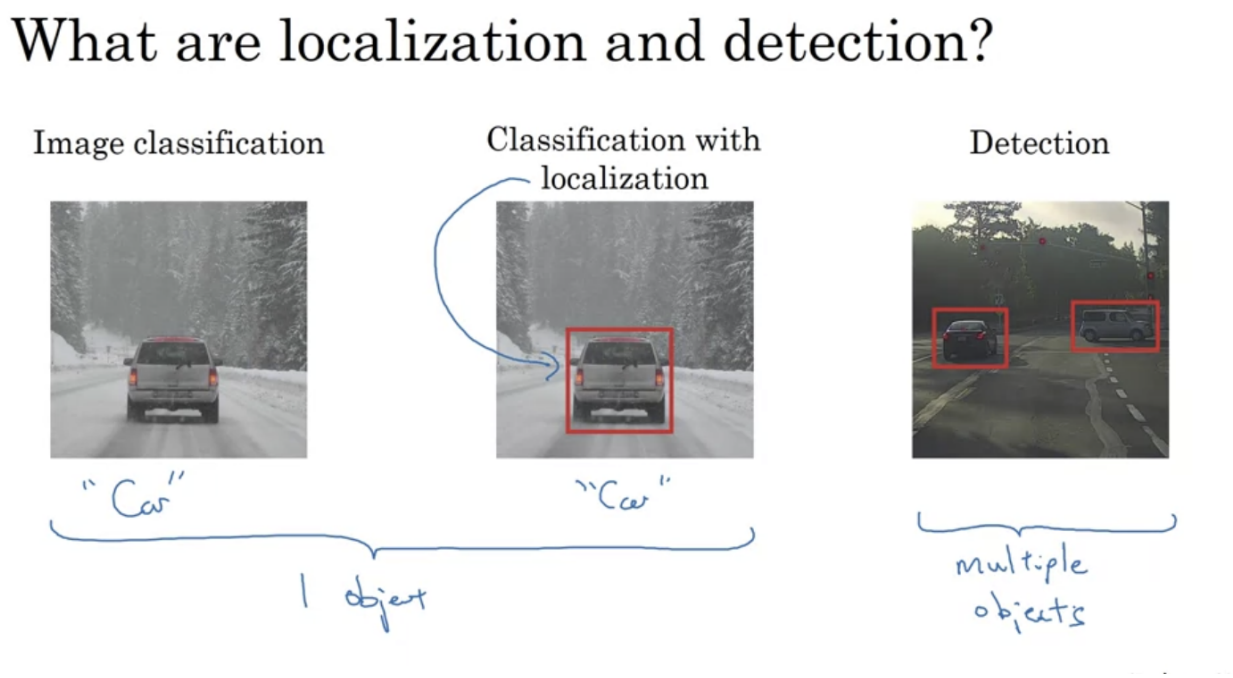
Convolutional Neural Networks DeepLearning AI

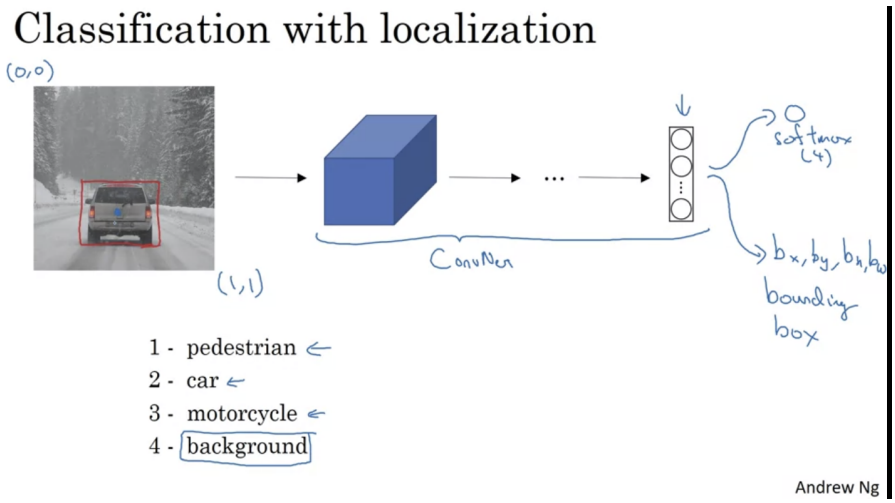
Detection Algorithms

Alec Dewulf

Object Localization



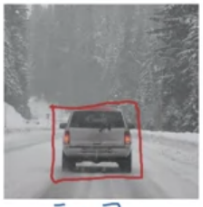
You can have a network output four more numbers to parameterize a bounding box to surround the detected object.



Defining the target label y

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

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

$x =$ 

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

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

Andrew N

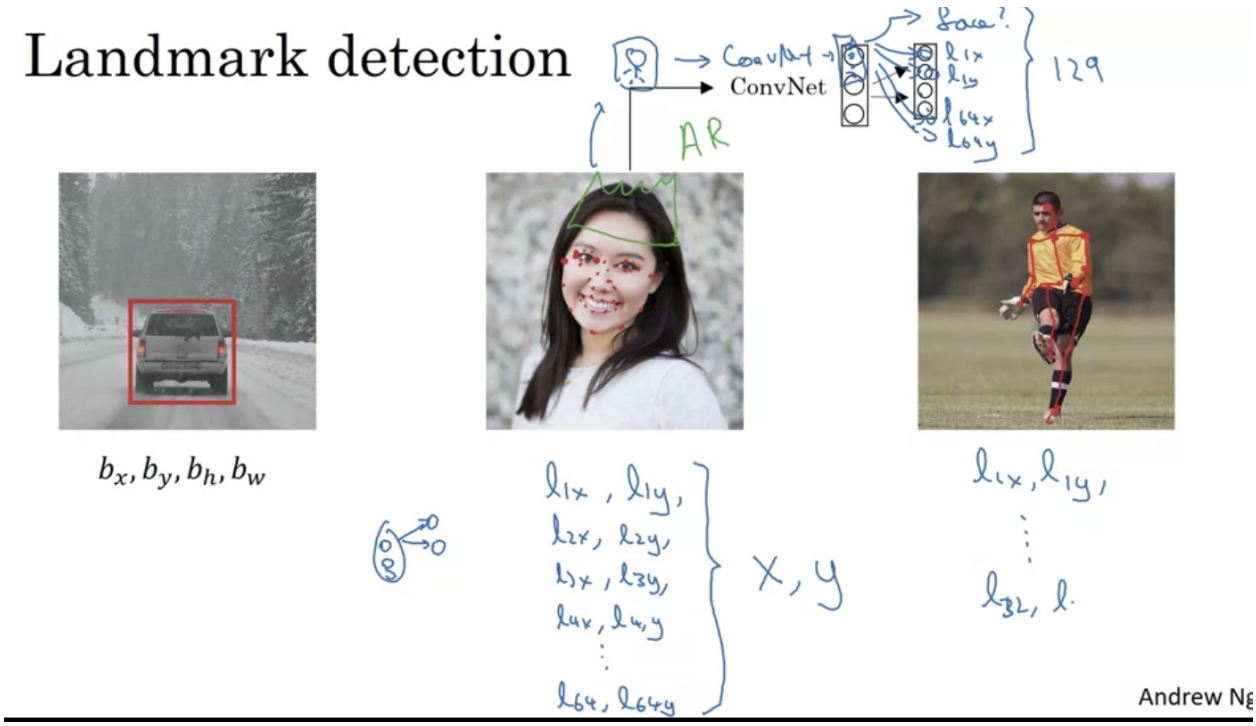
$$\mathcal{L}(\hat{y}, y) = \begin{cases} (\hat{y}_1 - y_1)^2 + (\hat{y}_2 - y_2)^2 + \cdots + (\hat{y}_8 - y_8)^2 & \text{if } y_1 = 1 \\ (\hat{y}_1 - y_1)^2 & \text{if } y_1 = 0 \end{cases}$$

Loss function has two cases. If there's no object we only care whether or not there was an object. Not what the predictions were.

Landmark Detection
Landmarks are important points in an image that you might want the algorithm to recognize. This can be represented by two numbers: the x and y coordinates.

You can output a vector of landmarks.

You can also define key positions if you are interested in pose estimation. By outputting landmarks in key positions you can output the pose of a person.



The landmark labels have to be consistent across images.

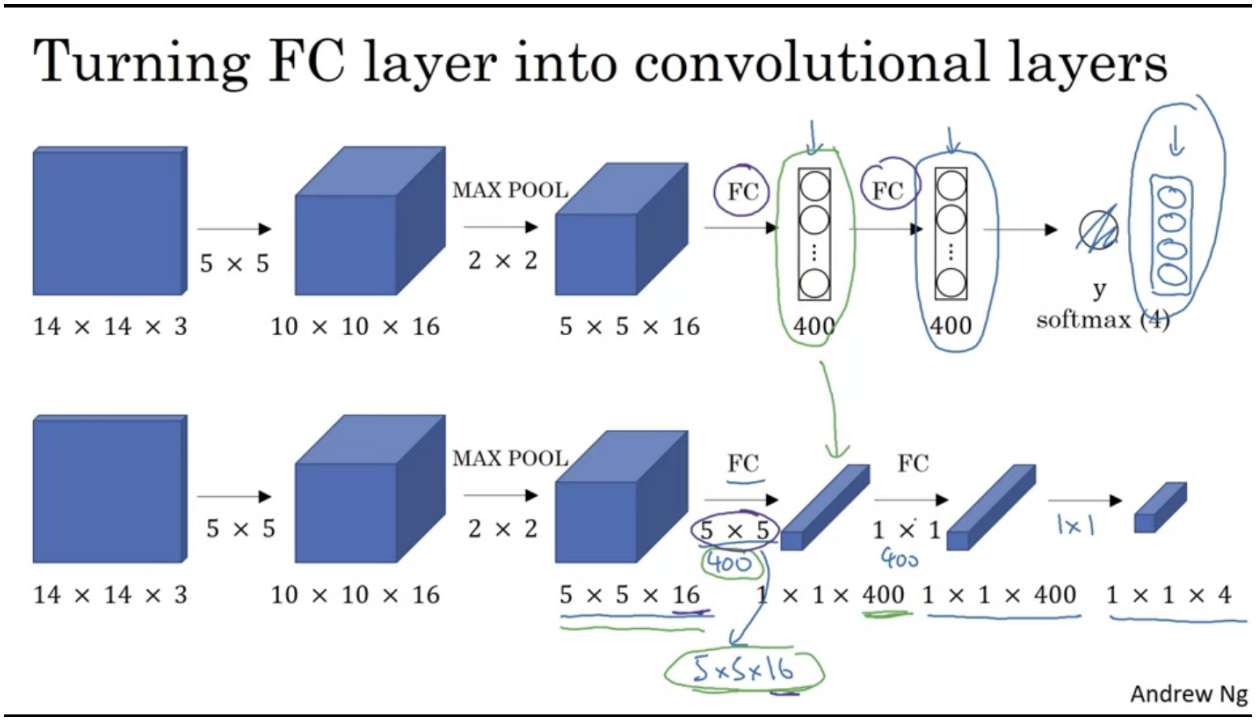
Object Detection
Train a ConvNet with closely cropped images to use with sliding window detection. The area outlined by the sliding window will be input to this model.

Use the sliding window multiple times with larger windows each time. Using sliding windows has a huge computational cost that is infeasible.

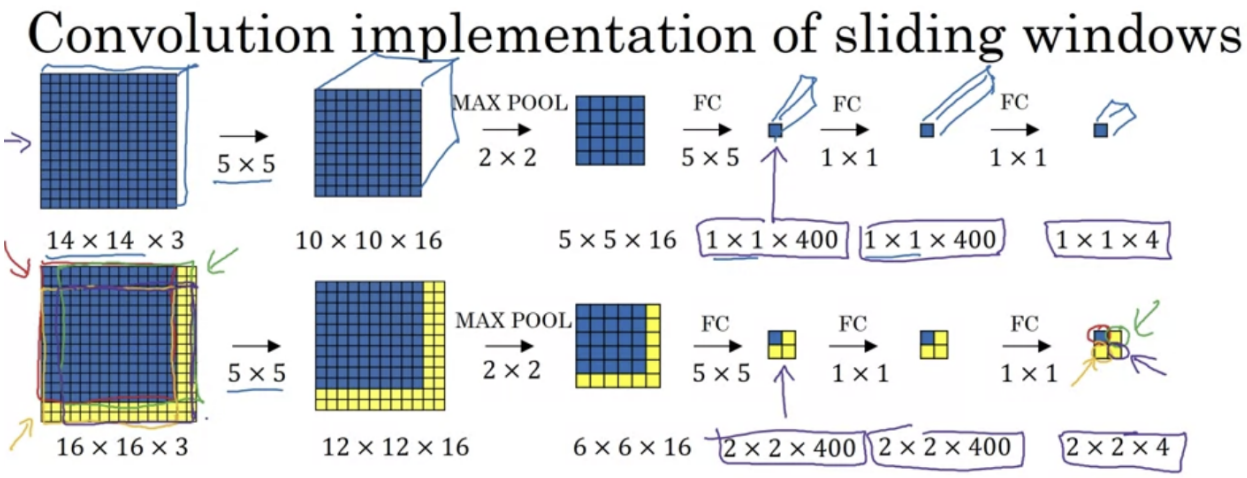
The sliding window's object detector can be implemented convolutionally which is much more efficient.

Convolutional Implementation of Sliding Windows

We can replace a FC with a Conv layer by using a large number of filters to build a 1x1xn_C volume.



A convolutional implementation allows sliding window steps to share computation.



The output volume has channels that give the same information as implementing the full sliding window process.

The max pooling dimension dictates the corresponding stride of the equivalent sliding window.

Bounding Box Predictions

The convolutional implementation of sliding windows doesn't output very accurate bounding boxes.

The YOLO (You Only Look Once) algorithm outputs more accurate bounding boxes. The idea is to divide the image into boxes and apply the image localization to each grid cell.

You have an output for each grid cell as shown in the image below.

The object is assigned to the cell that contains its midpoint.

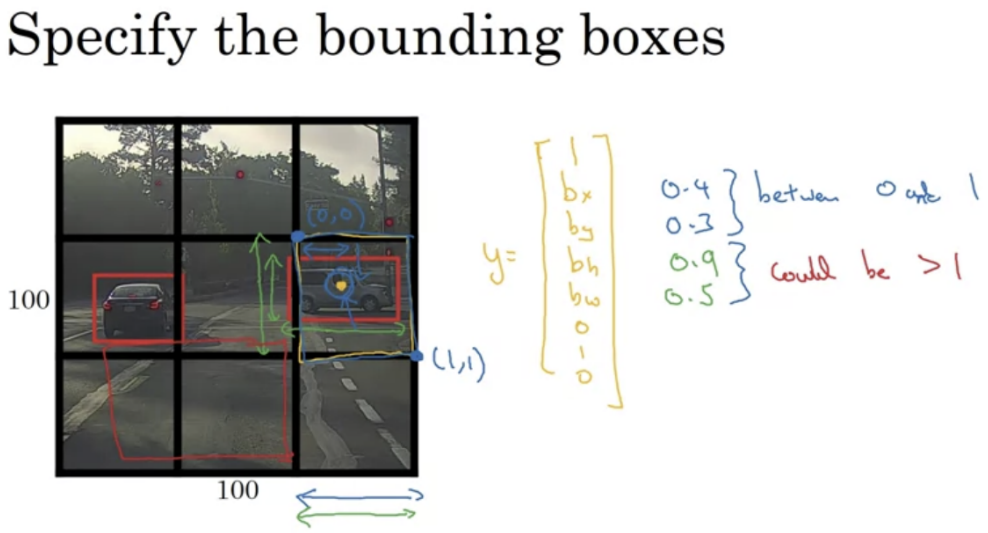
The outputs are concatenated into an output volume where each channel is an output vector for a specific grid cell.

This algorithm works well as long as there is only one object in each grid cell.

The coordinate system allows the algorithm to output a bounding box of non-square dimensions.

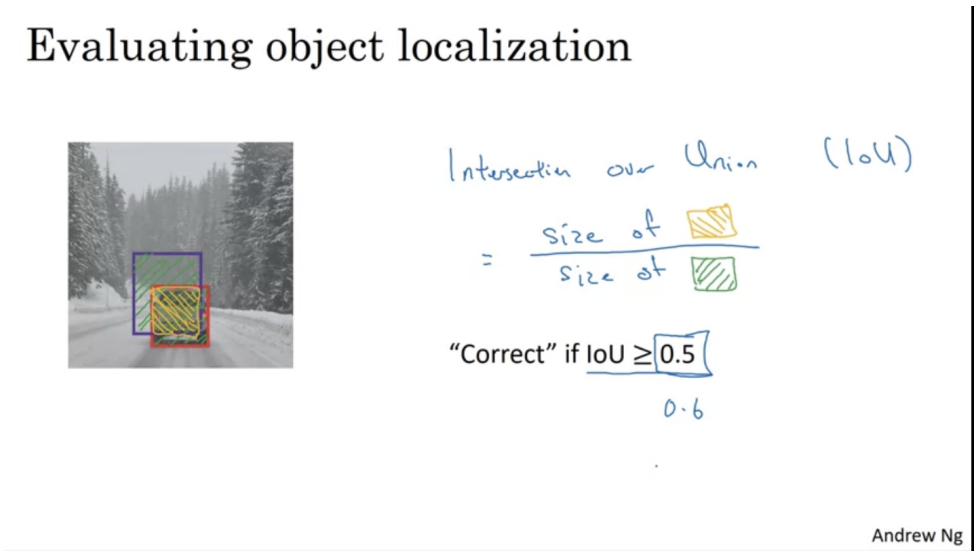
This algorithm runs very fast because it's a convolutional implementation.

The convention in the YOLO algorithm is to specify the upper left corner of the grid cell as (0,0). The midpoint coordinates are specified relative to this coordinate and the height and width are specified relative to the overall dimensions of the grid cell.



Intersection Over Union

The IoU function computes the intersection over union of the best bounding best and the bounding box outputted by the model. This is the ratio of the size of the union to the size of the remaining area in the output box. This ratio would be one if it was perfect.



Non-max Suppression

This is a way to make sure your algorithm detects each object only once.

It first looks at the probabilities associated with each of the detections. It takes the largest one and then looks at the bounding boxes with high IoUs. These will be suppressed because they mostly overlap.

Non-max suppression algorithm

19× 19

Each output prediction is: p_c, b_x, b_y, b_h, b_w

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 IoU ≥ 0.5 with the box output in the previous step

Andrew Ng

If you have more than one object, you should carry out non-max suppression on each of them.

Anchor Boxes

Anchor box algorithm

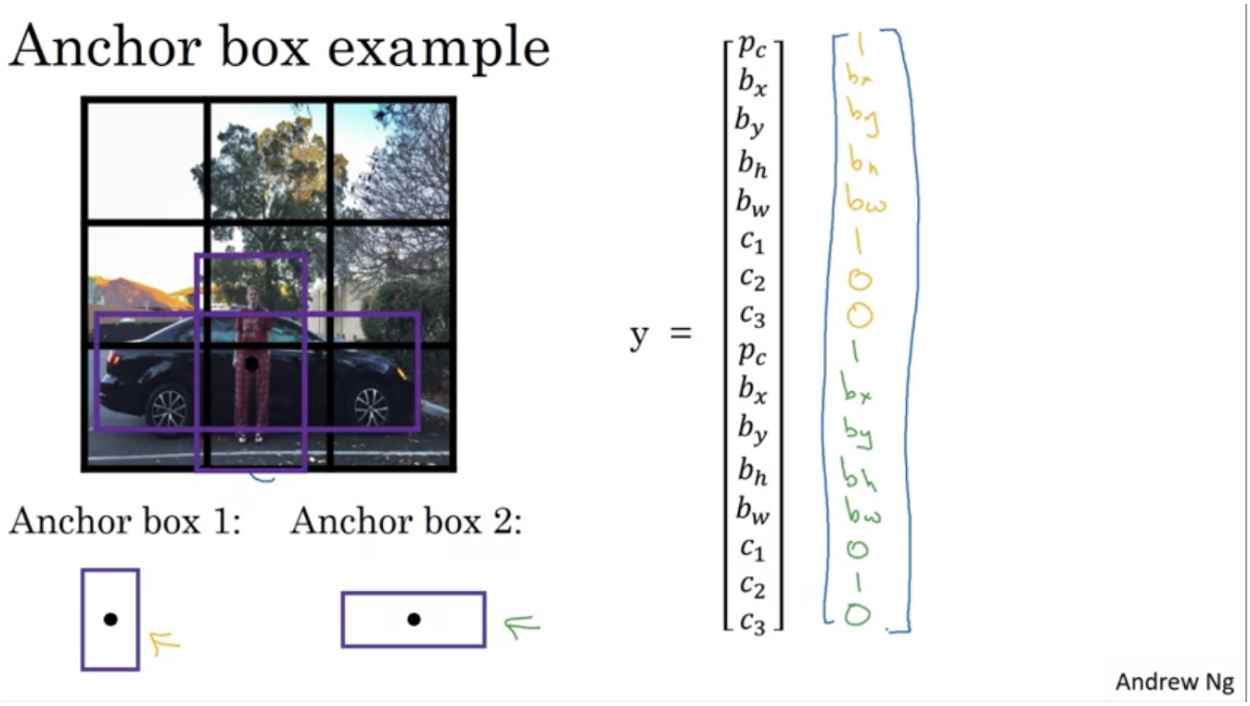
Previously:

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

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

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.

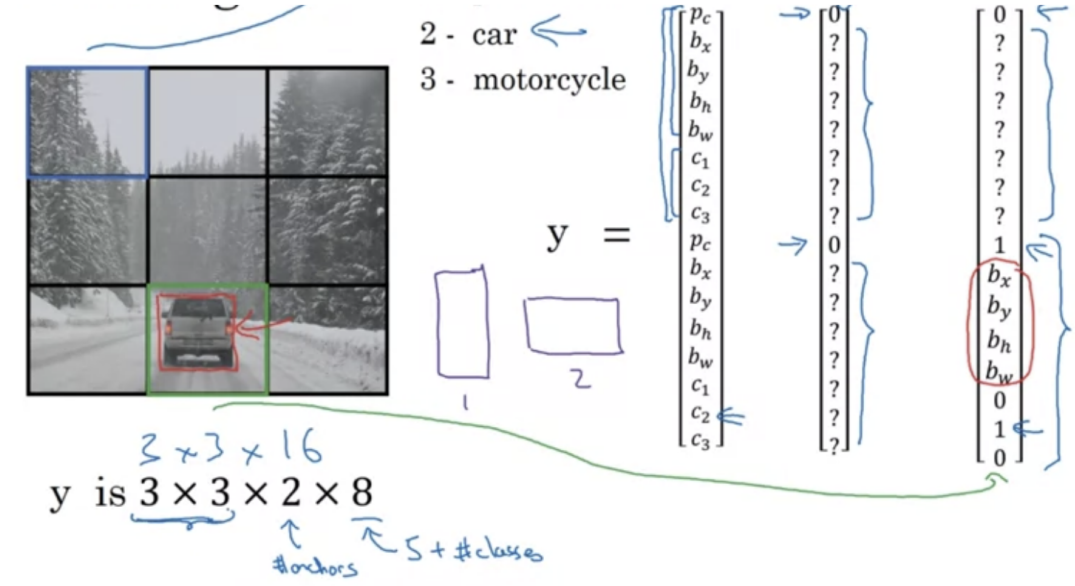


This algorithm doesn't handle the case where two objects are in the same grid cell and have a similar anchor box.

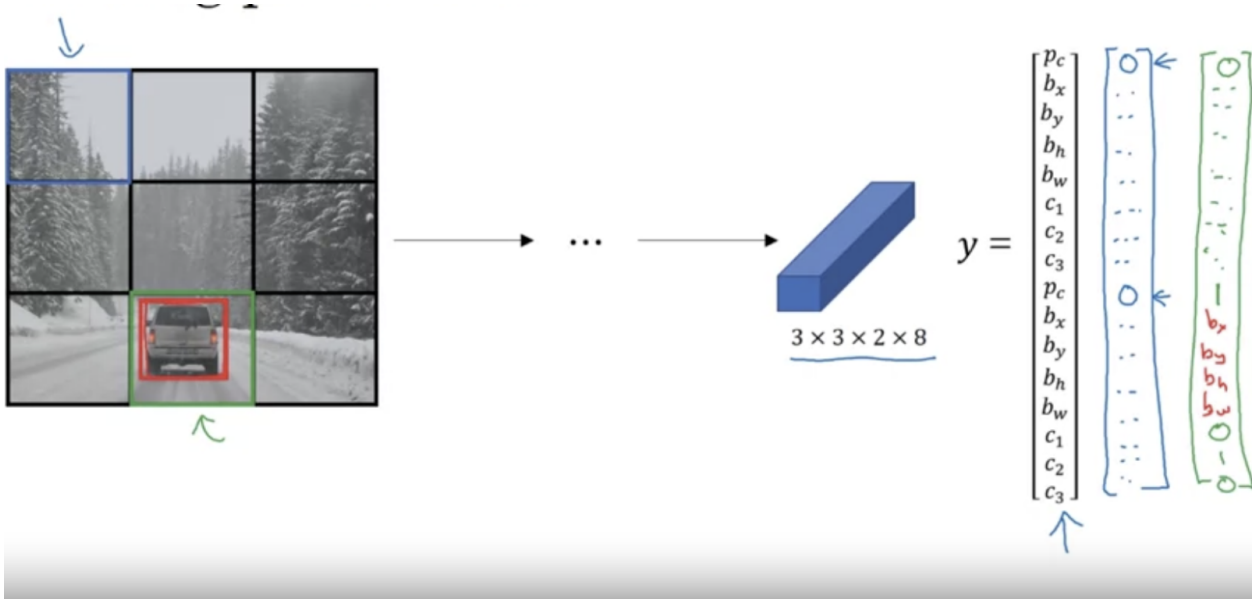
This is a method to deal with the case when two objects appear in the same grid cell (which doesn't happen often).

It also allows your algorithm to specialize. For example, in detecting tall skinny things and wide things. People usually choose the anchor boxes by hand.

YOLO Algorithm



For each grid cell you attempt to find an object and see which bounding box fits it best. You then output parameters for each bounding box (which is why y is double the size you may expect).



Outputting the non-max supressed outputs

- For each grid call, 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.

Region Proposals

The sliding window algorithm tends to look at a lot of grid cells that have nothing interesting in them.

The R-CNN algorithm tries to pick a few regions where it makes more sense to run the sliding window classifier.

They run a segmentation algorithm to find "blobs" that might be interesting. Bounding boxes are then placed around the interesting blobs and the ConvNet is run on these boxes.

Faster algorithms

R-CNN:	Propose regions. Classify proposed regions one at a time. Output label + bounding box.
Fast R-CNN:	Propose regions. Use convolution implementation of sliding windows to classify all the proposed regions.
Faster R-CNN:	Use convolutional network to propose regions.

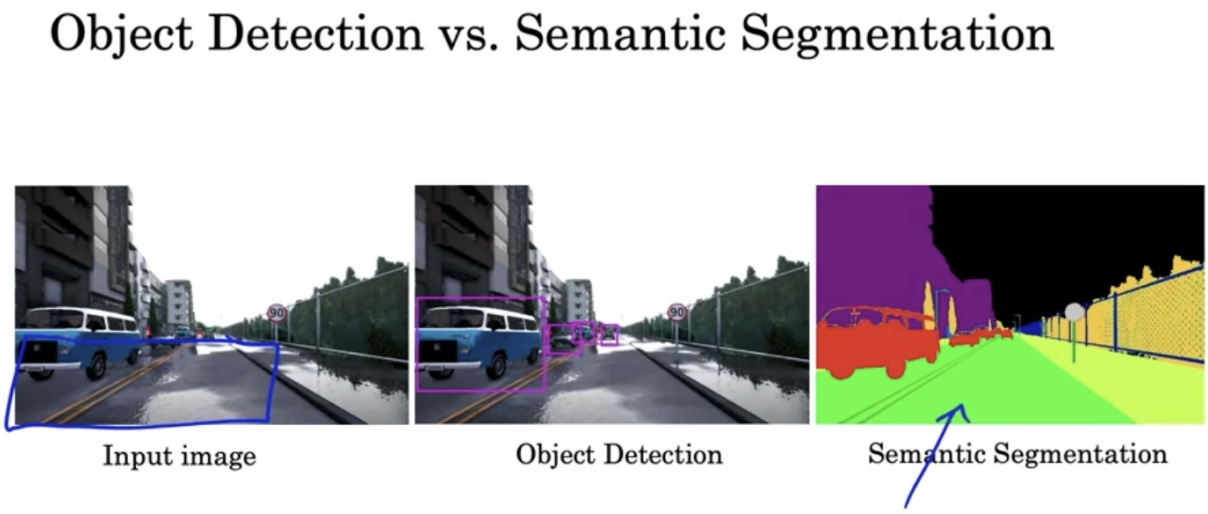
Girshik et. al, 2013. Rich feature hierarchies for accurate object detection and semantic segmentation]
Girshik, 2015. Fast R-CNN]
Ren et. al, 2016. Faster R-CNN: Towards real-time object detection with region proposal networks]

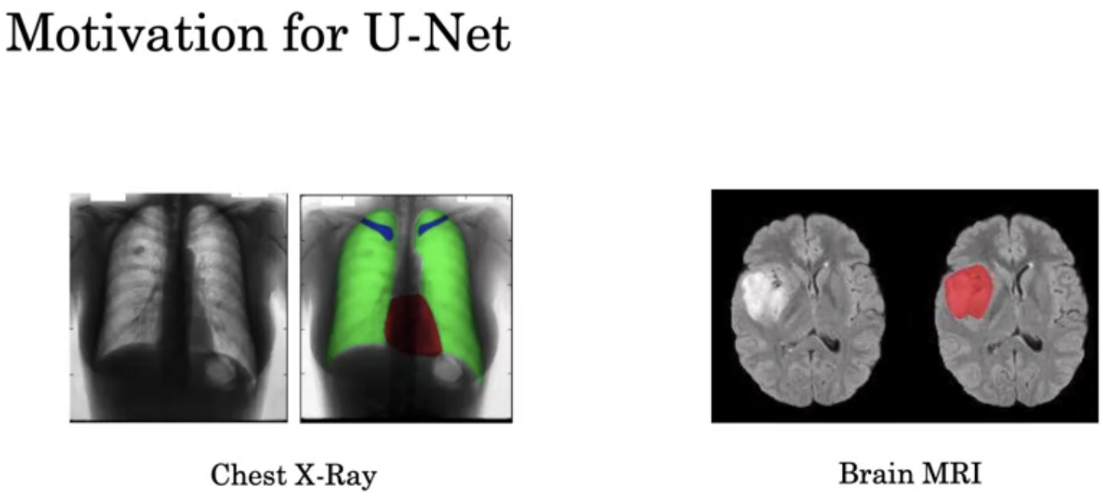
Andrew Ng

Andrew Ng thinks region proposal is a cool idea but the two-step process doesn't seem promising.

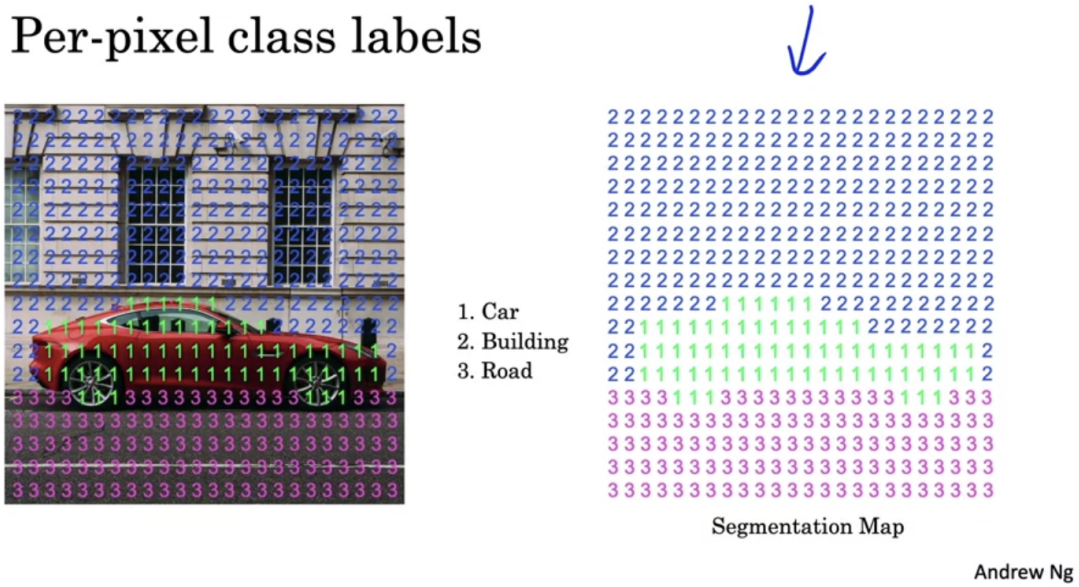
Semantic Segmentation with U-Net

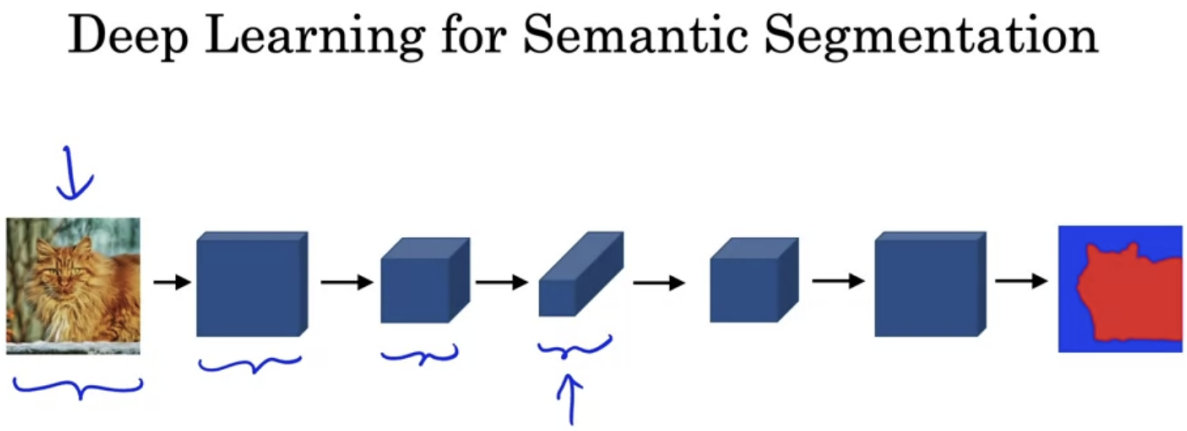
The goal of semantic segmentation is to draw a careful outline on the detected object.



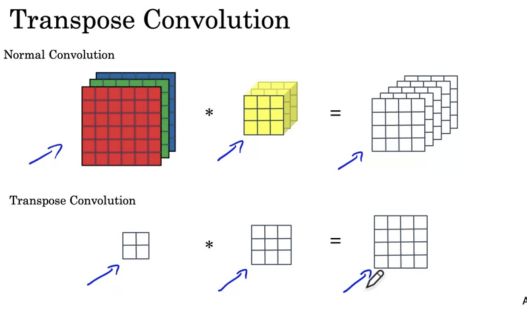


The job of the U-Net algorithm is to output a value for every pixel depending on its class. You may have two or more classes.



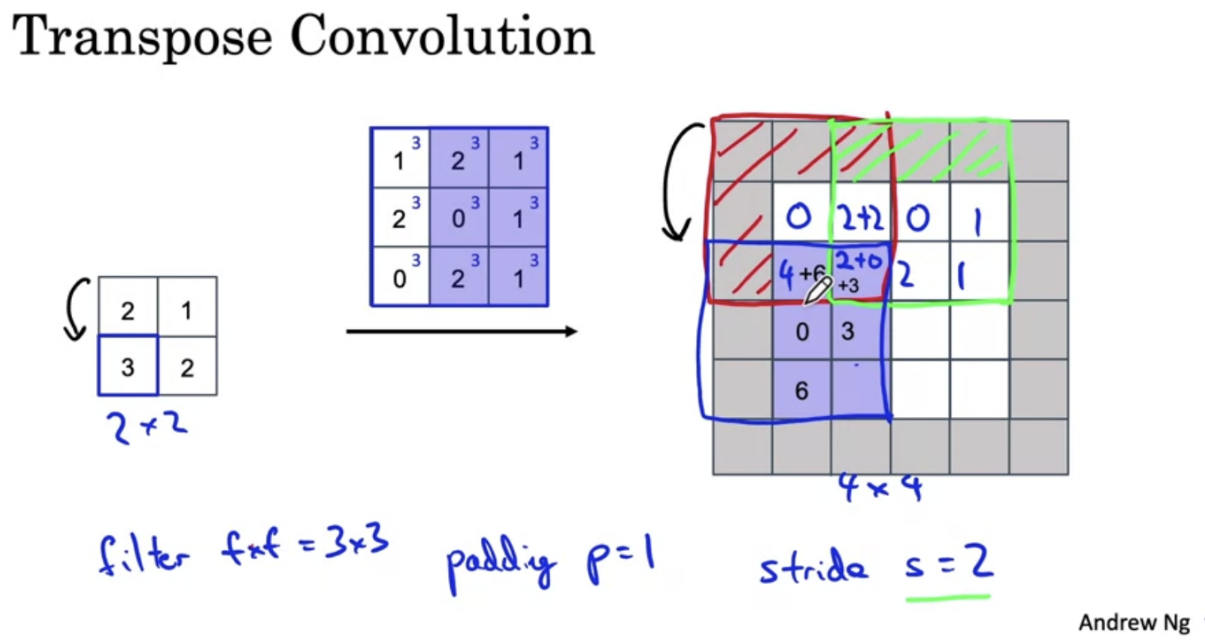


Transpose Convolutions



A transpose convolution increases the size of the matrix in contrast to the normal convolution.

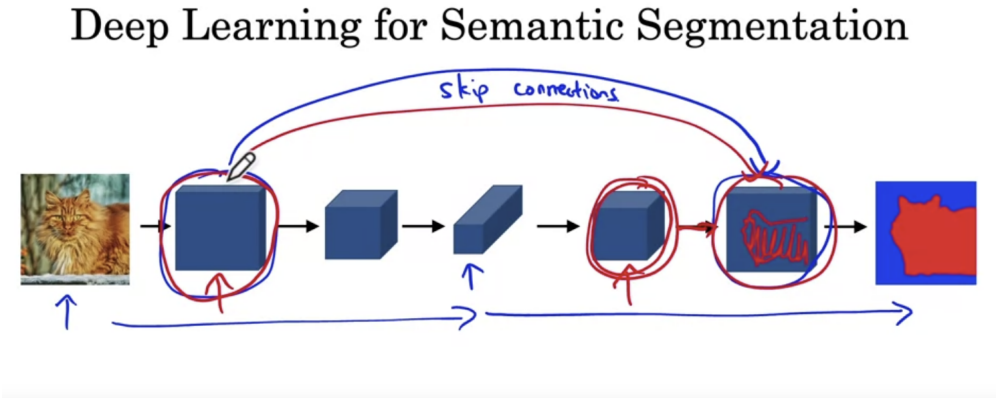
You place the filter on the output rather than the input. You multiply the input by everything in the filter to get a matrix which you paste in the output matrix. Padding removes values from the output matrix (they aren't added outside the output).



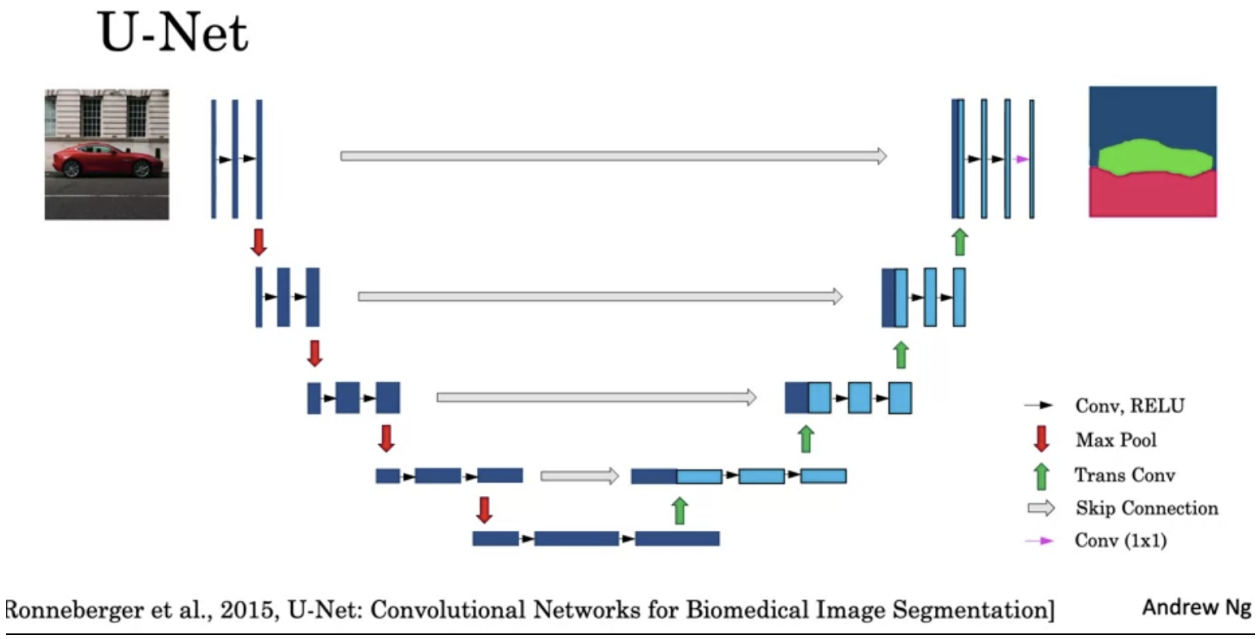
You add the elements in the overlapping sections.

U-Net Architecture Intuition

The skip connection in U-Net allows the model to take the high resolution low level information and pass it to the later layer. The last layer then has the high level and low-level features.



U-Net Architecture



The skip connection always runs by appending the set of activations to the connected layer deeper in the network.