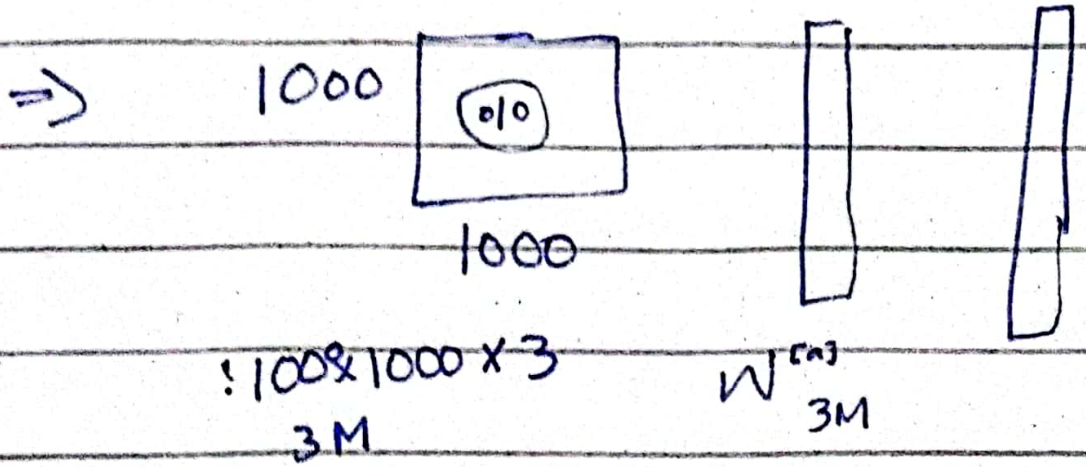
24/4/25 (After mids)

CONVOLUTIONAL NEURAL NETWORK



$W^{[a]} = W^{[1]}, W^{[2]}$ (1000, 3 million)

3 billion

⇒ Overfitting (less) ⇒ Computational (less) ⇒ Flattening

⇒ By Default, Convolutional neural network nature is to shrink.

$n-f+1$: Now Everything is practical

⇒ Convolutional first layer finds edge, middle layer finds objects small parts (cat ⇒ eye, ear etc....) and last layer determine whole object.

⇒ If filter and input are similar then answer will be big otherwise small.

⇒ Sparse Connection and parameter sharing is property of convolutional neural network.

⇒ $n=6, f=3, s=1, p=1$

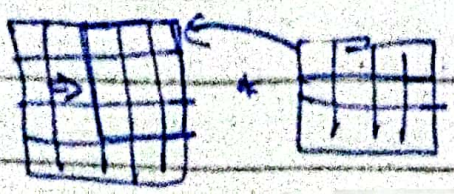
$$n+2p-f+1$$

$$: 6+2-3+1$$

$$: 6 \quad (6 \text{ by } 6)$$

⇒ **Strided Convolution:**

→ skips the column to reduce image size and get smaller area.



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

W.W ⇒ $n=6, f=3, s=2, p=(6+3-2)/2$

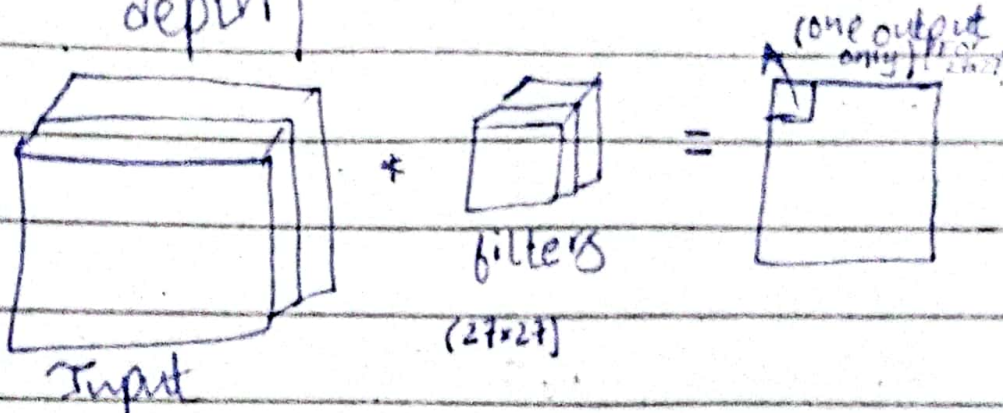
$=3.5$

(Identify Padding for that stride)

⇒ Convolutions over volume:

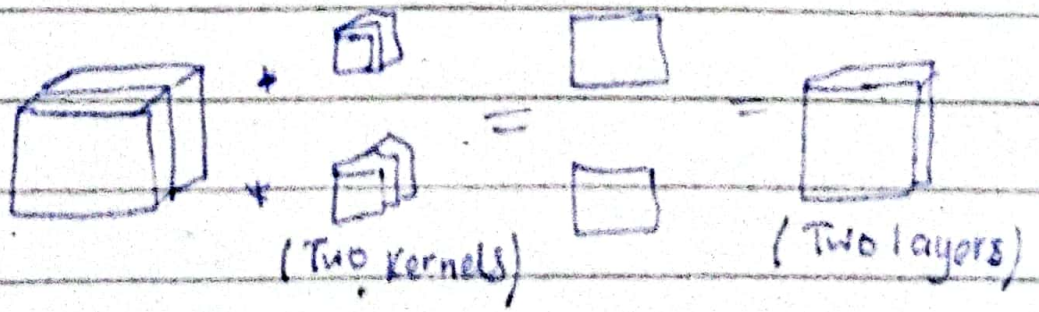
⊙ For RGB channels (Multiple channels)

⊙ Filter depth should be match. channel depth (input depth)



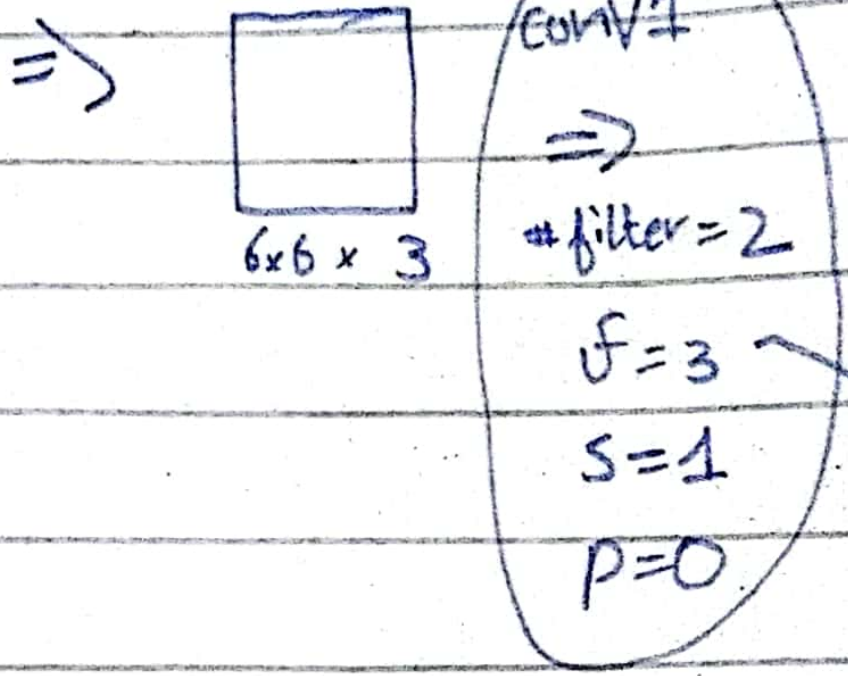
: (only one output for 27×27 (input & filter)

⊙ If more filters are applied more output will be there:



: $A = \text{ReLU}((W^T \cdot x) + b)$

(2 Biased for 2 filters)



(No. of Parameters)

$(3 \times 3 \times 3 + 1) = 28$

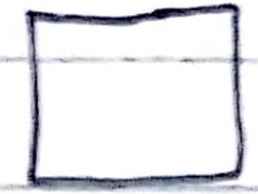
$(3 \times 3 \times 3 + 1) = 28$

$= 56$

: If 10 filters, then

No. of parameters (280)

6/5/25

(*) 

32 x 32 x 3

if = 5

S = 1

P = 0

# of filters = 6

$$\frac{n + 2p - f}{s} + 1$$

$$\rightarrow \frac{32 + 2(0) - 5}{1} + 1$$

$$32 - 5$$

$$27 + 1 = 28 \times 28 \times 6$$

Activation size

⊙ For CNN

: no. of parameters

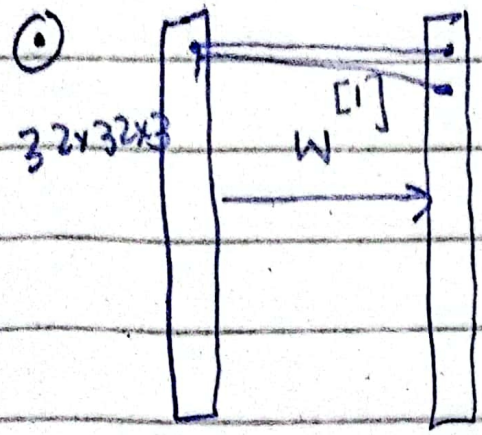
$= 6 \times (5 \times 5 \times 3 + 1)$

$= \boxed{456}$

(less Parameters)

(Because of sparse connection & sharing weights)

(No overfitting)

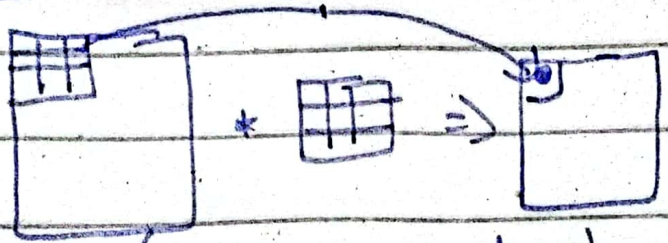


$n^{[2]} = 32 \times 32 \times 3$ $\quad 28 \times 28 \times 6$

$w^{[1]} = (28 \times 28 \times 6 \;,\; 32 \times 32 \times 3)$

$= 14$ million (More Parameters)

⊙ In CNN.



(less Parameters)

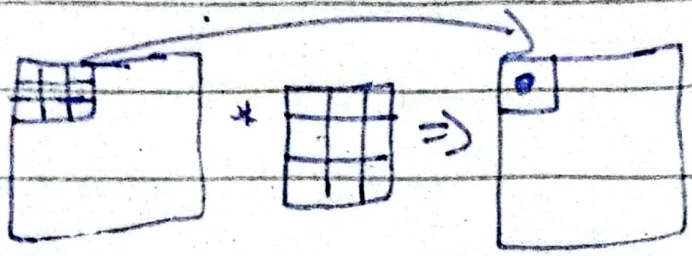
⇒ In CNN we see: (i) Convolutional layer (ii) Pooling Layer (iii) Fully connected Layer.

⊙ Its structure is input layer, convolutional layer, pooling layer, convolutional Layer .....

(Structure of CNN)

⇒ CL → PL → CL → PL ..... → FCL

(1) Receptive Field: How much portion of image a neuron has seen.



(Receptive fields of neuron is 9)

⇒ Receptive Size = $1 + L \times (f-1)$

E.g: $7 = 1 + L \times (3-1) = 1 + L * 2$

$6 = L * 2$

$\boxed{L = 3}$

E.g: $1000 = 1 \times L(3-1)$

$1000 = 1 \times 2L$

$\frac{499}{2} = L$

$L \approx 500$

⇒ For this, we use pooling and strided

⇒ Dense layer are learnable & pooling layer are not learnable.

(For Exam) ∴ In Pooling, variables or parameters will be zero

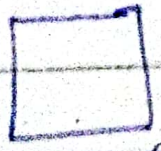
⇒ Max pooling (Pick Max) & Avg Pooling (Take Average)

⇒ Convolutional and pooling shrink the input layer.
(As we more forward, it spatial decreases and depth increases)

⇒ In CNN, first layer edges are find and in intermediate layer we find small objects and last layers find whole object.
(Didilect)

⊛ Homework: (LeNet-5)
↓
(Learnable layers) / (CL & FCL)

(Question for Exam) Find activation size, No. of parameters on each layers



Hyper Parameters → Experiments

: Make Correct Table for HW, Quiz, Exam

8/5/25

⊙ Architecture Designing:

→ ① LeNet-5 (for digit recognition)
: pooling layers are not included in counting because it is not learnable layer. (1998)

→ AlexNet (8 layer architecture):
- ⊙ 2012.
- ⊙ Calculation after filter is independent

⊙ Hardware and software available in that time period and 3 more layers added which makes network complex but pattern is same like now (CON → PL → CON → PL → FC)

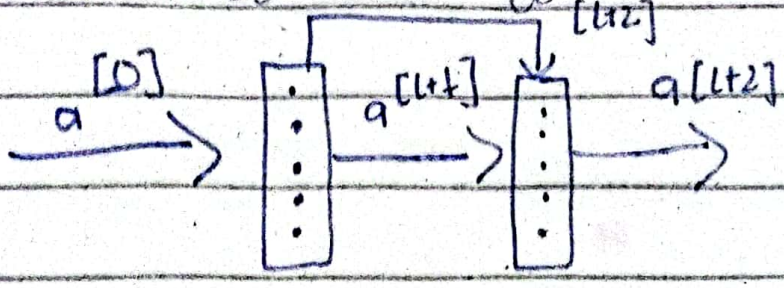
→ VGG-16:

⊙ simplified architecture

⊙ Depth becomes double but size remains same and by pooling dimension becomes half.

→ ResNet (Residual Network):

⊙ skip connection idea is used in it



$$a^{[l+2]} = RELU\left(W^{[l+2]} * a^{[l+1]} + a^{[l]}\right)$$

$$a^{[l+2]} = ReLU\left[a^{[l]}\right] = a^{[l]}$$

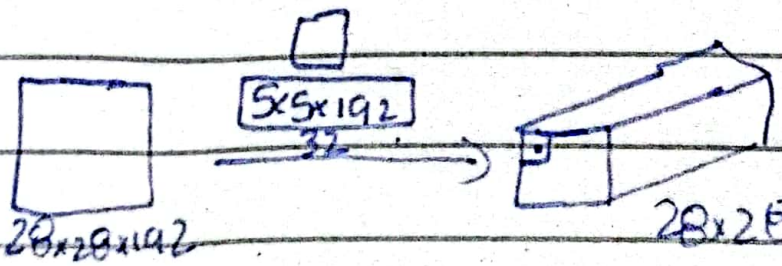
: RELU (0, a)
- +

→ (i) Scratch Training (ii) Pretrained

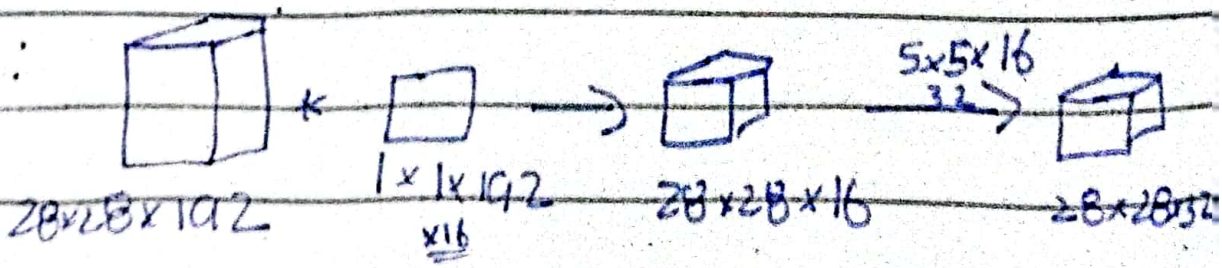
```
: import torchvision.models
resnet models = models.resnet34
                (pretrained = True)
                " if false then scratch.
```

→ Inception network :

- ⊙ Convolution operation. 1×1 is used (which is doing scaling)
- ⊙ Depth work is done using it (Retain, Increase ...)
- ⊙ In this, many filters are applied on one layer.



⇒ 28×28×32×5×5×192 = 120 million



→ 1×1×192×28×28×16 ≃ 12

: Skip Connection, 1×1 ...

13/5/25

① Object Detection:

→ Classification and localization
(Not just identify but also locate it)

→ First layer find edges and intermediate layer find small objects.

→ AlexNet is trained for imageNet (1000 images)(14 billion parameters)

→ (4096,1)

→ 10 images of same object and their features are close to each other
(If pixel level is compared then patterns of same objects with different background will be different)

→ Feature comparison is better than pixel comparison.

→ Last layer contains complete semantic whole image.

→ Block (CL→PL)(CL→CL-PL)

⊙ Transfer Learning:

→ To overcome overfitting, dataset size increases.

→ If dataset size is small then use transfer learning.

→ Three ways:

(i) ⊙ If dataset size < 10000, then last layer will be fine tuned (trained) and all other (Many layer) will be freezed instead of last. (3 neurons layers in last layer only)

(ii) ⊙ If dataset size > 10000, last layer will be removed, first layer freeze, last four layers will be fine tuned and transfer it.

: CelebriDataset

(iii) ⊙ If dataset is 50000 or 100000 and classes 3, last layer will be removed and train all remaining layers.

→ If Model is there, but not trained and we have three classes and dataset small, then train it on related large dataset and then train it on your own data.

→ Assignment. (ResNet)

⊙ Train last layer sign Dataset (Last layer contains 10 (classes) neurons)

→ Image Classification → Classification with localization → Detection (centralization)

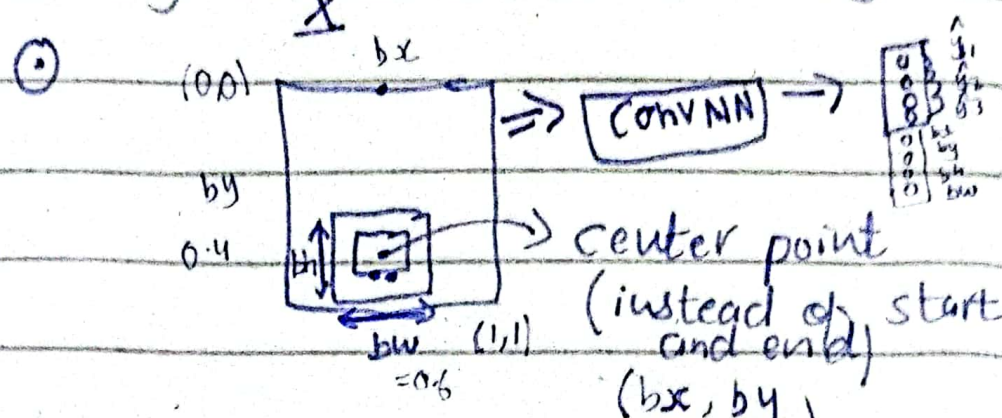
15/5/25

⊛ Self-Driving Car uses object Detection

⊛ Classification (one object) → object detection (one image but multiple images)

→ YOLO:

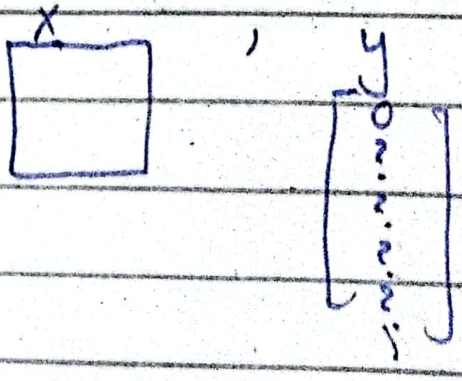
- You only look once
- Self Driving uses this algorithm to detect objects.



1) Person
2) Car
3) Motorcycle
4) Background (Not all above three)

$$\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

$$y = \begin{bmatrix} pc \\ bx \\ by \\ bh \\ bw \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 0.5 \\ 0.7 \\ 0.4 \\ 0.6 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

→ 

: ? (Garbage

: If we use sigmoid, then we use BCE

⊙ Loss Function:

if $(y_1 == 1)$

$L(y, \hat{y}) = (y_1 - \hat{y}_1)^2 + (y_2 - \hat{y}_2)^2 \dots (y_n - \hat{y}_n)^2$

$L(y, \hat{y}) = BCE$
$= y \log \hat{y} + (1-y)\log(1-\hat{y})$

else : // $(y_1 == 0)$

$L(y_1, \hat{y}) = (y_1 - \hat{y}_1)^2$

→ For First node apply BCE after sigmoid

→ For other nodes (coordinates) we apply MSE.

→ For last 3 nodes, classes probabilities apply softmax and use Cross Entropy.

② Landmarks Detection

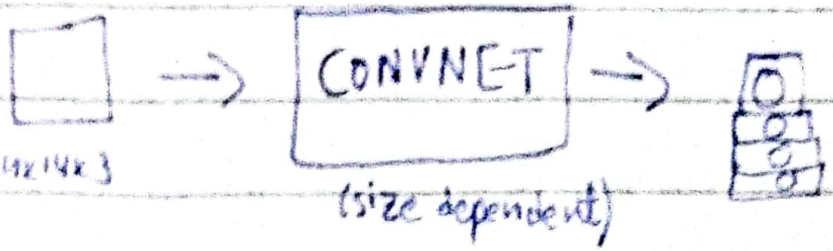
⊙ 68 **Landmarks** on human faces which is used by different filters etc... (glasses) to fit on face.

→ We can decide pose by Landmarks.

: For Smile Detection check Lips landmark Distance

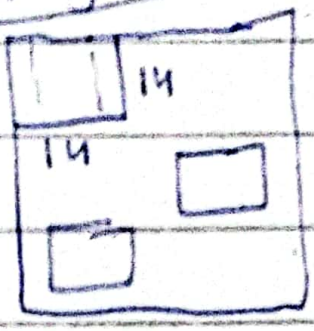
: For Death Detection Check Eye Landmark Distance.

⊙ Car detection :



→ Closely cropped image (only images in box train)

sliding window

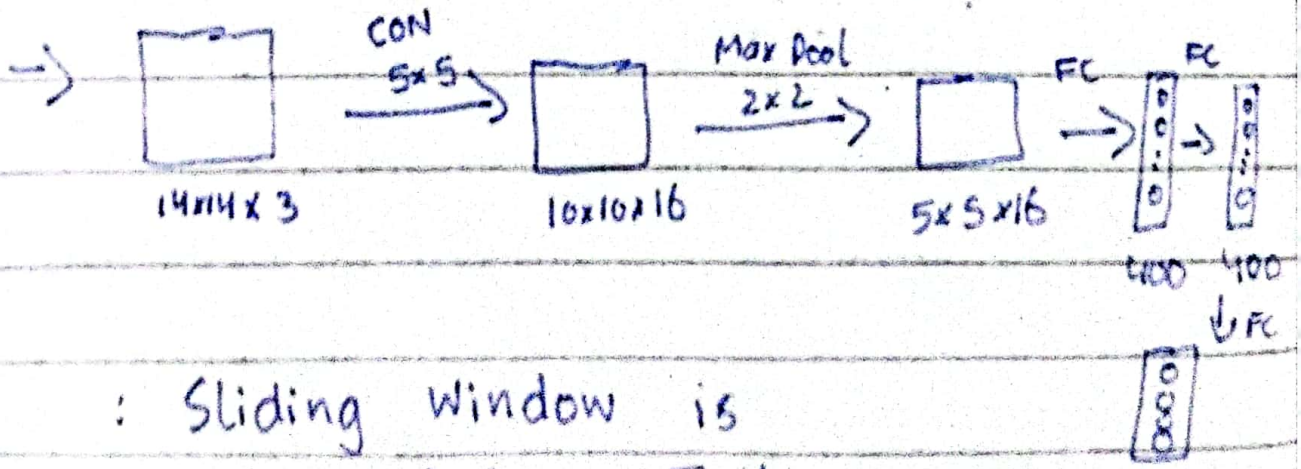


: If image of car size is greater than 14x14, than image size will be lesser to detect properly.
: (Computations Increases)

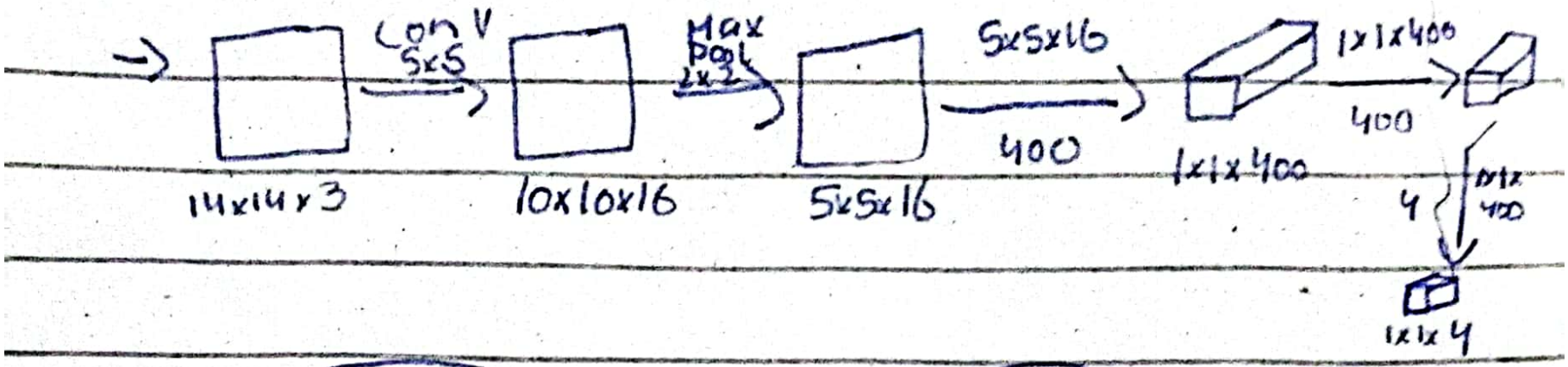
: Big Image (1000x1000) (Contains multiple objects)

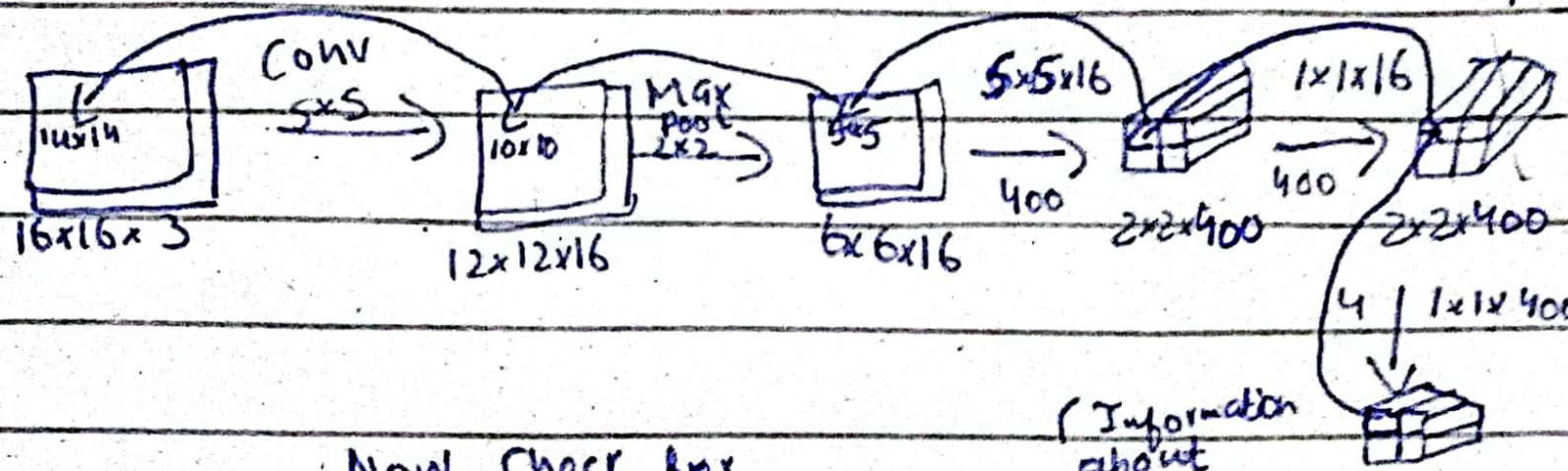
: $\left[\frac{n+2p-f}{s}+1\right]$

→ Solution is YOLO which looks once.



: Sliding window is replaced by Fully Convolutional Neural Network.





(Information about 4 region)

: Now Check for receptive field

(You only Look Once) (Classify done) (Box Done)

→ It is refined by max suppression etc...

20/5/25

Evaluation:

→ Label and prediction

intercept region

→ Sample IOU.

$$IOU = \frac{\text{size of intercepted area}}{\text{size of union area}}$$

: IOU >, 0.5 (satisfactory)

IOU >, 0.7 (Good)

IOU >, 0.9 (Best)

⊙ Non-Maximum Suppression:

→ Drawing tangent
→ We see the region where center is there
→ The values which are not of big
→ The closest or neighborhood region should be greater.
· If values is 0.7, suppress it. If values is 0.17, then finish it and leave
→ Select maximum and suppress others.

: Exam Imp Question

⊙ Example:

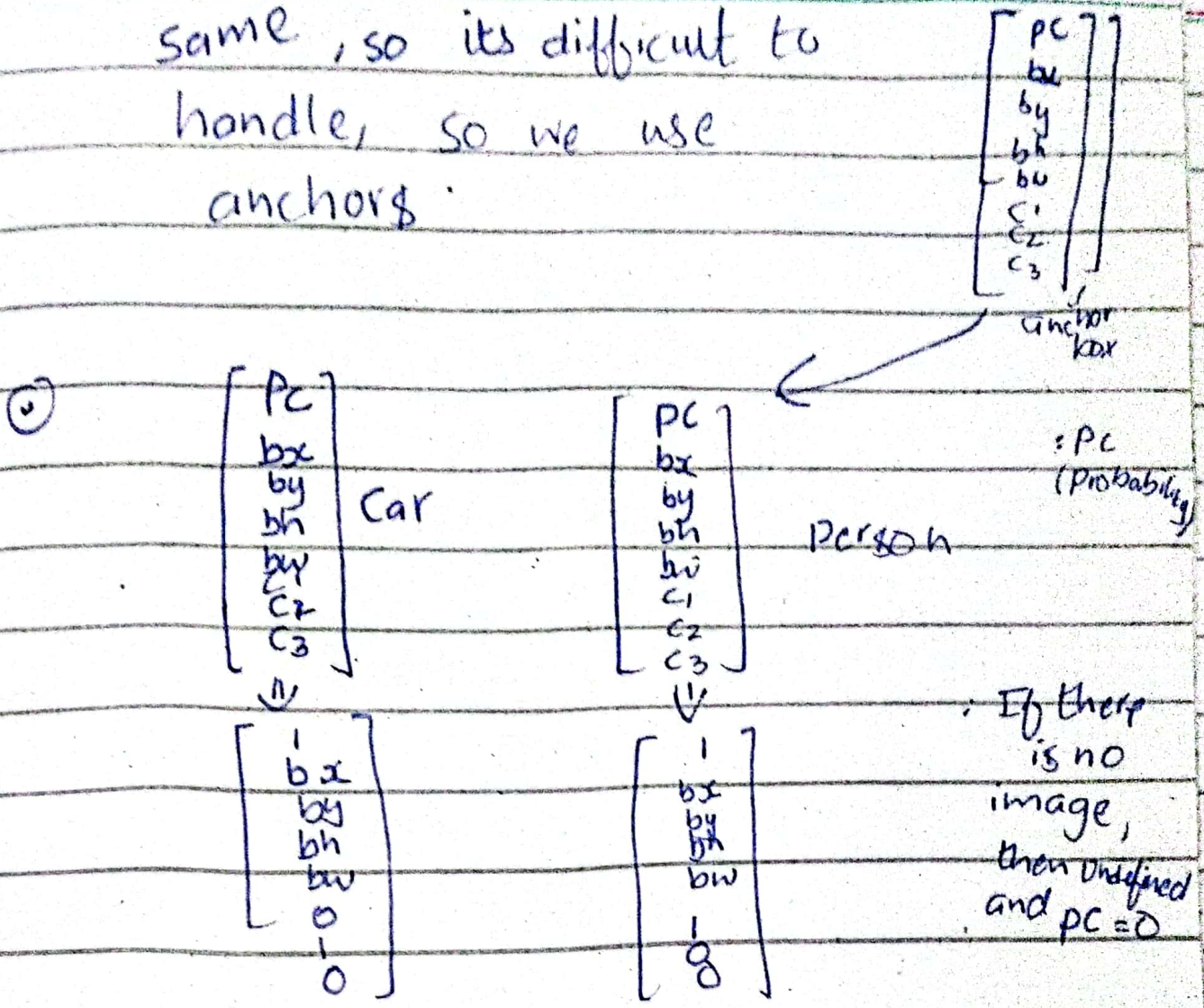
→ There are 4 rectangles
0.9, 0.8, 0.75, 0.7
1) Select highest one 0.9
2) Find, its IOU with others. Suppress angles with <0.7 and output will be 0.9

: Intercept over Union

⊙ Anchor Boxes:

→ If center of gravity of two images is

same, so its difficult to handle, so we use anchors.



- Objects can be 3 but anchors will be 2
- Call NMS for Person object and then for Car.
- **Mean Average Precision:**
  - -> Important Metric
  - -> It follows classification and IOU and then gives value.

: classification and localization

→ Model is trained and now we wanted to test it using (mAP)

→ Now check for each image ∃ confidence, TP or FP

(i) TP if (Pc > 0.5 && IOU between label and prediction is > 0.5)

→ (i) Write all images score together.

(ii) Sort in descending order

(iii) Precision or Recall.

Predict \ GT	+	-
+	TP	FP
-	FN	TN

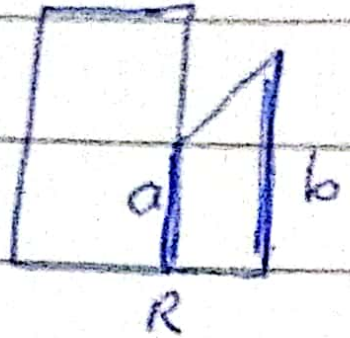
∴ Precision = $\frac{TP}{TP+FP}$ ∴ Recall = $\frac{TP}{TP+FN}$

∴ Accumulated TP ∴ Accumulated FP

∴ precision = $\frac{accTP}{accTP+accFP} = \frac{1}{1+1} = \frac{1}{2}$

∴ GT = 4 (objects, images) ∴ Recall = $\frac{TP}{(TP+FN)_{GT}}$

(iv) Plot the Precision-Recall graph
(Any Side of Precision or Recall on Graph)



: Calculate from graph

22/5/25

* Face Recognition :

→ One to one match

→ Checks face data from database (Mobile Face Recognition)

⊛ Face Verification :

→ Multiple faces or data present in database. (multiple faces)

∴ : Link for face detection project present in slides.

* Face Recognition System :

→ Employee data of company (4 employees and wanted to recognize their face and apply neural network).

→ If employees increases, then again training and testing is required. If data is less, training underfit. So more data is required to train. (One-shot learning)

→ The solution to one-shot Learning is to learn similarity function.

→ If one image is passed one feature is made and if again that image is passed, then feature comparison is done and check for less difference.

→ Just trained the model over big dataset and if new person comes, then compare it on feature vector and check for difference.

· code

nn. sequential (Run layers in sequence)

⊙ Triplet loss:

(for faces)

→ To train Siamese network, we use set of 3 images:

: Anchor (original image) + Positive (same Image) + Negative (Different Image)

$$d(A,P) + \alpha - d(A,N) \leq 0$$

$0.5 \quad 0.2 \quad 0.51$

$$0.2 \leq 0$$

→ Two Branches in Siamese, in which we are training model

→ Loss function:

$$L(A,P,N) = \max\left(d(A,P) + \alpha - d(A,N), 0\right)$$

(Positive)	0.5	0.2	0.51
(negative) (0)	0.5	0.2	0.9

→ Selection of triplet:

⊙ Anchor and Positive should be different in look to train network hard

⊙ Anchor and Negative look same to train hard.

⊙ $z = w[x] + b$

$$z = w\left(\frac{f(x)^{(i)} - f(x)^{j}}{\|f(x)^{(i)} - f(x)^{j}\|^2}\right) + b$$

• Feature Learning By Autoencoder: → Image retrieval

⊙ DeepFace:

⇒ If matched 1, otherwise 0

so deep face is much better than face Net.

⊙ FaceNet Pre computed and pass through neural network and check difference for comparison.

⊙ Autoencoder Architecture:

→ It has two parts:

(i) Encoder (Compress info in small vector)

(ii) Decoder (Generate original image)

E.g

: 4 → Encoder → [ ] → Decoder → 4

h

⊙ CNN can also be used and layers of convolution is used.

⊙ Application:

→ Noisy image given or good image given, then give noisy image to encoder and it makes original image. : It learn identity function

→ Image colorization (old images into RGB

→ Anamoly detection (Input and output image is same just one same image)

⊙ Difference will be more then anamoly.