~~Week 4~~

Course 4 :- Convolutional Neural Networks. [23.01.19].

week 1 :- Foundation of Convolutional Neural Networks.

L-1 :- Computer vision. [10.00 Am 23.01.19]

Computer vision problems.

- image classification.
- object detection.
- Neural style transfer. e.t.c.

** main challenge of computer vision is higher input space.

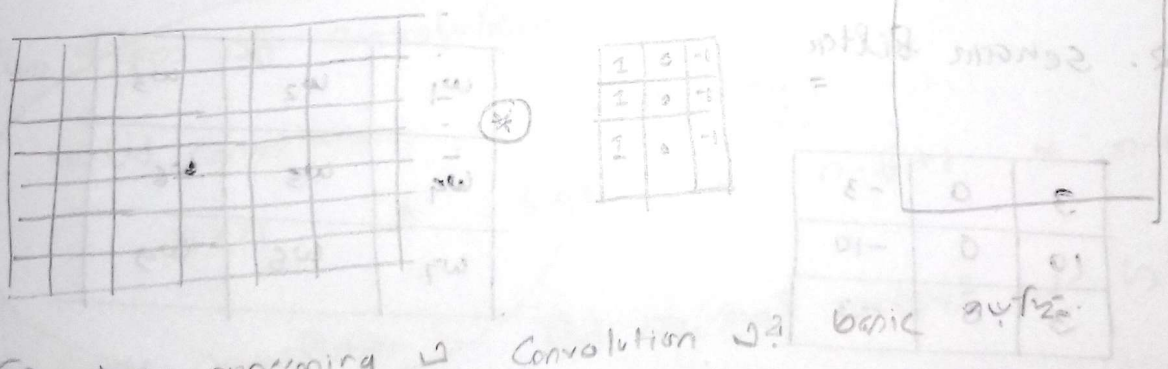
[Convolution operation is the fundamental building block of Convolutional neural network]

L-2 : Edge detection example. [10.10 Am 23.01.19]

python :- Conv_forward
tensorflow : tf.nn.conv2d.
keras : conv2D.

vertical edge detection :-

6 × 6 image



(**) image processing ↳ Convolution ↳ basic operation.

* A vertical edge is a region where bright region on the left and dark region on the right.

(simple operation)

L3: More Edge detection. [10:25 Am 23.01.19]

Matrix for detecting vertical edge →

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

(vertical edge detector)

matrix for detecting horizontal edges →

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

(Horizontal edge detector)

other filter to detect edges.

1. sobel filter.

** Add extra weight to the center of the image.

1	0	-1
2	0	-2
1	0	-1

2. Scharr filter.

3	0	-3
10	0	-10
3	0	-3

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

** Our goal is to find best choices of this nimes parameters.

→ use backpropagation to find out best parameters.

[it is one of the most powerful ideas of computer vision].

L-4 padding [10:40 Am 23.01.19]

If we convolve a $(n \times n)$ matrix with a $(f \times f)$ matrix then size of resultant matrix will be. $(n-f+1) \times (n-f+1)$.

$$\begin{cases} n \times n \quad \circledast \quad f \times f \rightarrow (n-f+1) \times (n-f+1) \\ 6 \times 6 \quad * \quad 3 \times 2 \rightarrow 4 \times 4. \end{cases}$$

Disadvantages:-

1. It shrinks the ~~img~~ image after each convolution.
2. Corners pixels of an image touched only once. [A lot of information on the edges are thrown away].

** To solve these problems we use padding.

** Adding extra pixels around the image is called padding. Normally we add 0 as padding pixel.

Two types of convolutions,

1. valid convolutions :- (No padding)

$$n \times n \quad * \quad f \times f \rightarrow n-f+1 \quad * \quad n-f+1.$$

2. Same convolution :- Pad so that output size is the same as the input size.

$$\text{input size} \rightarrow n+2p-f+1 \quad \times \quad n+2p-f+1.$$

[p = Padding amount]

If we want input and output size same,

$$n+2p-f+1 = n.$$

$$\therefore p = \frac{f-1}{2}.$$

[By using this equation we can find suitable padding size].

** filter size is usually odd.

L-5 Strided Convolutions. [11:10 Am 23.01.19]

[we will skip/jump over S → Stride number of rows and columns.]

$n \times n$ image $f \times f$ filter.

Padding p stride s.

output size:

$$\left[\frac{n+2p-f}{s}+1\right] \times \left[\frac{n+2p-f}{s}+1\right]$$

→ Floor sign of it is not an integer then we will take floor.

→ Cross-correlation vs. Convolution.

[Convolution a filter matrix को normally double flip करके गुणा करते हैं।]

L-6 Convolutions Over Volume. [11:25 Am 23.01.19]

(RGB image पर convolution कैसे करते हैं।)

→ Convolution on RGB images:

→ channel of image and filter must be equal.

[Computation समझें]

* we can use multiple filters on the image.

number of channel

$n \times n \times n_c$ * $f \times f \times n_c$ → $n-f+1 \times n-f+1 \times n'_c$

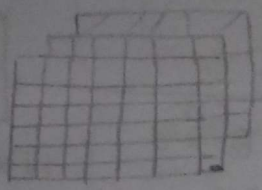
$6 \times 6 \times 3$ $3 \times 3 \times 3$ → $4 \times 4 \times 2$

number of filters.

* depending on the number of filter we can detect/use multiple features. (number of features we are detecting)

L-7 one layer of convolutional Network [11:45 Am 24-01-19]

Example of a layer :-



6×6×3

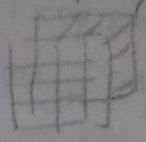
Filters

* 3×3×3 → ReLU (4×4 + b) → 4×4

$w^{[1]} a^{[0]}$ R

3×3×3 → ReLU (4×1+b)

4×4



4×4×(2)

→ because of 2 filters

** If you have 10 filters that are 3×3×3 in one layer of neural network, how many parameters does that layer have?

3×3×3 = 27 parameters.

+ bias = 28 "

(28×10) = 280 parameters.

** one property of convolutional neural net is that number of parameters is independent of the (input/size) of the image.

Summary of notation :-

If layer l is a convolutional layer :-

$f^{[l]}$ = filter size.

$p^{[l]}$ = padding.

$s^{[l]}$ = stride

$n_c^{[l]}$ = number of filters.

Each filter size = $f^{[l]} \times f^{[l]} \times n_c^{[l-1]}$

weights : $f^{[l]} \times f^{[l]} \times n_c^{[l-1]} \times n_c^{[l]}$.

bias : $1 \times 1 \times 1 \times n_c^{[l]}$.

Activation : $a^{[l]} \rightarrow n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$

Activations from Previous Layer:

Input :- $n_H^{[l-1]} \times n_W^{[l-1]} \times n_c^{[l-1]}$

output :- $n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$.

↳ Number of filters in layer l

Size :-

$$n_H^{[l]} = \left\lfloor \frac{n_H^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \right\rfloor$$

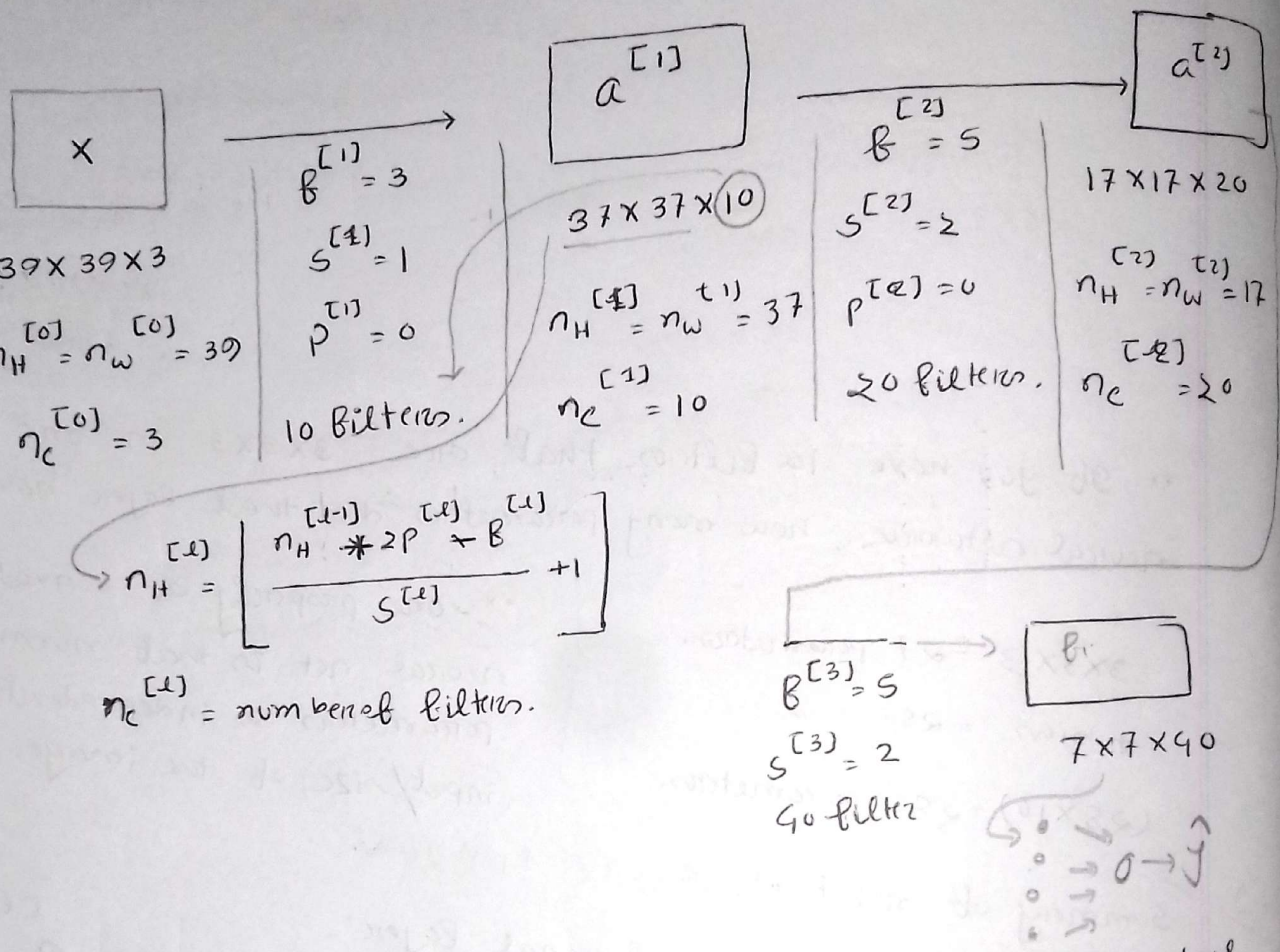
$$n_W^{[l]} = \left\lfloor \frac{n_W^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \right\rfloor$$

$A^{[l]} = m \times n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$

↳ vectorized.

L-8 Simple Convolutional Network Example. [12.20 pm 24.01.19]

Example ConvNet :- (Earlier Notation Quiz Example from Week 4)



** As we go deeper of the network size decreases but the number of channel increases.

** Some of the hyperparameters of the convolutional neural network are.

1. Filter size.
2. Number of filter.
3. Padding size.
4. Stride size.

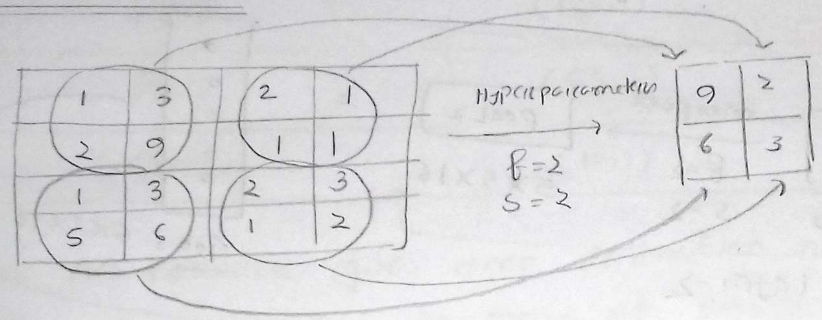
Types of layer in a convolutional network :-

- Convolution. (conv)
- pooling (POOL)
- Fully connected. (FC)

L-9 pooling layers. [12.40 pm 24.01.19]

Max pooling :- (used frequently).

[max pooling works better in different applications] এখনো কোন Theory নাই।



** No parameters to learn.

→ আগের মতই

$$\left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor$$

** size of the max pooled matrix

ঐ region এর maximum

** Matrix size, stride এবং padding choose করে filter হবে

Average pooling :- In average pooling we do not take the maximum but take average of the number inside a window

Summary of pooling.

$n_H \times n_w \times n_c$

↓ (Assume $p=0$)

$$\left\lfloor \frac{n_H - f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n_w - f}{s} + 1 \right\rfloor \times n_c$$

Hyperparameters :-

1. f : filter size. } usually f = 2, s = 2.
2. s : stride.

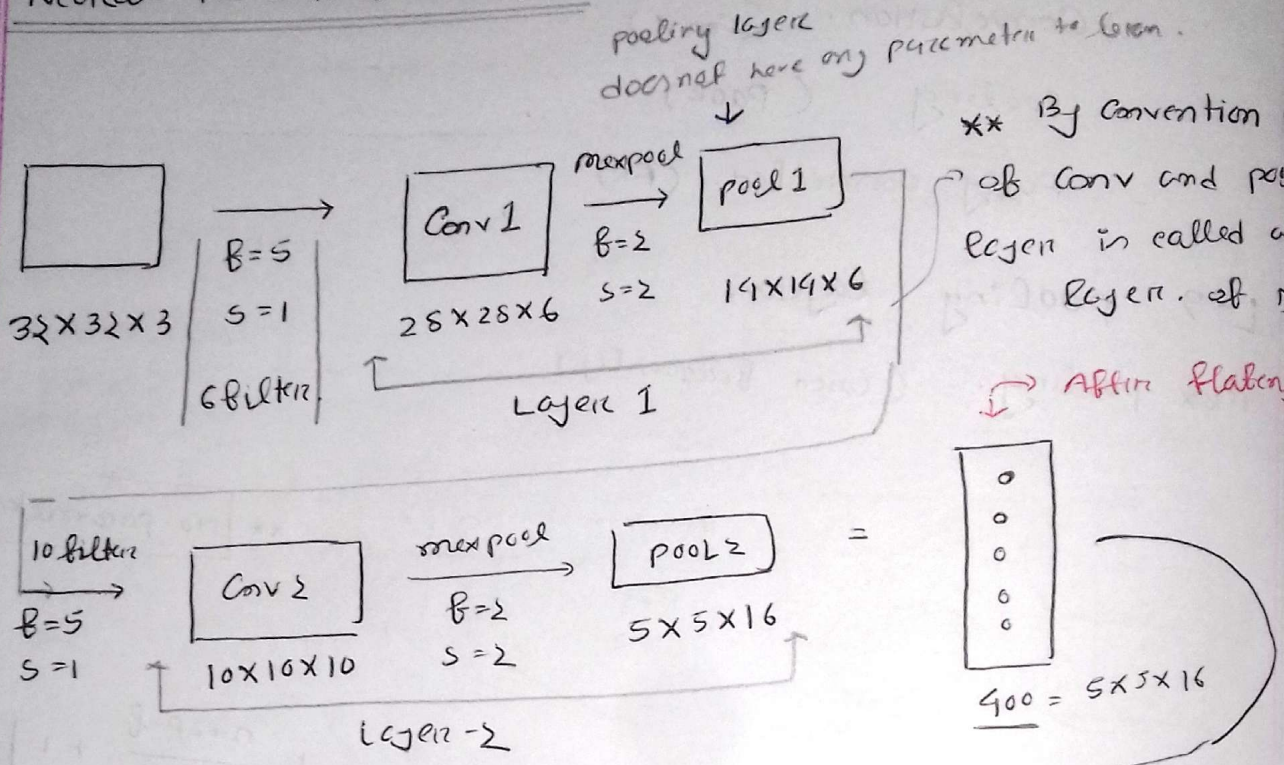
max or average pooling.

** [max pooling usually do not use padding]

** [pooling is applied in each channel independently]

L-10 CNN Example. [1.00 pm 24.01.19]

Neural Network example. [Lenet-5].



Common pattern :- [Conv-pool, Conv-pool. fc-fc-fc - softmax].

** See the concrete example of how others do it and try to use it in your applications.

	Activation Shape	Activation Size	# parameters.
input :	(32, 32, 3)	3072	0
Conv1 (f=5, s=1)	(28, 28, (8)) 8 filters.	6272	208 (25+1)×8
pool1 (f=2, s=2)	(14, 14, 8)	1568	0
Conv2 (f=5, s=1)	(10, 10, (16)) 16 filters	1600	416 (25+1)×16
pool2 f=2, s=2	(5, 5, 16)	400	0
FC3	(120, 1)	120	48001 - (400×120)+1
FC4	(84, 1)	84	10,081 - (120×84)+1
Softmax	(10, 1)	10	841 (84×10+1)

observations

- ** As network goes deep activation size shrinks.
- ** No parameters to learn at pooling layers.
- ** Maximum parameters are in fully connected layers.

L-11 why convolutions? [6.45 pm 29.01.19].

Main advantages of Convolution layer over fully connected layer are.

1. Parameter sharing :- A feature detector (such as vertical edge detector) that's useful in one part of image is probably useful in another part of the image.

2. Sparsity of Connections :- In each layer, each output value depends only on a small number of inputs.

** Convolutional layer requires small number of parameter compare to fully connected layer.

→ Translation invariance.

→ Training set $(x^{(1)}, y^{(1)}), \dots (x^{(m)}, y^{(m)})$

image → Conv → pool → FC → $\hat{y}$

Cost $J = \frac{1}{m} \sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)})$

use gradient descent to optimize parameters to reduce J.

→ Programming Assignment.

Conv-forward ( ) → এই function এ pad এবং stride calculate করে implement করেছি

In this assignment, we will implement

1. helper functions that will use when implementing tensorflow model.

2. Implement a fully connected ConvNet using tensorflow.

[Build and train a ConvNet in Tensorflow for a classification problem]

** built in functions for implementing forward propagation in tf

1. tf.nn.conv2d (X, W1, strides = [1,s,s,1], padding = 'SAME') :-

→ given an input X and group of filters W1, this function convolves W1's filter on 'X'. Third input [1,s,s,1] represents stride for each dimension of the input (m, n_H_prev, n_W_prev, n_C_prev).

2. tf.nn.max_pool (A, ksize = [1,f,f,1], strides = [1,s,s,1], padding = 'SAME')

→ given an input A, this function uses a window of size (f,f) and strides of size (s,s) to carry out max pooling over each window.

3. tf.nn.relu(Z1).

→ Computes elementwise relu of Z1 which can be any shape.

4. tf.contrib.layers.flatten (P).

→ Given an input P, this function flattens each example into a 1D vector while maintaining batch size. It returns a flattened tensor with shape [batch_size, k] (k → After flattening size of the vector.)

5. tf.contrib.layers.fully_connected (F, num_outputs).

→ give the flattened input F, it returns the output computed using a fully connected layer.

** (In fully connected layer we do not need to initialize weights, tensorflow model automatically initialize it).

6. tf.nn.softmax_cross_entropy_with_logits (logits = z3, labels = Y).

→ gt calculates softmax entropy loss. This function both computes softmax activation function and resulting loss.

7. tf.reduce_mean

→ Computes mean of elements across dimension of a tensor. use this to sum the losses over all the examples to get overall cost.

[open source Community to [illegible]] to get noticed.

Week 2 : Deep Convolutional models : Case studies. [26.01.19]

P1 : Case studies.

L-1 : Why look at case studies. [4.00 pm 26.01.19]

Outline :

1. Classic networks :
   - LeNet-5.
   - Alex Net
   - VGG

   } You can read research papers for these neural network architectures which may be useful for your task also.

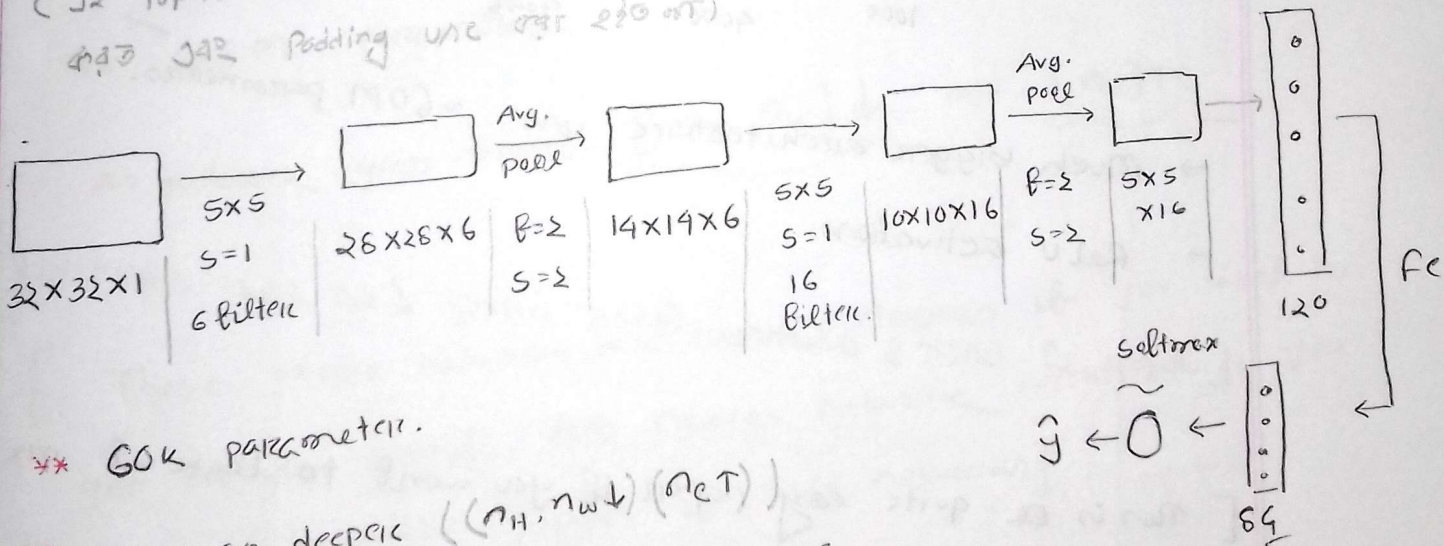
2. ResNet (Residual Network) [It has 152 layers]
3. Inception Neural Network.

L-2 : Classic Networks. [4.05 pm 26.01.19]

→ Paper এর নাম (1998)

1. LeNet-5. (useful for recognizing handwritten digit)

(এই Paper এখন implement করা প্রয়োজন তাহলে Avg. pool use করতে হবে Padding use করা হয় না)



** 60K parameters.

** As we go deeper ($n_H, n_W \downarrow$) ($n_C \uparrow$)

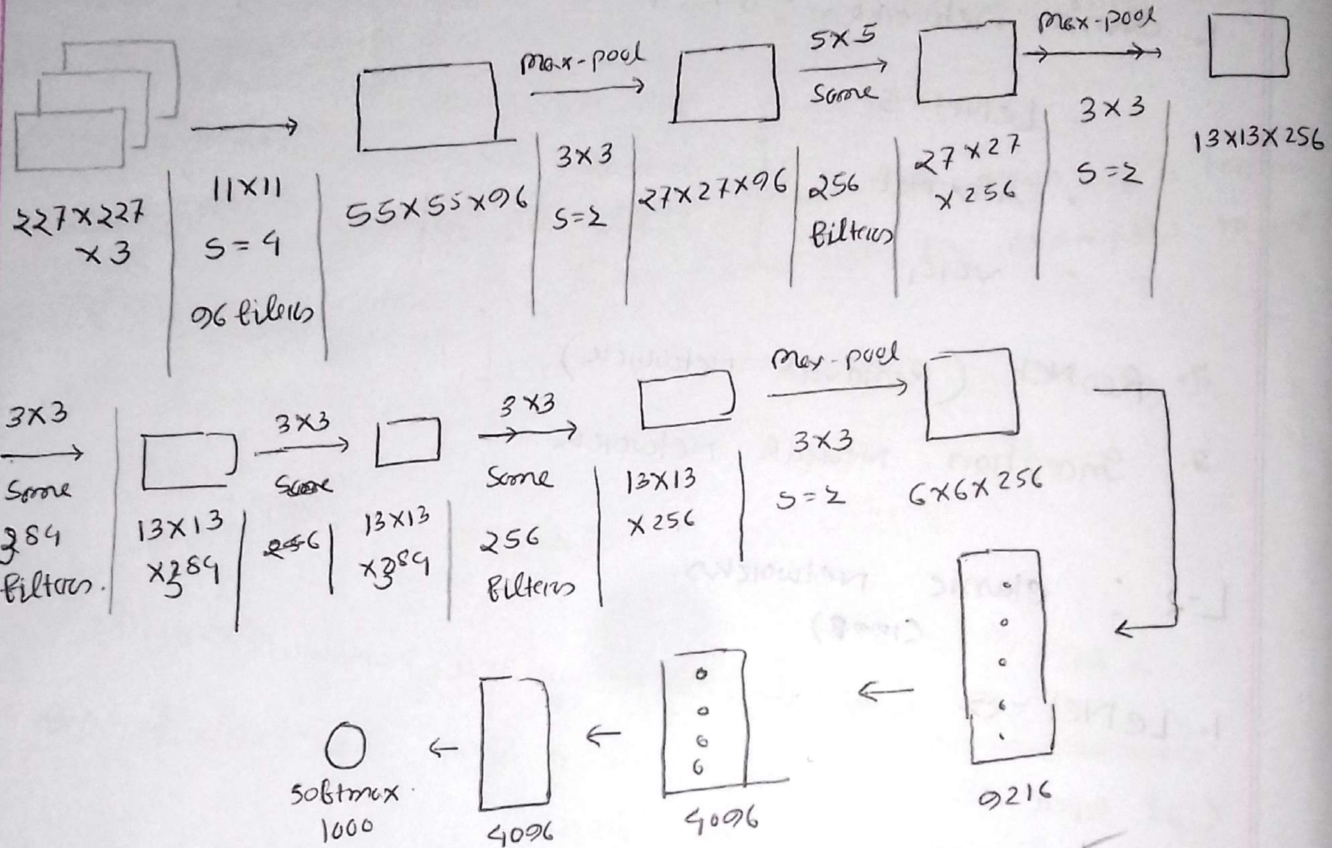
** Conv pool conv pool fc fc output.

Scanned by CamScanner

→ ImageNet classification with Deep Convolutional Neural Networks.

2. AlexNet.

Architecture of AlexNet.



→ much bigger architecture use ~60M parameters.

→ ReLU activations.

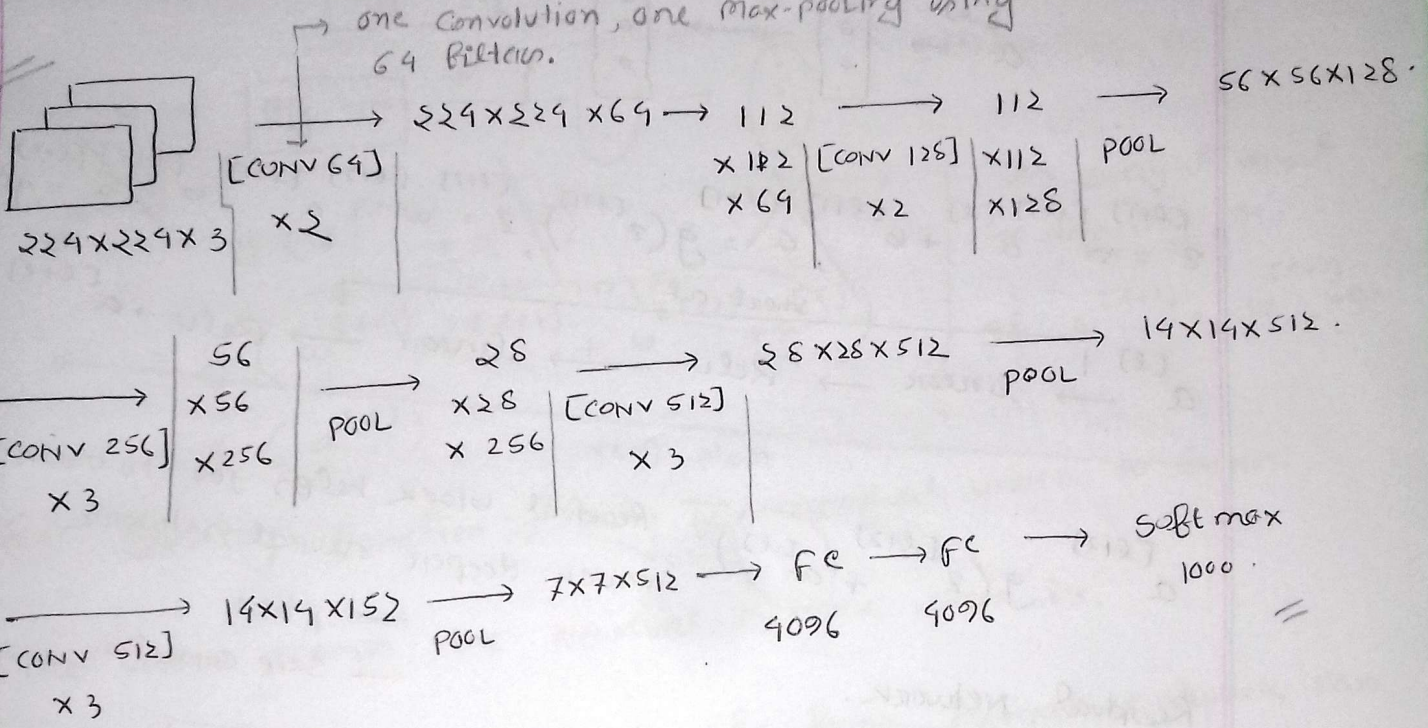
[This is a quite easy paper if you want to look at or start with]

(very Deep Convolutional networks for Large-scale Image Recognition).

↳ Paper wise

3. VGG-16

CONV = 3×3 filters, s=1, same MAX-POOL = 2×2, s=2.



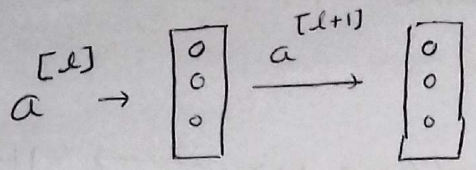
** 138 M parameters.

* As Network goes deeper $[n_H, n_W] \downarrow$ and $[n_c] \uparrow$

[These three networks are classic networks if you want to get an idea about deep neural network architecture you can go through papers of this networks]

L-3 ResNets. [8.00 pm 26.01.19]

[useful to train a very deep neural networks]



$z^{[l+1]} = w^{[l+1]} \cdot a^{[l]} + b^{[l+1]}$, $a^{[l+1]} = g(z^{[l+1]})$, $z^{[l+2]} = w^{[l+2]} \cdot a^{[l+1]} + b^{[l+2]}$, $a^{[l+2]} = g(z^{[l+2]})$

shortcut / skip connection.

$a^{[l]} \rightarrow$ linear $\rightarrow$ ReLU $\rightarrow$ linear $\rightarrow$ ReLU $\rightarrow a^{[l+2]}$.

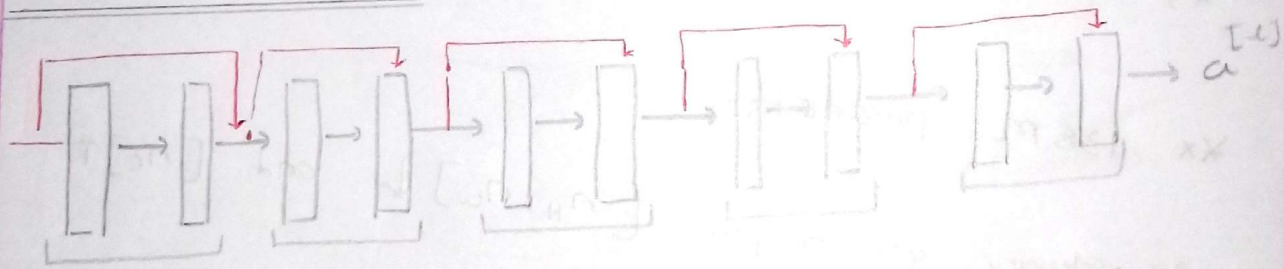
Main path.

$a^{[l+2]} = g(z^{[l+2]} + a^{[l]})$ Residual block helps you to train much deeper neural network.

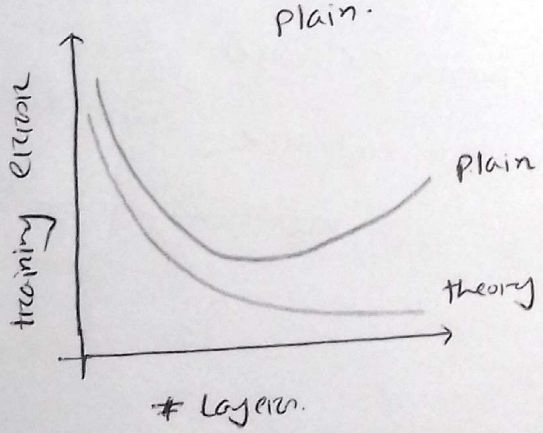
Residual network.

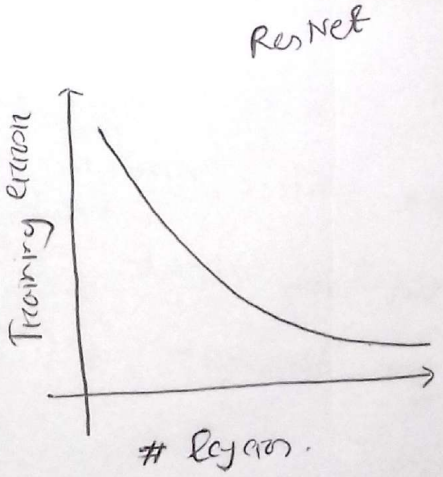
plain

skip connection



** Residual blocks.

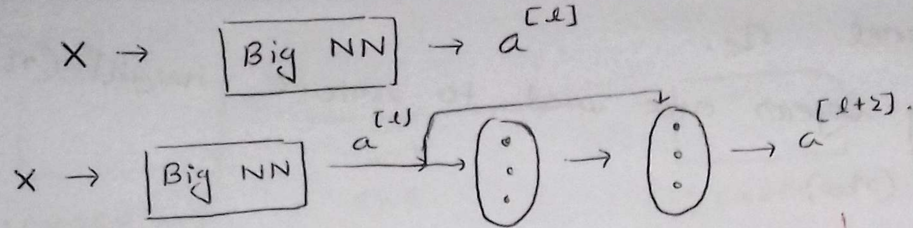


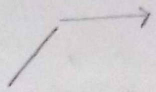


[This really helps with vanishing and exploding gradient Problems and allows to train much deeper neural network]

L-4 why ResNet works. [8:20 pm 26.01.19]

why do residual network work?





$$a^{[l+2]} = g(z^{[l+2]} + a^{[l]})$$
$$= g(w^{[l+2]} . a^{[l+1]} + b^{[l+2]} + a^{[l]}).$$
$$= g(a^{[l]})$$

As weight must be decaying so.
If $w^{[l+2]} = 0$, $b^{[l+2]} = 0$.

(**) Identity function is easy for residual block to learn.

The main reason the residual network work is,

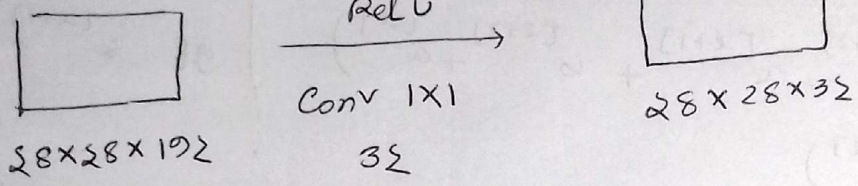
1. Extra layers of the ~~identity func~~ neural network can learn identity function. easily.
2. Do not hurt the performance of NN, also helps to improve

[Residual Network এর মধ্যে input এর output dimension যদি same থাকে ও দিতে সমান ধরতে হবে]

L-5 Networks in Networks and 1×1 Convolutions. [3:50 pm 27.01.19]

** 1×1 convolutions are used to shrink number of channel. n_c.

** Pooling layers are used to shrink height (n_H) and width (n_W).

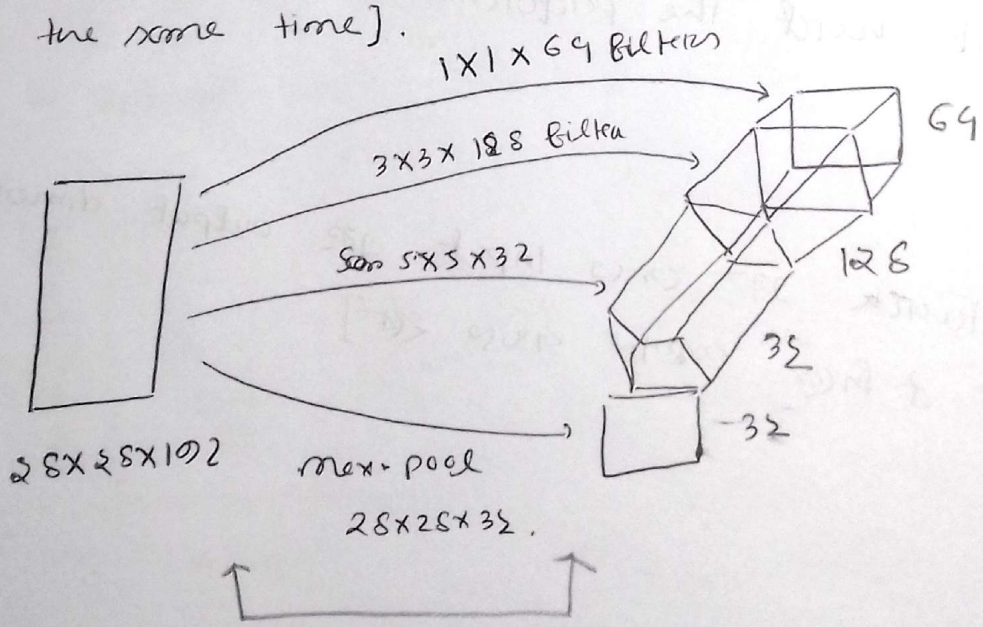


[shrinks the number of channel from 192 to 32.

[→ Allows to increase, decrease or keep same number of Channels of the volume]

L-6 Inception Network motivation. [4.05 pm 27.01.19]

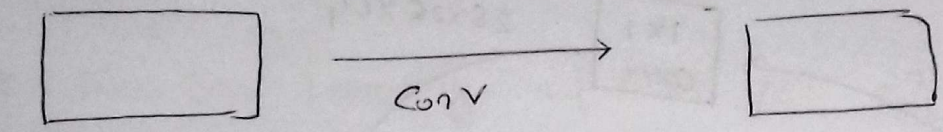
[The heart of inception network is that do convolutions with multiple size filter, max-pooling in one layer at the same time].



** To keep dimension same in input and output use same convolution.

* Here we are Convolving and pooling with multiple size of filters at the same time

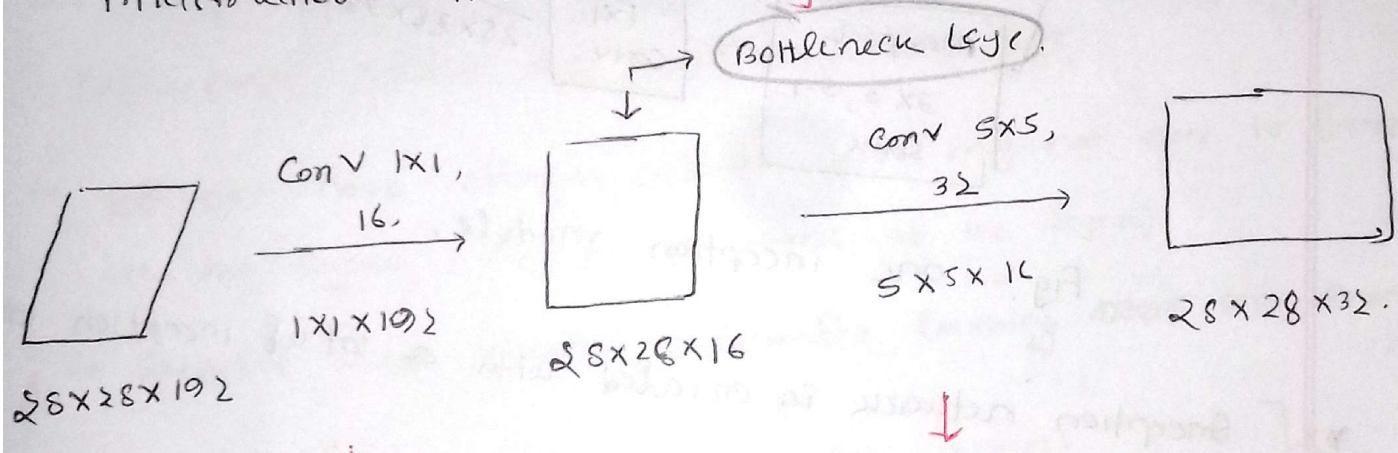
[Computational cost is much higher].



→ Huge number of multiplication

Total multiplication = $28 \times 28 \times 32 \times 5 \times 5 \times 192 = 120\,m$

** [This computation can be reduced by factor of 10 by using intermediate 1×1 Convolutions.]



$28 \times 28 \times 16 \times 192 = 2.4\,m$

$28 \times 28 \times 32 \times 5 \times 5 \times 16 = 10\,m.$

∴ Total multiplication = 12.4 m.

→ [Shrinking the number of channel in bottleneck layer does not hurt the performance much].

L-7 Gnception Network. [4.30 pm 27.01.19]

Gnception module.

(একটি ব্লকের মতো)

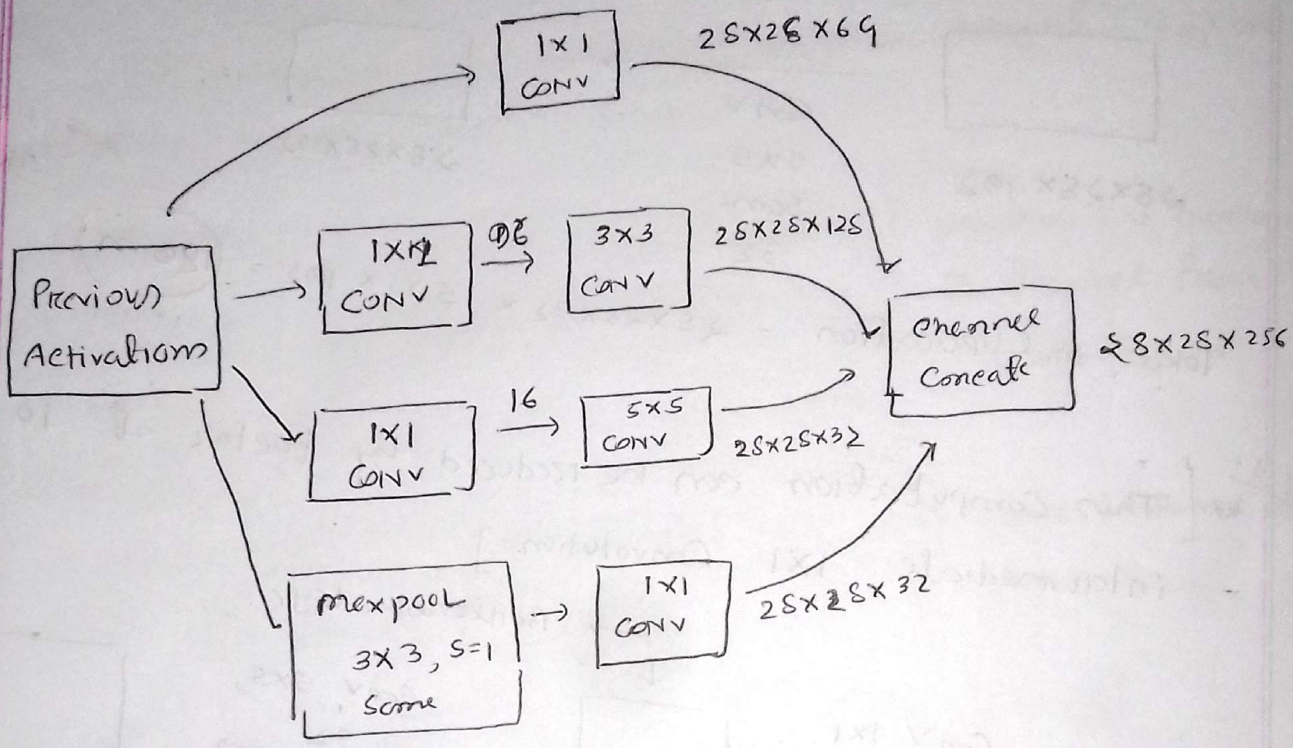


Fig :- one inception module.

** [Gnception network is created with a lot of inception block]

[Slide 9 এর ছবি দিয়ে কিছু ব্যাখ্যা দেওয়া আছে]

[inception Network related paper আছে]

P2: Practical advices for using ConvNets.

L-1 using open source implementation. [11.50 pm 27.01.19]

[open source implementation কিভাবে use করব].

→ open source implementation নিয়ে project start করা better.

L-2 Transfer Learning. [11.57 pm 27.01.19]

[** weights net থেকে download করে কিভাবে network এ pre-train করে transfer learning হিসেবে ব্যবহার করব.

** নিজের training set এ data কম থাকলে pretrain weight গুলো download করে just নিজের মত softmax layer change করে নিব.

** There are several ways to freeze intermediate layers. তবে কোন search করে implement করতে হবে.

→ If you have enough dataset you may not need to freeze all the layers. (may freeze some of the layers).

[** Computer vision problem এ transfer learning ব্যবহার করা possible].

L-3 Data Augmentation. [12.10 Am 28.01.19].

[Data augmentation is often used to improve performance of a computer vision system]

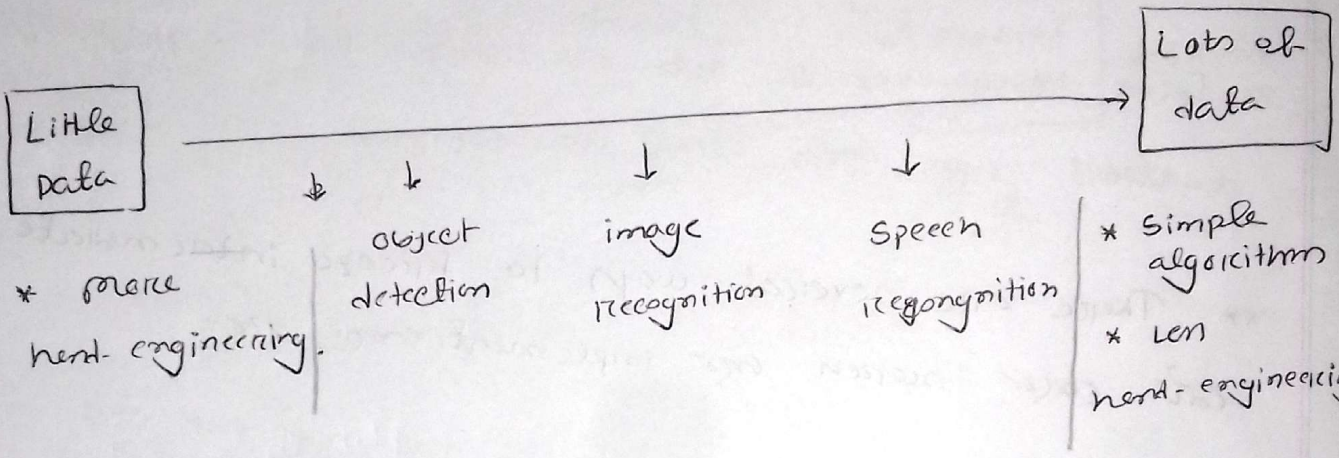
Common data augmentation method.:

1. mirroring.
2. Random Cropping.
3. Rotation.
4. Shearing e.t.c.

** Color shifting is another data agumentation method. Changing the (R, G, B) values of the pictures.

L-4 State of Computer vision [12.30 pm 28.01.19]

Data vs. hand engineering.



Two sources of knowledge.

- Labeled data.
- Hand engineered features / network architecture / other components.

→ If we have little data then learning may help.

** Tips for doing well on benchmarks / winning Competitions.

1. Ensembling.

→ Train several networks indepenty and average their outputs.

2. Multi-Crop at test time.

- Run classifier on multiple versions of test images and average results.

3. Use open source code.

- Use architecture of networks published in the literature.
- Use open source implementation if possible.
- Use pretrained models and fine-tune on your dataset.

Programming Assignment :-

- will learn how to use keras.

→ keras is even higher level language than tensorflow.

** To train and test a model, there are 4 steps in keras,

1. create the model.
2. Compile the model calling,

model.compile(optimizer="...", loss="...", metrics=["accuracy"])

3. Train the model on train data by calling,

model.fit(x="....", y=...., epochs=...., batch_size=....).

4. Test the model on test data by calling

model.evaluate(x=...., y=...)

[Resnet implementation github a unzip).

ResNet-50 model :- (Computer vision & Apply pernicle)

- Zero-padding pads the input with a pad of (3,3)

Stage 1 :-

→ The 2D Convolution has 64 filters of shape (7,7) and uses a stride of (2,2). name Conv1.

↔ BatchNorm is applied to channel axis of the input.

- Maxpooling with (3,3) window and (2,2) stride.

Stage 2 :-

→ The convolutional block uses three set of filters of size [64,64,256], "f" is 3, "s" is 1 and block is "a".

→ The 2 identity block uses three set of filters of size [64,64,256], "f" is 3, and blocks are "b" and c.

Stage 3 :-

→ The convolutional block uses three set of filters of size [128,128,512], "f" is 3, "s" is 2 and block is "a".

→ The 3 identity block uses three set of filters of size [128,128,512], 'f' is 3 and block as "b", "c", "d"

Stage 4 :-

→ The convolutional block uses three set of filters of size [256, 256, 1024], "f" is 3, "s" is 2 and block is "a".

→ The 5 identity block use three set of filters [256, 256, 1024] "f" is 3, blocks are "b", "c", "d", "e", "f".

Stage 5 :-

→ The convolutional block uses three set of filters of size [512, 512, 2048], "f" is 3, "s" is 2 and block is "a".

→ The 2 identity block uses three set of filters of size. [512, 512, 2048], "f" is 3 and blocks are "b" and "c".

- The 2D Average pooling uses a window of shape (2,2) and its name is avg_pool.
- Then flatten the output.
- The fully connected layer reduces its input to the number of classes using a softmax activation.

[Implementation details of ResNet-50]

v.v.g.

** [deep-learning model repository of github v.v.g]

1. VGG 16
2. VGG 19
3. ResNet50
4. Inception v3.
5. CRNN for music tagging.

अलग implement
कर सकते

week-3 object detection. [

L-1 object loclization [11.00 pm 30.01.19].

[Localization is the process of finding any object within an image]. → we have to put a bounding box around the object.

classification :- cal classify an image based on some category. such as whether an image contains a car or not.

classification with localization :- Find an object within the image. There will be only one image some where middle of the image. and put a bounding box around the image.

Detection :- Localize multiple ~~image~~ object of multiple category within an image.

[slide এ ছবি দিয়ে ধারণা দেওয়া হয়েছে].

Defining the target label y.

1. pedestrain
2. car.
3. motorcycle.
4. background.

Need to output b_x, b_y, b_h, b_w, class label (1-4)

$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

p_c → Is there any object ?

b_x, b_y, b_h, b_w } bounding box parameters.

c_1, c_2, c_3 } class parameters.

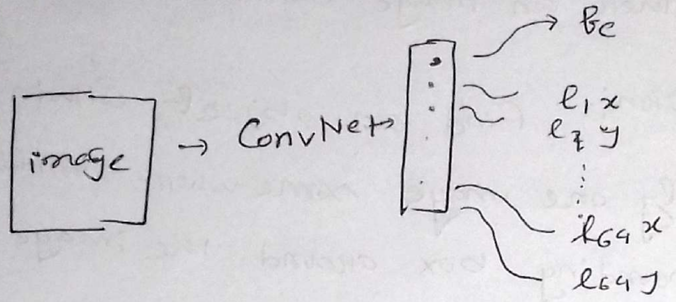
$$\mathcal{L}(\hat{y}, y) = \begin{cases} (\hat{y}_1 - y_1)^2 + (\hat{y}_2 - y_2)^2 + \cdots + (\hat{y}_8 - y_8)^2 & \text{if } y_1 = 1. \\ (\hat{y}_1 - y_1)^2 & \text{if } y_1 = 0. \end{cases}$$

↑ Number of parameters

L-2 Landmark Detection [11.25 pm 30.01.19]

[we can find the landmarks of the face within an image by using neural networks].

Suppose we have to find 64 landmark within an image



** 129 parameters.

** people pose detection using landmarks [research].
[find emotion from picture]

L-3 object Detection [11.35 pm 30.01.19]

[object detection using sliding window].

→ sliding window এর ব্যাপারটা বুঝছি. প্রথমে আমার training image কে ছোট করে crop করে train করবো.

→ Take a window of fixed size ~~and~~ and slide it over the image with some stride for almost every part of the region.

→ using this process computational cost is really high. because we have to crop the image so many times and pass it to the ConvNet.

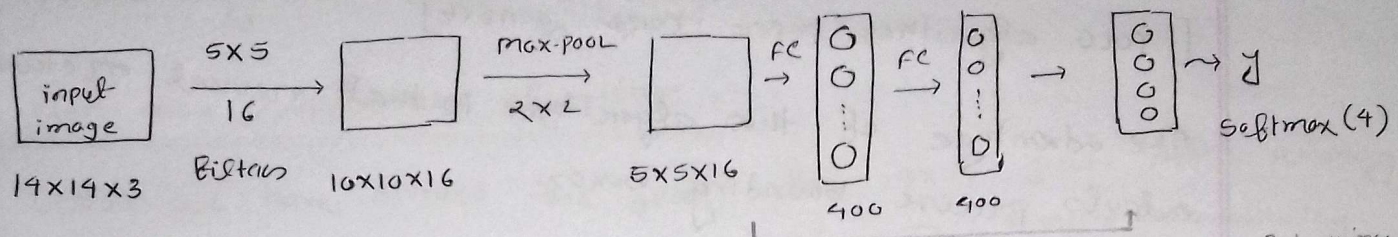
→ This problem can be solved.

Scanned by CamScanner

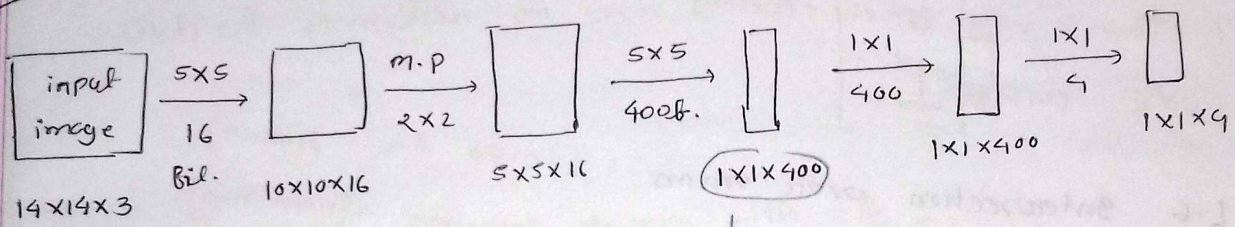
L-4 Convolutional implementation of sliding window.

[10:45 pm 30.01.19]

Turning FC layer into convolutional layers :-



→ Alternate



** Alternate implementation of FC layer using Convolution.

↳ mathematically it is equivalent to 400 fully connected unit

Convolutional implementation of sliding window :-

[Sliding window shares a lot of computations between windows ;
Convolutional implemetation of sliding window do not recompute the shared computation as a result it become quite faster]

[উদাহরণ] slide ও graph নিয়ে উদাহরণ graph এর ব্যাখ্যা কর।

⇒ [This algorithm has one problem that position of the bounding boxes will be not so accurate] .

L-5 Bounding box prediction [1.00 Am 31.01.19]

[ConvNet does not quite accurately predict bounding box]

[YOLO algorithm paper থেকে এসেছে]

The advantage of this algorithm is that neural network outputs precise bounding boxes.

[image will be divided into grid cells.

[This is very much efficient

[একটি ছোট্ট]

=

L-6 Intersection over union [10.50 Am 31.01.19]

[It is used for evaluating object detection algorithm]

Intersection over union (IoU)

$$= \frac{\text{Size of intersection area}}{\text{Size of union area.}}$$

"Correct" if $IoU \geq 0.5$.

It is measure that shows whether an object is correctly localized or not.

IoU is a measure of the overlap between two bounding boxes.

L-7 Non-Max- Suppression. [11:00 Am 31.01.19]

[Non-max suppression ensures that ~~an~~ algorithm detects each object only once within an image.

Non-max suppression algorithm :-

Let's we have divided the ~~grid~~ image into 19X19 grid. After running algorithm

Output prediction on each cell :

$$\begin{bmatrix} P_c \\ b_x \\ b_y \\ b_h \\ b_w \end{bmatrix}$$

× output in considering that we have to detect only one object

Now,

1. → discard all the boxes with $P_c \leq 0.6$.

2. while there are any remaining boxes :-

   → Pick the box with largest P_c. output that as a prediction.

   → Discard any remaining box with IoU ≥ 0.5 with the box output in the previous step.

   [যদি আমার Predicted bounding box এর সাথে high overlap হলে ঐ bounding ও object টা আমার detect হয়েছে আমরা ধরে নিচ্ছি। তাই আমরা এটা আর রাখব না।]

⇒ [If there ~~are~~ multiple object in an image we have to apply non-max suppression for each algorithm independently]

L-8 Anchor Boxes. [11:20 Am 31.01.19]

[Anchor boxes are predefined shapes of different objects].

Anchor box algorithm :-

Previously :- Each object in training is assigned to grid cell that contains that object's midpoint.

So, output y = 3 X 3 X 8 [Consider 9 grid cells with image can have 3 objects]

With two anchor boxes :-

Each object in training is assigned to grid cell that contains object's midpoint and anchor box for the grid cell with highest IoU.

[object এর anchor box গুলোর মধ্যে যার সাথে সবচেয়ে বেশি similarity থাকবে ঐ টা হবে আমাদের desired bounding box]

output, y = 3 X 3 X 8 X number of anchor box.

** we can define any number of anchor box.

** [If two objects have same midpoint within same grid cell then anchor boxes help us to find the accurate bounding box for these objects.]

~~vantage~~ ** It is not work so well when number of anchor boxes are less then number of object in an image.

**

↳ YOLO algorithm. [11.45 Am 31.01.19]

Training :- [sliding এ ছবি নিচে ধারণা দেওয়া আছে]

Suppose we have to detect 3 object with 2 anchor boxes.

objects :-

1. pedestrain
2. Car
3. motor cycle.

output y = (grid size) $3 \times 3 \times 2 \times 8$

$= 3 \times 3 \times 2 \times (5+3)$

- 2 → No of anchor box
- 3 → no. of class.

$$y = \begin{bmatrix} P_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ P_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

- P_c → is object ?
- b_x, b_y, b_h → position
- c_1, c_2 → which object
- First 8 entries (P_c to c_3) → Anchor box 1
- Next 8 entries (P_c to c_3) → Anchor box 2

Making prediction :-

[slide এ ছবিসহ দেওয়া]

Output the non-max suppression outputs.

1. For each cell, get 2 (# no of anchor box) predicted bounding boxes.
2. Get rid of low probability predictions.
3. For each class (pedestrain, car, motorcycle) use non-max suppression to generate final predictions.

[That is YOLO object detection algorithm]

L-10 Region proposals. [11.55 AM 31.01.19]

Region proposal : R-CNN

[This algorithm tries to pick few regions to run convolutional network.

Apply segmentation algorithm to find the area of interest and then apply CNN only to those areas.

Faster algorithm.

R-CNN : Propose regions. classify proposed regions one at a time. [slow]

Fast R-CNN : Propose regions. use convolution implementation of sliding windows to classify all proposed regions.

[time consume to propose regions].

Faster R-CNN : use convolutional network to propose region.

⇒ [YOLO algorithm is much faster]. (v.v.g)

Programming Assignment :-

1. use object detection on a car detection dataset.
2. Deal with bounding boxes.

(notebook এ অনেক detail ব্যাখ্যা দেয়া হয়েছে. এটা good for reading)

→ calculate dimension carefully.

Steps of implementing non-max suppression :-

1. Select the box that has highest score.
2. Compute its overlap with all other boxes and remove boxes that overlap is more than iou-threshold.
3. Go back to step 1 and iterate untill there is no more boxes with a lower score than the current selected box.

To implement this tensorflow has two built in function.

- tf.image.non_max_suppression()
- K.gather()

} Net থেকে Description দেখে নিও 275

[কোর্স এ yolo'র pretrain model implement করে দেখাইছে]

[কোর্স এ yolo'র pretrain model ব্যবহার করতে পারব]

↳ যদি আমাদের সম্পূর্ণ হয় তাহলে তারপরে

** Important github repository link দেয়া আছে.

Summary of YOLO.

1. Input image (608, 608, 3).
2. The input image goes through a CNN, resulting in a (19, 19, 5, 85) dimensional output.
3. After flattening the last two dimensions, the output is a volume of shape (19, 19, 425)
   - Each cell in a 19x19 grid over the input image gives 425 numbers.
   - 425 = 5x85, because each cell contains prediction of 5 ~~for~~ 5 anchor boxes.
   - 85 = 5 + 80 where 5 is because $(p_c, b_x, b_y, b_h, b_w)$ has 5 numbers and 80 is the number class we want to detect.
4. Select bounding boxes based on,
   - → Score thresholding :- throw away boxes that have detected a class with a score less than the threshold.
   - → Non-max suppression :- Compute intersection over union and avoid selecting overlapping boxes.

[This is YOLO (You Only Look Once) algorithm]

week-4 : Special applications : Face recognition & Neural style transfer.

P1 : Face Recognition.

L-1 what is face recognition. [10.20 Am 01.02.19]

Face verification vs. face recognition.

verification :-

→ Input image, name/ID.

→ output whether the input image is that of the claimed person.

(Recognition is much harder than verification).

Recognition :-

→ Has a database of K persons.

→ Get an input image.

→ output ID if the image is any of the K persons or "not recognized".

(For correct recognition accuracy of verification step must be higher).

L-2 one shot learning. [10.30 Am 01.02.19]

[one shot learning problem is that our system must be able to recognize a person given just one single image]

** It is difficult to do well on a deep learning problem with only one example of a class.

→ Learning from one example to recognize the person again.

** If a new employee joins the number of output will be change and we have to retrain our whole system..

[It is not a good way].

Learning a "similarity" function.

$d(img1, img2)$ = degree of difference between images

If $d(img1, img2) \leq \tau$ ← Threshold → "same"

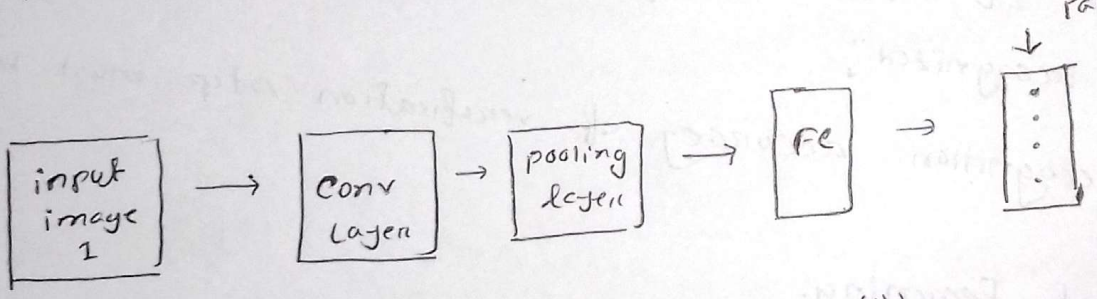
$> \tau$ → "different"

** This similarity function solves the problem of one-shot learning.

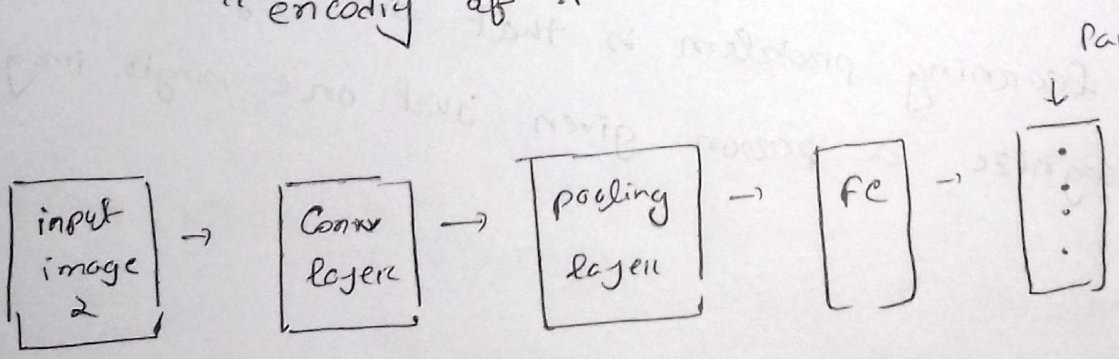
L-3 Siamese Network. [10.45 AM 01.02.19]

[This network is used to train similarity function]

Here we have encode each image by definate number of parameters to find similarity function.



"encoding of $x^{(1)}$" = $f(x^{(1)})$.



"encoding of $x^{(2)}$" = $f(x^{(2)})$

Fig: Siamese Network.

Goal of Learning :-

Parameters of NN define an encoding $f(x^{(i)})$ of $(x^{(i)})$ ↳ this is encoding

Learn parameters so that :

→ if $x^{(i)}, x^{(j)}$ are the same person, $\|f(x^{(i)}) - f(x^{(j)})\|^2$ is small.

→ if $x^{(i)}, x^{(j)}$ are different persons, $\|f(x^{(i)}) - f(x^{(j)})\|^2$ is large.

L-4 Triplet loss. [11.00 Am 01.02.19]

[At each time we have to look at three images anchor, positive and negative. This is where triplet terminology comes from].

objective function :-

$$\|f(A) - f(P)\|^2 \leq \|f(A) - f(N)\|^2$$

$d(A,P)$ $d(A,N)$

$$\|f(A) - f(P)\|^2 - \|f(A) - f(N)\|^2 \leq 0$$

If $f(A), f(P), f(N)$ are identical then $f(img) = \vec{0}$.

To avoid this we have to add a margin (α) so that positive image has certain degree of similarity greater than threshold.

margin

So, $\|f(A) - f(P)\|^2 + \alpha \leq \|f(A) - f(N)\|^2$

$\therefore \|f(A) - f(P)\|^2 - \|f(A) - f(N)\|^2 + \alpha \leq 0.$]

Loss function :-

Given 3 images A, P, N.

$$L(A,P,N) = \max\left(\|f(A) - f(P)\|^2 - \|f(A) - f(N)\|^2 + \alpha, 0\right)$$

$$J = \sum_{i=1}^{m} L\left(A^{(i)}, P^{(i)}, N^{(i)}\right)$$

Choosing the triplets A,P,N :- $\begin{cases} (A,P) \rightarrow \text{same person.} \\ (A,N) \rightarrow \text{different person.} \end{cases}$

During training if (A,P,N) are chosen randomly

$d(A,P) + \alpha \leq d(A,N)$ is easily satisfied. As a result our network will not learn much.

So we have to choose triplet in such way that are hard to train on.

$$d(A,P) \approx d(A,N).$$

Of we do that computational efficiency will increase.

popular way of naming algorithm in deep learning world.

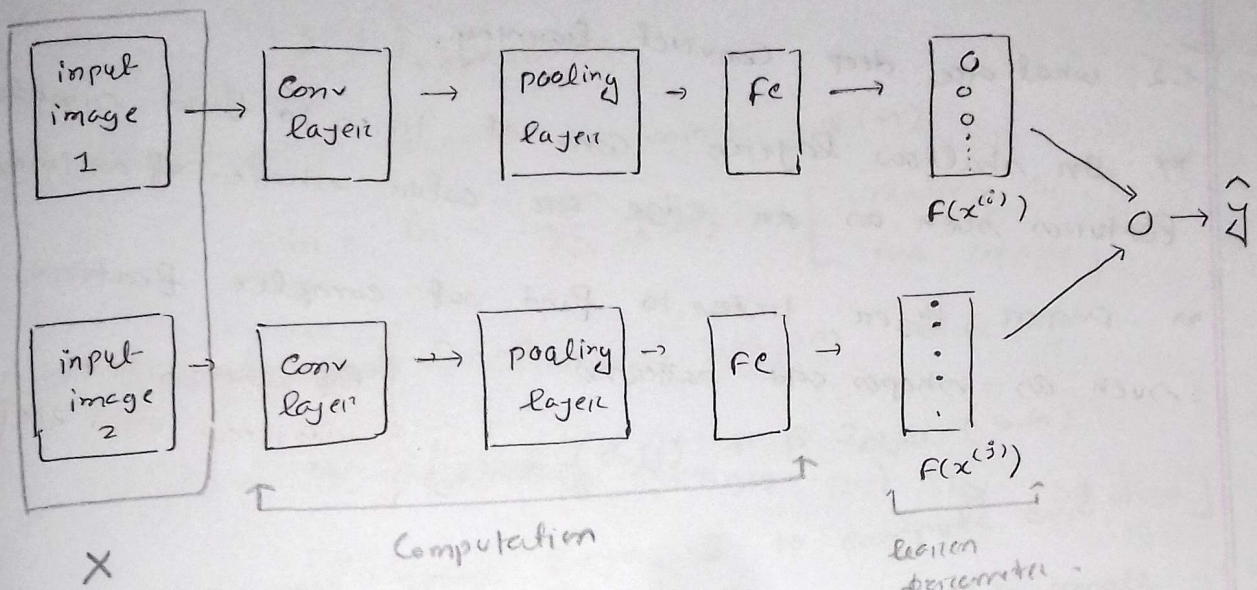
1. _____ Net

2. _____ Deep _____.

[online → pretrained model ব্যবহার করে]

L-5 Face verification and Binary classification. [11:40 AM 01.02.19]

[Similarity function can also be learnt by using logistic Regression].



$$\hat{y} = \begin{cases} 1 & \text{if } x^{(i)} \text{ and } x^{(j)} \text{ same person.} \\ 0 & \text{if } x^{(i)} \text{ and } x^{(j)} \text{ different person.} \end{cases}$$

→ number of parameters.

$$\hat{y} = \sigma\left(\sum_{k=1}^{128} w_i \left| f(x^{(i)})_k - f(x^{(j)})_k \right| + b_i \right)$$

This can also replaced by $= \dfrac{(f(x^{(i)})_k - f(x^{(j)})_k)^2}{f(x^{(i)})_k + f(x^{(j)})_k}$ [χ^2]

P2 : Neural style Transfer

L1 : what is neural style transfer. [12.30 pm 01.02.19]

[যদি Content image হতে আমরা style image এর ফিচার represent করা].

L2 : what are deep ConvNet learning. [12.40 pm 01.02.19]

veido ta important

** In shallow layers ConvNet tries to find simple features such as an edge or color shade of an image.

** Deeper layer tries to find out complex features such as shapes and patterns.

[কিছু বুঝতে চাইলে veido recap দেখতে হবে]

[veido te ছবি দিয়ে ভালোভাবে বুঝানো হয়েছে]

L3 : cost function [12.50 pm 01.02.19].

For neural style transfer we have to formulate a cost function where from content image (c) and style image (s) we will generate image (G).

cost of generated image (G).

$$J(G) = \alpha \, J_{content}(C, G) + \beta \, J_{style}(S, G)$$

$J_{content}$ → How similar content image and generated image

J_{style} → How similar content image and style image

α, β hyperparameter for relative weighting of content image and style image

Scanned by CamScanner

Find the generated image G.

1. Initialize G randomly.

G_1 : 100 × 100 × 3.

2. use gradient descent to minize $J(G)$

$$G = G - \frac{\sigma}{\sigma G} J(G).$$ [updating each pixel of the image]

L-4 Content cost function. [1.05 pm 01.02.19]

$$J(G_1) = \alpha J_{content}(C, G) + \beta J_{style}(S, G).$$

→ Say you use hidden layer l to compute cost.

– [value of l is choose in middle if l is small it will force generated image similar to content image. if l is large generated image will not like content image].

→ use pre-trained ConvNet. (E.g. VGG network).

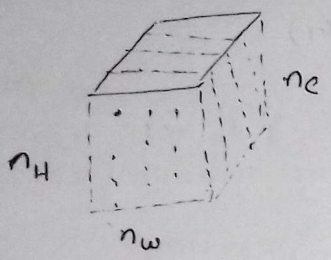
→ Let $a^{[l](C)}$ and $a^{[l](G)}$ be the activation of layer l on the images.

→ If $a^{[l](C)}$ and $a^{[l](G)}$ are similar both image must have similar content.

[$J_{content}(C, G) = \| a^{[l](C)} - a^{[l](G)} \|^2$] → elementwise L2-Norm.

L-5 style cost function [7.45 PM 01.02.19]

Suppose you are using layer l's activation to measure "style". Define style as correlation between activations across channels.



How correlated are the activations across different channels?

⇒ [Degree of correlation tells how often the high level feature occurs together and don't occur together in different parts of an image]

Style matrix :-

Let $a^{[l]}_{i,j,k}$ = activation at (i, j, k)

$n_H \; n_w \; n_c$

$G^{[l]}$ is $n_c^{[l]} \times n_c^{[l]}$ (style matrix)

style image

$$G^{[l](S)}_{kk'} = \sum_{i=1}^{n_H^{[l]}} \sum_{j=1}^{n_w^{[l]}} a^{[l](S)}_{ijk} \; a^{[l](S)}_{ijk'} \qquad \left[k, k' = 1, \ldots, n_c^{[l]}\right]$$

Generated image

$$G^{[l](G)}_{kk'} = \sum_{i=1}^{n_H^{[l]}} \sum_{j=1}^{n_w^{[l]}} a^{[l](G)}_{i,j,k} \; a^{[l](G)}_{ijk'}$$

Style cost function :-

(number of channel)

$$J^{[l]}_{style}(S,G) = \frac{1}{\left(2 n_H^{[l]} \cdot n_W^{[l]} \cdot n_C^{[l]}\right)^2} \sum_{k}^{n_c} \sum_{k'}^{n_c} \left(G^{[l](S)}_{kk'} - G^{[l](G)}_{kk'} \right)^2$$

we will get visually pleasant result if we take style cost function from different layers, → some hyper parameters.

$$J_{style}(S,G) = \sum_{l} \lambda^{[l]} \cdot J^{[l]}_{style}(S,G)$$

So overall cost function.

$$J(G) = \alpha J_{content}(C,G) + \beta J_{style}(S,G).$$

Now apply algorithm to minimize cost.

[8:10 pm 01-02.19]

L-6 1D and 3D Generalization.

1D ও 3D data তে Convolutional network apply [2D ছাড়াও 1D ও 3D data তে করা যায়].

[1D data এর জন্য Recurrent ~~data~~ neural network ব্যবহার হয়]

Programming Assignment :-

→ Implement the neural style transfer algorithm.

→ Implement novel artistic images using your algorithm.

[The idea of using a network trained on a different task and applying it to a new task is called transfer learning. Neural style transfer (NST) uses this technique.

Steps of a neural style transfer :-

1. Create an interactive session.
2. Load the content image.
3. Load the style image
4. Randomly initialize the image to be generated.
5. Load the VGG-16 model.
6. Build tensorflow graph.
   - Run the content image through the VGG-16 model and compute content cost.
   - Run the style image through the VGG16 model and compute style cost.
   - Compute total cost
   - Define optimizer and learning rate.
7. Initialize the tensorflow graph and run it for a large number of iterations, updating generated image at every step.

** Previously train model use করে Face recognition কাজ হয়ে.

what you should remember:

1. Face verification solves an easier 1:1 matching problem, face recognition addresses a harder 1:k matching problem.

2. Triplet loss is an effective loss function for training a neural network to learn an encoding of a face image.

3.

v.v.y [FaceNet and DeepFace are two important papers for face verification and recognition.]

** Also look at official repository of Facenet implementation.

[Completed 7.45 pm 02.02.19].