

Ahmad Hussameldin Hamed Hassan

Shared Git-hub link: <https://github.com/ahmadhassan1993/sharing-github>

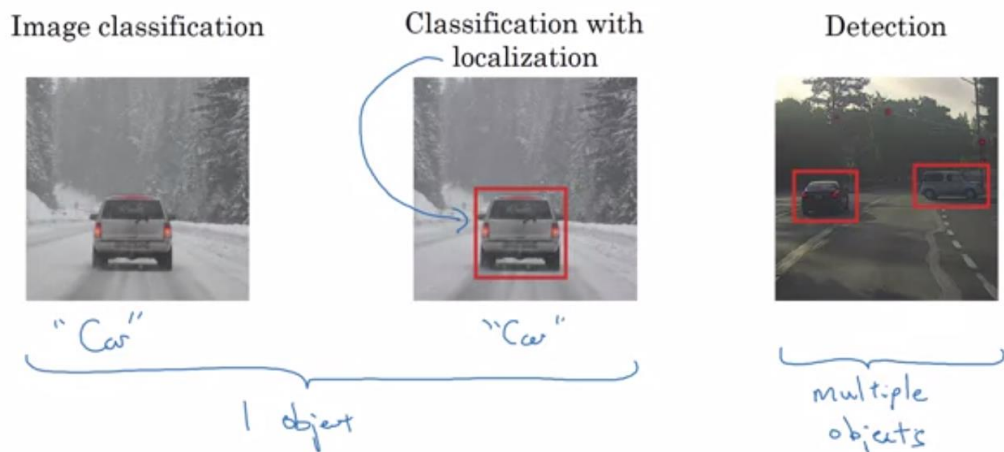
Convolutional Neural Network

Week 3 Summary

Detection Algorithms

Object Localization vs. Multi-Objects Detection:

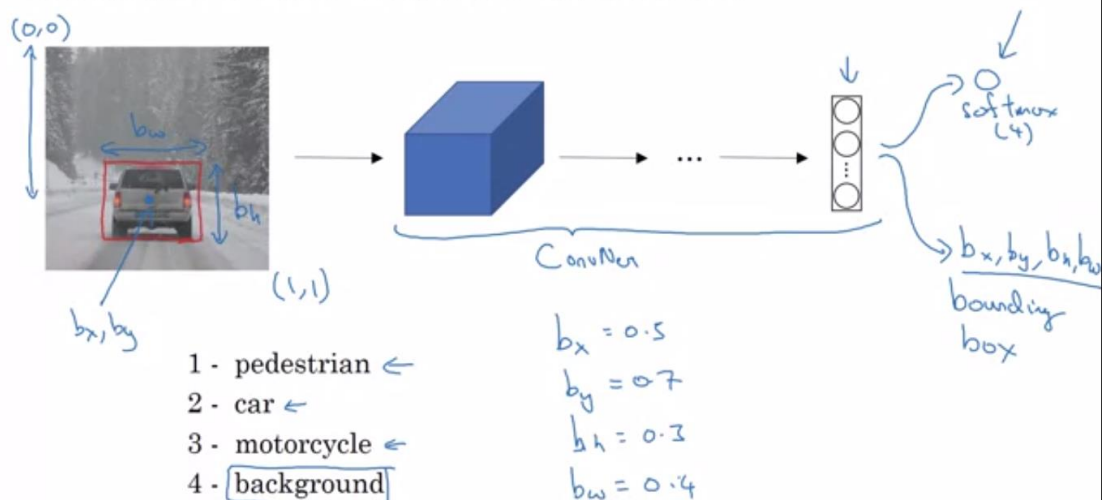
What are localization and detection?



Andrew Ng

1- Object Localization: by supervised learning

Classification with localization



Andrew Ng

Two examples with their outputs and loss functions:

Defining the target label y

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

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

$L(\hat{y}, y) = \begin{cases} (\hat{y}_1 - y_1)^2 + (\hat{y}_2 - y_2)^2 + \dots + (\hat{y}_n - y_n)^2 & \text{if } y_i = 1 \\ (\hat{y}_1 - y_1)^2 & \text{if } y_i = 0 \end{cases}$

$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$ is there an object?

(x, y)

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

$\begin{bmatrix} 0 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \end{bmatrix}$ ← "don't care"

Andrew Ng

Square error was used here for simplicity. In practice, we can use log error for classes C1, 2 and 3. Logistic regression for P_c and square error for bounding box.

2- Landmark Detection (specific point (x, y) in the image):

For example, defining specific part of a face or the capture pos (or emotion) of a picture:

Landmark detection

ConvNet → ConvNet → 129

AR

b_x, b_y, b_h, b_w

$\begin{bmatrix} l_{1x}, l_{1y} \\ l_{2x}, l_{2y} \\ l_{3x}, l_{3y} \\ l_{4x}, l_{4y} \\ \vdots \\ l_{64x}, l_{64y} \end{bmatrix} \begin{matrix} \left. \vphantom{\begin{matrix} l_{1x}, l_{1y} \\ l_{2x}, l_{2y} \\ l_{3x}, l_{3y} \\ l_{4x}, l_{4y} \\ \vdots \\ l_{64x}, l_{64y} \end{matrix}} \right\} X, Y \end{matrix}$

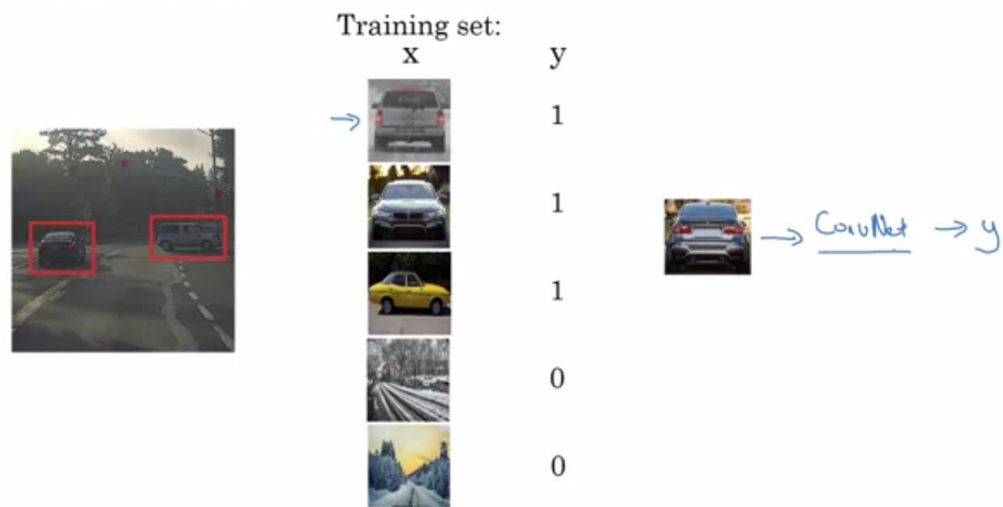
$\begin{bmatrix} l_{1x}, l_{1y} \\ \vdots \\ l_{31x}, l_{31y} \end{bmatrix}$

Andrew Ng

3- Object Detection:

1) Training:

Car detection example



Andrew Ng

2) Sliding window algorithm of CNN:

Sliding windows detection



Then large the window size and repeat:

Sliding windows detection

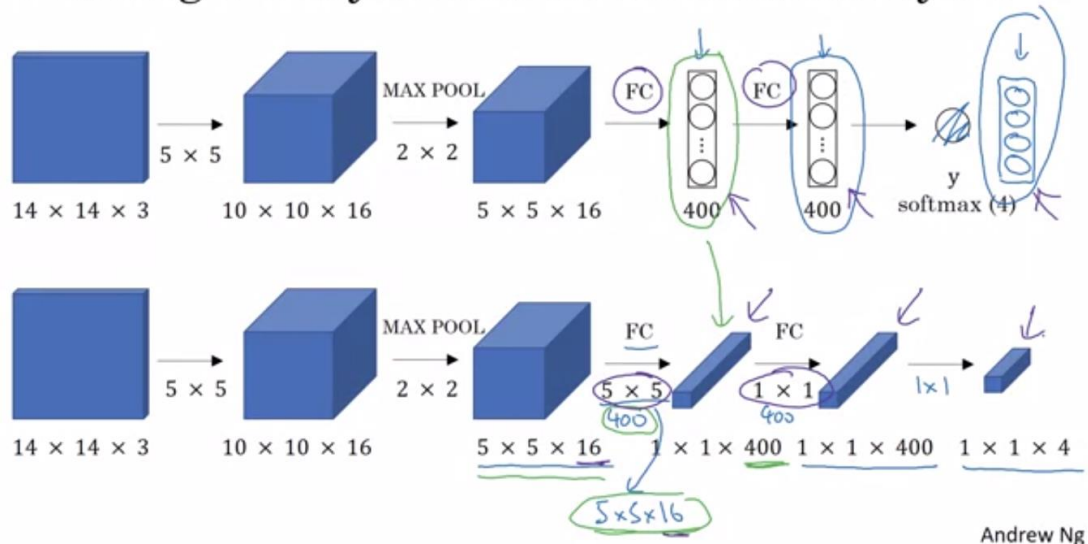


Andrew Ng

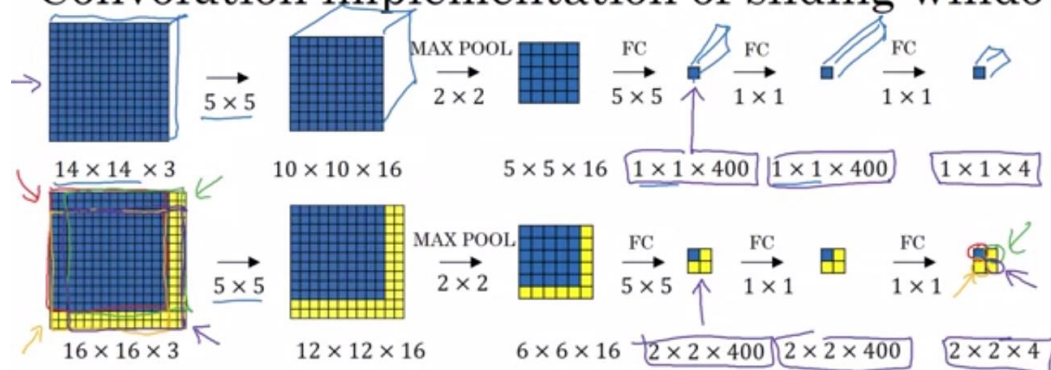
This was used before CNN era and there was a tradeoff between computation cost and performance depending on the window size choose.

3) Convolution implementation of sliding window:

Turning FC layer into convolutional layers



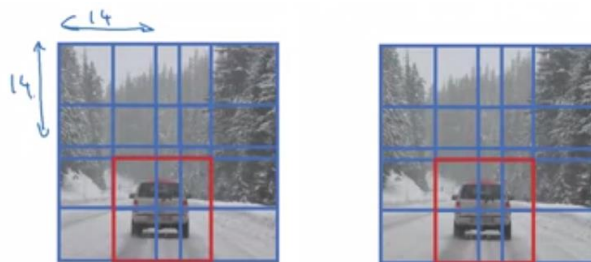
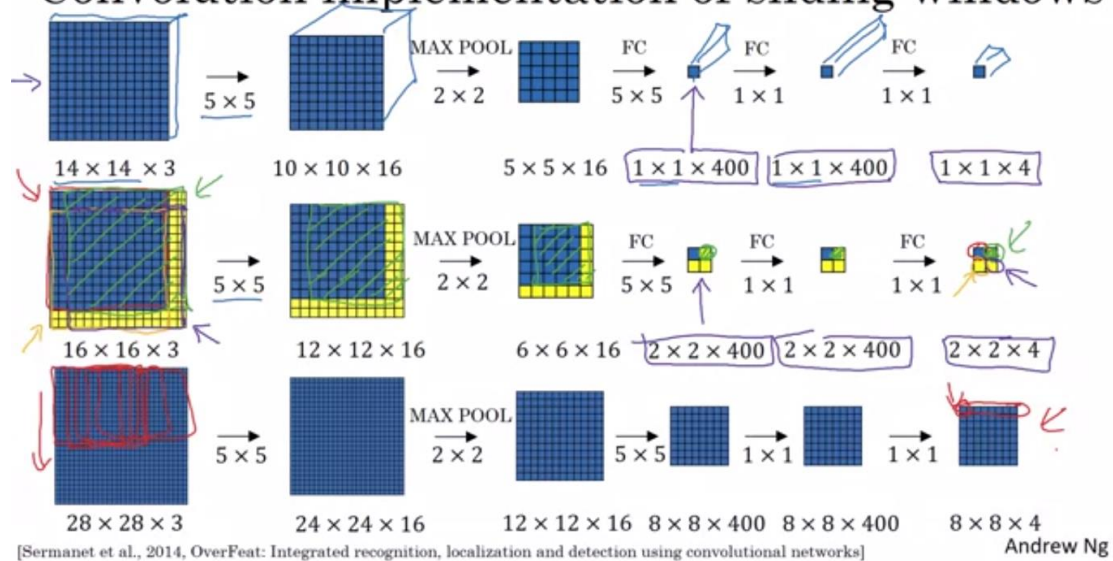
Convolution implementation of sliding windows



So, convolution made the computation cost of sliding window less because of sharing made by convolution.

Another example:

Convolution implementation of sliding windows

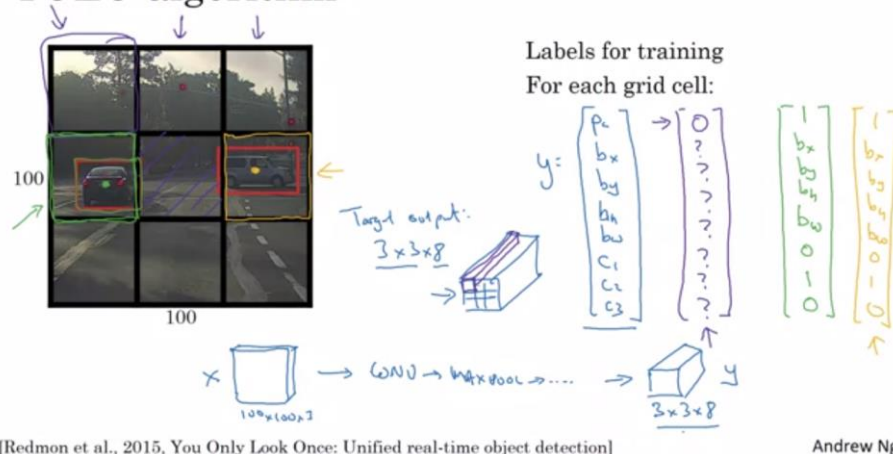


Andrew Ng

4) Output more accurate bounding boxes (YOLO (you only look once) Algorithm):

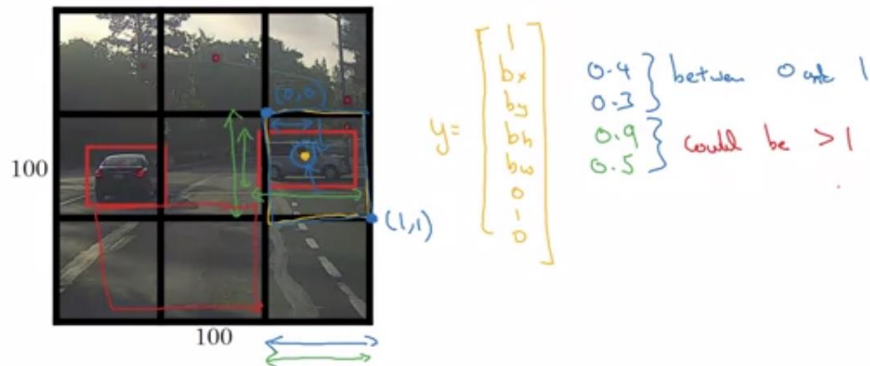
- (1) Apply grid cells on the image (more cell numbers, more accuracy)
- (2) Apply object localization for each cell (we must be sure that each cell has single object (in labeling phase before training))

YOLO algorithm



We should specify the bounds with respect to the cell dimensions:

Specify the bounding boxes



[Redmon et al., 2015, You Only Look Once: Unified real-time object detection]

Andrew Ng

5) Evaluating the output bond box accuracy (Intersection over union, IoU):

The higher the IoU, the more accurate is the bound box from the algorithm:

Evaluating object localization

Intersection over Union (IoU)

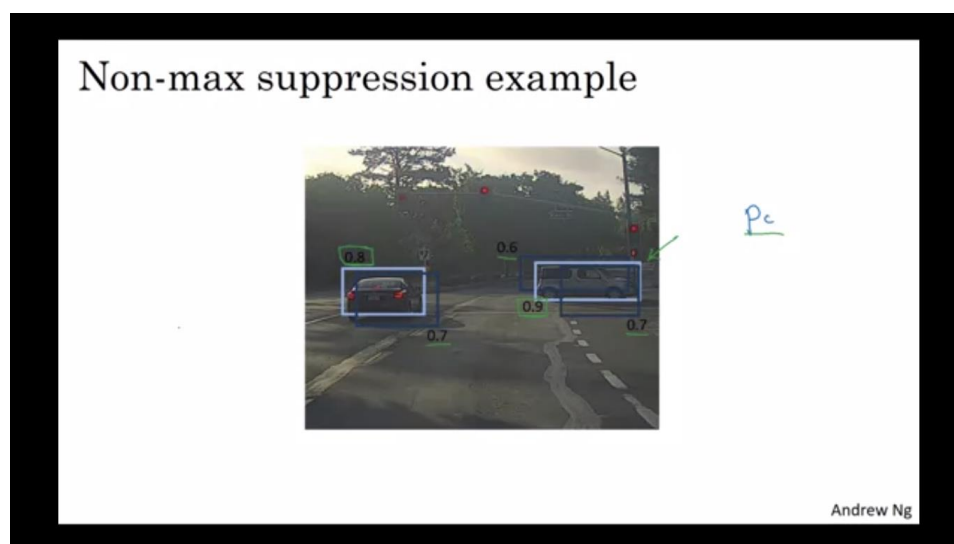
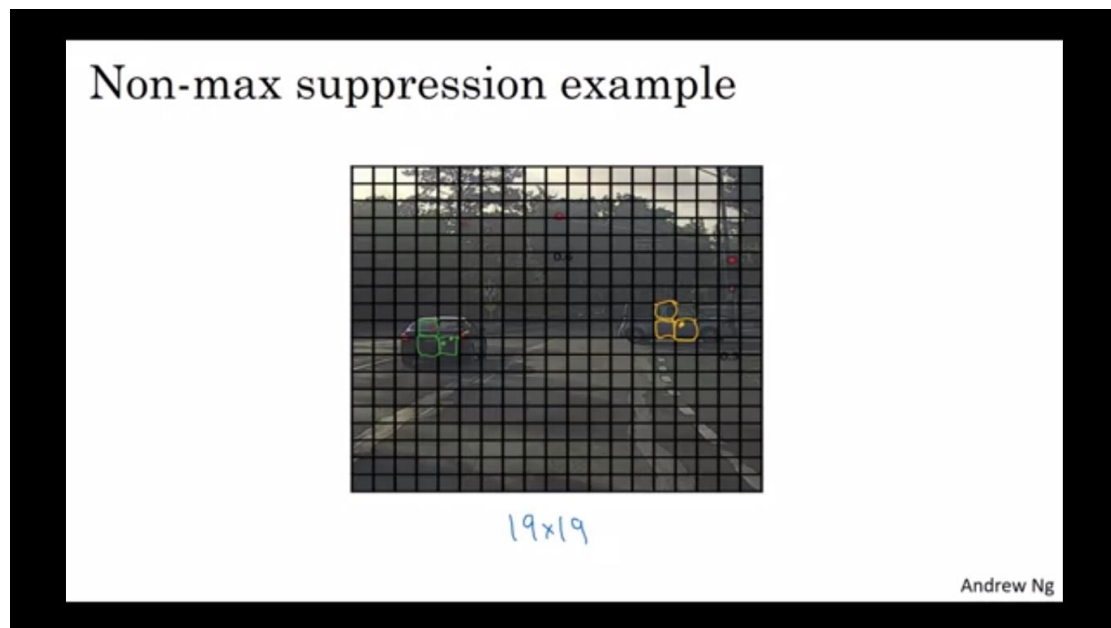
$$= \frac{\text{size of } \text{[yellow box]}}{\text{size of } \text{[green box] + [red box] - [yellow box]}}$$

"Correct" if $\text{IoU} \geq 0.5$

0.6

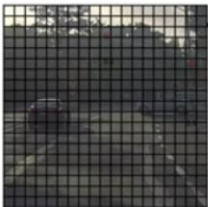
Andrew Ng

6) Detecting each object only once (Non-Max Suppression):



The algorithm:

Non-max suppression algorithm



19x19

Each output prediction is:

p_c
 b_x
 b_y
 b_h
 b_w

c_1
 c_2
 c_3

Discard all boxes with $p_c \leq 0.6$

→ While there are any remaining boxes:

- Pick the box with the largest p_c
Output that as a prediction.
- Discard any remaining box with $\text{IoU} \geq 0.5$ with the box output in the previous step

Andrew Ng

7) Two Objects detection in single cell (Anchor boxes Algorithm):

Overlapping objects:

Anchor box 1:

Anchor box 2:

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

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

[Redmon et al., 2015, You Only Look Once: Unified real-time object detection] Andrew Ng

Anchor box algorithm

Previously:

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

Output y :
 $3 \times 2 \times 8$

With two anchor boxes:

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

Output y :
 $3 \times 3 \times 16$
 $3 \times 3 \times 2 \times 8$

(grid cell, anchor box)

Andrew Ng

Anchor box example

Anchor box 1:

Anchor box 2:

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

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

Anchor box 1:

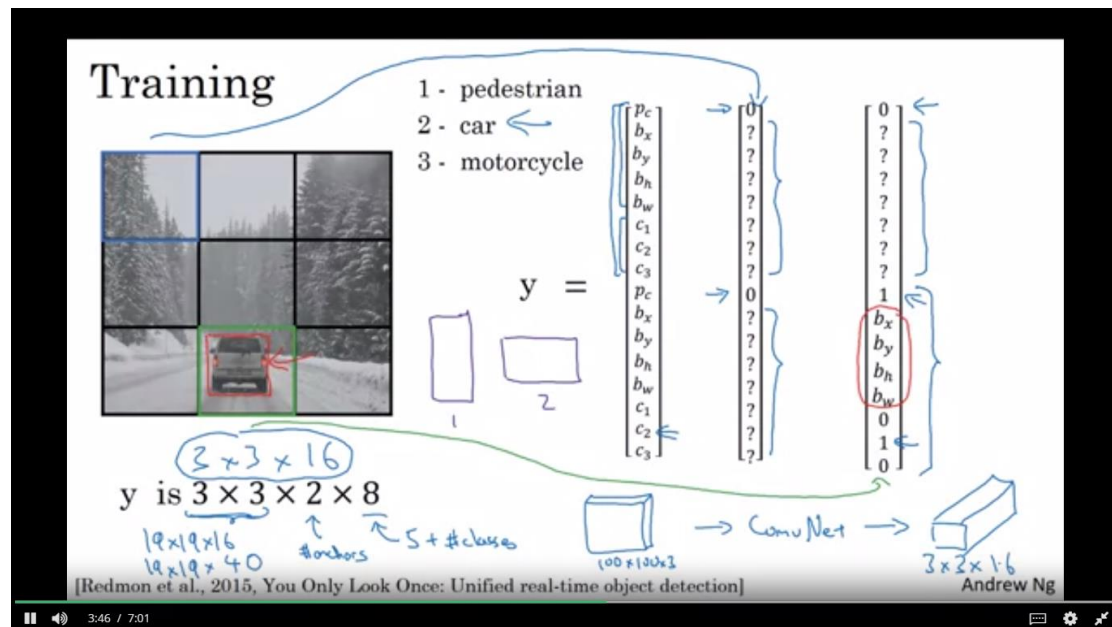
Anchor box 2:

Andrew Ng

This algorithm will not work:

- If we have more than two objects in a single cell
- If two objects but with same anchor box

Summary:



Outputting the non-max suppressed 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 suppression to generate final predictions.

Andrew Ng

Shared git-hub: [https://github.com/ahmadhassan1993/sharing-github/blob/master/Autonomous_driving_application_Car_detection_v3a%20\(1\).ipynb](https://github.com/ahmadhassan1993/sharing-github/blob/master/Autonomous_driving_application_Car_detection_v3a%20(1).ipynb)

Week 4 Summary

Face Recognition

Face verification vs. face recognition

→ Verification 1:1 99.9%

- Input image, name/ID
- Output whether the input image is that of the claimed person 99.9%

→ Recognition 1:K


- Has a database of K persons
- Get an input image K=100 ←
- Output ID if the image is any of the K persons (or “not recognized”)


Andrew Ng


We need to make the accuracy of face verification as high as possible to be able to implement face recognition.


Training problem (one-shot learning or single data set):


One-shot learning

→ 

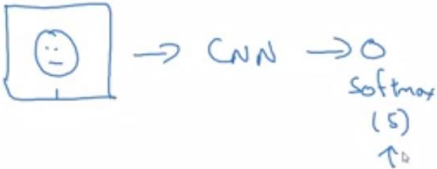
→ 

→ 

→ 

→ 

Learning from one example to recognize the person again



Andrew Ng

To solve this, we make the Similarity learning function:

Learning a “similarity” function

→ $d(\text{img1}, \text{img2})$ = degree of difference between images

If $d(\text{img1}, \text{img2}) \leq \tau$
 $> \tau$

“same”
 “different” } Verification.

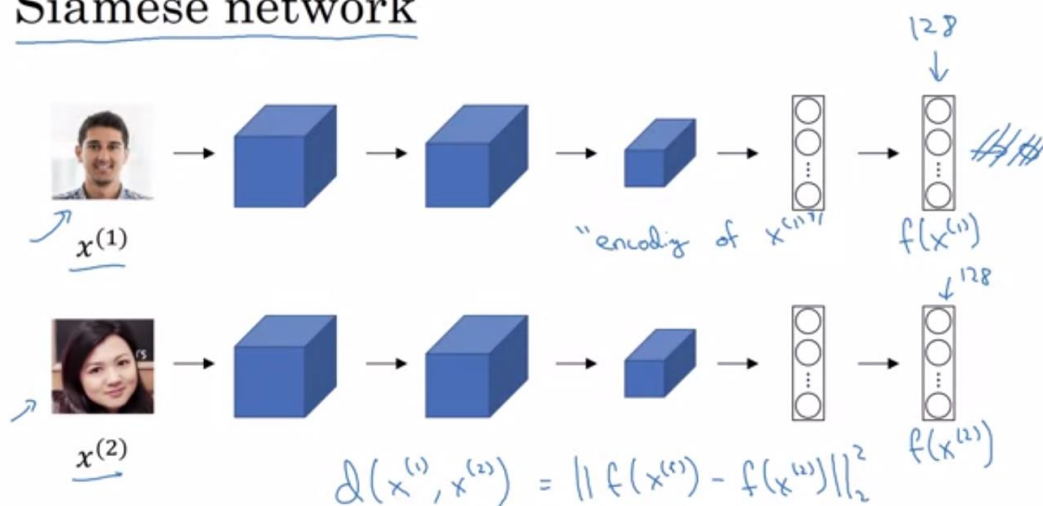


$d(\text{img1}, \text{img2})$

Andrew Ng

Siamese Network (implementation of similarity learning function):

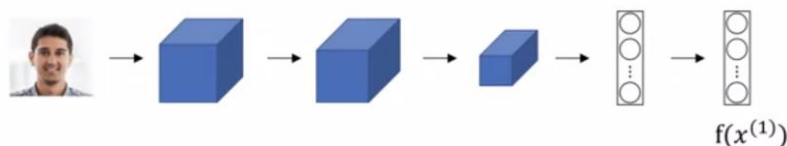
Siamese network



[Taigman et. al., 2014, DeepFace closing the gap to human level performance]

Andrew Ng

Goal of learning



Parameters of NN define an encoding $f(x^{(i)})$ 128

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.

A) Triplet Loss Function (learning NN's parameters to get good encoding):

- 1- We use margin α to make positive difference and negative difference functions away in values from each other to prevent trivial solution (each of them equals zero):

Learning Objective



The diagram shows two pairs of face images. The first pair consists of two identical images of a woman, labeled 'Anchor' (A) and 'Positive' (P). The second pair consists of two different images of women, labeled 'Anchor' (A) and 'Negative' (N). Below each pair, the distance is calculated as the squared L2 norm of the difference between their feature vectors: $d(A, P) = \|f(A) - f(P)\|^2$ and $d(A, N) = \|f(A) - f(N)\|^2$. Handwritten notes indicate that for positive examples, the distance should be small (e.g., 0.5), and for negative examples, it should be large (e.g., 0.7). The triplet loss formula is written as:
$$L(A, P, N) = \max(\|f(A) - f(P)\|^2 - \|f(A) - f(N)\|^2 + \alpha, 0)$$
 where α is the margin. A note states $f(\text{img}) = \vec{0}$.

[Schroff et al., 2015, FaceNet: A unified embedding for face recognition and clustering]

Andrew Ng

Positive is same example and negative is vice versa.

- 2- Loss and cost functions:

Loss function

Given 3 images A, P, N :

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

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

Training set: 10k pictures of 1k persons

The diagram shows arrows indicating that the training set provides multiple examples of positive pairs (A, P) and negative pairs (A, N) for the loss function.

[Schroff et al., 2015, FaceNet: A unified embedding for face recognition and clustering]

Andrew Ng

We should have many photos of single person (positive examples) to train NN.

We should take hard examples to train the NN in order to make the gradient descent very efficient:

Choosing the triplets A,P,N

During training, if A,P,N are chosen randomly,
 $d(A,P) + \alpha \leq d(A,N)$ is easily satisfied.

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

Choose triplets that're "hard" to train on.

$$\frac{d(A,P) + \alpha}{d(A,P)} \leq \frac{d(A,N)}{d(A,N)}$$

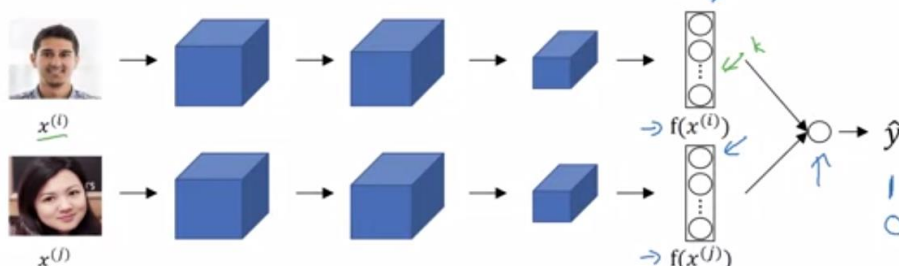
Face Net
Deep Face

[Schroff et al., 2015, FaceNet: A unified embedding for face recognition and clustering]

Andrew Ng

B) Binary classification (1 for same persons, 0 for different persons):

Learning the similarity function



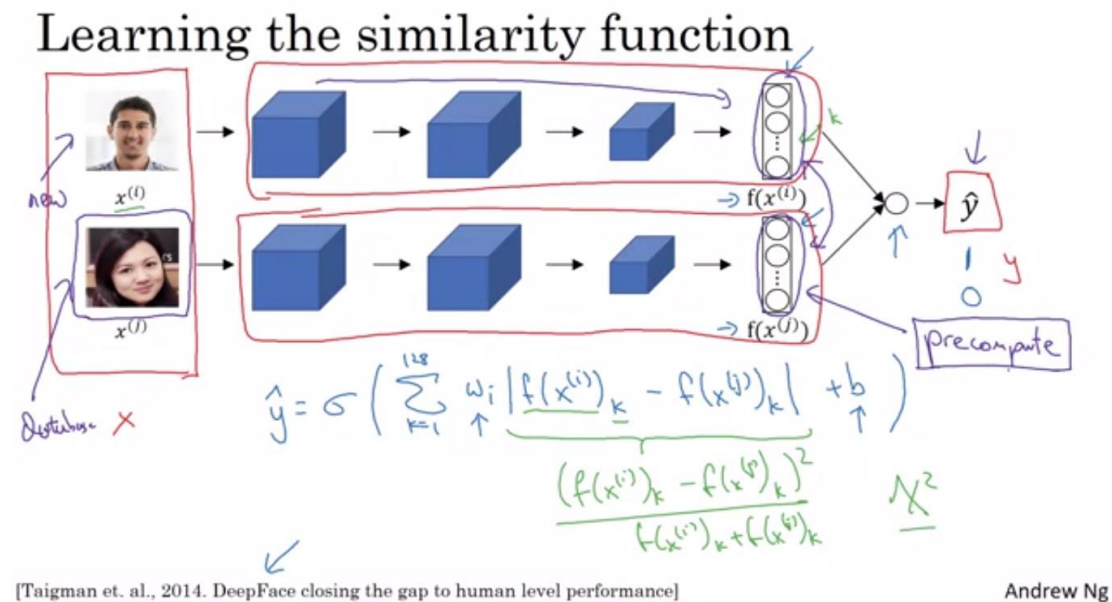
$$\hat{y} = \sigma \left(\sum_{k=1}^{128} w_k \frac{(f(x^{(i)})_k - f(x^{(j)})_k)^2}{f(x^{(i)})_k + f(x^{(j)})_k} + b \right)$$

[Taigman et. al., 2014. DeepFace closing the gap to human level performance]

Andrew Ng

Note that both networks have same parameters.

We can precompute one branch if we already know the image before, and this will reduce memory overhead for the system (when employees for example are very large):



Note that this can also be used in Triple loss function.

Shared git-hub: https://github.com/ahmadhassan1993/sharing-github/blob/master/Face_Recognition_v3a.ipynb