Introduction To Neural Networks

ECEN 478

Dr. Mahmoud Nabil Mahmoud
mnmahmoud@ncat.edu

North Carolina A & T State University

March 23, 2022

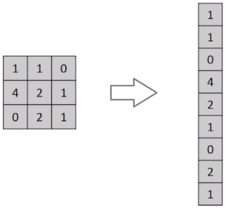
Outline

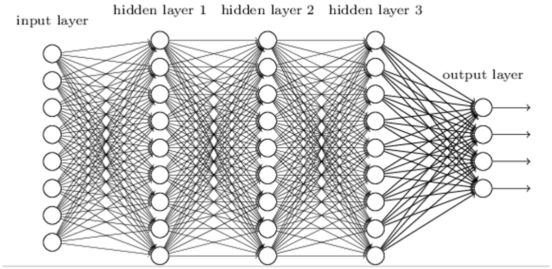
Convolutional Neural Networks (ConvNets)

- It is a special structure to deal with image inputs.

Convolutional Neural Networks (ConvNets)

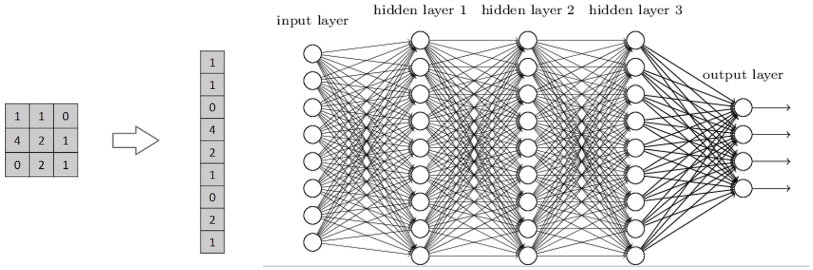
- It is a special structure to deal with image inputs.
- Why not just flatten the image and feed it to a Multi-Level Perceptron for classification purposes?





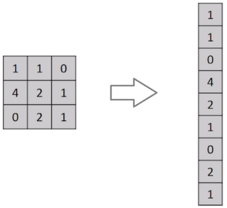
Convolutional Neural Networks (ConvNets)

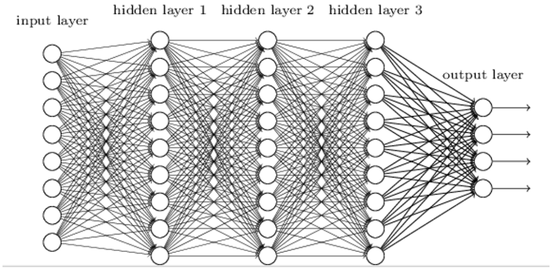
- It is a special structure to deal with image inputs.
- Why not just flatten the image and feed it to a Multi-Level Perceptron for classification purposes?



Convolutional Neural Networks (ConvNets)

- It is a special structure to deal with image inputs.
- Why not just flatten the image and feed it to a Multi-Level Perceptron for classification purposes?





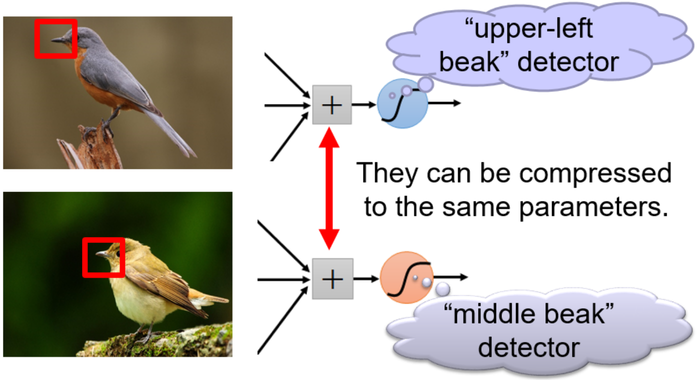
Do we really need all the edges?

Convolutional Neural Networks

- We've obtained a classification accuracy better than 98 percent, using training and test data from the MNIST handwritten digit data set.
- However, it's strange to use networks with fully-connected layers to classify images.
- Such an architecture does not take into account the spatial structure of the images. For instance, it treats input pixels which are far apart and close together exactly the same way.

Convolutional Neural Networks (ConvNets)

Spatial and Temporal dependencies between pixels. (Edges, patterns, curves ...)

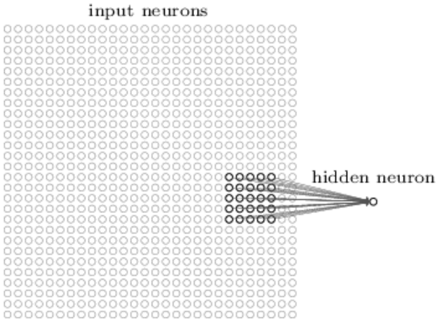

"upper-left beak" detector
They can be compressed to the same parameters.
"middle beak" detector

In MLP, Large network is needed for all possible positioning of the peak

Outline

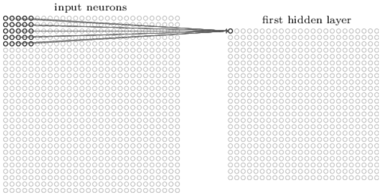
Local Receptive Fields

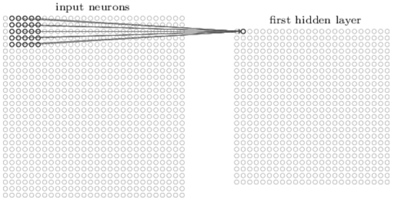
- We will not connect every input pixel to every hidden neuron. Instead, we only make connections in small, localized regions of the input image.
- That region in the input image is called the local receptive field for the hidden neuron.



Local Receptive Fields

- It's a little window on the input pixels. Each connection learns a weight.
- We then slide the local receptive field across the entire input image.
- Sometimes a different stride length is used. For instance, we might move the local receptive field 2 pixels at time.





Shared weights and biases

- It should be noted that, all the hidden neurons share the same weights and biases connected to its local receptive field.

Shared weights and biases

- It should be noted that, all the hidden neurons share the same weights and biases connected to its local receptive field.
- In other words, for the j, k^{th} hidden neuron, the output is

$$\sigma_{j,k}\left(b + \sum_{l=0}^{4}\sum_{m=0}^{4} w_{l,m} a_{j+l,k+m}\right)$$

Shared weights and biases

- It should be noted that, all the hidden neurons share the same weights and biases connected to its local receptive field.
- In other words, for the j, k^{th} hidden neuron, the output is

$$\sigma_{j,k}\left(b + \sum_{l=0}^{4} \sum_{m=0}^{4} w_{l,m} a_{j+l,k+m}\right)$$

  - σ is the neural activation function

Shared weights and biases

- It should be noted that, all the hidden neurons share the same weights and biases connected to its local receptive field.
- In other words, for the j, k^{th} hidden neuron, the output is

$$\underset{j,k}{\sigma}\left(b + \sum_{l=0}^{4}\sum_{m=0}^{4} w_{l,m}a_{j+l,k+m}\right)$$

  - σ is the neural activation function
  - $w_{l,m}$ is shared window of weights of size 5 by 5.

Shared weights and biases

- It should be noted that, all the hidden neurons share the same weights and biases connected to its local receptive field.
- In other words, for the j, k^{th} hidden neuron, the output is

$$\sigma_{j,k}\left(b + \sum_{l=0}^{4}\sum_{m=0}^{4} w_{l,m} a_{j+l,k+m}\right)$$

  - σ is the neural activation function
  - $w_{l,m}$ is shared window of weights of size 5 by 5.
  - b is the shared bias

Shared weights and biases

- It should be noted that, all the hidden neurons share the same weights and biases connected to its local receptive field.
- In other words, for the j, k^{th} hidden neuron, the output is

$$\sigma_{j,k}\left(b + \sum_{l=0}^{4}\sum_{m=0}^{4} w_{l,m} a_{j+l,k+m}\right)$$

  - σ is the neural activation function
  - $w_{l,m}$ is shared window of weights of size 5 by 5.
  - b is the shared bias
  - $a_{x,y}$ denote the input activation at position x,y

Shared weights and biases

- It should be noted that, all the hidden neurons share the same weights and biases connected to its local receptive field.
- In other words, for the j, k^{th} hidden neuron, the output is

$$\underset{j,k}{\sigma}\left(b + \sum_{l=0}^{4}\sum_{m=0}^{4} w_{l,m} a_{j+l,k+m}\right)$$

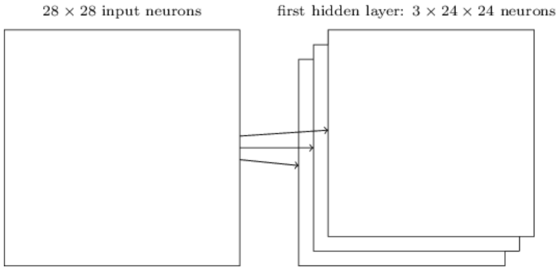
  - σ is the neural activation function
  - $w_{l,m}$ is shared window of weights of size 5 by 5.
  - b is the shared bias
  - $a_{x,y}$ denote the input activation at position x,y
- This means that all the neurons in the first hidden layer detect exactly the same feature just at different locations of the input image.

Popular Definitions

- We sometimes call the map from the input layer to the hidden layer a feature map.
- We call the weights defining the feature map the shared weights.
- We call the bias defining the feature map in this way the shared bias.
- The shared weights and bias are often said to define a kernel or filter.

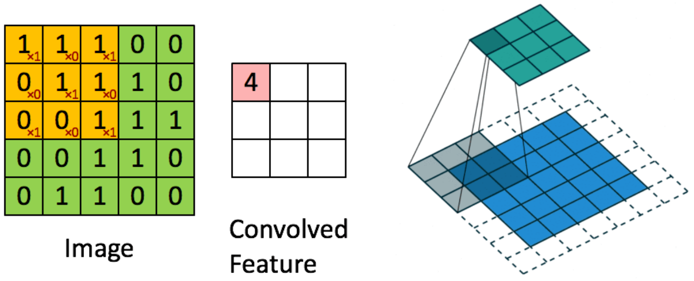
Network Structure

- One kernel (local receptive field) can detect just a single kind of localized feature.
- To do image recognition we'll need more than one feature map.


28×28 input neurons
first hidden layer: $3 \times 24 \times 24$ neurons

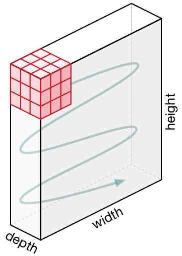
The Kernel (1/2)

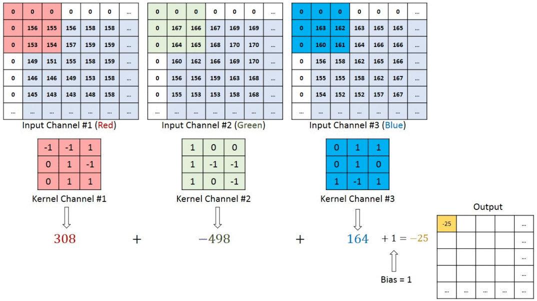
It is a small matrix of learnable weights which have a purpose to detect certain feature in the image (e.g. Horizontal edes).



The Kernel (2/2)

For RGB images kerenls can be 3d matrices.





0	0	0	0	0	0	...
0	156	155	156	158	158	...
0	153	154	157	159	159	...
0	149	151	155	158	159	...
0	146	146	149	153	158	...
0	145	143	143	148	158	...
...	...	...	...	...	...	...

Input Channel #1 (Red)

0	0	0	0	0	0	...
0	167	166	167	169	169	...
0	164	165	168	170	170	...
0	160	162	166	169	170	...
0	156	156	159	163	168	...
0	155	153	153	158	168	...
...	...	...	...	...	...	...

Input Channel #2 (Green)

0	0	0	0	0	0	...
0	163	162	163	165	165	...
0	160	161	164	166	166	...
0	156	158	162	165	166	...
0	155	155	158	162	167	...
0	154	152	152	157	167	...
...	...	...	...	...	...	...

Input Channel #3 (Blue)

-1	-1	1
0	1	-1
0	1	1

Kernel Channel #1

1	0	0
1	-1	-1
1	0	-1

Kernel Channel #2

0	1	1
0	1	0
1	-1	1

Kernel Channel #3

$$308 \quad + \quad -498 \quad + \quad 164 \quad + 1 = -25$$

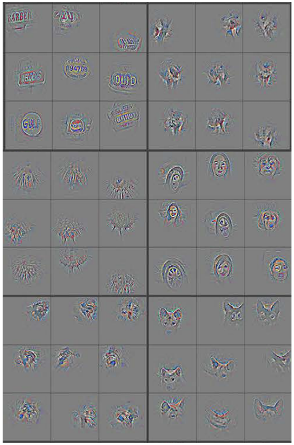
Bias = 1

Output

-25				...
				...
				...
				...
...	...	...	...	...

What are these kernels is learning

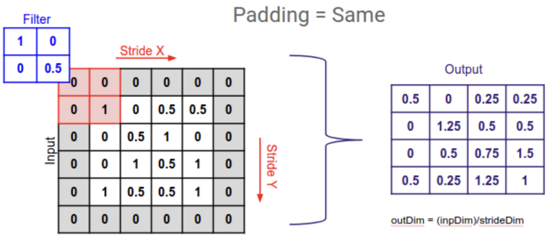
- Spatial features are learned many of the features have clear sub-regions if visualized.
- I suggest reading the paper Visualizing and Understanding Convolutional Networks by Matthew Zeiler and Rob Fergus (2013).



Outline

Padding

The purpose of padding is to preserve (as much as possible) the size of the input to prevent losing information.



1	0
0	0.5

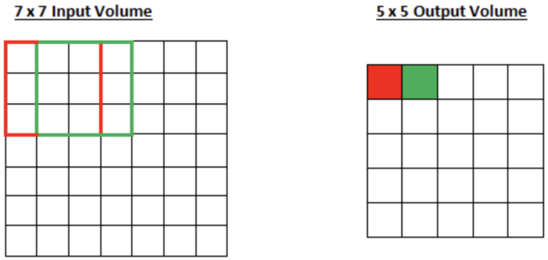
0	0	0	0	0	0
0	1	0	0.5	0.5	0
0	0	0.5	1	0	0
0	0	1	0.5	1	0
0	1	0.5	0.5	1	0
0	0	0	0	0	0

0.5	0	0.25	0.25
0	1.25	0.5	0.5
0	0.5	0.75	1.5
0.5	0.25	1.25	1

Padding is added to the outer frame of the image to allow for more space for the filter to cover in the image.

Stride

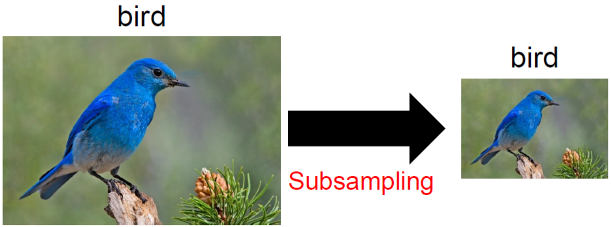
Stride is a parameter of the neural network's filter that modifies the amount of movement over the image



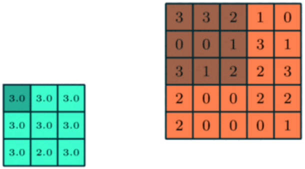
As the stride, or movement, is increased, the resulting output will be smaller.

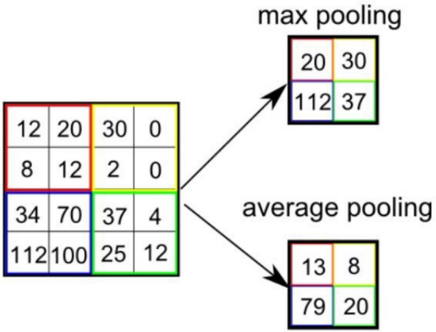
Pooling Layer

- Pooling layer is responsible for reducing the spatial size of the Convolved Feature.
- This is to decrease the computational power required to process the data.
- Sub-sampling pixels will not change the object.
- Simply pooling layer takes each feature map and prepares a condensed feature map.



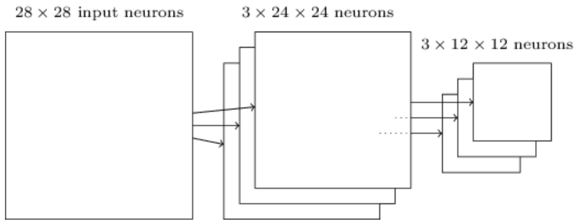
Pooling Types





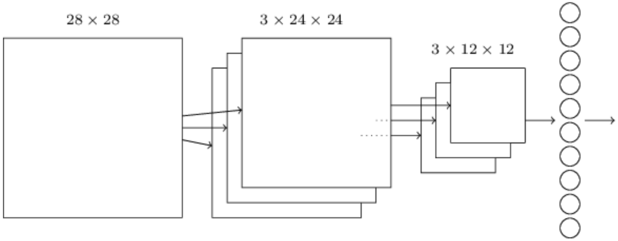
Pooling Intuition

First we detect the localized feature using our kernel, then we can throw away the exact positional information.


28×28 input neurons
$3 \times 24 \times 24$ neurons
$3 \times 12 \times 12$ neurons

This helps reduce the number of parameters needed in later layers

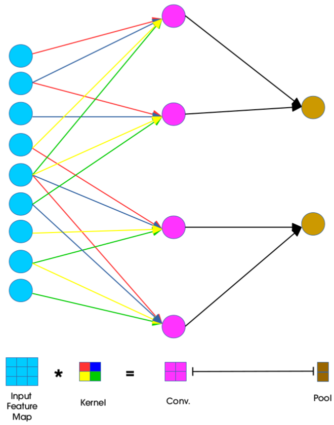
Putting it all together



Note, the final layer of connections in the network is a fully-connected layer (flattened).

CNN vs MLP

A **big** advantage of sharing weights and biases is that it greatly reduces the number of parameters involved in a convolutional network.



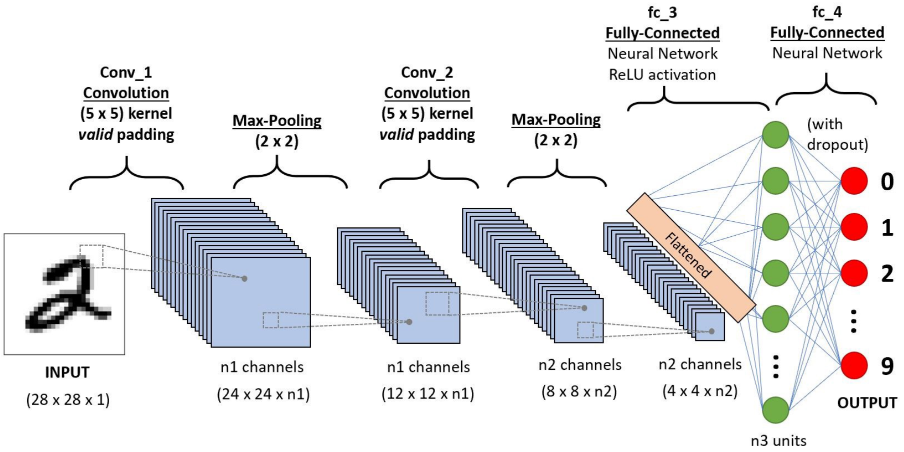
In CNN, very few weights to learn compared to MLP without loss of

Exercise

For MNIST dataset each image is 28x28 pixels, compare the number of parameters needed by only the first layer in case of MLP with 100 hidden units and CNN with 20 kernels each is 5x5.

Convolutional Neural Networks Architecture (ConvNets)

ConvNets have the ability to learn filters/characteristics.

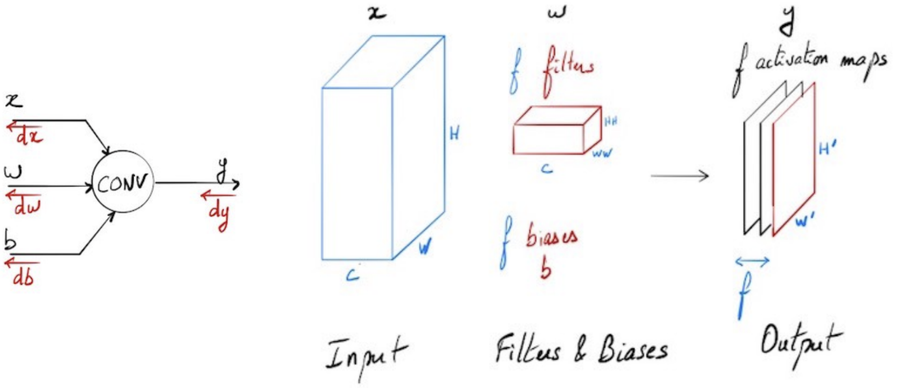


Outline

The Backward Pass Intution

- The backward pass is where convolutions get a bit trickier.
- We need to compute two set of gradients for each convolutional layer
    - The partial derivative of the loss with respect to each element of the input to the convolution operation
    - The partial derivative of the loss with respect to each element of the filter

CNN Forward Pass



- x: Input data of shape (N, C, H, W)
- w: Filter weights of shape (F, C, HH, WW)
- b: Biases, of shape $(F,)$
- y: Output data shape (N, F, H', W')

Outline

Trivial case: input x is a vector (1 dimension)

We are looking for an intuition of how it works

Input

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

$$w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$$

$$b$$

Output

$$y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

$$\begin{aligned} y_1 &= w_1x_1 + w_2x_2 + b \\ y_2 &= w_1x_2 + w_2x_3 + b \\ y_3 &= w_1x_3 + w_2x_4 + b \end{aligned}$$

Back propagation

Lets denote the gradient of our cost function L with respect to y as $\frac{dL}{dy}$:

$$dy = \begin{bmatrix} \frac{\partial L}{\partial y_1} & \frac{\partial L}{\partial y_2} & \frac{\partial L}{\partial y_3} \end{bmatrix}$$
$$dy = \begin{bmatrix} dy_1 & dy_2 & dy_3 \end{bmatrix}$$

We are looking for

$$dx = \frac{\partial L}{\partial x}, dw = \frac{\partial L}{\partial w}, db = \frac{\partial L}{\partial b}$$

Bias Gradient

$$db = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial b} = dy \cdot \frac{\partial y}{\partial b}$$

Using the chain rule

$$db = \sum_j \frac{\partial L}{\partial y_j} \cdot \frac{\partial y_j}{\partial b} = \begin{bmatrix} dy_1 & dy_2 & dy_3 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

$$\Rightarrow db = dy_1 + dy_2 + dy_3$$

As expected the bias gradient is scalar

Weights Gradient

$$dw = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial w} = dy \cdot \frac{\partial y}{\partial w}$$

$$\frac{\partial y}{\partial w} = \begin{bmatrix} \frac{\partial y_1}{\partial w_1} & \frac{\partial y_1}{\partial w_2} \\ \frac{\partial y_2}{\partial w_1} & \frac{\partial y_2}{\partial w_2} \\ \frac{\partial y_3}{\partial w_1} & \frac{\partial y_3}{\partial w_2} \end{bmatrix} \qquad \frac{\partial y}{\partial w} = \begin{bmatrix} x_1 & x_2 \\ x_2 & x_3 \\ x_3 & x_4 \end{bmatrix}$$

$$\begin{bmatrix} dy_1 & dy_2 & dy_3 \end{bmatrix} \cdot \begin{bmatrix} x_1 & x_2 \\ x_2 & x_3 \\ x_3 & x_4 \end{bmatrix}$$

$$\begin{aligned} dw_1 &= x_1 dy_1 + x_2 dy_2 + x_3 dy_3 \\ dw_2 &= x_2 dy_1 + x_3 dy_2 + x_4 dy_3 \end{aligned}$$

As expected the gradient is two dimension

Weights Gradient

We can notice that dw is a convolution of the input x with a filter dy.

$$dw = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} * \begin{bmatrix} dy_1 \\ dy_2 \\ dy_3 \end{bmatrix}$$

Input Gradient

$$dx = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial x} = dy^T \cdot \frac{\partial y}{\partial x}$$

$$\frac{\partial y}{\partial x} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \frac{\partial y_1}{\partial x_3} & \frac{\partial y_1}{\partial x_4} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} & \frac{\partial y_2}{\partial x_3} & \frac{\partial y_2}{\partial x_4} \\ \frac{\partial y_3}{\partial x_1} & \frac{\partial y_3}{\partial x_2} & \frac{\partial y_3}{\partial x_3} & \frac{\partial y_3}{\partial x_4} \end{bmatrix} \qquad \frac{\partial y}{\partial x} = \begin{bmatrix} w_1 & w_2 & 0 & 0 \\ 0 & w_1 & w_2 & 0 \\ 0 & 0 & w_1 & w_2 \end{bmatrix}$$

$$\begin{aligned} dx_1 &= w_1 dy_1 \\ dx_2 &= w_2 dy_1 + w_1 dy_2 \\ dx_3 &= w_2 dy_2 + w_1 dy_3 \\ dx_4 &= w_2 dy_3 \end{aligned}$$

It is four dimension as expected.

Input Gradient

Once again, we can have a convolution. But it is more complex this time.

$$dx = \begin{bmatrix} 0 \\ dy_1 \\ dy_2 \\ dy_3 \\ 0 \end{bmatrix} * \begin{bmatrix} w_2 \\ w_1 \end{bmatrix}$$

Outline

Two Dimension CNN

Input

$$x = \begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \\ x_{41} & x_{42} & x_{43} & x_{44} \end{bmatrix}$$

$$w = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix}$$

$$b$$

Output

$$y = \begin{bmatrix} y_{11} & y_{12} & y_{13} \\ y_{21} & y_{22} & y_{23} \\ y_{31} & y_{32} & y_{33} \end{bmatrix}$$

$$\begin{aligned} y_{11} &= w_{11}x_{11} + w_{12}x_{12} + w_{21}x_{21} + w_{22}x_{22} + b \\ y_{12} &= w_{11}x_{12} + w_{12}x_{13} + w_{21}x_{22} + w_{22}x_{23} + b \\ &\dots \end{aligned}$$

Where the output can be written as:

$$y_{ij} = \left(\sum_{k=1}^{2} \sum_{l=1}^{2} w_{kl} x_{i+k-1, j+l-1} \right) + b \quad \forall (i,j) \in \{1,2,3\}^2$$

Bias Gradient

The incoming gradient from the output is

$$dy_{ij} = \frac{\partial L}{\partial y_{ij}}$$

The bias gradient

$$db = dy_{ij} \cdot \frac{\partial y_{ij}}{\partial b}$$

Summation on i and j. And we have:

$$\forall(i,j) \quad \frac{\partial y_{ij}}{\partial b} = 1$$

$$db = \sum_{i=1}^{3} \sum_{j=1}^{3} dy_{ij}$$

Weight gradient

$$dw = \frac{\partial L}{\partial y_{ij}} \cdot \frac{\partial y_{ij}}{\partial w} = dy \cdot \frac{\partial y}{\partial w}$$

In other words

$$dw_{mn} = dy_{ij} \cdot \frac{\partial y_{ij}}{\partial w_{mn}}$$

We are looking for

$$\frac{\partial y_{ij}}{\partial w_{mn}}$$

Weights Gradient

Using the convolution formula

$$\frac{\partial y_{ij}}{\partial w_{mn}} = \sum_{k=1}^{2}\sum_{l=1}^{2} \frac{\partial w_{kl}}{\partial w_{mn}} x_{i+k-1,j+l-1}$$

All terms

$$\frac{\partial w_{kl}}{\partial w_{mn}} = 0$$

Except for (k,l)=(m,n) where it's 1, case occurring just once in the double sum. Hence:

$$\frac{\partial y_{ij}}{\partial w_{mn}} = x_{i+k-1,j+l-1}$$

Weights Gradient

We now have:

$$dw_{mn} = dy_{ij} \cdot x_{i+k-1,j+l-1}$$

$$\Rightarrow dw_{mn} = \sum_{i=1}^{3} \sum_{j=1}^{3} dy_{ij} \cdot x_{i+k-1,j+l-1}$$

If we look closely, this is a convolution function!

Weights Gradient

We now have:

$$dw_{mn} = dy_{ij} \cdot x_{i+k-1,j+l-1}$$

$$\Rightarrow dw_{mn} = \sum_{i=1}^{3} \sum_{j=1}^{3} dy_{ij} \cdot x_{i+k-1,j+l-1}$$

If we look closely, this is a convolution function! Thus,

$$dw = \begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \\ x_{41} & x_{42} & x_{43} & x_{44} \end{bmatrix} * \begin{bmatrix} dy_{11} & dy_{12} & dy_{13} \\ dy_{21} & dy_{22} & dy_{23} \\ dy_{31} & dy_{32} & dy_{33} \end{bmatrix}$$

$$dw = x * dy$$

Input gradient

Again we have

$$dx_{mn} = dy_{ij} \cdot \frac{\partial y_{ij}}{\partial x_{mn}}$$

This time, we are looking for

$$\frac{\partial y_{ij}}{\partial x_{mn}}$$

Using the convolution equation we want to claculate:

$$\frac{\partial y_{ij}}{\partial x_{mn}} = \sum_{k=1}^{2} \sum_{l=1}^{2} w_{kl} \frac{\partial x_{i+k-1,j+l-1}}{\partial x_{mn}}$$

Input gradient

Again we have

$$dx_{mn} = dy_{ij} \cdot \frac{\partial y_{ij}}{\partial x_{mn}}$$

This time, we are looking for

$$\frac{\partial y_{ij}}{\partial x_{mn}}$$

Using the convolution equation we want to claculate:

$$\frac{\partial y_{ij}}{\partial x_{mn}} = \sum_{k=1}^{2} \sum_{l=1}^{2} w_{kl} \frac{\partial x_{i+k-1,j+l-1}}{\partial x_{mn}}$$

Input gradient

We now have:

$$\frac{\partial x_{i+k-1,j+l-1}}{\partial x_{mn}} = \begin{cases} 1 & \text{If } m = i+k-1 \text{ , } n = j+l-1 \\ 0 & \text{Otherwise} \end{cases}$$

$$\begin{cases} m = i+k-1 \\ n = j+l-1 \end{cases}$$

$$\Rightarrow \begin{cases} k = m-i+1 \\ l = n-j+1 \end{cases}$$

In our example, range sets for indices are:

$m, n \in [1,4]$	inputs
$k, l \in [1,2]$	filters
$i, j \in [1,3]$	outputs

When we set k=m-i+1, we are going to be out of the defined boundaries:

$$(m\text{-}i+1) \in [-1,4]$$

Input gradient

Thus,

$$\frac{\partial y_{ij}}{\partial x_{mn}} = w_{m-i+1,n-j+1}$$

Aggregating all the gradients affect $x_{m,n}$

$$dx_{mn} = \sum_{i=1}^{3} \sum_{j=1}^{3} dy_{ij} \cdot w_{m-i+1,n-j+1}$$

An example,

$$\begin{aligned}
dx_{11} &= \sum_{i=1}^{3} \sum_{j=1}^{3} dy_{ij} \cdot w_{2-i,2-j} \\
&= \sum_{i=1}^{3} dy_{i1} w_{2-i,1} + dy_{i2} w_{2-i,0} + dy_{i3} w_{2-i,-1}, \\
&= dy_{11} w_{1,1} + dy_{12} w_{1,0} + dy_{13} w_{1,-1} \\
&+ dy_{21} w_{0,1} + dy_{22} w_{0,0} + dy_{23} w_{0,-1} \\
&+ dy_{31} w_{-1,1} + dy_{32} w_{-1,0} + dy_{33} w_{-1,-1}
\end{aligned}$$

Note that all weights are zero in this example except $w_{1,1}$

Input Gradient

It was concluded that:

$$dx = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & dy_{11} & dy_{12} & dy_{13} & 0 \\ 0 & dy_{21} & dy_{22} & dy_{23} & 0 \\ 0 & dy_{31} & dy_{32} & dy_{33} & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} w_{22} & w_{21} \\ w_{12} & w_{11} \end{bmatrix}$$

$$dx = dy\_0 * w'$$

Summary of backprop equations

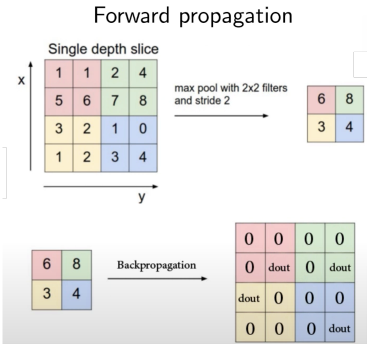
$$db = \sum_{i=1}^{3}\sum_{j=1}^{3} dy_{ij}$$

$$dw = \begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \\ x_{41} & x_{42} & x_{43} & x_{44} \end{bmatrix} * \begin{bmatrix} dy_{11} & dy_{12} & dy_{13} \\ dy_{21} & dy_{22} & dy_{23} \\ dy_{31} & dy_{32} & dy_{33} \end{bmatrix} = x * dy$$

$$dx = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & dy_{11} & dy_{12} & dy_{13} & 0 \\ 0 & dy_{21} & dy_{22} & dy_{23} & 0 \\ 0 & dy_{31} & dy_{32} & dy_{33} & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} w_{22} & w_{21} \\ w_{12} & w_{11} \end{bmatrix} = dy\_0 * w'$$

Gradient of the Pooling Layer

- There are no weights to update, only have to propagate the gradient through
- In max pooling, backpropagated gradient is assigned only to the **Wining Pixels** i.e., the one which had the maximum value in the pooling block. This can be kept tracked of in the forward pass.
- In the average pooling, the backpropagated gradient is **divided by the area of pooling block** $(k \times k)$ **and equally assigned to all the pixels in the block**



Questions