cnn mathematics-

Convolutional operation takes a patch of the image, and applies a filter by performing a dot product on it. The convolution layer is similar to fully connected layer, but performs convolution operation on input rather than matrix multiplication.

The convolutional layer takes an input volume of:

Number of input N

The depth of input C

Height of the input H

Width of the input W

These hyperparameters control the size of output volume:

Number of filters K

Spatial Extent F

Stride length S

Zero Padding P

The spatial size of output is given by $(H-F+2P)/S+1 \times (W-F+2P)/S+1$

Note: When $S=1, P=(F-1)/2$ preserves the input volume size.

Forward Propagation-

As stated earlier, convolutional layer replaces the matrix multiplication with convolution operation. To compute the pre non linearity for i,j^{th} neuron on l layer, we have:

$$Z^{l}_{ij}=\sum_{a=0}^{m-1}\sum_{b=0}^{m-1}W_{ab}a^{l-1}_{(i+a)(j+b)}$$

Naively, for doing our convolutional operation we loop over each image, over each channel and take a dot product at each $F\times F$ location for each of our filters. For the sake of efficiency and computational simplicity, what we need to do is gather all the locations that we need to do the convolution operations and get the dot product at each of these locations.

Backward Propagation-

We know the output error for the current layer ∂out which in our case is $\partial C / \partial Z^{l}_{ij}$ as our layer is only computing pre non linearity output Z . We need to find the gradient

$\partial C / \partial W^{l}_{ab}$ for each weight .

$$\partial C / \partial W^{l}_{ab} =$$

$$\sum_{i=0}^{N-m}\sum_{j=0}^{N-m} \partial C / \partial Z^{l}_{ij} \times \partial Z^{l}_{ij} / \partial W_{ab}{}^{l}=\sum_{i=0}^{N-m}\sum_{j=0}^{N-m} \partial C / \partial Z^{l}_{ij} \times a^{l-1}_{(i+a)(j+b)}$$

Notice that $\partial Z^{l}_{ij} / \partial W^{l}_{ab} = a^{l-1}_{(i+a)(j+b)}$ is from the forward propagation above, where a^{l-1} is the output of the previous layer and input to our current layer.

For bias gradient, we simply accumulate the gradient as with backpropagation for fully connected layers. So,

$$\partial C / \partial b^{l} = \sum_{i=0}^{N-m} \sum_{j=0}^{N-m} \partial C / \partial Z^{l}_{ij}$$

Now to backpropagate the errors back to the previous layer, we need to compute the input gradient ∂X which in our case is $\partial C / \partial a^{l-1}_{ij}$.

$$\partial C / \partial a^{l-1}_{ij} = \sum_{a=0}^{m-1} \sum_{b=0}^{m-1} \partial C / \partial Z_{(i-a)(j-b)} \times \partial Z^{l}_{(i-a)(j-b)} /$$
$$\partial a^{l-1}_{ij} = \sum_{a=0}^{m-1} \sum_{b=0}^{m-1} \partial C / \partial Z_{(i-a)(j-b)} \times W_{ab}$$

Notice this looks similar to our convolution operation from forward propagation step but instead of $Z_{(i+a)(j+b)}$ we have $Z_{(i-a)(j-b)}$, which is simply a convolution using W which has been flipped along both the axes.