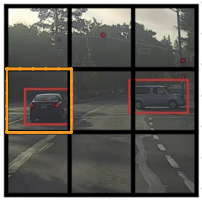
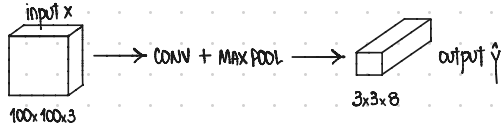
BOUNDING BOX PREDICTIONS

* YOLO (you only look once) algorithm: you divide your image into (+) regions, and with the localization algorithm, and for each of the grids, you specify a label y (8-dimensional vector).

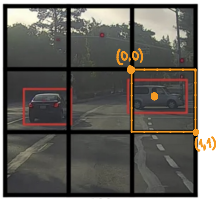


$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} \quad \begin{bmatrix} 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

* Target output will be $3 \times 3 \times 8$ → LABELS
  → GRID CELLS (REGIONS)



* the way you assign an object to a grid cell → you look at the midpoint of an object and then you assign that object to whichever one grid cell contains the midpoint. Even if the object is in multiple grids, that object is only assigned to one of the cells.



* how do you specify the bounding box?

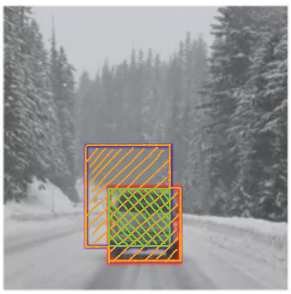
$$\begin{bmatrix} 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 0.4 \\ 0.3 \\ 0.9 \\ 0.5 \end{bmatrix}$$

BETWEEN 0 AND 1

THESE ARE SPECIFIED RELATIVE TO THE GRID CELL

CAN BE >1

INTERSECTION OVER UNION

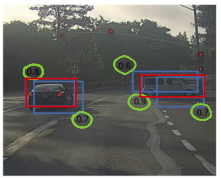


* it measures the overlap between two bounding boxes
* intersection over union = $\frac{\text{Size of } \square \text{ (intersection)}}{\text{Size of } \square \text{ (union)}}$
* bounding box is "correct" if IoU ≥ 0.5

NON-MAX SUPPRESSION

* one of the problems with object detection is that the algorithm may find multiple detections of the same object. Rather than detecting it once, it may detect it multiple times.

* it may be the case that the algorithm thinks the object midpoint (•) is in multiple cells. So you may end up with something like this:



* what max-suppression does is that it cleans up the detections so you end up with one detection per car.

* it looks at the p_c component in the y label and the IoU of the regions.
  - ↳ IS THIS A CAR? (PROBABILITY)

* you output your maximal probabilities classification, but suppress those that are non-maximal.

* each output prediction is: $\begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \end{bmatrix}$

(#1) discard all boxes with $p_c \leq 0.6$

(#2) while there are any remaining boxes:
- pick the box with the largest p_c and output as prediction
- discard any remaining box with IoU ≥ 0.5 with the box output in the previous step.

ANCHOR BOXES

* what if the objects are overlapping?

Anchor box 1:

Anchor box 2:

* you can create an output label y containing the elements of box anchor boxes.

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

ANCHOR BOX 1

ANCHOR BOX 2

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

* output y will be of shape 3×3×2×8
  → # OF ANCHOR BOXES

Anchor box example

Anchor box 1: Anchor box 2:

PEDESTRIAN CAR

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

PEDESTRIAN ANCHOR BOX

CAR ANCHOR BOX

YOLO ALGORITHM

* example:

Training

CLASSES
1 - pedestrian
2 - car
3 - motorcycle

$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$ 8

y is 3×3×2×8 → 5 + #CLASSES

GRID CELL #ANCHORS

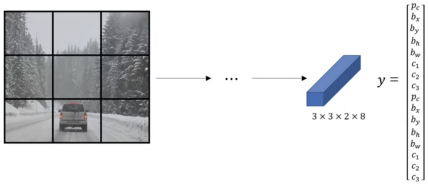
GRID 1 GRID 8

$\begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix} \quad \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix}$

THE CAR IS ASSOCIATED WITH ANCHOR BOX #2.

* for each of the 3×3 grids you come up with a 16-dimensional vector.

* making predictions:



* outputting the non-max suppressed outputs:

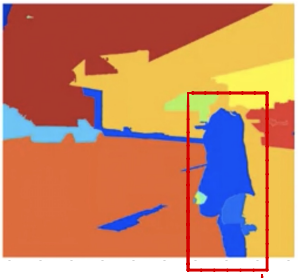
#1 for each grid cell, get 2 predicted bounding boxes.

#2 get rid of low probability predictions

#3 for each class use non-max suppression to generate final predictions.

REGION PROPOSALS

* it tries to pick some regions to run your (CNN) classifier on. Rather than running your sliding windows on every single grid cell, you select just a few and run it on those.



* you segmentate your image and run your algorithm on those regions.

FACE RECOGNITION

* face verification → input image, name ID
  → output whether the input image is that of the claimed person.

(VS)

* face recognition → has a database of (K) persons
  → get an input image
  → output ID of the image is any of the (K) persons (or "not recognized").

ONE-SHOT LEARNING

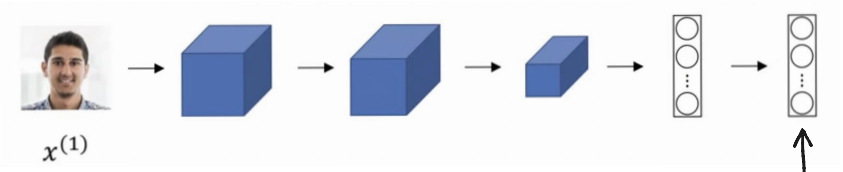
* one-shot learning problem → for most face recognition applications you need to be able to recognize the person with just one single image.

HOW TO SOLVE IT?

* learning "similarity" function → degree of difference between images: $d(img1, img2)$

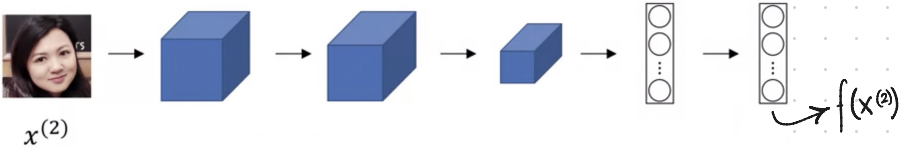
if $d(img1, img2) \leq \tau$: SAME
if $d(img1, img2) > \tau$: DIFFERENT

SIAMESE NETWORK



$f(x^{(1)})$: encoding of $x^{(1)}$
128 features

* you run two identical (CNN) on two (≠) inputs and compare them



$$d(x^{(1)}, x^{(2)}) = \| f(x^{(1)}) - f(x^{(2)}) \|^2$$

LEARNING SIMILARITY FUNCTION

* the parameters of the (NN) define an encoding $f(x^{(1)})$

you want to learn parameters so that:

- if $x^{(i)}$ and $x^{(j)}$ are the same person, $\| f(x^{(i)}) - f(x^{(j)}) \|^2$ is SMALL.
- if $x^{(i)}$ and $x^{(j)}$ are different persons, $\| f(x^{(i)}) - f(x^{(j)}) \|^2$ is LARGE.

TRIPLET LOSS

* good way of learning the parameters so that it gives you a good encoding for the pictures of faces.

$$\underbrace{\| f(A) - f(P) \|^2}_{d(A,P)} \leq \underbrace{\| f(A) - f(N) \|^2}_{d(A,N)}$$

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

→ TO AVOID TRIVIAL SOLUTIONS (MARGIN)

* given 3 images A, P, N:

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

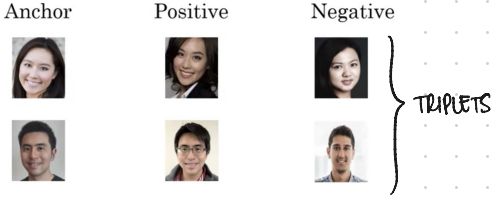
SO LONG AS YOU MAKE IT ≤ 0, THE LOSS WILL BE 0. INSTEAD, IF IT IS > 0, THEN THE LOSS WILL END UP BEING POSITIVE.

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

* how do you choose the triplets (A,P,N)?
  - during the training, if A,P,N are chosen randomly, $d(A,P) + \alpha \leq d(A,N)$ is easily satisfied.
  - you want to choose triplets that are "hard" to train on, so that $d(A,P) \approx d(A,N)$

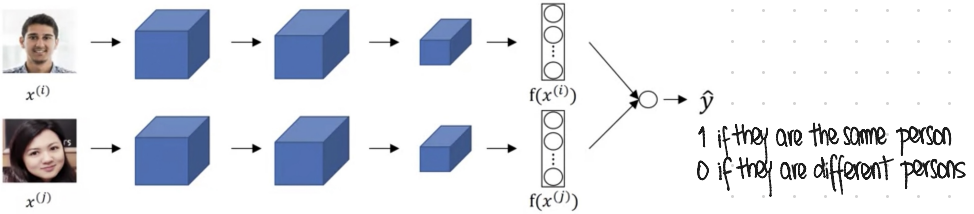
THE (NN) WILL HAVE TO PUSH IT DOWN

THE (NN) WILL HAVE TO PUSH IT UP

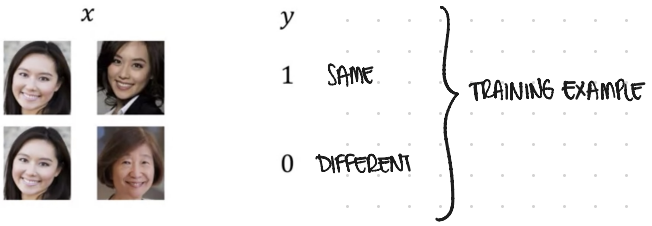


FACE VERIFICATION AND BINARY CLASSIFICATION

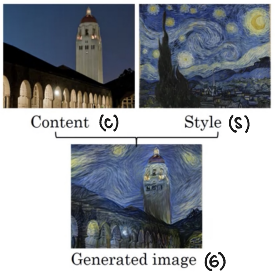
* another way of learning the parameters of a continent for face recognition



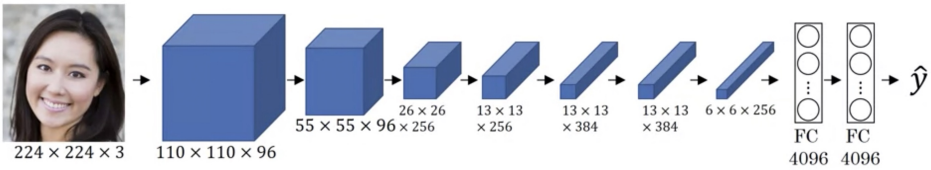
$$\hat{y} = \sigma\left(\sum_{k=1}^{128} w_k |f(x^{(i)})_k - f(x^{(j)})_k| + b\right)$$



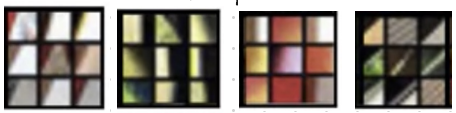
NEURAL STYLE TRANSFER



* in order to implement Neural Style Transfer, you need to look at the features extracted by ConvNet at various layers, the shallow, and the deeper layers of a ConvNet.



* visualizing what a deep network is learning:
  - pick a unit in layer 1 and find the image patches that maximize the unit's activation.
  - repeat for other units.



* in the deeper units the (NN) will see more general regions of the image.