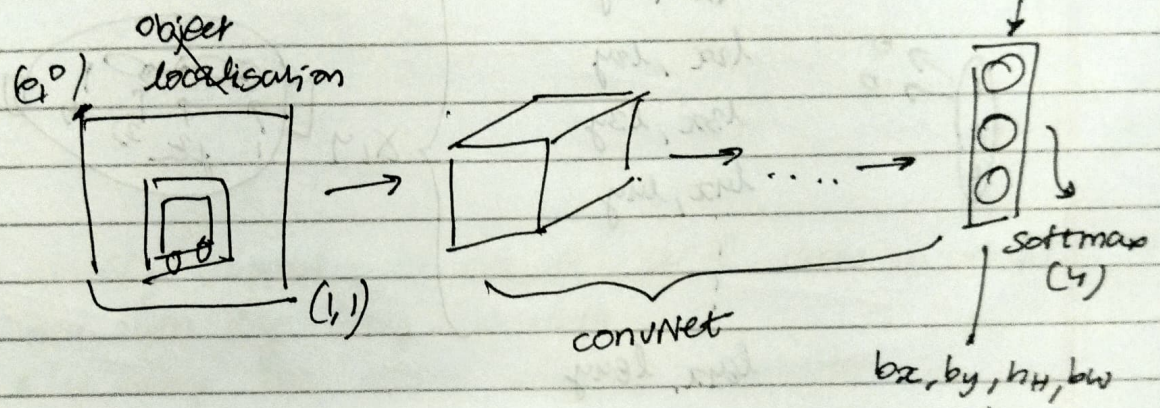
object detection

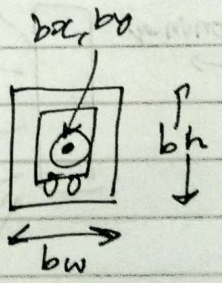
Image classification → Classification with localisation → Detection

Image classification: 1 object

Multiple objects



1 - pedestrian
2 - car
3 - motorcycle
4 - background



bx = 0.5
by = 0.7
bh = 0.3
bw = 0.4

$y = \begin{bmatrix} Pc \\ bx \\ by \\ bh \\ bw \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$ Pc → Is there any object?

c_1 — Pedestrian
c_2 — car
c_3 — motorcycle
} any one of them almost 1

for car: $\begin{bmatrix} 1 \\ bx \\ by \\ bh \\ bw \\ 0 \\ 1 \\ 0 \end{bmatrix}$ (x, y)

For no object: $\begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix}$

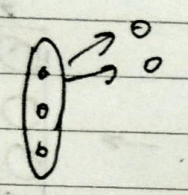
$L(\hat{y}, y) =$

if $y_1 = 1$ $\quad (\hat{y}_1 - y_1)^2 + (\hat{y}_2 - y_2)^2 + \dots (\hat{y}_8 - y_8)^2$

if $y_1 = 0$ $\quad (\hat{y}_1 - y_1)^2$

↑ value or PC

Landmark Detection



l_{1x}, l_{1y}

l_{2x}, l_{2y}

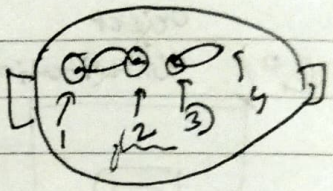
l_{3x}, l_{3y}

l_{4x}, l_{4y}

⋮

l_{64x}, l_{64y}

x, y



Convnet →

face = ?

l_{1x}

l_{2x}

l_{3x}

l_{61x}

l_{64y}

129

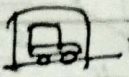
x, y

AR Augmented Reality

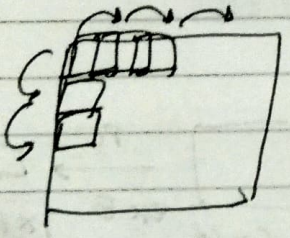
Sliding windows

car detection example

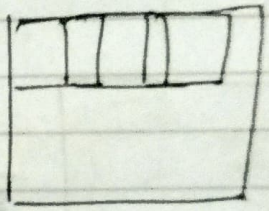
Traing set

x	y
	1
	0
	1
	1

closely cropped images of car & train on them
and then try using test data
a by using the sliding windows detection



& then try using bigger windows

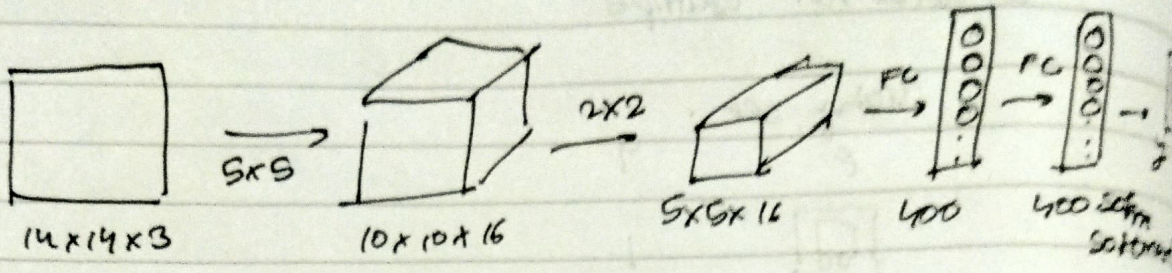


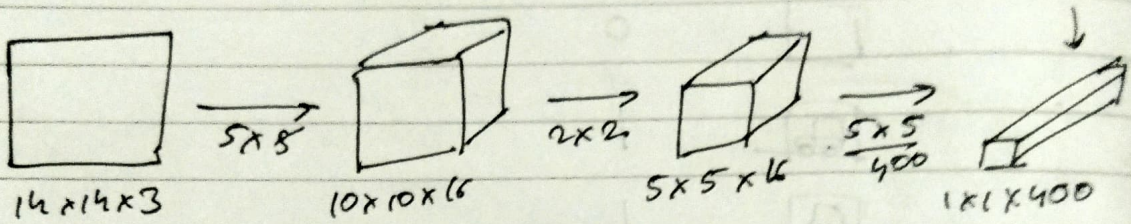
& so on

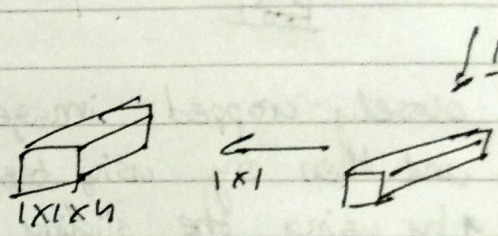
till we detect a car

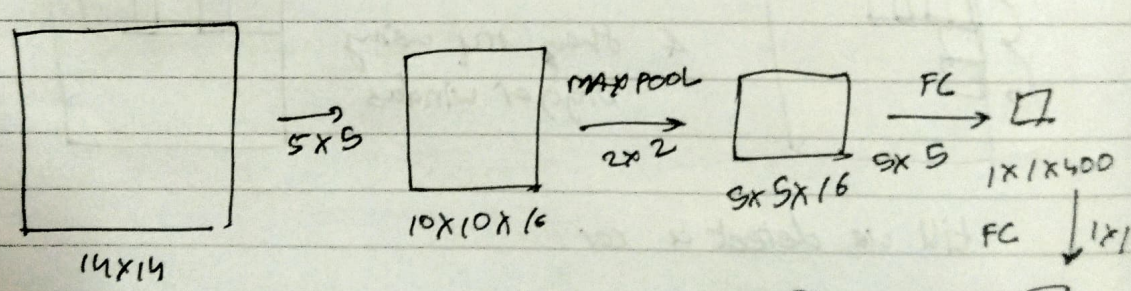
computation cost !!!

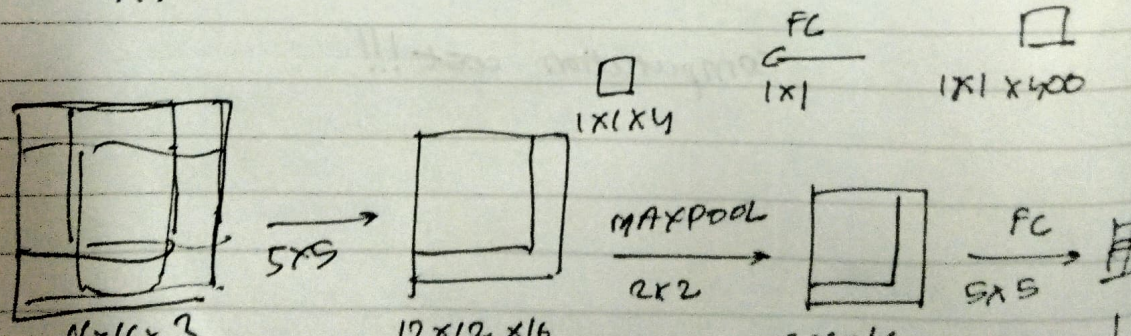
How to implement algo conv.





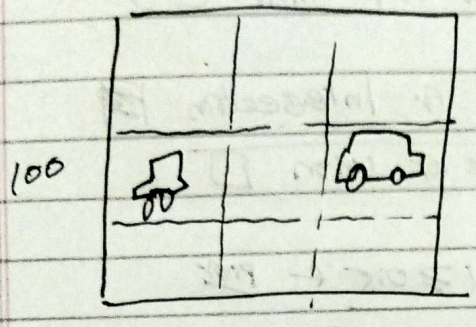






Bounding Box Prediction

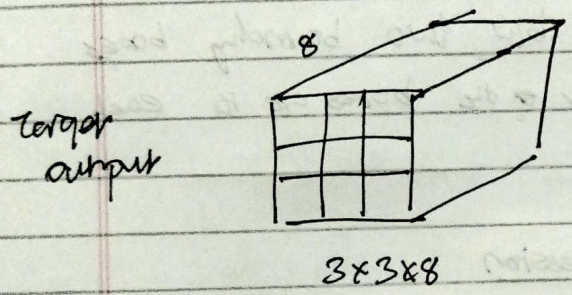
YOLO Algorithm — You look only once



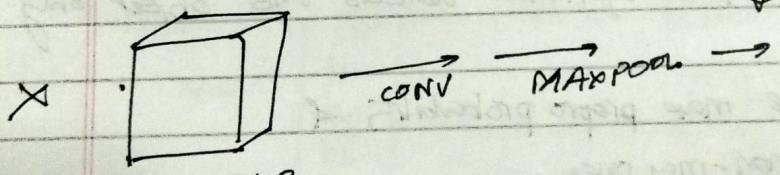
→ Apply image classification & localisation to each of 9 grid cells.

For each grid cell:

$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} \quad \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix} \quad \begin{bmatrix} 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix}$$



target output

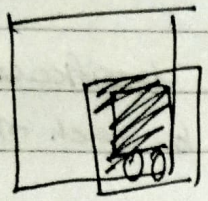


Q] How do you compute b_x, b_y, b_h, b_w?

Ans

0.4 } b/w 0 & 1
0.3 }
0.5 } could be > 1
0.9 }

Intersection over Union



Intersection over Union (IoU)

$$= \frac{\text{size of Intersection}}{\text{size of Union}}$$

"correct" if $IoU \geq 0.5$ ← prior

0.6 ←

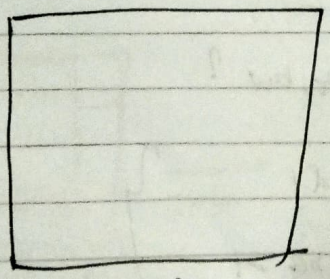
→ IoU is the measure of overlap b/w two bounding boxes

→ way of measuring how similar the boxes are to each other

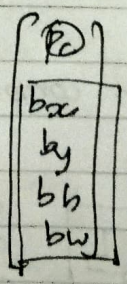
Non-max Suppression

way that your object algorithm detects the object only once.

commutes the max propro probability & supress the non-max ones



19 x 19



Discard all boxes with $p \leq 0.6$

while there are remaining bounding box

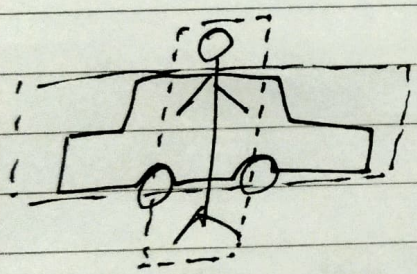
- Pick the box with the largest P_c output that as a prediction

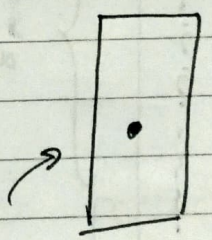
- Discard any remaining box with IoU ≥ 0.5 with the box output in previous step

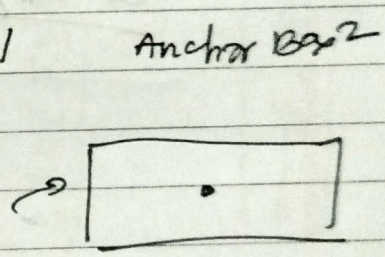
Independently carry out for every possible case.
① Car
② Pedestrian

Anchor Boxes

Overlapping objects







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

$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ p_c \\ b_x \\ \vdots \\ c_3 \end{bmatrix}$ anchor box 1, anchor box 2

Previously:

Previously

Each object in training image is assigned to grid cell that contains object midpoint

3x3x8

with 2 anchor boxes

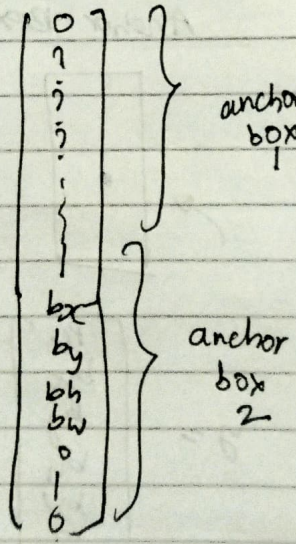
that contains object's midpoint and anchor box for the grid cell with highest IOU

output

3x3x16

3x3x2x8

car only



How to choose anchor boxes?

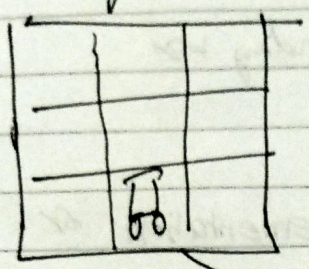
Previously hand picked

Now algorithms

Training

YOLO Algorithm

Training



1 → Pedestrian

2 → car

3 → motorcycle

$y =$ $\begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$ $\begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ \\ \\ \\ 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix}$

y is $3 \times 3 \times 2 \times 8$

- 3 × 3 → #anchors
- 2 → 5 + # classes

Outputting the non max supressed outputs

→ For each grid cell, get 2 predicted bounding boxes

→ Get rid of low probability predictions

→ For each class (pedestrian, car, motorcycle) use non-max supression to generate final predictions

Region Proposals

R-CNN Tries to pick just few windows where to run compner classifier

First run's through segmentation & find possible blobs and runs classifier on them

→ R-CNN Propose Regions. Classify proposed regions one at a time
Output label + bounding box

→ Fast RCNN → Propose regions -
Use convolutional implementation of sliding windows to classify all the proposed regions

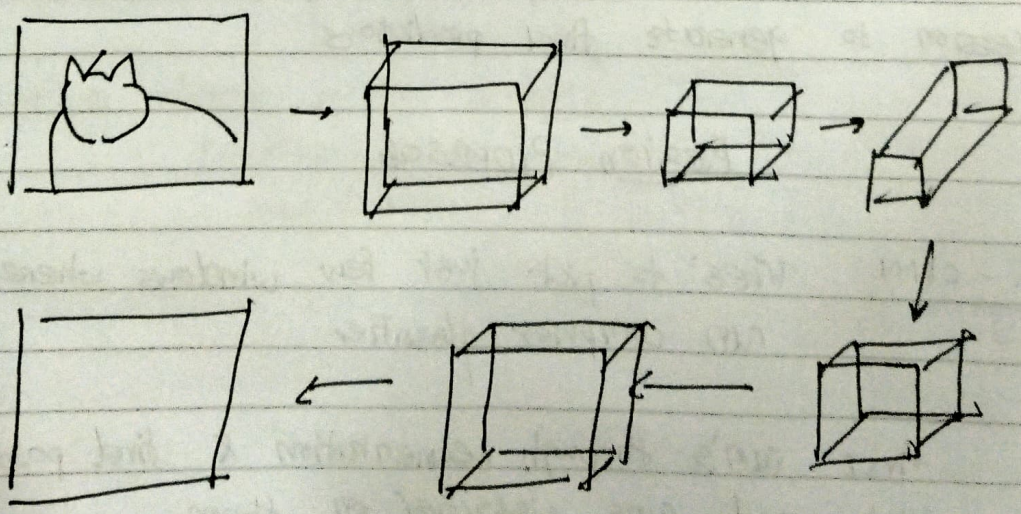
→ Faster RCNN Use convolutional network to propose regions

Semantic Segmentation with U-Net

UNet

Per Pixel class labels

1. Car
2. Building
3. Road

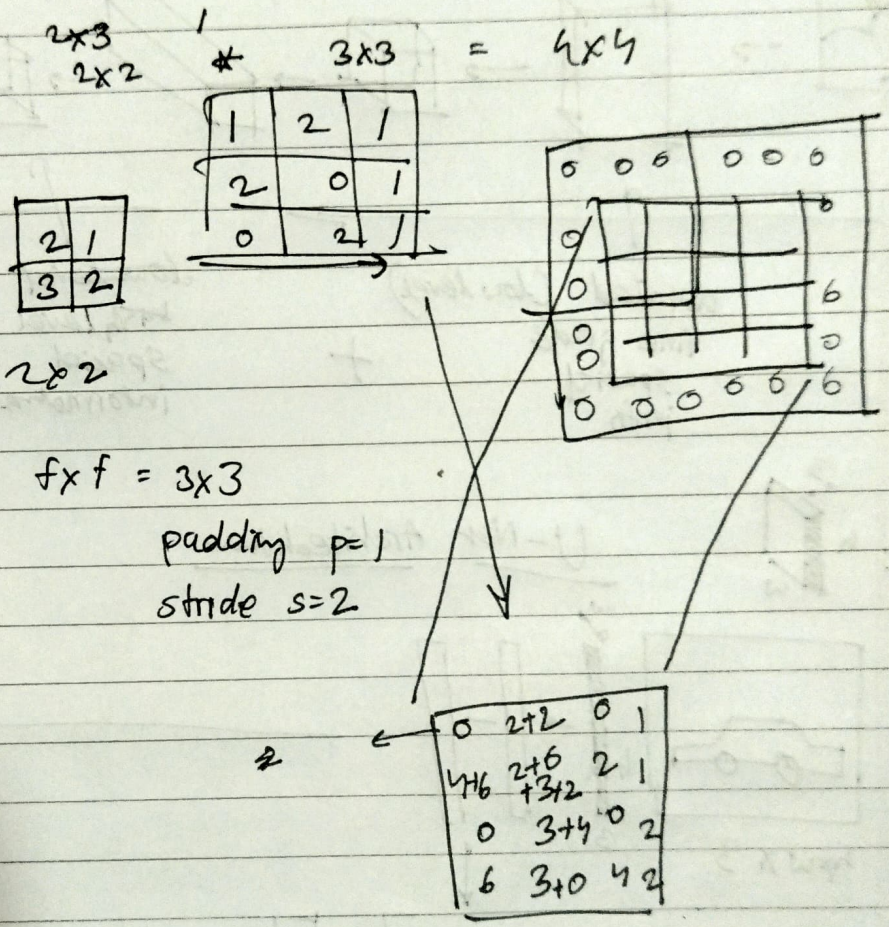


Transpose convolution

Normal convolution

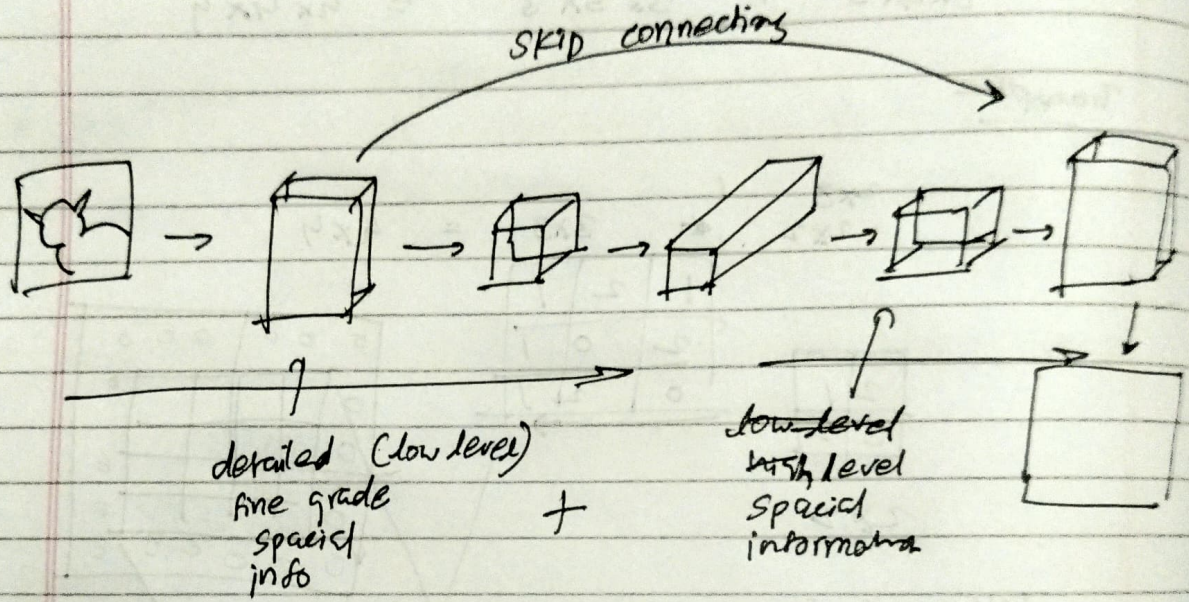
$6 \times 6 \times 3 \quad * \quad 3 \times 3 \times 3 \quad = \quad 4 \times 4 \times 4$

Transpose

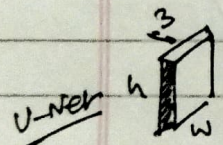


U Ner Architechture intution

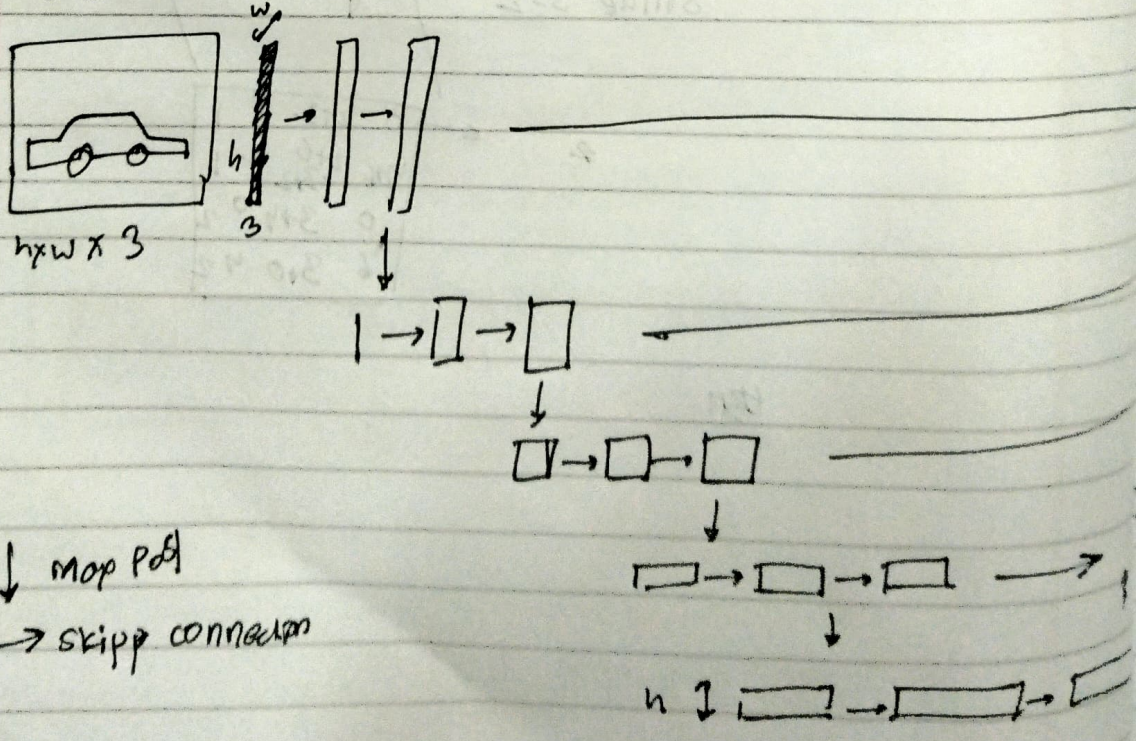
first compress image then second half uses transpose convolution to blow up

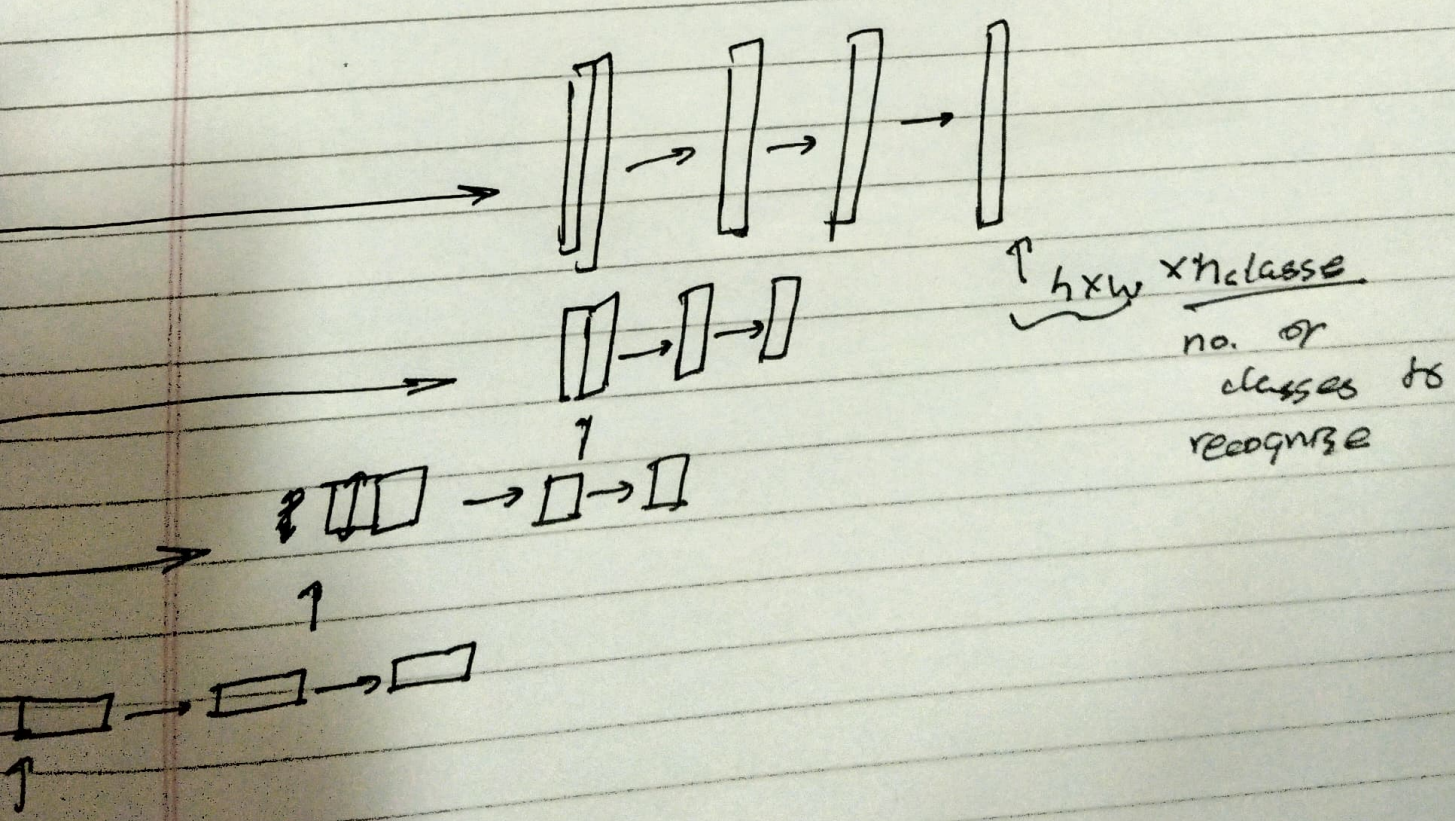


U-Ner



U-Ner Architechture




h x w x no. classes
no. of classes to recognize