Convolutional Neural Network

This CNN implementation is done for classification problem specifically handwritten digit recognition problem.
Training and testing data set: mnist dataset
Structure:

Layers: 6

Layer 1:

Name: Input layer
Size: 28 X 28
Slice: 1
Weights: no
Bias: no

Layer 2:

Name: Convolution Layer
Size of each slice: 28X28
Number of slices: 6
Weights each slice: 3 X 3
Bias: 1 for each slice
Receptive field: 3X3

Layer 3:

Name: Sub Sampling
Size of each slice: 14X14
Number of slices: 6
Weights each slice: NO
Bias: No

Layer 4:

Name: Convolution Layer
Size of each slice: 14X14
Number of slices: 12
Weights each slice: 3 X 3 X 6
Bias: 1 for each slice
Receptive field: 3X3

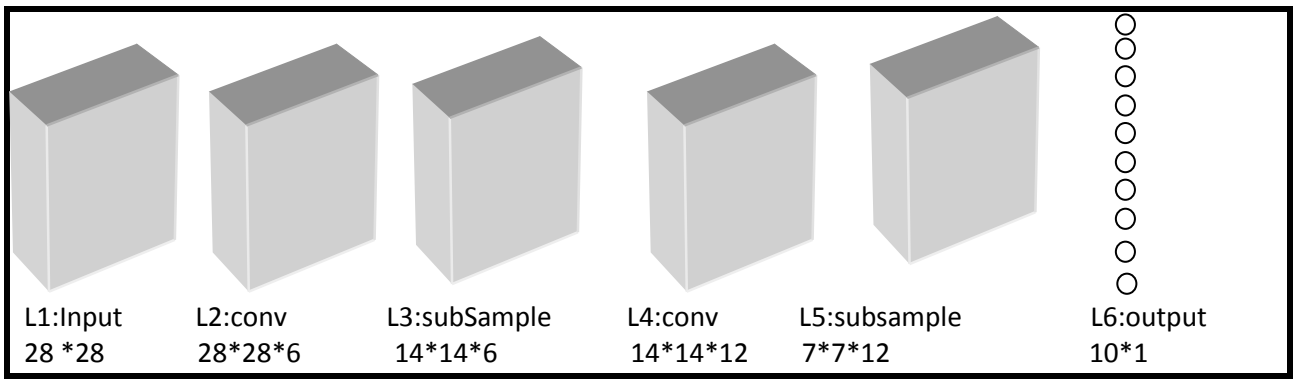
Layer 5:

Name: Sub Sampling
Size of each slice: 7X7
Number of slices: 12
Weights each slice: No
Bias: No

Layer 6:

Name: Output

Size of each slice: 1X1
Number of slices: 1
Weights each slice: 7 X 7 X 12
Bias: 1 for each slice
Receptive field: ?



Forward Propagation:

In general forward propagation each unit is weighted sum of its input and then passes through non-linear function

$$a_j = \sum_i w_{ji} z_i$$

1. L1: input image of size 28*28
2. L2 conv : it has 6 slices of size 28*28 each of slice is formed by convolving the input image with 3 by 3 weights (parameters) note that weights are shared and same for one slice so in L2 there are 3 * 3 * 6 weights in total 3* 3 for each slice and one bias for each slice and then passes through non linear function.
3. L3 subsample: this layer also have six slices of size 14*14 and it is formed by max pooling mean out of 4 neurons one is selected which value is greater among 4 it has no weights.
4. L4 conv: this layer have 12 slices of size 14*14 each slice is formed by convolution of 3*3*6 weights of L3 layer so here are we have 3*3*6*12 weights and one bias for each slice so here are 12 biases.
5. L5 subsample: this layer have 12 slices of size 7*7 and it is formed by max pooling of each slice of previous layer
6. L6 output: it has 10 neurons and each neuron is fully connected through all neurons of previous layer mean 7*7*12 so it has total 7*7*12*10 weights.

Back Propagation:

Compute delta (errors) and gradient s of weights and bias.

Last layer delta simply computed by difference of our forward prop output and training set target

Delta = yk – tk;

Hidden layer‘s delta computed formula is:

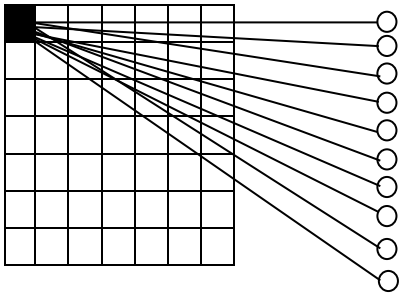
$$\delta_j = h'(a_j) \sum_k w_{kj}\delta_k$$

Means:

Delta of current layer neuron = derivative of output of that neuron* sum of (product of that weights which is connected to next layer’s neuron and delta of next layer neurons)

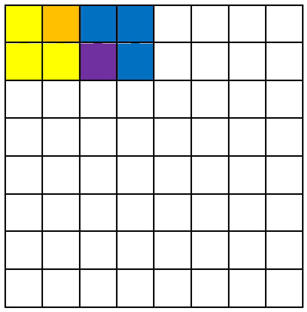
So; to calculate the delta of neurons of each layer we have to note down the connection between neurons to next layer.

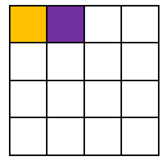
Layer 6:
Delta: yk-tk:
Layer5:
Current layer: subsample layer
Next layer: output layer
Delta:
Each neuron is connected to 10 neurons in 6th layer so



As in above figure one neuron of fifth layer is connected to all 10 neurons so delta of this neuron is calculated by product of deltas , weights (from which this neuron is connected) and derivative of current layer activation function.

L4 layer :
Current layer: conv layer (14 by 14)
Next layer: subsample layer (7 by 7)
Delta:
In forward prop maximum value out of four neurons is sent to next layer’s one neuron as shown in the following figure.





Delta should be transferred only to those neurons that send their effect to next layer. For this purpose we should have location of those neurons that send their effect. Here; we should modify our forward propagation in such a way during max-pooling of conv layer we have to save the maxmap in convLayer. By using maxmap we can easily send deltas of subsample layer to conv layer.

Layer 3:
Current layer: subsample
Next layer:conv layer

In **forward prop** conv to subsample we conv the 4[th] layer with weights
Let we have first subsample layer and second conv layer and only focus on neuron#28 REMINDER: THIS IS FORWARD PROP.

1	2	3	4	5	6	7	8
9	10 w11	11 w12	12 w13	13	14	15	16
17	18 W21	19 W22	20 W23	21	22	23	24
25	26 W31	27 W32	28 W33	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	**26**	**27**	**28**	29	30	31	32
33	**34**	**35**	**36**	37	38	39	40
41	**42**	**43**	**44**	45	46	47	48

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	**35**	36	37	38	39	40
41	42	43	44	45	46	47	48

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	**27**	**28**	**29**	30	31	32
33	34	**35**	**36**	**37**	38	39	40
41	42	**43**	**44**	**45**	46	47	48

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	**36**	37	38	39	40
41	42	43	44	45	46	47	48

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	**28**	**29**	**30**	31	32
33	34	35	**36**	**37**	**38**	39	40
41	42	43	**44**	**45**	**46**	47	48

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	**37**	38	39	40
41	42	43	44	45	46	47	48

So; in back prop we can say that #28 neuron send its effect/connected to 9 neurons of next layer's neuron. Following is visualization of BACK PROP

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19 W33	20 W32	21 W31	22	23	24
25	26	27 W23	28 W22	29 W21	30	31	32
33	34	35 W13	36 W12	37 W11	38	39	40
41	42	43	44	45	46	47	48

As according to figures 28 is connected to 3by3 (9) neurons and #28 is multiplied to w33 when 19 Produced 28 is multiplied to w32 when 20 , 28 is* to w31 when 21 and so on 28 is multiplied to w11 when 37 so when we have to calculate the delta of 28th neuron it can be calculated by dot product of weights(3 by 3) with deltas of (19 20,21 , 27,28,29,35,36,37) s , when we go to 29th it can be calculated by dot product of weights by deltas of neuron #(20,21,22,28,29,30,36,37,38).

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20 W33	21 W32	22 W31	23	24
25	26	27	28 W23	29 W22	30 W21	31	32
33	34	35	36 W13	37 W32	38 W31	39	40
41	42	43	44	45	46	47	48

As weights are shared so delta of conv layer can be calculated by convolution of deltas by the weights.

Layer 2:
Current layer: conv
Next layer: subsample
Same as Layer4

Gradients of weights and biases:

$$\frac{\partial E_n}{\partial w_{ji}} = \delta_j z_i.$$

Neurons are connected through weights so gradient of that weight is calculates by (delta * input of that neuron (previous layer))