = delta in flattening layer

$$\delta_{flatten} = W_1^T \times \delta_1$$

$$\delta C_n^2(i,j) = \frac{1}{4}\,\delta_{flatten}\left(\lceil i/2 \rceil, \lceil j/2 \rceil\right) \cdot f'$$

$i,j = 1, 2 \ldots 8$

f' — derivative of activation function

$\delta_{flatten}$ is copied to corresponding places in δC_n^2 and divided by 4

Computing $\Delta K_{m,n}^2$ is tricky. Note that the equation for forward pass for C_n^2 is (from page 3)

$$C_n^2(i,j) = f\left(\sum_{m=1}^{4} \sum_{u=-2}^{2} \sum_{v=-2}^{2} C_{P_m}^1(i-u, j-v) \cdot K_{m,n}^2(u,v) + b_n^2\right)$$

$$\Delta K_{m,n}^2(u,v) = \sum_{i=1}^{8} \sum_{j=1}^{8} \delta C_n^2(i,j) \cdot \frac{\partial C_n^2(i,j)}{\partial K_{m,n}^2(u,v)}$$

This is because each kernel element is involved in computing the 8x8 values in C_n^2

$$\Delta K_{m,n}^2(u,v) = \sum_{i=1}^{8} \sum_{j=1}^{8} \delta C_n^2(i,j) \cdot \frac{\partial}{\partial K_{m,n}^2(u,v)} \left(\sum_{m=1}^{4} \sum_{u=-2}^{2} \sum_{v=-2}^{2} C_{P_m}^1(i-u, j-v) \cdot K_{m,n}^2(u,v) + b_n^2\right)$$

derivative is for a particular value of u, v and an i, j

$$\Delta K_{m,n}^2(u,v) = \sum_{i=1}^{8} \sum_{j=1}^{8} \delta C_n^2(i,j)\, C_{P_m}^1(i-u, j-v)$$

for convolution, it needs to be $u-i, j-v$ so rotate it by 180°

$$\boxed{\Delta K_{m,n}^2(u,v) = \delta C_n^2(i,j) \otimes \left(C_{P_m}^1 \text{ rotated by } 180^\circ\right)}$$

$\otimes$ — convolution

To determine Δb_n^2, note that it is of size 1×1 (5)

$$\Delta b_n^2 = \sum_{i=1}^{8}\sum_{j=1}^{8} \delta c_n^2(i,j) \times \frac{\partial}{\partial b_n^2}\left(\sum_{m=1}^{4}\sum_{u=-2}^{2}\sum_{v=-2}^{2} cp_m^1(i-u, j-v) \cdot k_{m,n}^2(u,v) + b_n^2\right)$$

$$\boxed{\Delta b_n^2 = \sum_{i=1}^{8}\sum_{j=1}^{8} \delta c_n^2(i,j)}$$

To determine $\Delta k_{1,m}^1$, first we need to determine δcp_m^1 and δc_m^1

$$\delta cp_m^1(i,j) = \sum_{n=1}^{6}\sum_{u=-2}^{2}\sum_{v=-2}^{2} \delta c_n^2(i+u, j+v) \cdot \frac{\partial}{\partial cp^1(i,j)}\left(\sum_{m=1}^{4}\sum_{u=-2}^{2}\sum_{v=-2}^{2} cp^1(i,j) \cdot k_{m,n}^2(u,v) + b_n^2\right)$$

$$= \sum_{n=1}^{6}\sum_{u=-2}^{2}\sum_{v=-2}^{2} \delta c_n^2(i+u, j+v) \cdot k_{m,n}^2(u,v)$$

$$= \sum_{n=1}^{6}\sum_{u=-2}^{2}\sum_{v=-2}^{2} \delta c_n^2(i-(-u), j-(-v)) \cdot k_{m,n\ \text{rot. }180^\circ}^2(-u,-v)$$

$$\boxed{\underbrace{\delta cp_m^1}_{12\times12} = \sum_{n=1}^{6} \underbrace{\delta c_n^2}_{8\times8} \otimes \underbrace{k_{m,n\ \text{rot. }180^\circ}^2}_{5\times5}}$$

→ Note that this is full convolution.

$$\delta c_m^1(i,j) = \frac{1}{4}\, \delta cp_m^1(\lceil i/2 \rceil, \lceil j/2 \rceil) \cdot f'$$

$i, j = 1..24$

derivative of activation function

$$\Delta k_{1,m}^1(u,v) = \sum_{i=1}^{24}\sum_{j=1}^{24} \delta c_m^1(i,j) \cdot \frac{\partial}{\partial k_{1,m}^1(u,v)}\left(\sum_{u=-2}^{2}\sum_{v=-2}^{2} I(i-u, j-v) \cdot k_{1,m}^1(u,v) + b_m^1\right)$$

$$= \sum_{i=1}^{24}\sum_{j=1}^{24} \delta c_m^1(i,j) \cdot I(i-u, j-v)$$

$$\boxed{\Delta k_{1,m} = \delta c_m^1 \otimes I_{\text{rot. }180^\circ}}$$

$$\Delta b_m^1 = \sum_{i=1}^{24}\sum_{j=1}^{24} \delta c_m^1(i,j) \cdot \frac{\partial}{\partial b_m^1}\left(\sum_{u=-2}^{2}\sum_{v=-2}^{2} I(i-u, j-v) \cdot k_{1,m}^1(u,v) + b_m^1\right)$$

$$\boxed{\Delta b_m^1 = \sum_{i=1}^{24}\sum_{j=1}^{24} \delta c_m^1(i,j)}$$

The model parameters will be updated as:

$$K_{1,m}^{1} = K_{1,m}^{1} - \lambda \cdot \Delta K_{1,m}^{1}$$

$$b_{m}^{1} = b_{m}^{1} - \lambda \cdot \Delta b_{m}^{1}$$

$$K_{m,n}^{2} = K_{m,n}^{2} - \lambda \cdot \Delta K_{m,n}^{2}$$

$$b_{n}^{2} = b_{n}^{2} - \lambda \Delta b_{n}^{2}$$

$$W_1 = W_1 - \lambda \Delta W_1$$

$$b_1 = b_1 - \lambda \Delta b_1$$

$$W_2 = W_2 - \lambda \Delta W_2$$

$$b_2 = b_2 - \lambda \Delta b_2$$