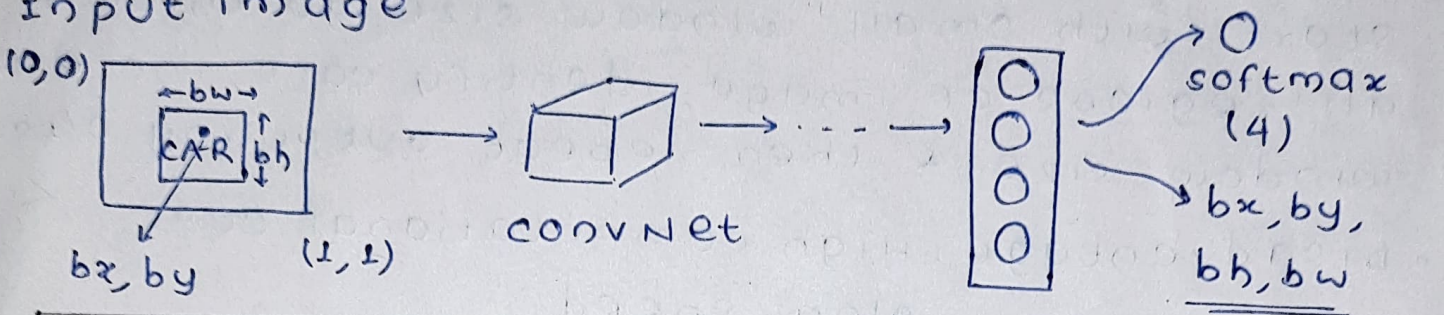
WEEK-3

- Image classification → 1 object
- classification with localization → 1 object
- object detection → multiple object (box around obj)



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

* Target label (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 an obj?
  - $P_c = 1$ for classes (1, 2, 3)
  - $P_c = 0$ for class 4
- If $P_c = 1$, these will be assigned values,
- If $P_c = 0$, these will be "?" → don't care
- Any 1 will be 1

* Loss $\mathcal{L}(\hat{y}, y) = (\hat{y_1} - y_1)^2 + (\hat{y_2} - y_2)^2 + \cdots + (\hat{y_8} - y_8)$ ⇒ $y_1 = 1$, $P_c = 1$
(squared error) $= (\hat{y_1} - y_1)^2$ ⇒ $P_c = y_1 = 0$

- LANDMARK DETECTION (landmark = imp. points)

eg: For 64 landmarks, no of output units = 129
1 = face / not face? & 128 = landmarks

$(l_{1x}, l_{1y}), (l_{2x}, l_{2y}), \ldots, (l_{64x}, l_{64y})$

* Landmark labels → consistent across images

* OBJECT DETECTION

	x	y
Labeled training set ⇒	closely cropped images of cars	0/1

then input image → convnet → y

- **sliding window detection**
  start with small window size, go through all regions of image, identify cars, increase window size & then repeat outputing 0/1s
- disadvantage: High computational cost
  slow speed

* Fully connected layer representation as convolutional layer

$14 \times 14 \times 3 \xrightarrow{5\times5,\ \text{filters}=16} 10\times10\times16 \xrightarrow[2\times2]{\text{Max pool}} 5\times5\times16 \xrightarrow{FC} 400 \xrightarrow{FC} 400 \rightarrow (y)\ 0$
softmax [4]

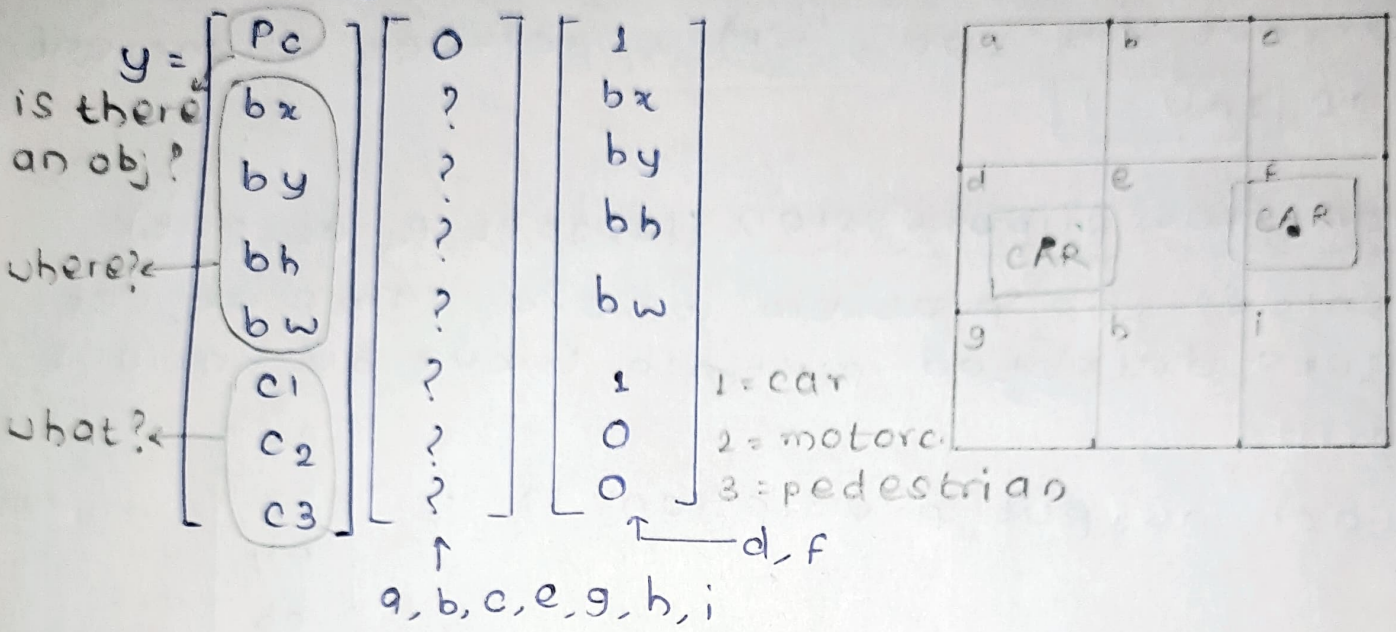
$14\times14\times3 \xrightarrow[16\ \text{filt.}]{5\times5} 10\times10\times16 \xrightarrow[2\times2]{\text{Max pool}} 5\times5\times16 \xrightarrow[5\times5,\ 400\ \text{filt}]{FC} 1\times1\times400\ (\text{vol.}) \xrightarrow[1\times1,\ 400\ \text{filt}]{FC} 1\times1\times400\ (\text{volume}) \xrightarrow{1\times1} 1\times1\times4\ (\text{volume})$

- sliding windows using convolution
  ⇒ process entire image at once.

problem ⇒ bounding boxes are not accurate

• YOLO ALGORITHM (You only look once)

→ To achieve accurate bounding boxes
→ Input images are layered with a grid
→ Training labels for each grid:



Target output = 3 × 3 × 8 ← dimensional target vector
(3 × 3 = grid cells)

$$y = \begin{bmatrix} \vdots \\ \vdots \\ \vdots \end{bmatrix}$$

* (TRAINING) FAST & EFFICIENT (useful for real time.)

Input [X] 100×100×3 → conv → pool → .... → output [Y] 3 × 3 × 8

* (Specify bounding box)

$$y = \begin{bmatrix} 1 \\ bx \\ by \\ bh \\ bw \\ 0 \\ 1 \\ 0 \end{bmatrix} \begin{matrix} \\ 0.4 \\ 0.3 \\ 0.5 \\ 0.9 \\ \\ \\ \end{matrix} \begin{matrix} \\ \} \text{ b/w 0 \& 1} \\ \\ \} \text{ can be } > 1 \\ \\ \\ \end{matrix}$$

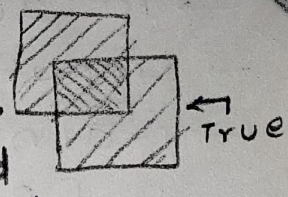
YOLO

NOTE → only one forward propagation pass

• Intersection over union → accuracy

→ To evaluate object detection algorithm

→ compute IOU = $\frac{\text{size of intersection}}{\text{size of union}}$



→ correct for IOU ≥ 0.5 & perfect alignment has IoU = 1

• Non-max suppression (largest P_c output)

→ solves the problem where same object gets detected multiple times by various grids

→ Each output prediction is $y = \begin{bmatrix} P_c \\ b_x \\ b_y \\ b_h \\ b_w \end{bmatrix}$

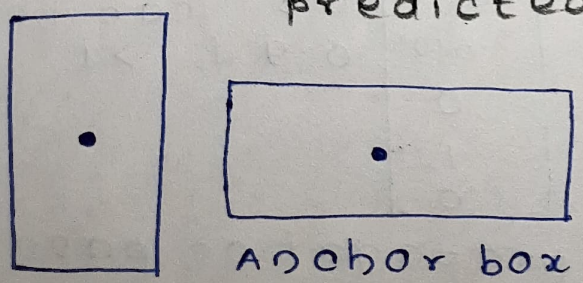
• c_1, c_2, c_3 not included as we are detecting car only

discard boxes with $P_c \leq 0.6$

For remaining boxes:

- pick box with largest P_c, output that as a prediction
- discard any remaining box with IoU ≥ 0.5 with box output in previous step. (largest P_c)

• Anchor boxes (To detect 2 objects in one grid) → suitable shape w.r.t output to be predicted.



$\Rightarrow y = \begin{bmatrix} P_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ P_c \\ \vdots \\ c_3 \end{bmatrix}$ Anchor box 1 / Anchor box 2

OUTPUT

~~3 × 3~~ × 8

↓

3 × 3 × 16

Grid

y

3 × 3 × 2 × 8

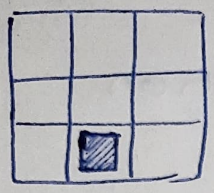
◦ YOLO ALGORITHM

? ⇒ random nos. or noise

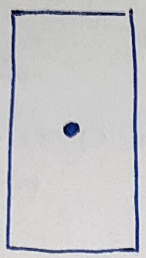
* Training

Input

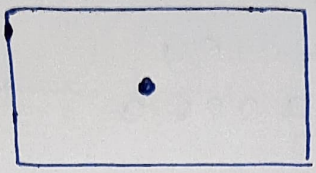
Image & 3×3 Grid



1 - pedestrian
2 - car
3 - motorcycle



A1



A2

$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ p_c \\ \vdots \\ \vdots \\ c_3 \end{bmatrix} \begin{matrix} \} A_1 \\ \\ \\ \\ \\ \\ \\ \\ \} A_2 \\ \\ \\ \end{matrix}$$

Output : 3×3×16 or 3×3×2×8

* outputting non-max supressed outputs

→ each grid cell = 2 predicted bounding boxes
→ eliminate low probability predictions ($p_c \leq 0.5$)
→ use non-max supression to generate final output prediction for each class

◦ Region proposal (R-CNN)

→ perform segmentation & run classifier (sliding window) on only those blobs

R-CNN : propose regions, classify proposed regions one at a time, output → label + bounding box

Fast R-CNN : propose regions, classify using convolutional implementation of sliding windows

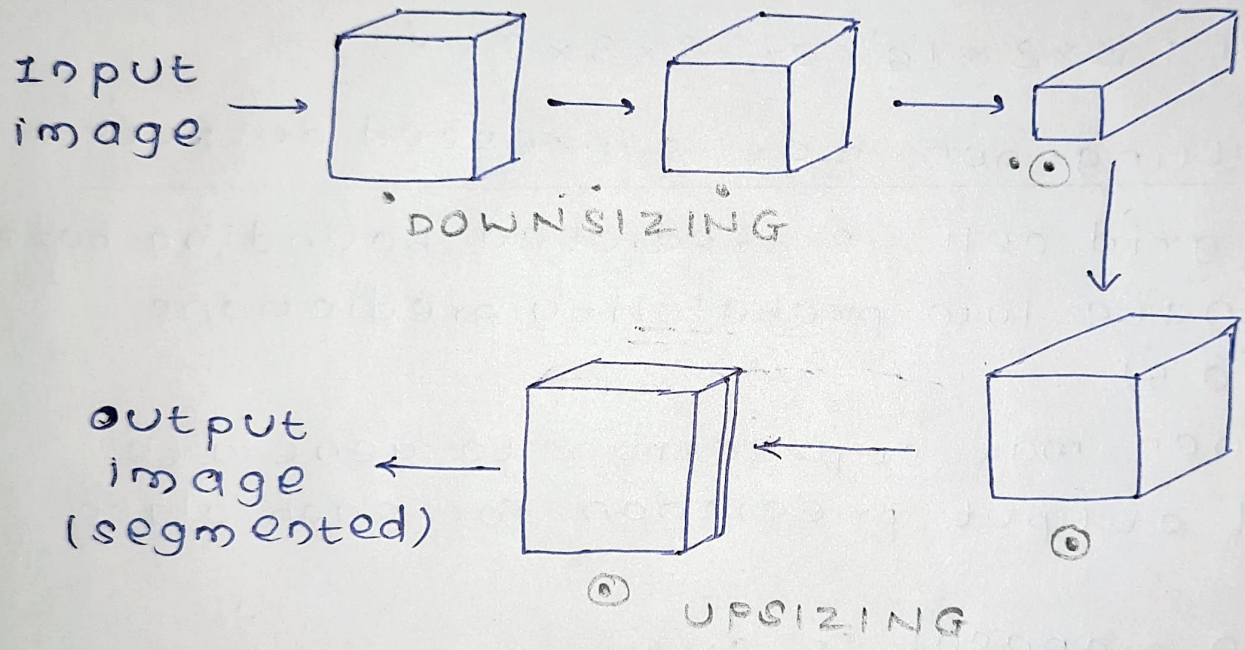
Faster R-CNN : use CNetwork to propose regions.

• SEMANTIC SEGMENTATION WITH U-NET

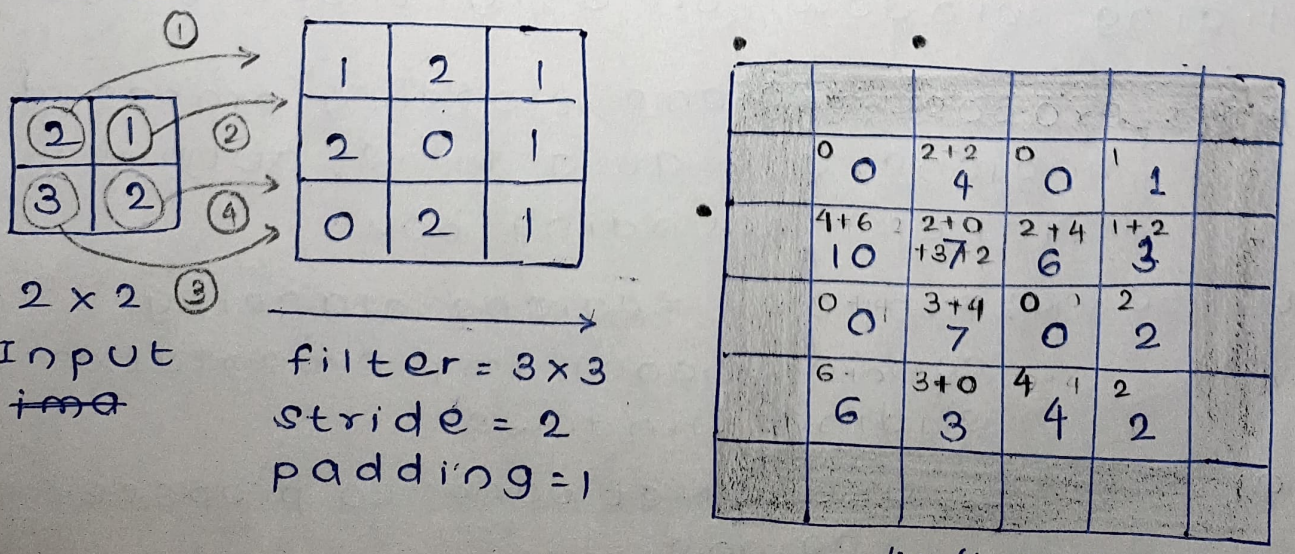
→ pixel categorization / labelling
→ draw exact outline around objects over use of bounding boxes.
→ uses : (i) self driving cars
(ii) Medical Imaging

→ input image → specify classes → segmentation map.

→ unet Architecture



* TRANSPOSE CONVOLUTION



- **U-NET**

ENCODER DECODER

Input image → convolution → Transpose convolutions → output image

* Skip connections (gives info from encoder to decoder)

Input image $h \times w \times 3$ → Encoder → Decoder → segmentation map $h \times w \times n$ classes

- for 3 classes o/p = $h \times w \times 3$

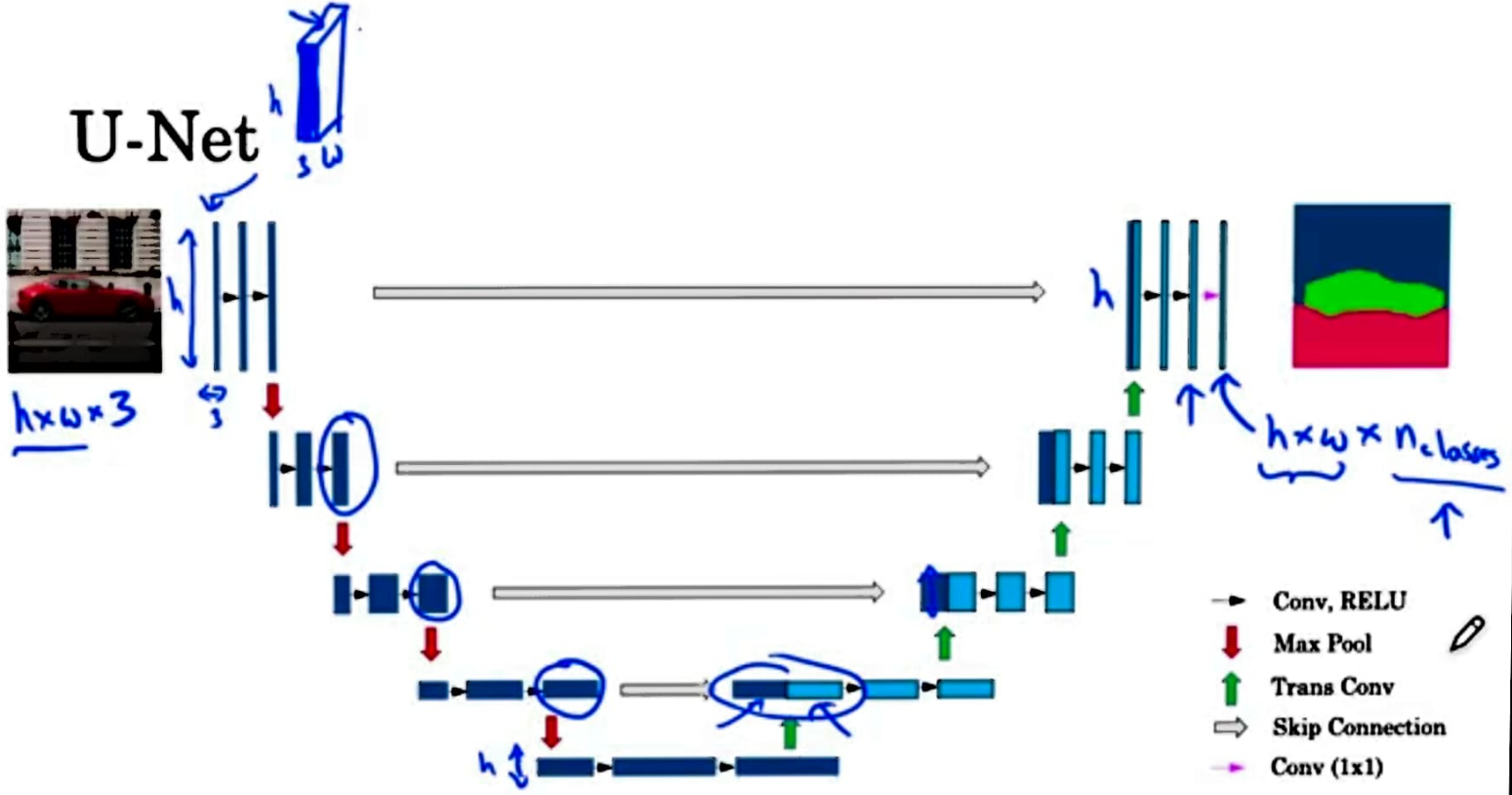
steps: (i) conv, ReLU →

(ii) Max Pool ⇩

(iii) Transconv ⇧

(iv) Skip connection ⇨

(v) conv (1×1) → ... (just before o/p)



[Ronneberger et al., 2015, U-Net: Convolutional Networks for Biomedical Image Segmentation]

Andrew Ng

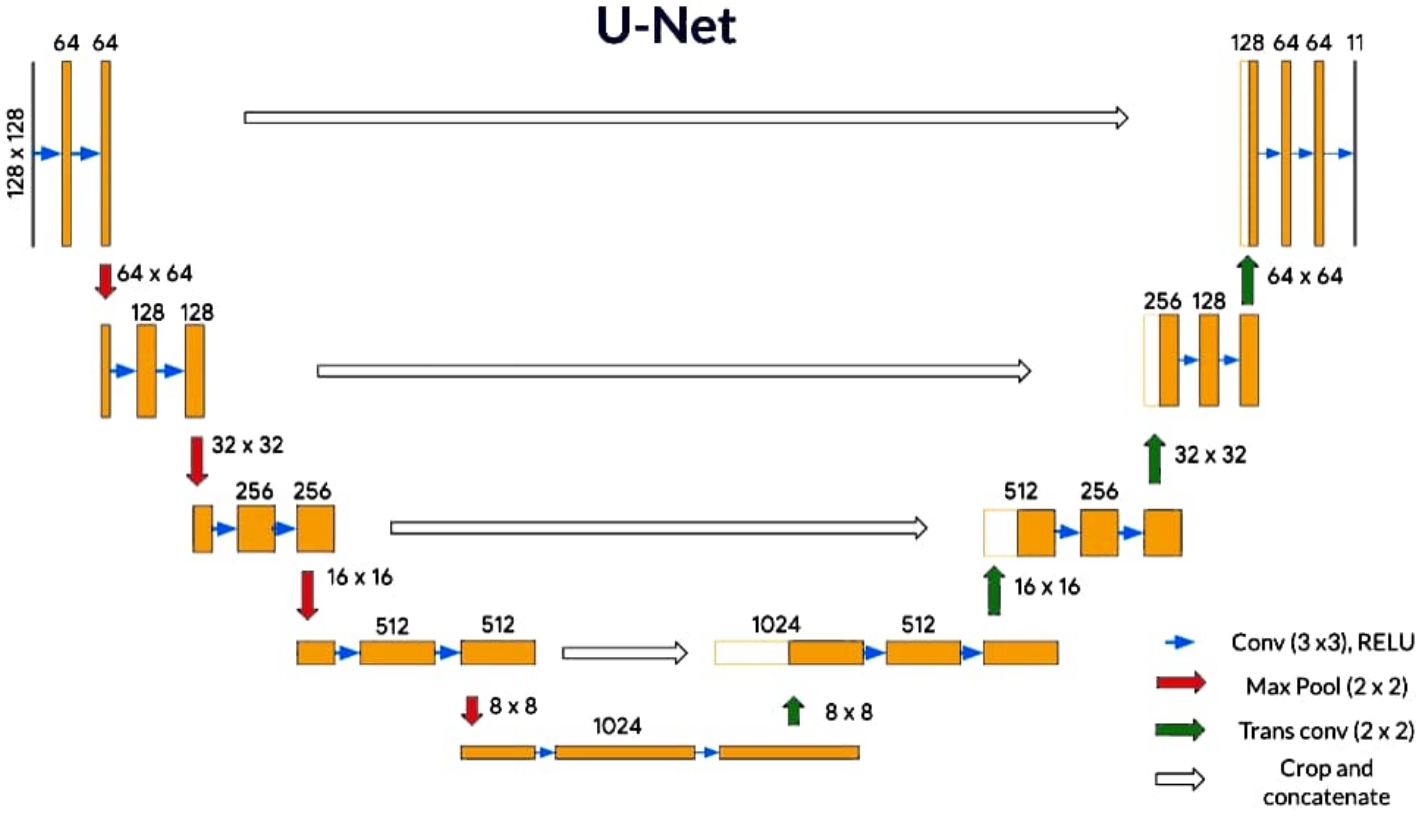


Figure 2 : U-Net Architecture

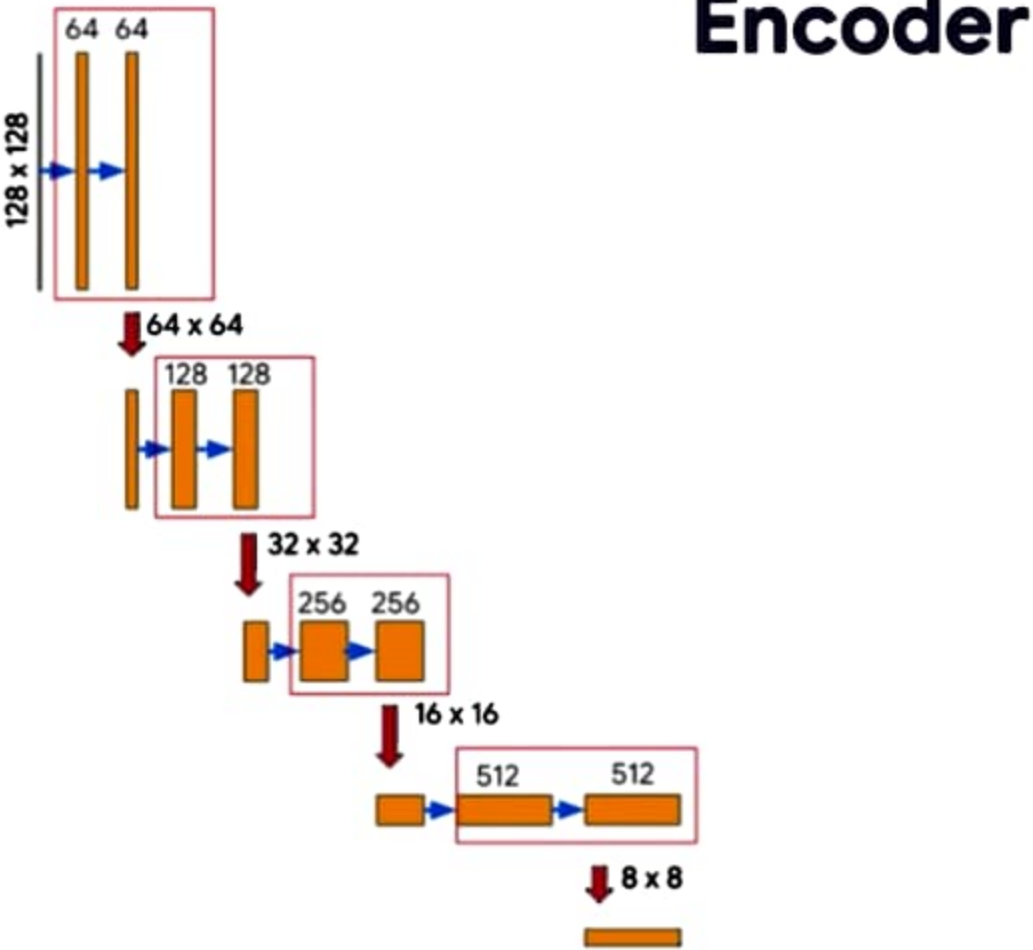


Figure 3: The U-Net Encoder up close