6-12-2021 Monday

Week#15 Lecture#19

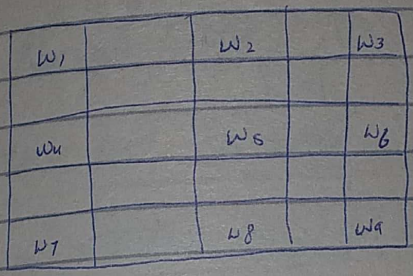
CNN

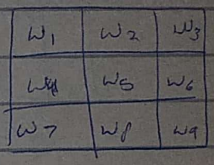
we could consider 1 filter as one Neuron

we won't reshape the image when sending into a filter layer / Convolution layer.

Dilated Convolution:

Dilation = 1





5x5 image

before Dilation = 0

Now

Resultant

i) Image size gets small size

ii) with small filter we scan bigger Image.

iii) less Computation

iv) bcz neighbour fields would be same, we scan quickly.

1x1 Convolution:

1x1 filter size

No padding required

with more channels

1x1x3 filter size

Used to reduce or enhance input size.

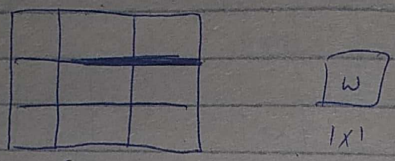


Image
3x3 Here no size reduction.

e.g:

consider 224x224x70 = output size of one convolutional layer.

we can convert it to

224x224x1

by using filter

1x1x70.

& increase size

32 x 32 x 3

Convert to

32 x 32 x N

by using filter

1 x 1 x 3

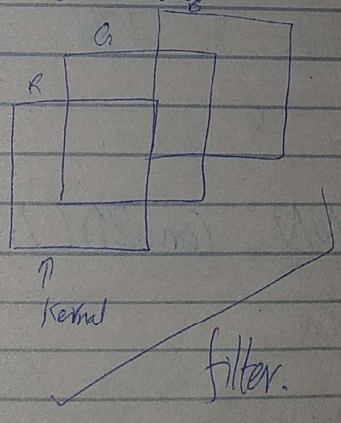
& No. of filters N in one convolution layer.

we increase & or decrease depth of input.

No. of filters = kernal

1 filter of 2D is called kernal

1 filter of 3D have 3 2D kernals



filter.

Conv1D() ; 1D filter

Conv2D() ; 2D filter

Conv3D() ;

↓

if input had size 32x32x20

Then filter size will be 3x3x20

instead ~~Size~~ in this we use

3x3x3 stride in channels (depth) as well

Stride was already in rows & columns in image.

when input & filter size depth, then automatically we use Conv3D

First make stride in depth dimension & we stride in row & column.

Code of CNN Conv2D():

image size = 4x4

filter size = 3x3

Stride = 1
Padding = 0

```
Conv2D(Image I, filter f):
    Feature_Map = np.zeros((I.height - f.height + 1,
                            I.width - f.width + 1))
    for i in range(
    Ir, Ic = I.shape[0], I.shape[1]
    fr, fc = f.shape[0], I.shape[1]
    out = np.zeros((Ir - fr + 1, Ic - fc + 1))
    for i in range(Ir - fr + 1):
        for j in range(Ic - fc + 1):
            out[i][j] = np.Sum(f * I[i:i+fr, j:j+fc])
            #out[i][j] = np.Sum(f * I[i:i+fr, j:j+fc, :])
```

all channels for more channels

$f * I[i:i+f_r, j:j+f_c, :]$

Zero strides in channels/depth

Gradients for filters.

a	b	c	d
e	f	g	h
i	j	x	l
m	n	o	p

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

first time gradients.

dw

a	b	c
ed	ef	g.
i	j	k

Second time gradients

dw

b	c	d
f	g	h
n	o	l

Now add these dw.

Code:-

```
gradeW[]
gradeW.append(IPart)
```

if Image size = 4×4 & filter size = 3×3

then we have 4 gradients & At the end we

Darknet "YOLO V1 & Implementation" Project

add all values in these to one 2x2 gradient

Correlation:

when we multiply filter with image part this process is called correlation

Convolution:

if we rotate filter 180° degrees then do correlation, This concept is called convolution

reverse order multiplication is called convolution

Assumption that filter is 180 degrees rotated already & we only do correlation