Intro to Object Detection

Sources: A. Ng, F. Lee, J. Johnson, YOLO V3

We will move from...

Using CNNs to predict…

... one result from a single image (image classification) to..

… prediction of one result within a single image (image classification with localization)

...and ultimately we will learn to detect and predict multiple results within a single image (object detection)

Image Classification: A core task in Computer Vision

(assume given set of discrete labels)
{dog, cat, truck, plane, ...}

cat

Classification + Localization: Task

Classification: C classes
Input: Image
Output: Class label
Evaluation metric: Accuracy

CAT

Localization:
Input: Image
Output: Box in the image (x, y, w, h)
Evaluation metric: Intersection over Union

(x, y, w, h)

Classification + Localization: Do both

Dependent Vars Change

Classification example with five categories:

- Train CNN model on one hot encoded data with five categories
- Model predicts probability for all cats in output:

cat	dog	frog	rabbit	horse
.01	.20	.55	.25	.19

Dependent Vars Change

Classification example with five categories:

- Train CNN model on one hot encoded data with five categories
- Model predicts probability for all cats in output:

cat	dog	frog	rabbit	horse
.01	.20	.55	.25	.19

- Use maximum value to predict category (i.e.-Frog)
- Localization adds four continuous data points to DV and consequently to predicted output

Dependent Vars Change

Classification with localization will generate:

- Predictions for five categories plus..
- Four numeric values indicating local rectangle surrounding local category.
- The local rectangle is called a bounding box.
    - x (center of bounding box on x axis)
    - y (center of bounding box on y axis)
    - h (height)
    - w (width)

So training y will include the actual category of an observation AND its bounding box data.

Model architecture adjustment

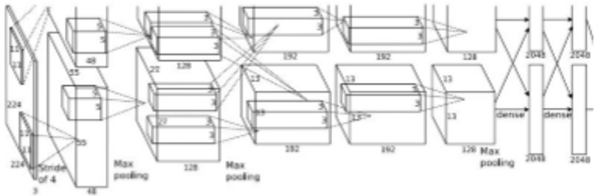
Now we are essentially training output to for classification AND regression tasks simultaneously!

- Need to adjust our loss calculations:
  - Simply need to calculate loss for softmax prediction (e.g.-log loss) AND regression prediction (e.g.-RSS)
  - So new loss value is Categorical loss + Regression loss
  - Goal: Minimize this value!

Model architecture (Image net example 1000 categs)

Classification + Localization

This image is CC0 public domain



Fully Connected: 4096 to 1000

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Correct label: Cat

Softmax Loss

Multitask Loss

+ → **Loss**

Vector: 4096

Fully Connected: 4096 to 4

Box Coordinates (x, y, w, h)

L2 Loss

Correct box: (x', y', w', h')

Treat localization as a regression problem!

Could use same approach to generate other localized predictions:

- Called “Landmark Detection” when we train model on multiple x,y locations in an image rather than training model to find bounding boxes

- Training images use predefined locations to train model such as:
  - Locations defining eyes,
  - Or different facial expressions
  - Or different bodily poses

Object Detection step by step: Sliding Windows

- To find multiple objects within a single image we could:
    i. Train a CNN model to predict categories from images that are closely cropped around the objects we wish to classify
        - Cropped image of t-rex head:

Object Detection step by step:
Sliding Windows Detection

- To find multiple objects within a single image we could:
    i. Train a CNN model to predict categories from images that are closely cropped around the objects we wish to classify

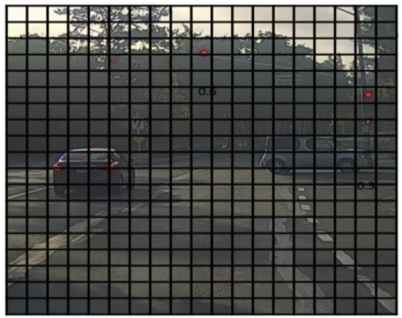
    i. Using this classification model we could:
        - Pick a window size
        - Slide a window over all parts of a larger image and make predictions each time with slide the window one step.
        - Generate predictions for each window.
        - Then repeat with different size windows

Object Detection step by step: Sliding Windows Detection

. . . .slide across every position...

Repeat process with different size windows to capture different size objects

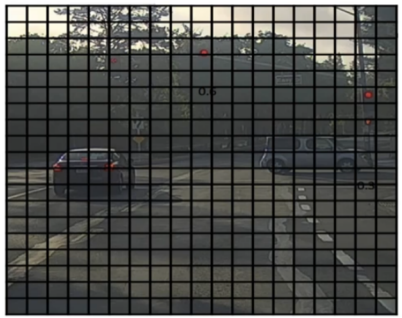
Object Detection step by step: Sliding Windows Detection



Practically speaking:

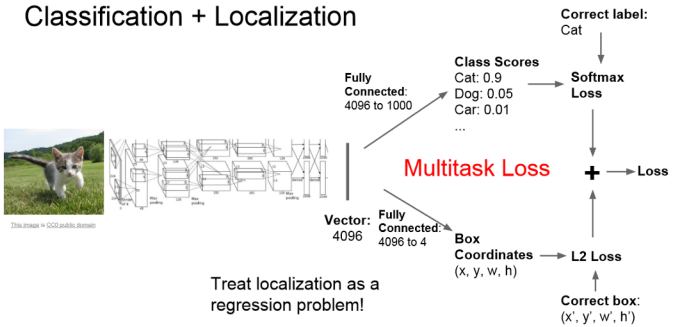
- We use a grid overlay where one cell equals an area we will utilize to fit parameters we can use to predict future bounding boxes.
- Our model will ultimately predict bounding boxes for objects with center points located in each cell.

Object Detection step by step: Sliding Windows Detection



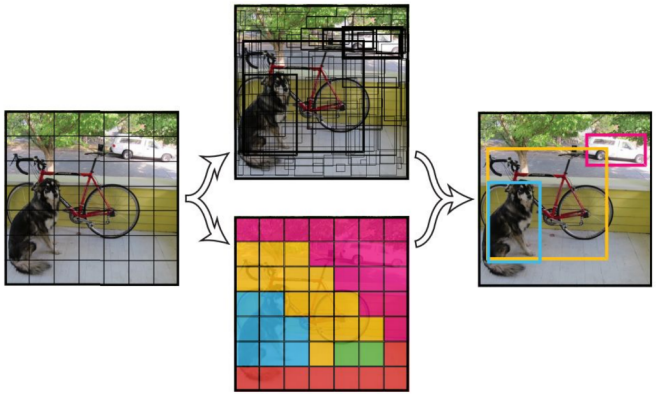
Practically speaking:

- So we can use a model like this:



- ...applied to the image within each cell…
- To return prediction with:
  - Predicted probs of categorical class AND
  - Numeric predicted values of bounding box x, y, h, and w

Object Detection step by step:
Prediction process still not complete!



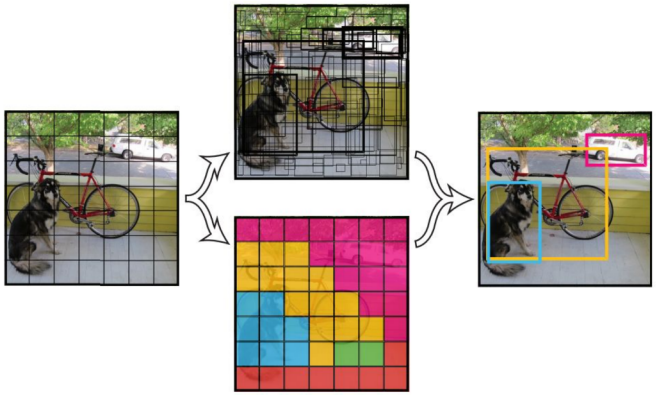
In practice we do not return all predicted bounding boxes!

- Utilize two approaches to calculate final predictions of objects
  - **Non Max Suppression**
    (used to get rid of unsuccessful object predictions)

    And

  - **Intersection Over Union (IOU)**
    (Used to select best bounding box prediction when same object is predicted in multiple cells)

Object Detection step by step:
Prediction process still not complete!



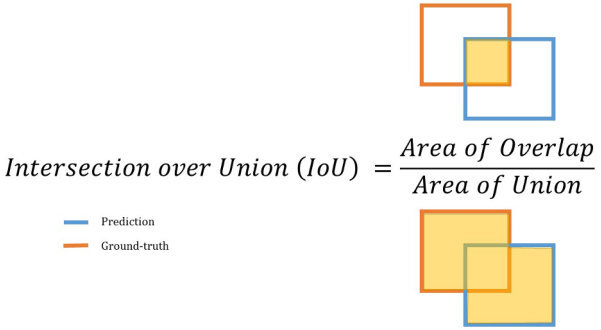
In practice we do not return all predicted bounding boxes!

- Utilize two approaches to calculate final predictions of objects
  - **Non Max Suppression**
    (used to get rid of unsuccessful object predictions)

    Delete all predicted bounding boxes / classes that have class maximum predicted probability less than some value (often > .6, but can be higher)

    - Goal: delete predictions for cells with no objects!

Object Detection step by step:
Prediction process still not complete!



In practice we do not return all predicted bounding boxes!

- Utilize two approaches to calculate final predictions of objects
  - **Intersection over union defined:**
    (Used to select best bounding box prediction when same object is predicted in multiple cells)
    - <u>What is IOU?</u>

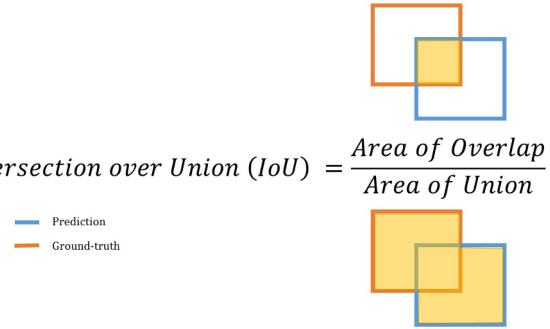
      Calculation used to compare two bounding boxes

      We use it as part of process to choose best bounding box to predict local object.

      How?

Object Detection step by step:
Prediction process still not complete!

$$Intersection\ over\ Union\ (IoU) = \frac{Area\ of\ Overlap}{Area\ of\ Union}$$



In practice we do not return all predicted bounding boxes!

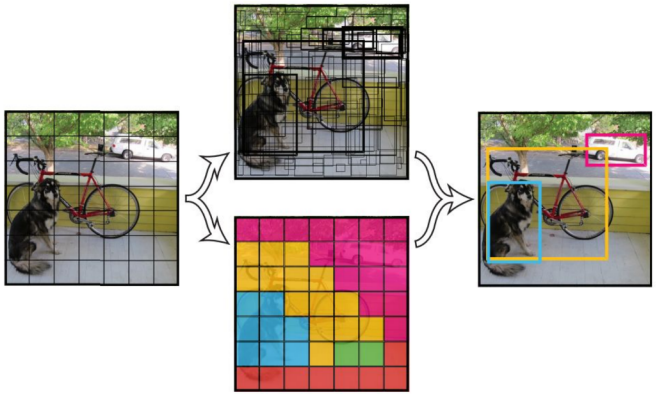
- Utilize two approaches to calculate final predictions of objects
  - **Intersection over union defined:**
    (Used to select best bounding box prediction when same object is predicted in multiple cells)
    - <u>What is IOU?</u>

      Calculation used to compare two bounding boxes

      We use it as part of process to delete overlapping bounding boxes.

      How?

Object Detection step by step:
Prediction process still not complete!



In practice we do not return all predicted bounding boxes!

- Utilize two approaches to calculate final predictions of objects

Using IOU to delete overlapping boxes

1. **For a each category we prune out overlapping boxes by:**
    a. Finding cell with highest predicted probability for category
    b. Calculating IOU between cell's predicted bounding box all overlapping bounding boxes.
    c. Deleting bounding boxes that overlap with best predicted box at or above IOU=.5
        i. .5 is common but can be adjusted.

Object Detection step by step:

Problem=Sliding Windows Detection Model Training Inefficient!

. . . .slide across every position...

- Building model parameters using MANY subimages.
  - Multiple grids overlaid over each image
- In practice we circumvent this computationally inefficient (and quite slow) process.

Solution:
We utilize a convolutional architecture that *can use shared parameters to predict categories and bounding boxes.*

Object Detection step by step:
Sliding Windows Detection Models Inefficient!

. . . .slide across
every position...

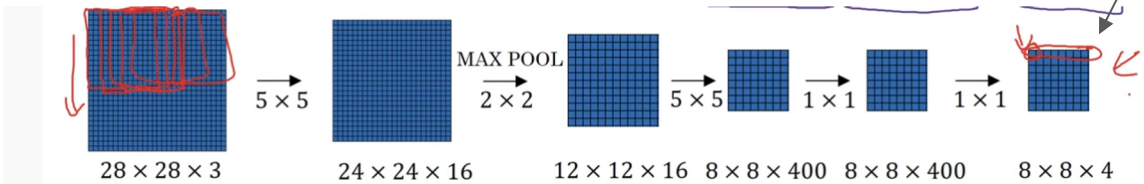
Convnet layer usage (rather than dense layers):

- uses shared parameters
- outputs all prediction bounding boxes
  For each cell simultaneously!

Much more efficient to train models!

Object Detection step by step: Convolution layers that output to grid cell def'n

Input original image:



Output predictions that utilized shared parameters to make predicted output to selected grid

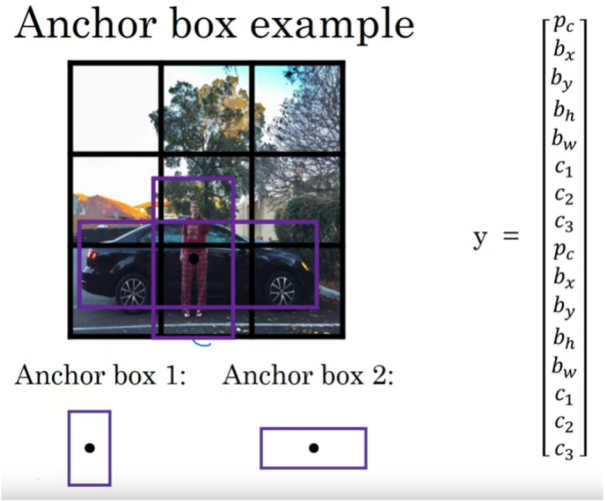
To recap:

It's the same result (categorical predictions + bounding box predictions at each grid cell),
But now the model is more computationally efficient

More details on this approach?
Read Overfeat paper here:https://arxiv.org/abs/1312.6229

Detecting multiple objects per cell (thus far only one) -Anchor Box Idea!

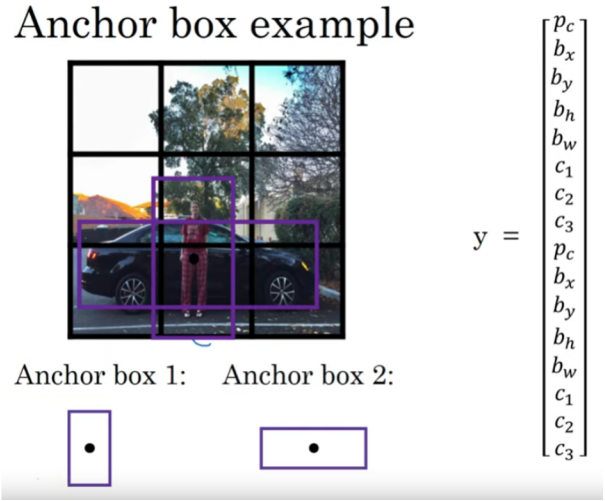
- Predefine rectangle shapes to fit bounding box (x,y, h , and w) to.
  - Example with two predefined shapes



- Fit two bounding boxes restricted to each anchor box shape per cell
- Use IOU with ground truth box to attach predicted category to anchor box with highest IOU.

Detecting multiple objects per cell (thus far only one) -Anchor Box Idea!

- Predefine rectangle shapes to fit bounding box (x,y, h , and w) to.
  - Example with two predefined shapes



- Fit two bounding boxes restricted to each anchor box shape per cell
- Use IOU with ground truth box to attach predicted category to anchor box with highest IOU.
- Now we can detect two categories with predictions from same grid cell
- Often use five or more predefined rectangles to maximize object detection effectiveness

Next: Implement most used object detection model in Keras

YOLO V3