unsa.maqui@nu.edu.pk

Computer Vision

* **Image**: a 2D array
  contains spatial coordinates i.e x and y
* **grayscale image**, an image is defined in black & white, the range is 0 – 256 i.e 2^8, it is the range of shades between white & black i.e grays.
* **Sampling**: to choose a specific dimension of image before it is preprocessed (resolution)
* **quantization**: to choose the pixel value range i.e 2^1, 2^2 ... , 2^8 etc. (bit quantization)
* **Image Enhancement**: improving image quality
  * noise reduction
  * edge highlighting

→ Types of image enhancement operations:
- **point/pixel** operations: operations done on specific pixels
- **local** operations: operations on neighbour pixels of a specific pixels as well.
- **global** operations: operations on whole image

* Common Pixel Operation:

→ **Image Negative**
  * reverse the gray level order
  * for L gray levels, transformation function is:

$$s = T(r) = (L-1) - r$$

'r' is the pixel value.

→ **Image Scaling**. (contrast adjustment)
  * $s = T(r) = a \cdot r$ where a is a constant

→ Log Transformation
→ Power Law Transformation } home tasks

* **Image Averaging for Noise Reduction** → home task
  → To remove noise, take many images and take their average, the resulting image has less noise.
* Image **subtraction**, **multiplication**, **OR operations**

Convolutional Neural Network

→ core idea: update the weights in order to minimize the loss of target parameter

NN Layers

1 - Fully connected layers
2 - convolution layers
3 - Batch Normalization
4 - Local response normalization layer (obsolete)

Multichannel:

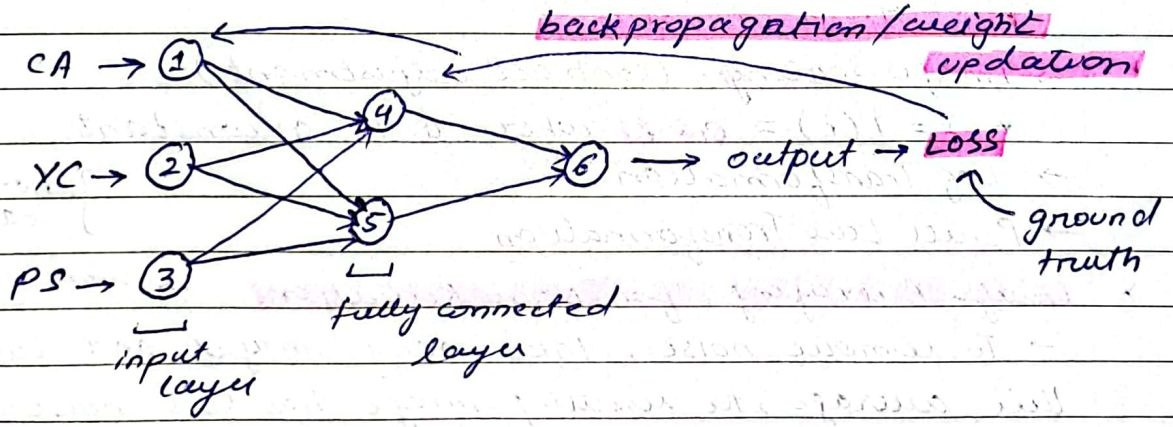
- dropout layer
- merge layer

Relu: if $x \leq 0$ $f(x) = 0$; else $f(x) = x$ $\Big\}$ $f(x) = \max(0, x)$

Example: predict price

covered area = CA
year of construction = YC
plot size = PS

Deep NN structure:



weights:

w_{14}, w_{15} w_{46}, w_{56}
w_{24}, w_{25}
w_{34}, w_3

for neuron 4:

$$N4 = (CA \cdot w_{14} + YC \cdot w_{24} + PS \cdot w_{34})\,AF$$

(AF: activation function)

for neuron 5:

$$N5 = AF(CA \cdot w_{15} + YC \cdot w_{25} + PS \cdot w_{35})$$

for neuron 6:

$$N6 = AF(N4 \cdot w_{46} + N5 \cdot w_{56})$$

1	2	1	0	2
2	0	0	1	0
1	0	2	1	0
0	1	0	2	1
0	2	1	0	2

image 5×5

convolution operation ↓ ×

1	0	1
0	1	0
1	0	1

Kernel: 3×3

=

5	3	6
2	6	2
5	3	7

→ multiply and add all

size of output image:

$$\text{output size} = \frac{\text{Input size} - \text{Kernel size} + 2(\text{padding})}{\text{stride}} + 1$$

$$= \frac{5 - 3 + 2(0)}{1} + 1 \quad \text{for } x \& y$$

$$= 3$$

e.g. 6 × 7 image, 3×3 kernel

$$x = \frac{6 - 3 + 2(0)}{1} + 1 = 4$$

$$y = \frac{7 - 3 + 2(0)}{1} + 1 = 5$$

Padding techniques:

→ Zero padding: add zeros in borders

→ Standard padding: add some border values as pads.

* To decrease the size of image i.e **down scale** we can use:
→ **stride** : increase the kernel slide count
→ **pooling**

* **Edge detection** : home task, convolution operation
vertical, horizontal, padding, pooling, stride

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

→ vertical edge detection kernel

* **Pooling:**

max-pooling

8	3	5	9
4	7	2	4
6	5	2	1
1	7	3	6

4×4

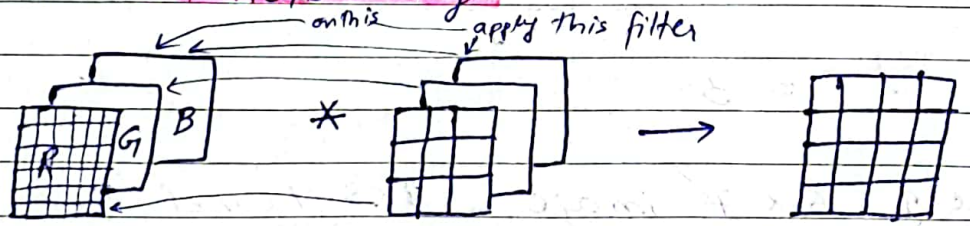
→

8	9
7	6

2×2

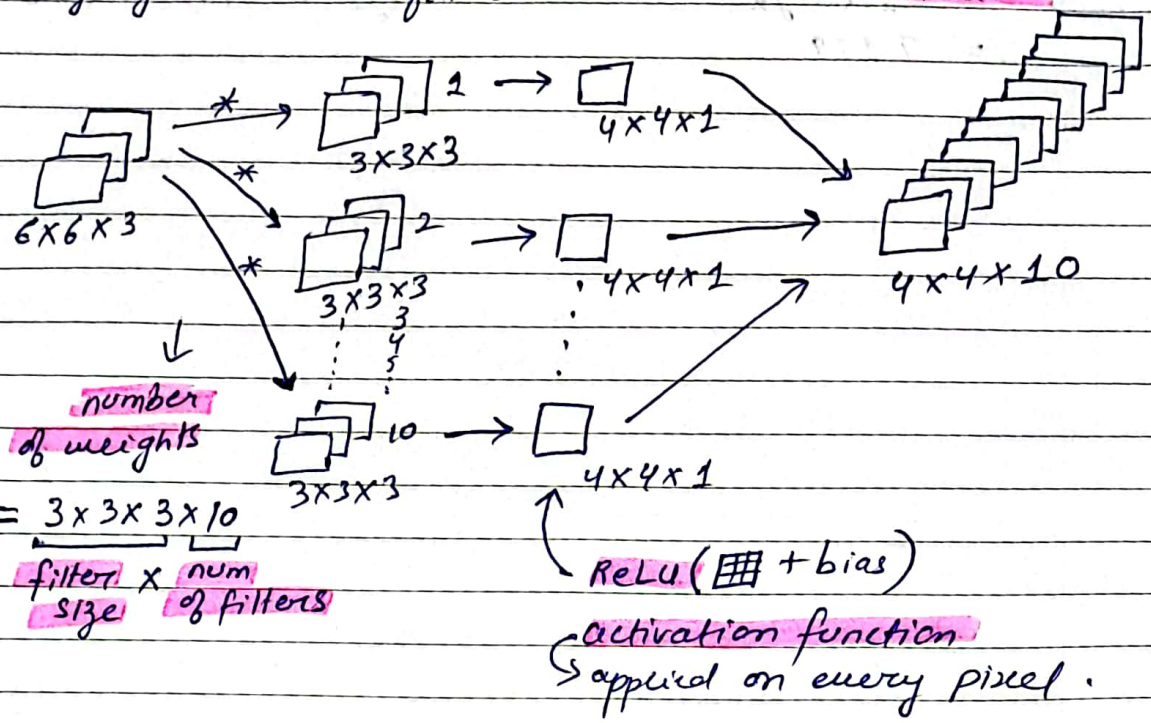
parameters : not customizable e.g kernel, weights
hyper-parameters : customizable e.g kernel size, stride, padding, pooling method.

Convolution on RGB Image



* **seperate filter** for every channel.
* value got from channel R = x, G = y, B = z
* so, final output value = x+y+z = (9).

e.g. you use 10 filters. this how the architecture works:



* pooling doesn't affect the number of the trainable parameters as there is no kernel

convolutional layer + pooling
is considered as a convolutional layer as a whole because the pooling layer has no trainable parameters.

$$Softmax(z_i) = \frac{e^{z_i}}{\underbrace{\sum_{j=1}^{n} e^{z_j}}_{PI}}$$ → probability distribution for outputs classes (logits)

Now, when you have outputs, calculate loss.

$$\text{cross entropy loss} = -\sum y_i \log(p_i)$$
(y_i: ground truth, p_i: predicted output)

Example:

output layer

○ → 4 → $softmax(4) = \frac{e^4}{e^4+e^9+e^{-3}} = 0.00669$

○ → 9 → $softmax(9) = 0.993 = \frac{e^9}{e^4+e^9+e^{-3}}$

○ → -3 → $softmax(-3) = 0.0000061 = \frac{e^{-3}}{e^4+e^9+e^{-3}}$

sum of all = 0.0069 + 0.99 + 0.0000061 => 1

$CEL = -1(\log_2(0.0069) - 0(\log_2(0.99)) - 0(\log_2(0.0000061))$

$= 7.179$

* Dense Layer:
  fully connected layer.

* AlexNet : 60 Million parameters.
  → overfits.
  to fix it: → data augmentation.
  ↓
→ Dropout layer.

→ Image normalization

* Inception model.

GoogleNet

FaceNet

① Face Recognition
② Contrastive Loss
③ Siamese Network
④ Clustering.

input
$6 \times 6 \times 32$

filter
$\#(3 \times 3 \times \#32)$ (6 filters)

output
$6 \times 6 \times 6$

$$\frac{6-6+2(0)}{1}+1$$

$$\frac{6-3+2(1)}{1}+1$$

$$\frac{6-1+2(0)}{1}+1$$

---

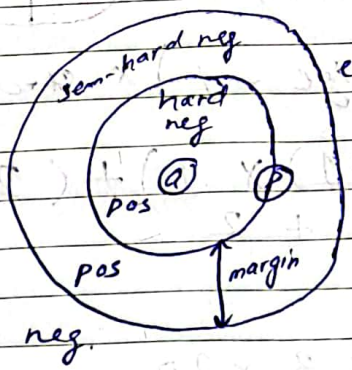
facenet.

d	$d(a,p)$	$d(a,n)$
45	32	99

anchor positive + margin < anchor negative.

$32 + 45 < 99$.

$77 < (99)$



(a) anchor
(P) positive.

* loss will be zero in easy negative.
* we want our model to not be trained on easy negatives as it won't contribute to model training.

hard ← $ap = 100$, $an = 90$, $\alpha = 10$ $\quad 10 < 20$

$100 - 90 + 10 = 20$ (loss = 20).

semi hard ← $ap = 100$, $an = 105$

$100 - 105 + 10 = 5$ (loss = 5). $\quad (10 > 5)$

$ap - an + \alpha = loss.$

d(a,p)	d(a,n)	α	loss.
32	99	45	$\max(0, -22)$ easy
24	51	45	18 semi
76	23	45	98 hard.
34	89	45	$\max(0, -10)$ easy

overall loss = 116

* if dist b/w a & p greater than a & n then hard triplet
* if dist b/w a&p + α and still less than a & n then easy
* if dist b/w a&p < a&n but a&p + α >= a&n semi-hard

* Transpose Convolution (depth wise)

output = [(input size − 1) x stride] + [(kernel size − 2 x padding)]

= ((2 − 1) x 1) + (2 − 2 x 0)

= 1 + 2

= 3

Input 100 x 100 x 3 x 20

x x y x d x n ← batch size.

filter 5 x 5 x 60, stride = 2

$\frac{199}{2}$

$\frac{198}{}$

$[(100 + 1) \times 2] + (5 - 2 \times 0)$

198 + 5

203

$= \frac{100 - 5 + 2(0)}{2} + 1 \Rightarrow 48$

output size = 48 x 48 x 3 x 20