Object Recognition

Tushar B. Kute,
http://tusharkute.com

Object Recognition

- Object recognition in image processing refers to the task of identifying and classifying objects within digital images or video frames.
- It involves the use of computer vision techniques and machine learning algorithms to automatically detect and recognize objects of interest in visual data.

Object Recognition

- The goal of object recognition is to assign meaningful labels or categories to objects in an image.
- This can involve identifying common objects such as cars, people, animals, or specific object instances such as a particular breed of dog or a specific model of car.

Object Recognition

- Object recognition typically involves several steps, including:
  - Preprocessing: This step involves preparing the image data for analysis by performing operations such as resizing, normalizing, or enhancing the image.
  - Feature Extraction: Features that represent discriminative characteristics of objects are extracted from the preprocessed image. These features can be low-level visual descriptors like edges, corners, or textures, or higher-level features learned from deep neural networks.

Object Recognition

- Classification or Detection: The extracted features are used to classify the objects or detect their presence in the image. Classification involves assigning a specific label or category to an entire image, while detection involves localizing and classifying objects within an image.
- Post-processing: After classification or detection, post-processing techniques are applied to refine the results, such as filtering out false positives, grouping detections, or estimating object poses and attributes.

Image Processing

- As such, we can distinguish between these three computer vision tasks:
  - Image Classification: Predict the type or class of an object in an image.
    - Input: An image with a single object, such as a photograph.
    - Output: A class label (e.g. one or more integers that are mapped to class labels).

Image Processing

- Object Localization: Locate the presence of objects in an image and indicate their location with a bounding box.
  - Input: An image with one or more objects, such as a photograph.
  - Output: One or more bounding boxes (e.g. defined by a point, width, and height).

Image Processing

- Object Detection: Locate the presence of objects with a bounding box and types or classes of the located objects in an image.
  - Input: An image with one or more objects, such as a photograph.
  - Output: One or more bounding boxes (e.g. defined by a point, width, and height), and a class label for each bounding box.

Image Processing

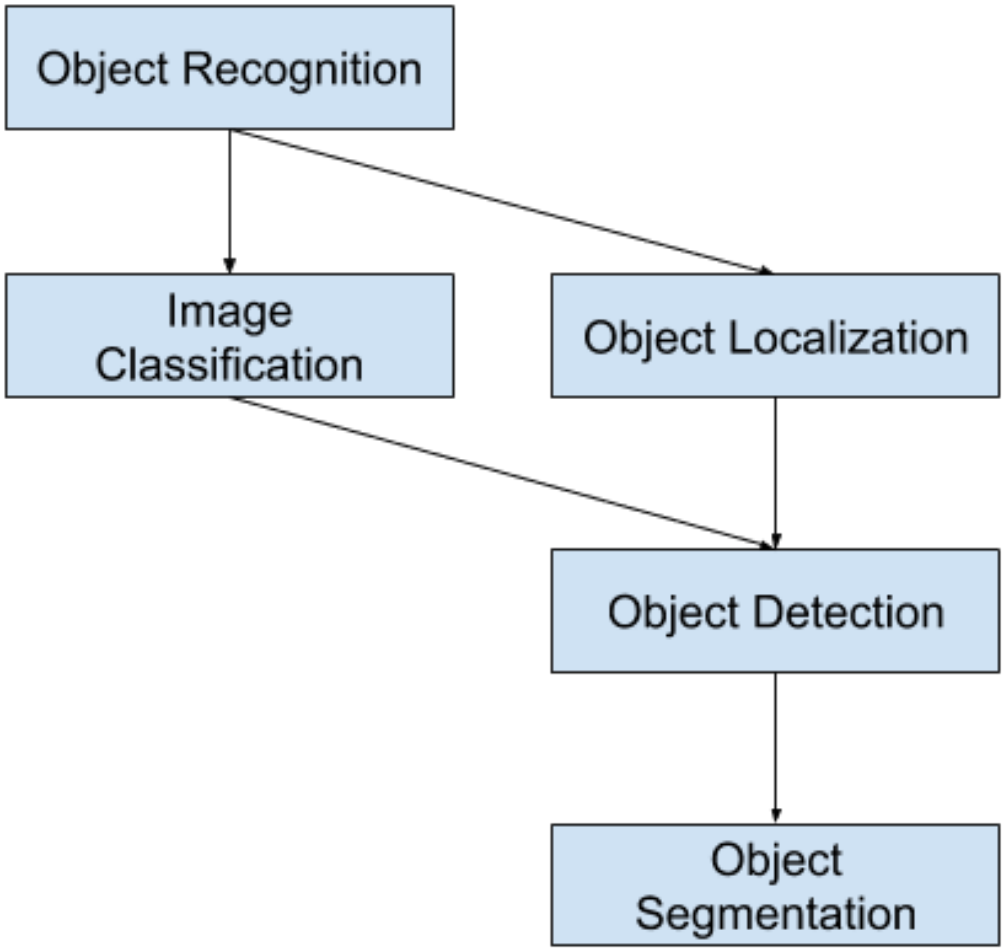
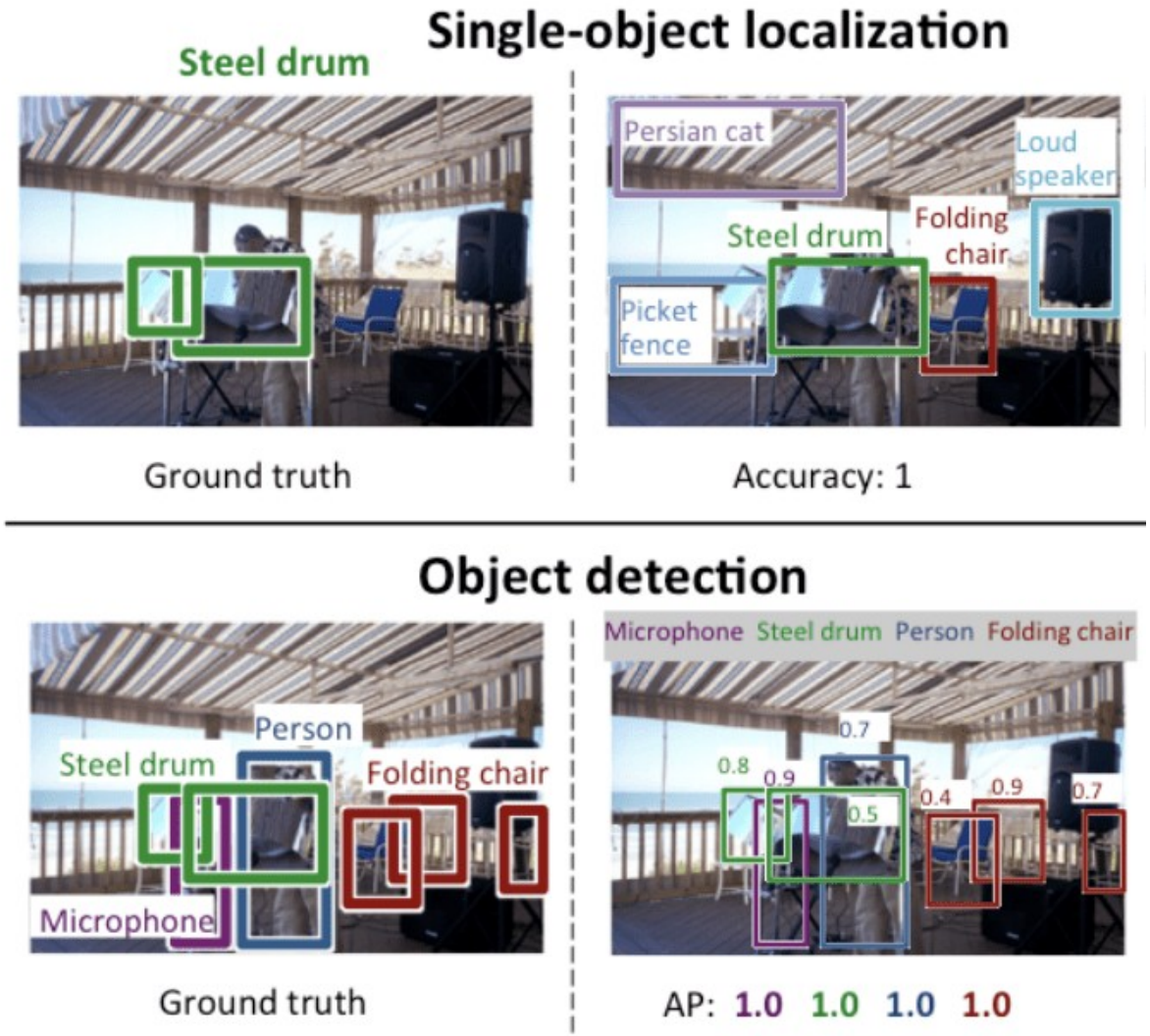


Image Processing



The RCNN Model Family

- The R-CNN family of methods refers to the R-CNN, which may stand for “Regions with CNN Features” or “Region-Based Convolutional Neural Network,” developed by Ross Girshick, et al.
- This includes the techniques R-CNN, Fast R-CNN, and Faster-RCNN designed and demonstrated for object localization and object recognition.

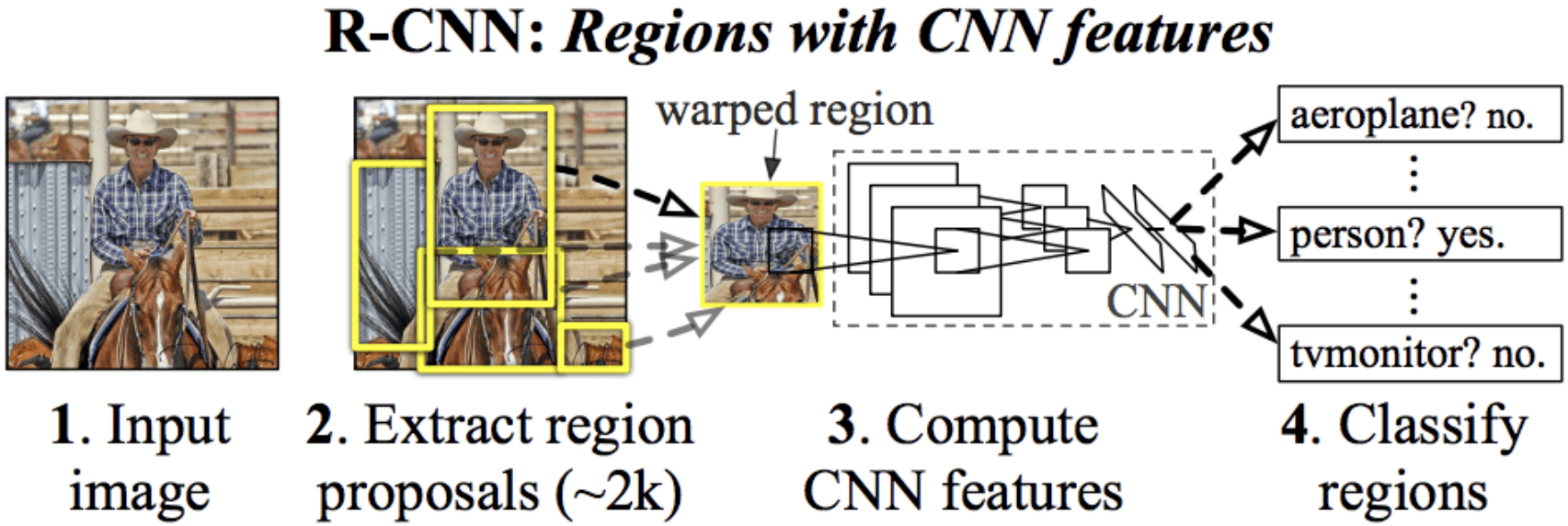
The RCNN

- The R-CNN was described in the 2014 paper by Ross Girshick, et al. from UC Berkeley titled "*Rich feature hierarchies for accurate object detection and semantic segmentation*."
- It may have been one of the first large and successful application of convolutional neural networks to the problem of object localization, detection, and segmentation.
- The approach was demonstrated on benchmark datasets, achieving then state-of-the-art results on the VOC-2012 dataset and the 200-class ILSVRC-2013 object detection dataset.

The RCNN

- Proposed R-CNN model is comprised of three modules; they are:
  - Module 1: Region Proposal. Generate and extract category independent region proposals, e.g. candidate bounding boxes.
  - Module 2: Feature Extractor. Extract feature from each candidate region, e.g. using a deep convolutional neural network.
  - Module 3: Classifier. Classify features as one of the known class, e.g. linear SVM classifier model.

The RCNN


R-CNN: Regions with CNN features
warped region
CNN
aeroplane? no.
person? yes.
tvmonitor? no.
1. Input image
2. Extract region proposals (~2k)
3. Compute CNN features
4. Classify regions

RCNN

- A computer vision technique is used to propose candidate regions or bounding boxes of potential objects in the image called “selective search,” although the flexibility of the design allows other region proposal algorithms to be used.
- The feature extractor used by the model was the AlexNet deep CNN that won the ILSVRC-2012 image classification competition.
- The output of the CNN was a 4,096 element vector that describes the contents of the image that is fed to a linear SVM for classification, specifically one SVM is trained for each known class.

Fast-RCNN

- Given the great success of R-CNN, Ross Girshick, then at Microsoft Research, proposed an extension to address the speed issues of R-CNN in a 2015 paper titled “Fast R-CNN.”
- The paper opens with a review of the limitations of R-CNN, which can be summarized as follows:
  - Training is a multi-stage pipeline. Involves the preparation and operation of three separate models.
  - Training is expensive in space and time. Training a deep CNN on so many region proposals per image is very slow.
  - Object detection is slow. Make predictions using a deep CNN on so many region proposals is very slow.

Fast-RCNN

- A prior work was proposed to speed up the technique called spatial pyramid pooling networks, or SPPnets, in the 2014 paper "*Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition*."
- This did speed up the extraction of features, but essentially used a type of forward pass caching algorithm.
- Fast R-CNN is proposed as a single model instead of a pipeline to learn and output regions and classifications directly.

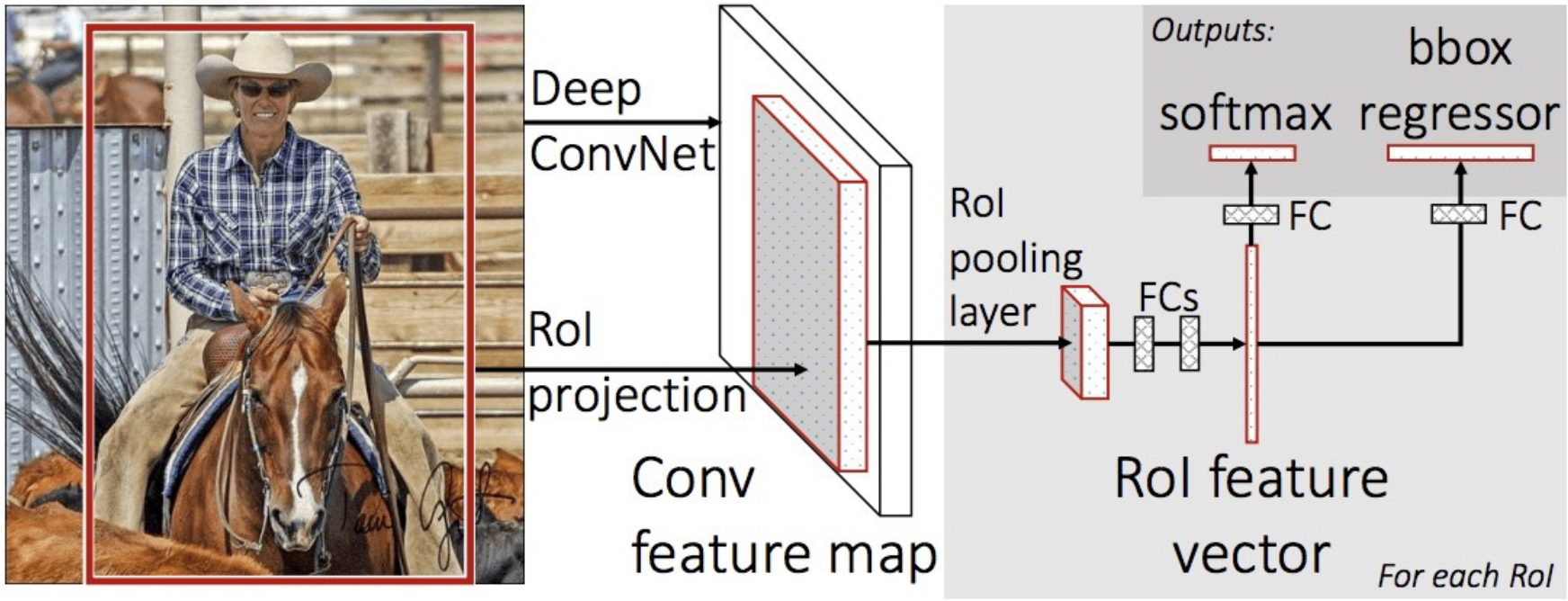
Fast-RCNN

- The architecture of the model takes the photograph a set of region proposals as input that are passed through a deep convolutional neural network.
- A pre-trained CNN, such as a VGG-16, is used for feature extraction.
- The end of the deep CNN is a custom layer called a Region of Interest Pooling Layer, or RoI Pooling, that extracts features specific for a given input candidate region.

Fast-RCNN

- The output of the CNN is then interpreted by a fully connected layer then the model bifurcates into two outputs, one for the class prediction via a softmax layer, and another with a linear output for the bounding box.
- This process is then repeated multiple times for each region of interest in a given image.

Fast-RCNN


Deep
ConvNet
RoI
projection
Conv
feature map
RoI
pooling
layer
FCs
Outputs:
softmax
bbox
regressor
FC
FC
RoI feature
vector
For each RoI

Faster-RCNN

- The model architecture was further improved for both speed of training and detection by Shaoqing Ren, et al. at Microsoft Research in the 2016 paper titled “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.”
- The architecture was the basis for the first-place results achieved on both the ILSVRC-2015 and MS COCO-2015 object recognition and detection competition tasks.

Faster-RCNN

- The architecture was designed to both propose and refine region proposals as part of the training process, referred to as a Region Proposal Network, or RPN.
- These regions are then used in concert with a Fast R-CNN model in a single model design.
- These improvements both reduce the number of region proposals and accelerate the test-time operation of the model to near real-time with then state-of-the-art performance.

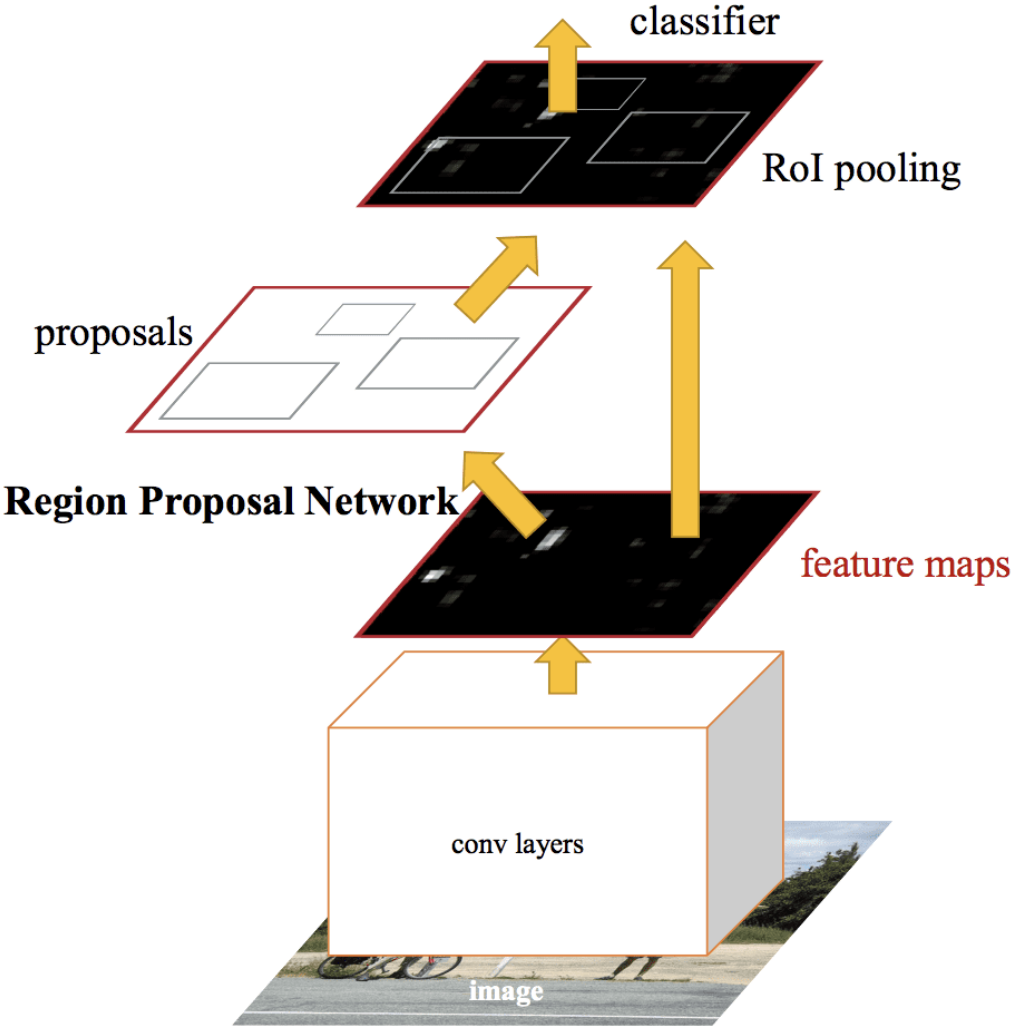
Faster-RCNN

- Although it is a single unified model, the architecture is comprised of two modules:
  - Module 1: Region Proposal Network. Convolutional neural network for proposing regions and the type of object to consider in the region.
  - Module 2: Fast R-CNN. Convolutional neural network for extracting features from the proposed regions and outputting the bounding box and class labels.

Faster-RCNN

- Both modules operate on the same output of a deep CNN.
- The region proposal network acts as an attention mechanism for the Fast R-CNN network, informing the second network of where to look or pay attention.

Faster-RCNN


classifier
RoI pooling
proposals
Region Proposal Network
feature maps
conv layers
image

Faster-RCNN

- The RPN works by taking the output of a pre-trained deep CNN, such as VGG-16, and passing a small network over the feature map and outputting multiple region proposals and a class prediction for each.
- Region proposals are bounding boxes, based on so-called anchor boxes or pre-defined shapes designed to accelerate and improve the proposal of regions.
- The class prediction is binary, indicating the presence of an object, or not, so-called “objectness” of the proposed region.

Faster-RCNN

- A procedure of alternating training is used where both sub-networks are trained at the same time, although interleaved.
- This allows the parameters in the feature detector deep CNN to be tailored or fine-tuned for both tasks at the same time.
- At the time of writing, this Faster R-CNN architecture is the pinnacle of the family of models and continues to achieve near state-of-the-art results on object recognition tasks.
- A further extension adds support for image segmentation, described in the paper 2017 paper "Mask R-CNN."

Image Segmentation

- This technique gives us a far more granular understanding of the object(s) in the image.
- The image shown below will help you to understand what image segmentation is:

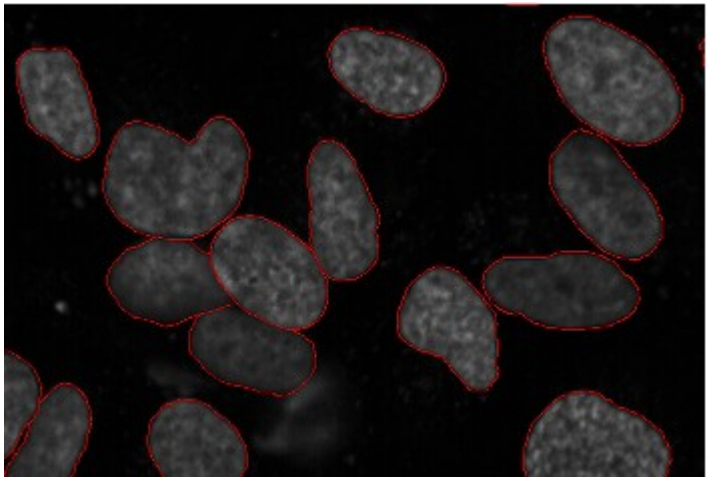
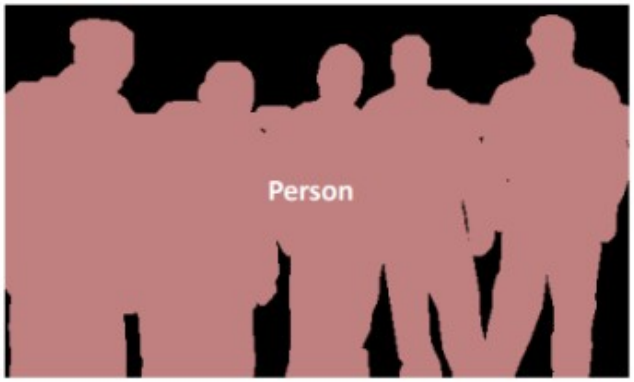
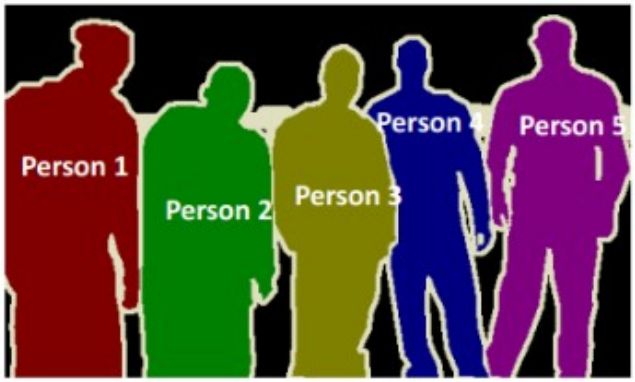


Image Segmentation

- All 5 objects in the left image are people. Hence, semantic segmentation will classify all the people as a single instance.
- Now, the image on the right also has 5 objects (all of them are people). But here, different objects of the same class have been assigned as different instances. This is an example of instance segmentation.



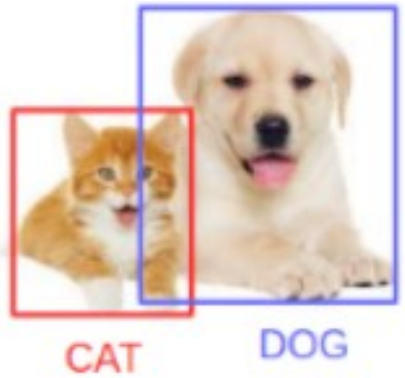
Semantic Segmentation



Instance Segmentation

Mask RCNN

- Mask R-CNN is basically an extension of Faster R-CNN. Faster R-CNN is widely used for object detection tasks.
- For a given image, it returns the class label and bounding box coordinates for each object in the image. So, let’s say you pass the following image:



Mask RCNN

- Let's first quickly understand how Faster R-CNN works. This will help us grasp the intuition behind Mask R-CNN as well.
  - Faster R-CNN first uses a ConvNet to extract feature maps from the images
  - These feature maps are then passed through a Region Proposal Network (RPN) which returns the candidate bounding boxes
  - We then apply an RoI pooling layer on these candidate bounding boxes to bring all the candidates to the same size
  - And finally, the proposals are passed to a fully connected layer to classify and output the bounding boxes for objects

Mask RCNN

- Backbone Model
  - Similar to the ConvNet that we use in Faster R-CNN to extract feature maps from the image, we use the ResNet 101 architecture to extract features from the images in Mask R-CNN.
  - So, the first step is to take an image and extract features using the ResNet 101 architecture. These features act as an input for the next layer.

Mask RCNN

- Region Proposal Network (RPN)
  - Now, we take the feature maps obtained in the previous step and apply a region proposal network (RPN).
  - This basically predicts if an object is present in that region (or not).
  - In this step, we get those regions or feature maps which the model predicts contain some object.

Mask RCNN: Region of Interest

- The regions obtained from the RPN might be of different shapes. Hence, we apply a pooling layer and convert all the regions to the same shape.
- Next, these regions are passed through a fully connected network so that the class label and bounding boxes are predicted.
- Till this point, the steps are almost similar to how Faster R-CNN works.
- Now comes the difference between the two frameworks. In addition to this, Mask R-CNN also generates the segmentation mask.

Mask RCNN: Region of Interest

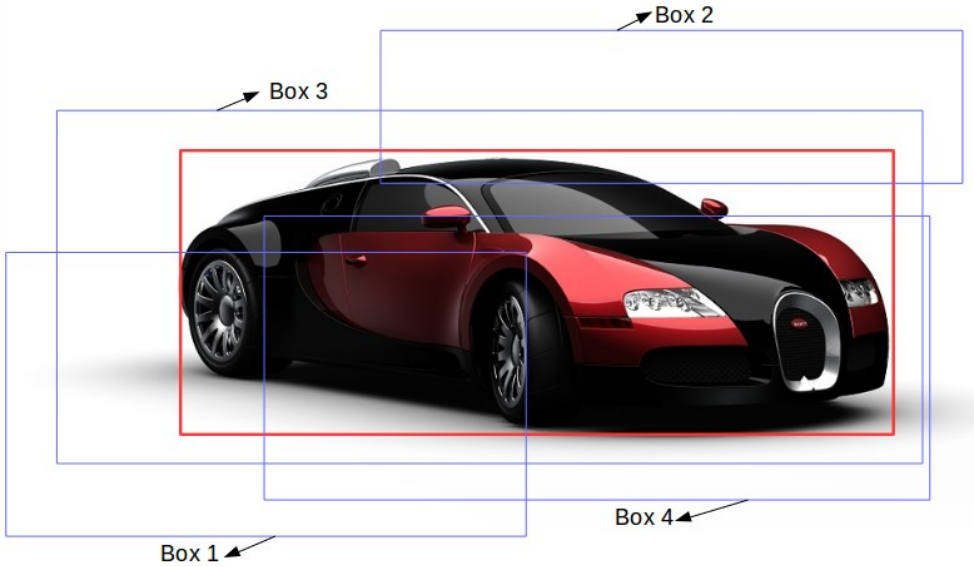
- For that, we first compute the region of interest so that the computation time can be reduced.
- For all the predicted regions, we compute the Intersection over Union (IoU) with the ground truth boxes. We can computer IoU like this:

  IoU = Area of the intersection / Area of the union

- Now, only if the IoU is greater than or equal to 0.5, we consider that as a region of interest. Otherwise, we neglect that particular region.
- We do this for all the regions and then select only a set of regions for which the IoU is greater than 0.5.

Mask RCNN: Region of Interest

Here, the IoU of Box 1 and Box 2 is possibly less than 0.5, whereas the IoU of Box 3 and Box 4 is approximately greater than 0.5. Hence. we can say that Box 3 and Box 4 are the region of interest for this particular image whereas Box 1 and Box 2 will be neglected.



Mask RCNN: Segmentation

- Once we have the RoIs based on the IoU values, we can add a mask branch to the existing architecture.
- This returns the segmentation mask for each region that contains an object.
- It returns a mask of size 28 X 28 for each region which is then scaled up for inference.

Mask RCNN: Segmentation

COCO Dataset

- COCO is a large-scale object detection, segmentation, and captioning dataset. COCO has several features:
  - Object segmentation
  - Recognition in context
  - Superpixel stuff segmentation
  - 330K images (>200K labeled)
  - 1.5 million object instances
  - 80 object categories
  - 91 stuff categories
  - 5 captions per image
  - 250,000 people with keypoints

Mask RCNN: Segmentation

- Here, our model has segmented all the objects in the image. This is the final step in Mask R-CNN where we predict the masks for all the objects in the image.
- Keep in mind that the training time for Mask R-CNN is quite high. It took me somewhere around 1 to 2 days to train the Mask R-CNN on the famous COCO dataset.
- We will instead use the pretrained weights of the Mask R-CNN model trained on the COCO dataset.
- Now, before we dive into the Python code, let's look at the steps to use the Mask R-CNN model to perform instance segmentation.

Mask RCNN: Implementation

- Step 1: Clone the repository
  - First, we will clone the mask rcnn repository which has the architecture for Mask R-CNN. Use the following command to clone the repository:

    git clone https://github.com/matterport/Mask_RCNN.git
- Once this is done, we need to install the dependencies required by Mask R-CNN.

Mask RCNN: Implementation

- Step 2: Install the dependencies
- Here is a list of all the dependencies for Mask R-CNN:

numpy	scipy
pillow	cython
matplotlib	scikit-image
tensorflow>=1.3.0	keras>=2.0.8
opencv-python	h5py
imgaug	IPython

Mask RCNN: Implementation

- Step 3: Download the pre-trained weights (trained on MS COCO)
- Next, we need to download the pretrained weights. You can use this link to download the pre-trained weights.
- These weights are obtained from a model that was trained on the MS COCO dataset.
- Once you have downloaded the weights, paste this file in the samples folder of the Mask_RCNN repository that we cloned in step 1.

Mask RCNN: Implementation

- Step 4: Predicting for our image
- Finally, we will use the Mask R-CNN architecture and the pretrained weights to generate predictions for our own images.
- Once you're done with these four steps, it's time to jump into your Jupyter Notebook!
- We will implement all these things in Python and then generate the masks along with the classes and bounding boxes for objects in our images.

YOLO Models

- Another popular family of object recognition models is referred to collectively as YOLO or "You Only Look Once," developed by *Joseph Redmon*, et al.
- The R-CNN models may be generally more accurate, yet the YOLO family of models are fast, much faster than R-CNN, achieving object detection in real-time.

YOLO Models

- The YOLO model was first described by Joseph Redmon, et al. in the 2015 paper titled "*You Only Look Once: Unified, Real-Time Object Detection*."
- Note that *Ross Girshick*, developer of R-CNN, was also an author and contributor to this work, then at *Facebook AI Research*.
- The approach involves a single neural network trained end to end that takes a photograph as input and predicts bounding boxes and class labels for each bounding box directly.
- The technique offers lower predictive accuracy (e.g. more localization errors), although operates at 45 frames per second and up to 155 frames per second for a speed-optimized version of the model.

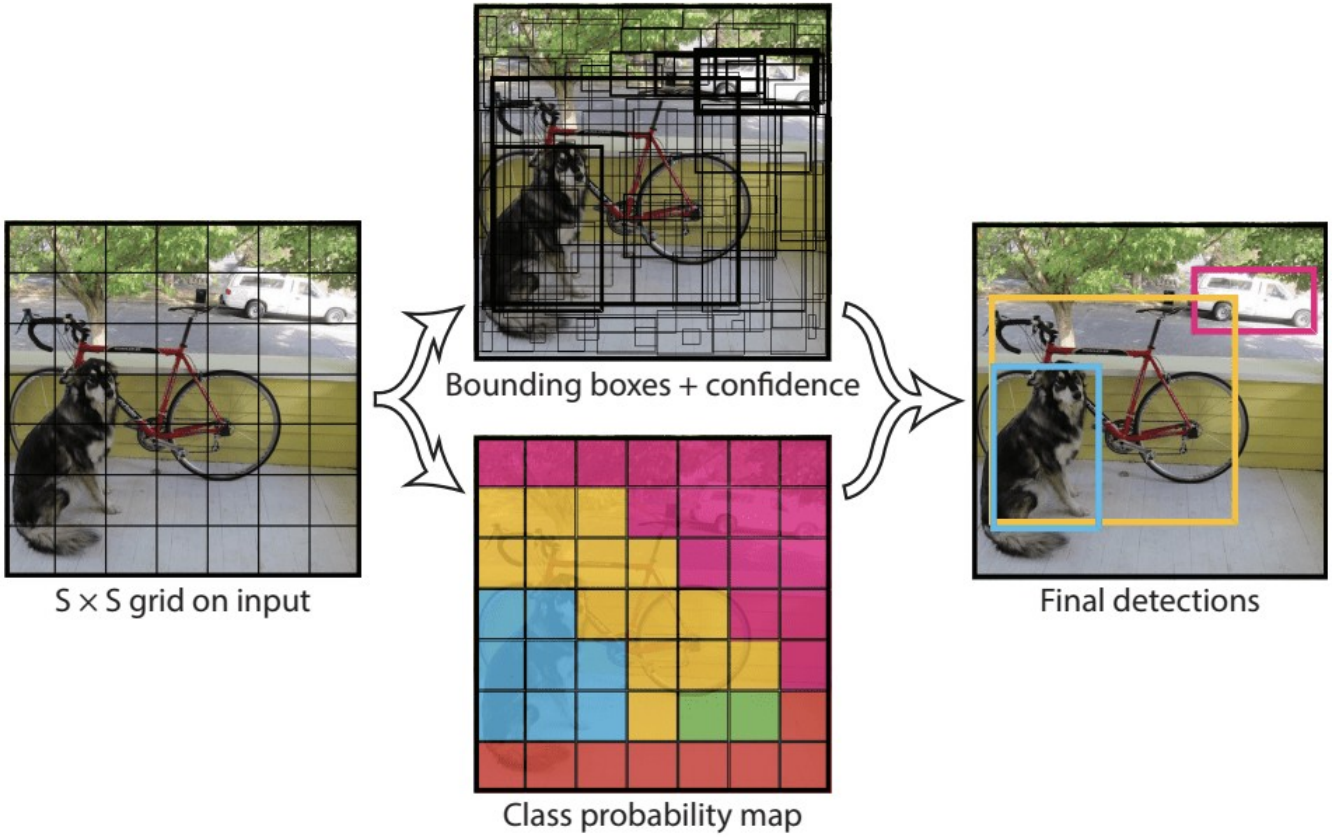
YOLO Models

- The model works by first splitting the input image into a grid of cells, where each cell is responsible for predicting a bounding box if the center of a bounding box falls within the cell.
- Each grid cell predicts a bounding box involving the x, y coordinate and the width and height and the confidence. A class prediction is also based on each cell.

YOLO Models

- For example, an image may be divided into a 7×7 grid and each cell in the grid may predict 2 bounding boxes, resulting in 94 proposed bounding box predictions.
- The class probabilities map and the bounding boxes with confidences are then combined into a final set of bounding boxes and class labels.
- The image taken from the paper below summarizes the two outputs of the model.

YOLO Models


Bounding boxes + confidence
S × S grid on input
Class probability map
Final detections

YOLO: Step by Step

- 1. Divide the image into a grid of cells.
  - The YOLO algorithm divides the image into a grid of cells, typically 7x7 or 13x13.
  - Each cell is responsible for predicting a set of bounding boxes and class probabilities.

YOLO: Step by Step

- 2. Predict bounding boxes and class probabilities for each cell.
  - For each cell, the YOLO algorithm predicts a set of bounding boxes and class probabilities.
  - The bounding boxes are represented as four coordinates: the top left corner, the bottom right corner, and the width and height of the box.
  - The class probabilities represent the probability that the object in the box belongs to a particular class.

YOLO: Step by Step

- 3. Apply non-max suppression.
  - The bounding boxes predicted by the YOLO algorithm may overlap.
  - To remove overlapping boxes, the YOLO algorithm applies a non-max suppression algorithm.
  - This algorithm keeps the box with the highest confidence score, and it removes all other boxes that have a high overlap with the selected box.

YOLO: Step by Step

- 4. Draw the bounding boxes and class labels on the image.
  - The final step is to draw the bounding boxes and class labels on the image.
  - The bounding boxes are drawn in a different color for each class, and the class labels are displayed next to the bounding boxes.

YOLO: Summary

- Divide the image into a grid of cells.
- Predict bounding boxes and class probabilities for each cell.
- Apply non-max suppression.
- Draw the bounding boxes and class labels on the image.

YOLOv2 (YOLO9000) and YOLOv3

- The model was updated by Joseph Redmon and Ali Farhadi in an effort to further improve model performance in their 2016 paper titled "*YOLO9000: Better, Faster, Stronger*."
- Although this variation of the model is referred to as YOLO v2, an instance of the model is described that was trained on two object recognition datasets in parallel, capable of predicting 9,000 object classes, hence given the name "*YOLO9000*."

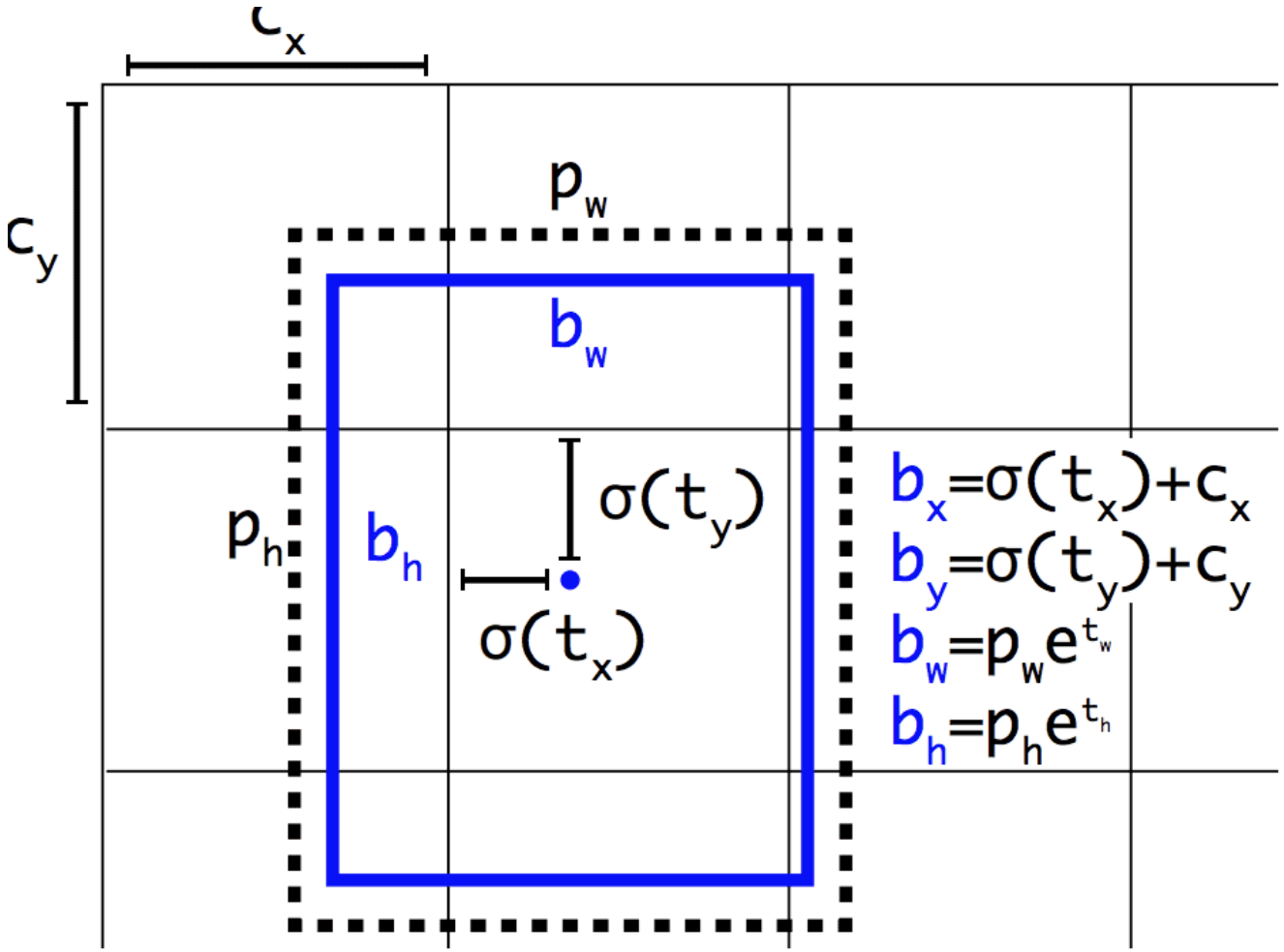
YOLOv2 (YOLO9000) and YOLOv3

- A number of training and architectural changes were made to the model, such as the use of batch normalization and high-resolution input images.
- Like Faster R-CNN, YOLOv2 model makes use of anchor boxes, pre-defined bounding boxes with useful shapes and sizes that are tailored during training.
- The choice of bounding boxes for the image is pre-processed using a k-means analysis on the training dataset.

YOLOv2 (YOLO9000) and YOLOv3

- Importantly, the predicted representation of the bounding boxes is changed to allow small changes to have a less dramatic effect on the predictions, resulting in a more stable model.
- Rather than predicting position and size directly, offsets are predicted for moving and reshaping the pre-defined anchor boxes relative to a grid cell and dampened by a logistic function.

YOLOv2 (YOLO9000) and YOLOv3



Applications of Computer Vision

- Gender Prediction
- Image Mapping
- Face / Object Recognition
- Image Recreation
- Image Matching

Thank you

@mitu_skillologies

@mITuSkillologies

@mitu_group

@mitu-skillologies

@MITUSkillologies

kaggle

@mituskillologies

Web Resources
https://mitu.co.in
http://tusharkute.com

@mituskillologies

contact@mitu.co.in

tushar@tusharkute.com