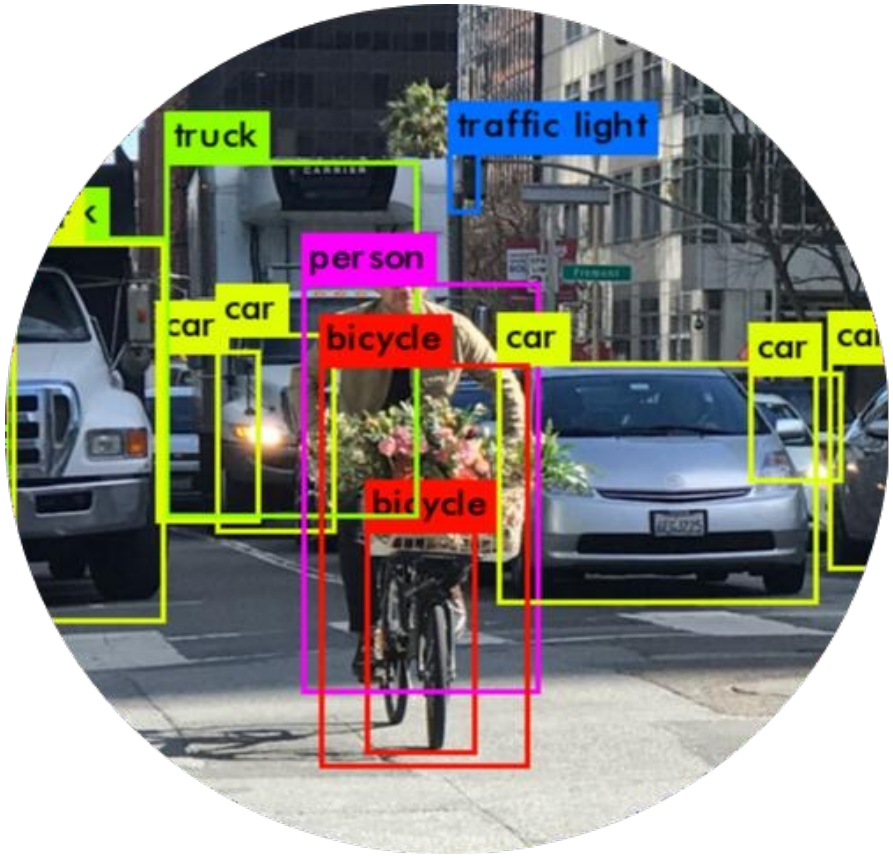
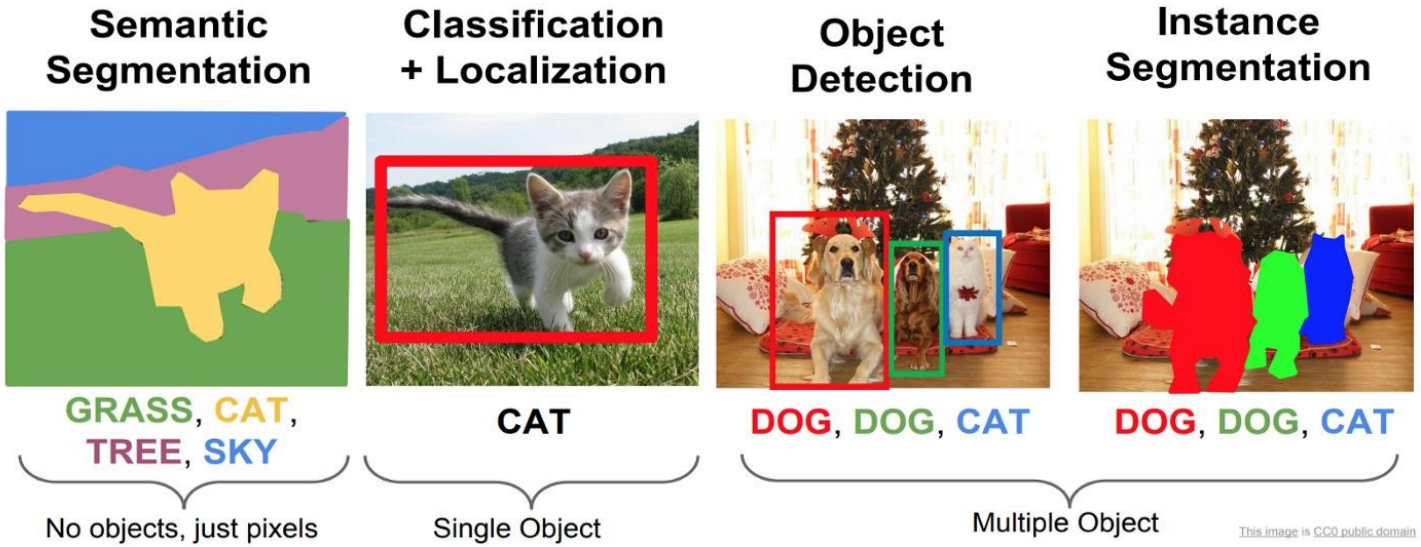
Object Detection & Instance Segmentation Evaluation



Object Detection

- We all know about the image **classification** problem. Given an image, the Net finds out the class to which the image belongs.

- Localizing an object in a picture means predicting a **bounding box** around the object and can be expressed as a **regression** task.



Object Detection

Problem: the dataset does not have bounding boxes around the objects, how can we train our model?

- We need to add them ourselves. This is often one of the hardest and most costly parts of a Machine Learning project: getting the labels.

- It is a good idea to spend time looking for the right tools.

Image Labeling Tool

An image labeling or annotation tool is used to label the images for bounding box object detection and segmentation.

Open-source image labeling tool like :

- VGG Image
- Annotator
- LabelImg
- OpenLabeler
- ImgLab

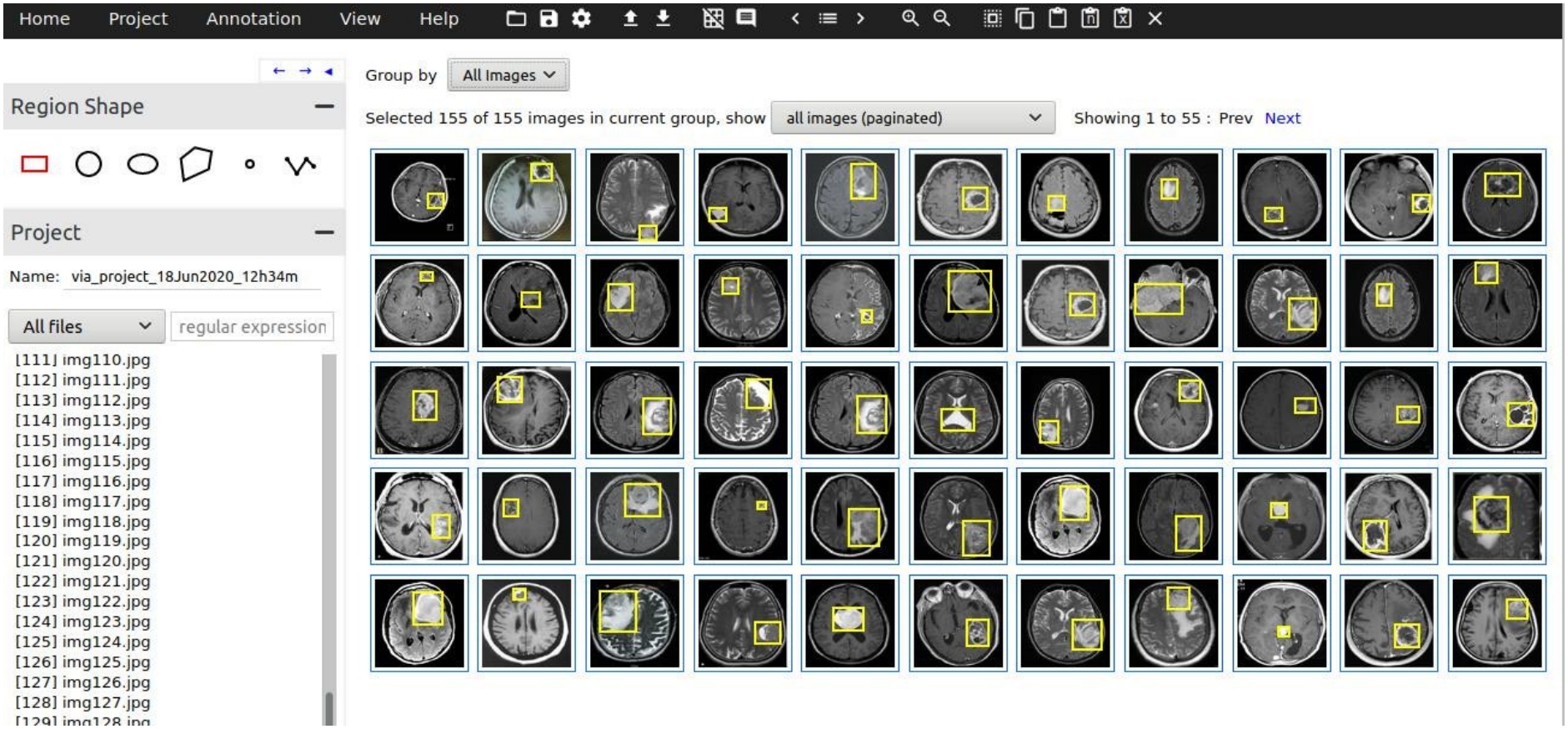
Commercial tool like :

- LabelBox
- Supervisely

Crowdsourcing Platform like :

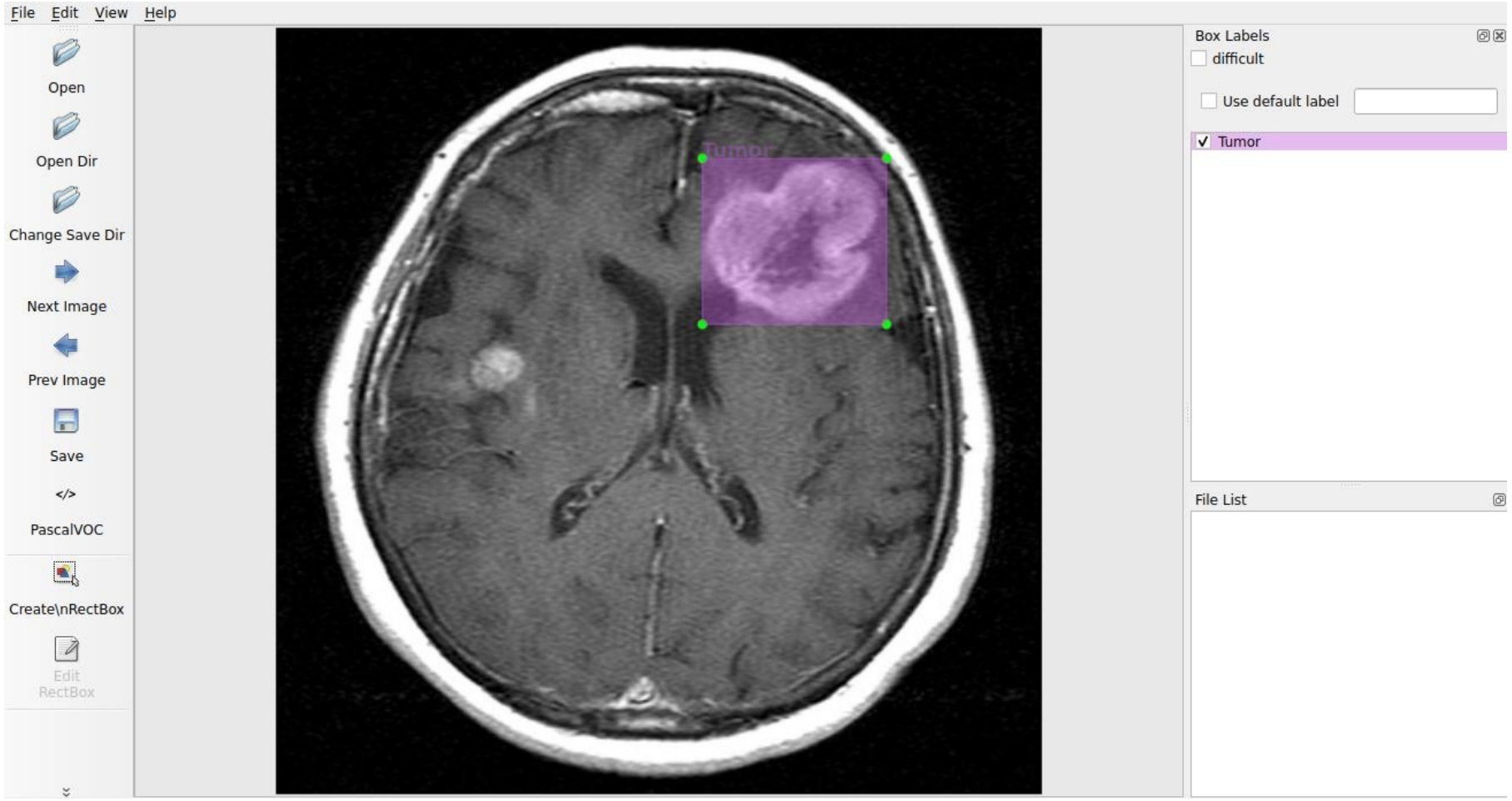
- Amazon Mechanical Turk

Image Labeling Tool - VGG Image



VGG Image: *http://www.robots.ox.ac.uk/~vgg/software/via/*

Image Labeling Tool - labelImg



labelimg

Object Detection - Notes

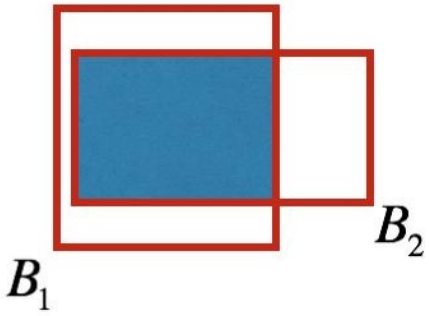
- The bounding boxes should be **normalized** so that the **horizontal** and **vertical** coordinates, as well as the **height** and **width**, all range from **0 to 1**.

- It is common to predict **the square root of the height and width** rather than the height and width directly: this way, a 10-pixel error for a large bounding box will not be penalized as much as a 10-pixel error for a small bounding box.

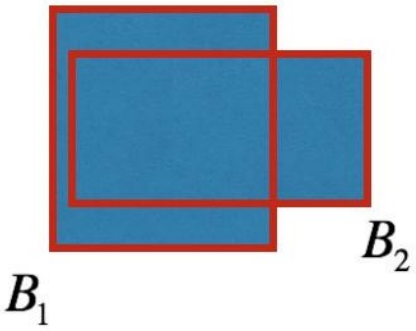
How to Evaluate Object Detection Model?

- The **MSE** often works fairly well as a **cost function** to train the model, but it is not a great metric to evaluate how well the model can predict bounding boxes.
- The most common metric for this is **the Intersection over Union (IoU)**.
- **tf.keras.metrics.MeanIoU**

Intersection

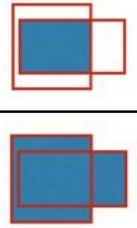


Union



Intersection over Union

$$IoU = \frac{B_1 \cap B_2}{B_1 \cup B_2} =$$



Evaluate Object Detection Model 2mAP

mean Average Precision (mAP)

In order to calculate **mAP**, we draw a series of **precision-recall curves** with the **IoU threshold** set at varying levels of difficulty. In COCO evaluation, the IoU threshold ranges from 0.5 to 0.95 with a step size of 0.05 represented as AP@[.5:.05:.95]

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{\text{TP}}{\#\text{ ground truths}}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{\text{TP}}{\#\text{ predictions}}$$

Matching detections to ground truth

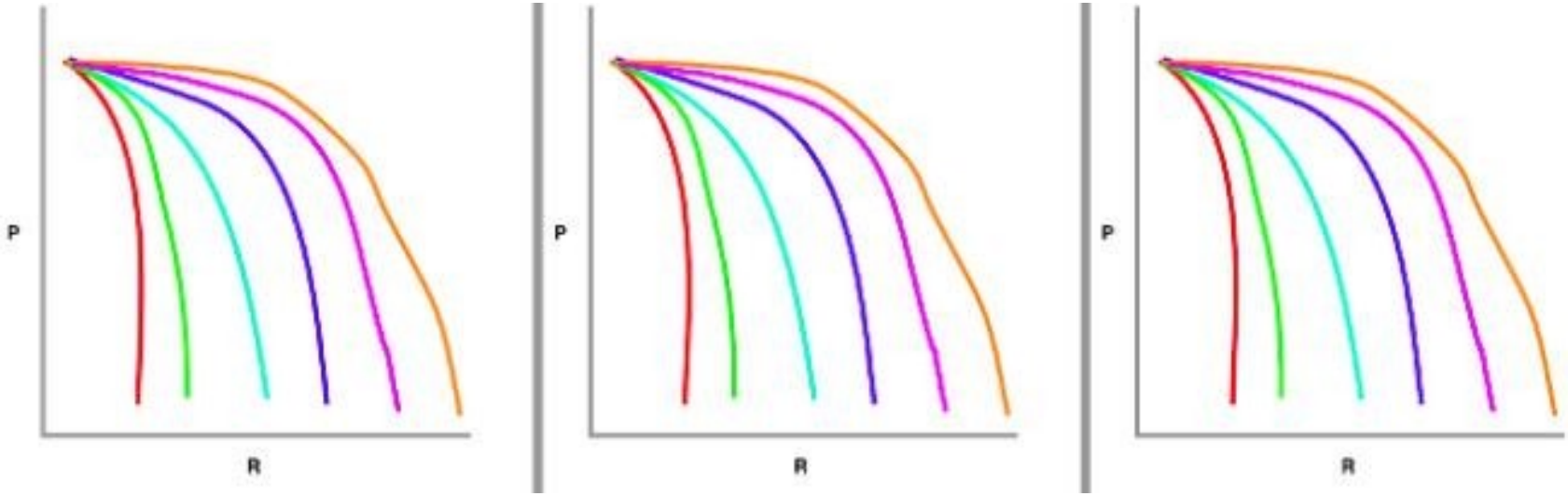
- Match detection to most similar ground truth
  - highest IoU
- If IoU > 50%, mark as correct
- If multiple detections map to same ground truth, mark only one as correct
- **Precision** = #correct detections / total detections
- **Recall** = #ground truth with matched detections / total ground truth

Tradeoff between precision and recall

- ML usually gives scores or probabilities, so threshold
- Too low threshold → too many detections → low precision, high recall
- Too high threshold → too few detections → high precision, low recall
- Right tradeoff depends on application
  - Detecting cancer cells in tissue: need high recall
  - Detecting edible mushrooms in forest: need high precision

Average precision

Evaluate Object Detection Model 2mAP



We draw these precision-recall curves for the dataset split out by class type (for example **3 classes** and **6 threshold ranges**).

Evaluate Object Detection Model 2mAP

These precision and recall values are then plotted to get a PR (precision-recall) curve. **The area under the PR curve** is called **Average Precision (AP)**.

- For each class, calculate AP at different IoU thresholds and take their average to get the AP of that class.

$$\text{AP[class]} = \frac{1}{\#\text{thresolds}} \sum_{\text{iou} \in \text{thresholds}} AP[class, iou]$$

Exp: in **PASCAL VOC challenge 2007**, it is defined as the mean of precision values at a set of **11** equally spaced recall levels [0,0.1,…,1] (0 to 1 at step size of 0.1).

$$AP = \frac{1}{11} \sum_{r \in (0, 0.1, \ldots, 1)} p_{interp(r)}$$

Object detection metrics : *https://github.com/rafaelpadilla/Object-Detection-Metrics*

Evaluate Object Detection Model 2mAP

- Calculate the final AP by averaging the AP over different classes.

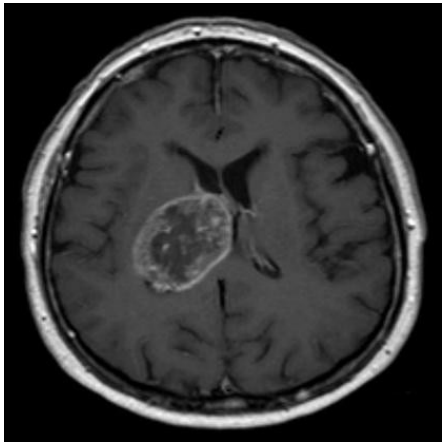
$$\text{AP} = \frac{1}{\#\text{classes}} \sum_{\text{class} \in \text{classes}} AP[class]$$

= **mAP-IoU_thresholds**

mAP - example COCO dataset

$$mAP_{\text{COCO}} = \frac{mAP_{0.50} + mAP_{0.55} + \ldots + mAP_{0.95}}{10}$$

Object Detection - Example



raw image
416*416

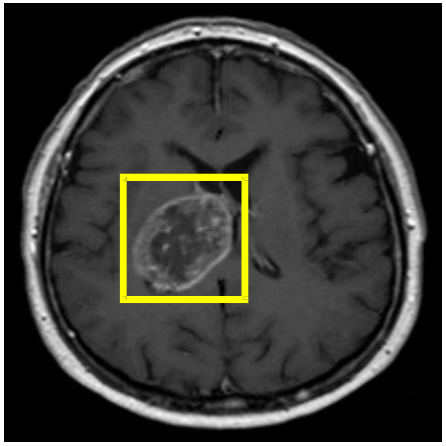
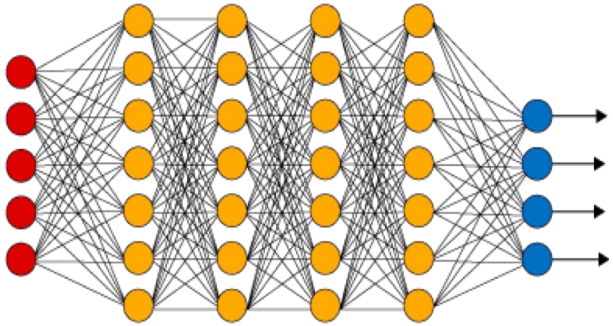


image labeling
(C, X,Y, W, H)



model

(C, X,Y, W, H)

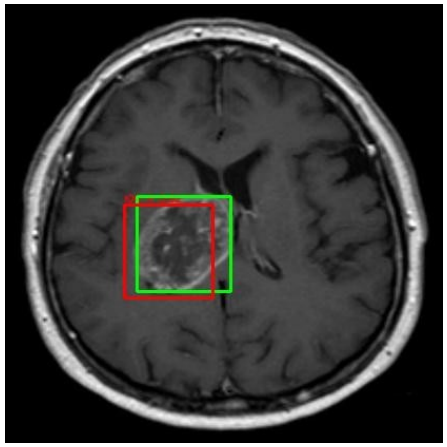


image result

- Each item should be a tuple of the form :

(images,
(class_labels, bounding_boxes))

Object Detection - Example

```
81  model = keras.models.Sequential()
82  model.add(keras.layers.Conv2D(filters=64, kernel_size=5, strides=1, padding="same",
83                                activation="relu", input_shape= (416,416,3)))
84  model.add(keras.layers.MaxPool2D(pool_size=2))
85  model.add(keras.layers.Conv2D(filters=128, kernel_size=3, strides=1, padding="same",
86                                activation="relu"))
87  model.add(keras.layers.MaxPool2D(pool_size=2))
88  model.add(keras.layers.Conv2D(filters=128, kernel_size=3, strides=1, padding="same",
89                                activation="relu"))
90  model.add(keras.layers.MaxPool2D(pool_size=2))
91
92  model.add(keras.layers.Flatten())
93  model.add(keras.layers.Dense(64, activation="relu"))
94  model.add(keras.layers.Dense(32, activation="relu"))
95  model.add(keras.layers.Dense(4))
96
97  opt = keras.optimizers.Adam(lr=0.001, beta_1=0.9, beta_2=0.999)
98  model.compile(loss="mean_squared_error", optimizer=opt, metrics=["mse"])
99  history = model.fit(X_train, y_train, epochs=10, batch_size=8)
```

Full code: *https://www.kaggle.com/hichemfelouat/braintumorlocalization*

Object Detection - Exp TL

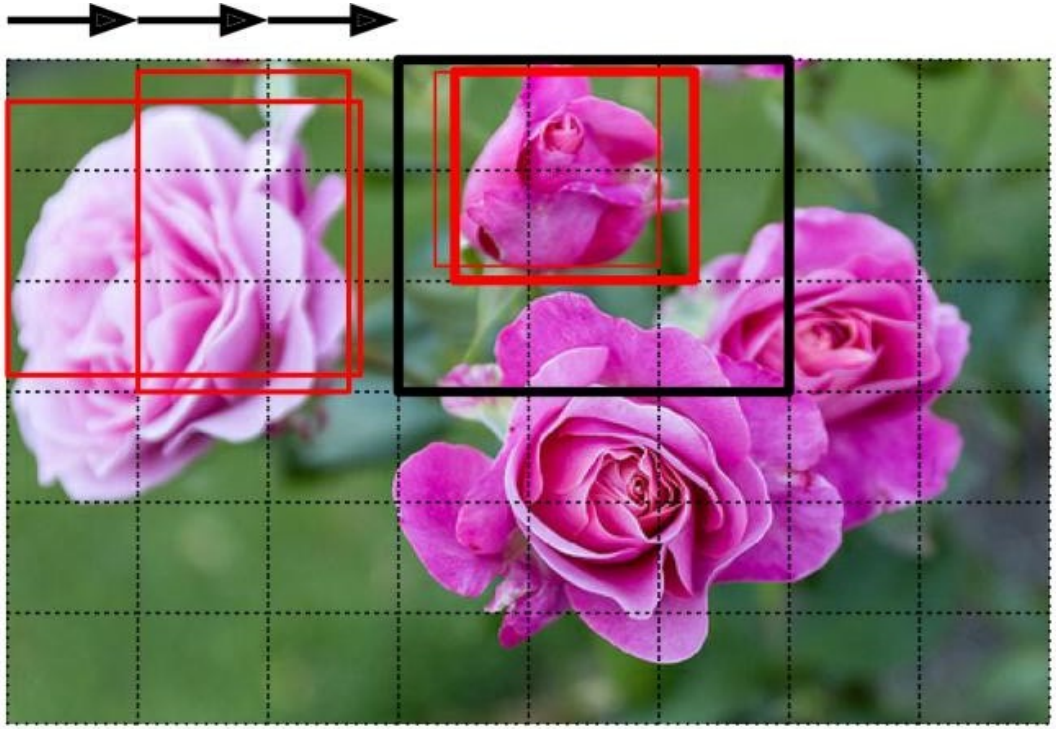
```
82  base_model = keras.applications.inception_resnet_v2.InceptionResNetV2(weights=
83                                                      "imagenet",include_top=False)
84  avg = keras.layers.GlobalAveragePooling2D()(base_model.output)
85  layer_h1 = keras.layers.Dense(64, activation="relu")(avg)
86  layer_h2 = keras.layers.Dense(32, activation="relu")(layer_h1)
87  output = keras.layers.Dense(4)(layer_h2)
88  model = keras.Model(inputs=base_model.input, outputs=output)
```

Full code: *https://www.kaggle.com/hichemfelouat/braintumorlocalization*

Multiple Objects Detection

The task of **classifying and localizing multiple objects** in an image is called **object detection**.

Detecting multiple objects by **sliding a CNN across the image**.



Multiple Objects Detection

1. You need to add an **extra objectness output** to your CNN, to estimate the probability that an object is indeed present in the image.

2. Find **the bounding box with the highest objectness score**, and get rid of all the other bounding boxes that overlap a lot with it (**IoU**).

3. **Repeat step two** until there are no more bounding boxes to get rid of.

Multiple Objects Detection

In general, object detectors have three (3) main components:

1) The **backbone** that **extracts features** from the given image.

2) The **feature network** that takes multiple levels of features from the backbone as input and **outputs a list of fused features** that represent salient characteristics of the image.

3) The final **class/box network** that uses the fused features to **predict the class and location** of each object.

Other performance metrics

- **Pixel accuracy**
  - Fraction of pixels correctly classified.
- **Mean pixel accuracy**
  - Pixel accuracy averaged over classes.
- **Dice coefficient**
  - Dice = $\frac{2|A \cap B|}{|A|+|B|}$
  - Similar to IoU, as is evident from the equation above.
  - Same as F1 score for binary segmentation (like foreground vs. background).
  - More popular in medical image segmentation domain. Used by UNet++ as loss.
- Many other metrics exist

Basic loss functions

- Semantic segmentation
  - Calculate cross entropy for each pixel.
  - Sum the losses of all the pixels.
  - Balance for class if required.
- Instance segmentation
  - Since it's typically trained jointly with object detection, the loss is the sum of all the three losses: classification loss, box regression loss, mask loss
  - The mask loss is calculated just like for semantic segmentation
- Loss functions are typically modified as per the need/problem.
- There are many application/dataset-specific loss functions out there.