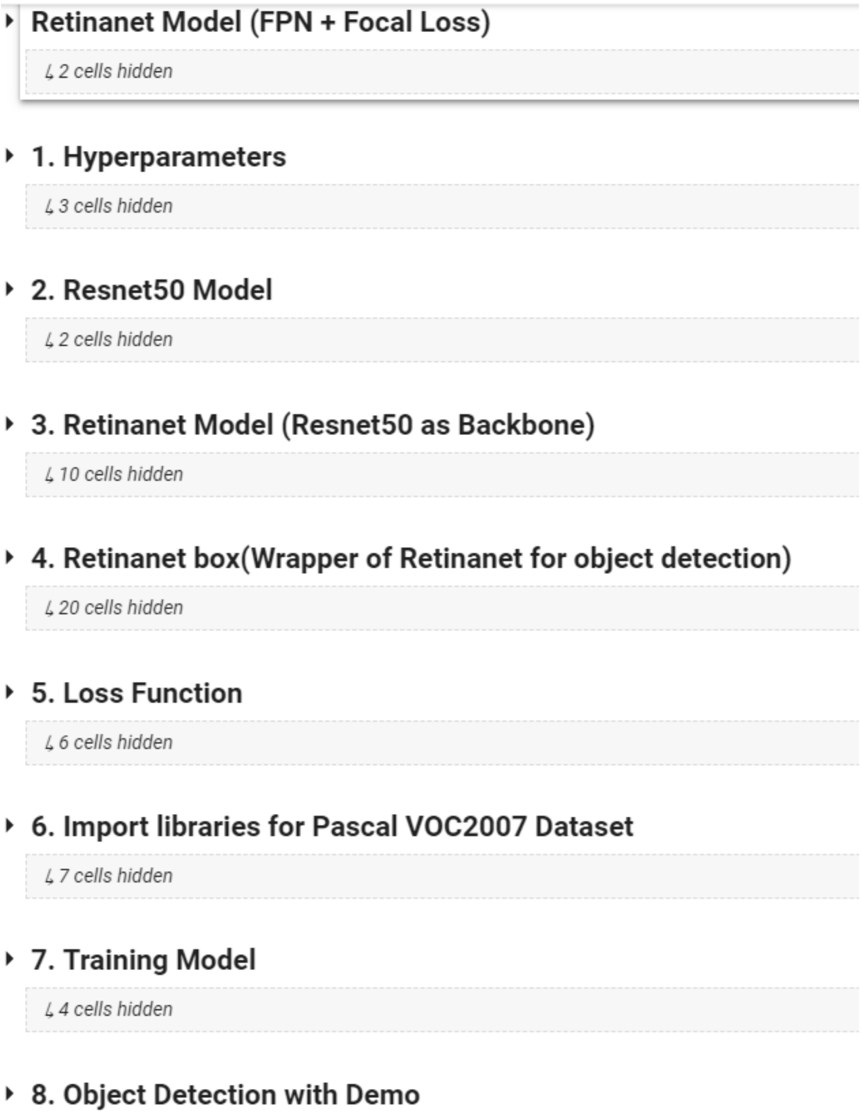
Using Focal Loss, RetinaNet able resolve issue of huge class imbalance problem.
RetinaNet use ResNet50 + FPN (Feature Pyramid Network) as backbone for better feature extraction

In this document, I have mainly explained my approach & code hierarchy.
In code, I have mentioned detail explanation of each logic block by adding more comments & diagrams. For more details, please refer code.

RetinaNet Code Hierarchy



▸ Retinanet Model (FPN + Focal Loss)

↳ 2 cells hidden

▸ 1. Hyperparameters

↳ 3 cells hidden

▸ 2. Resnet50 Model

↳ 2 cells hidden

▸ 3. Retinanet Model (Resnet50 as Backbone)

↳ 10 cells hidden

▸ 4. Retinanet box(Wrapper of Retinanet for object detection)

↳ 20 cells hidden

▸ 5. Loss Function

↳ 6 cells hidden

▸ 6. Import libraries for Pascal VOC2007 Dataset

↳ 7 cells hidden

▸ 7. Training Model

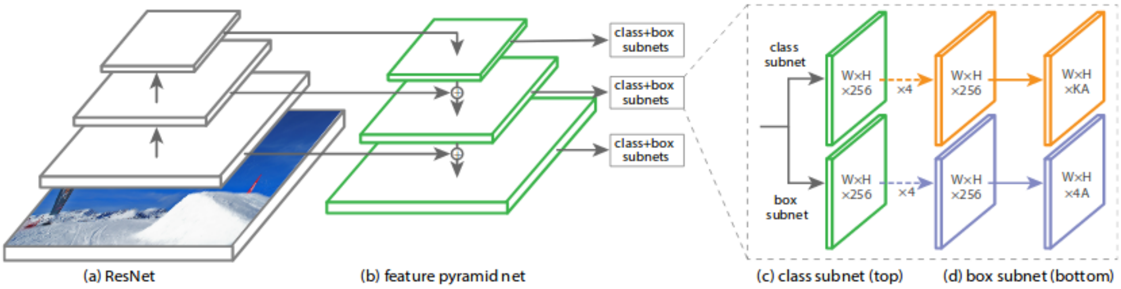
↳ 4 cells hidden

▸ 8. Object Detection with Demo

Main 4 parts of code:

1. RetinaNet Model (retinanet_model) *[Cover Topics 2 - 3]*
2. Inference Model (retinanet_box) on top of RetinaNet Model *[Cover Topic 4]*
3. Train retinanet_model *[Cover Topics 5 - 7]*
4. Object Detection/Test RetinaNet using Inference Model (retinanet_box) *[Cover Topics 8]*

1. RetinaNet Model



(a) ResNet (b) feature pyramid net (c) class subnet (top) (d) box subnet (bottom)

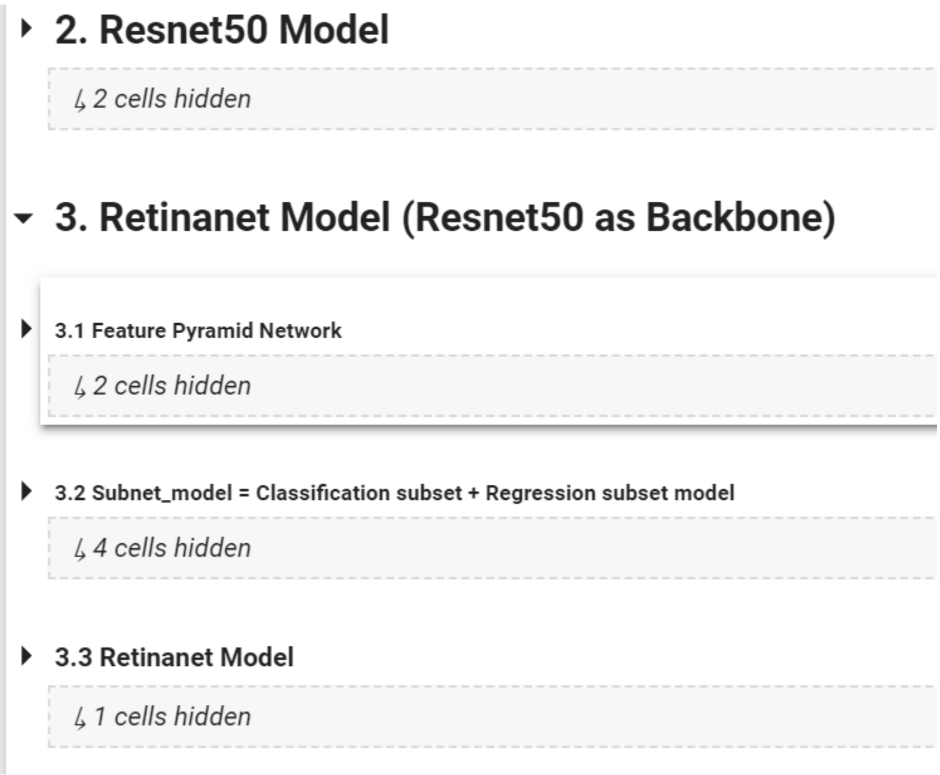
RetinaNet Model contains 4 main logic blocks

a) **ResNet50:** I imports ResNet50 model using Keras package (Only Architecture). We are using 3 encoding outputs of ResNet, called {C3, C4, C5}

b) **Feature Pyramid Network:** FPN use {C3, C4, C5} layers to generate {P3, P4, P5} which is multiscale & more semantically rich. P6, P7 layers are additionally calculated using std 3x3 convolution. Now We have {P3, P4, P5, P6, P7} layers which are connected to classification subnet & regression subnet

c) **Classification Subnet**: It applies four 3x3 conv layers (filter=256 & with relu), followed by a 3×3 conv layer with *K*A* filters. (*K* = No. of classes, *A*=9 anchors). It predicts objectness score for each A anchors & K classes at each spatial position of FPN feature map

Regression subnet: It applies four 3x3 conv layers (filter=256 & with relu), followed by a 3×3 conv layer with *4*A* filters. (*K* = No. of classes, *A*=9 anchors). It predicts the offset value for each anchor boxes compare to a nearby ground-truth object, if one exists



▸ 2. Resnet50 Model

↳ 2 cells hidden

▾ 3. Retinanet Model (Resnet50 as Backbone)

▸ 3.1 Feature Pyramid Network

↳ 2 cells hidden

▸ 3.2 Subnet_model = Classification subset + Regression subset model

↳ 4 cells hidden

▸ 3.3 Retinanet Model

↳ 1 cells hidden

Figure - RetinaNet Code Hierarchy

2. Inference Model (Retinanet_box)

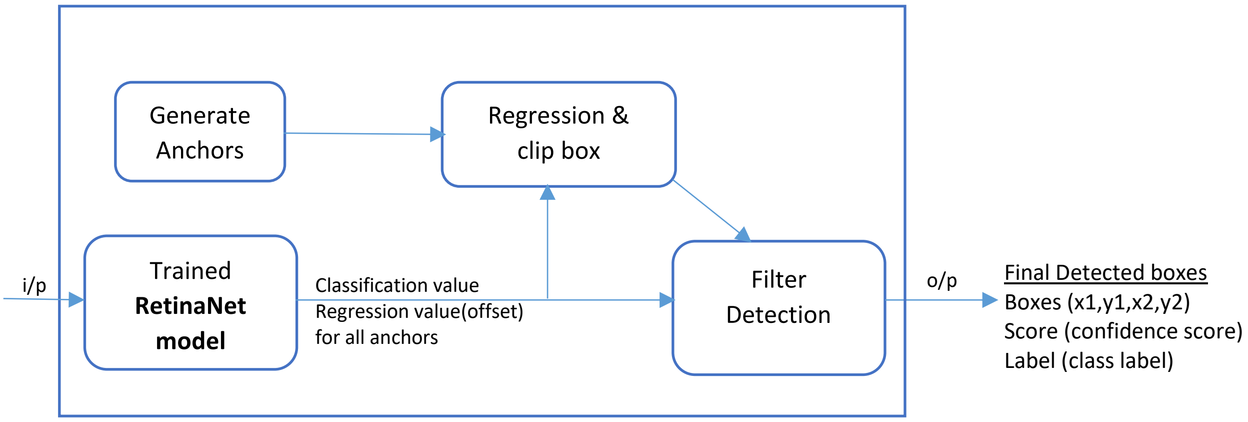
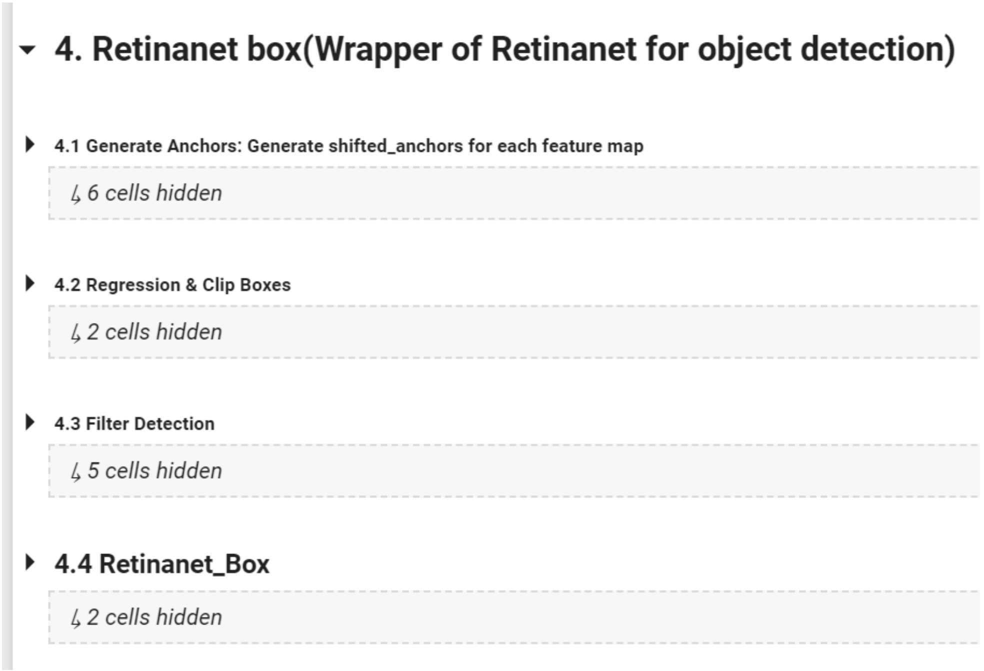


Figure - Inference model (retinanet_box)

Trained RetinaNet model can't be used for Inference. so I created Inference model by adding some extra layers to RetinaNet model.

Retinanet_box contains 4 main logic blocks

1. **Retinanet_model:** Main retinanet_model which is used while training
2. **Generate Anchors:** Generate final anchors for each feature maps of FPN based on given values of size, stride, scale & ratios
3. **Regression & clipbox:** Convert Boxes offset value to boxes absolute value (x1,y1,x2,y2) & clipped value which is outside of feature map
4. **Filter Detection:** Apply non_max_suppression (nms) & iou_threshold, to identify final top k detection



4. Retinanet box(Wrapper of Retinanet for object detection)

4.1 Generate Anchors: Generate shifted_anchors for each feature map

↳ 6 cells hidden

4.2 Regression & Clip Boxes

↳ 2 cells hidden

4.3 Filter Detection

↳ 5 cells hidden

4.4 Retinanet_Box

↳ 2 cells hidden

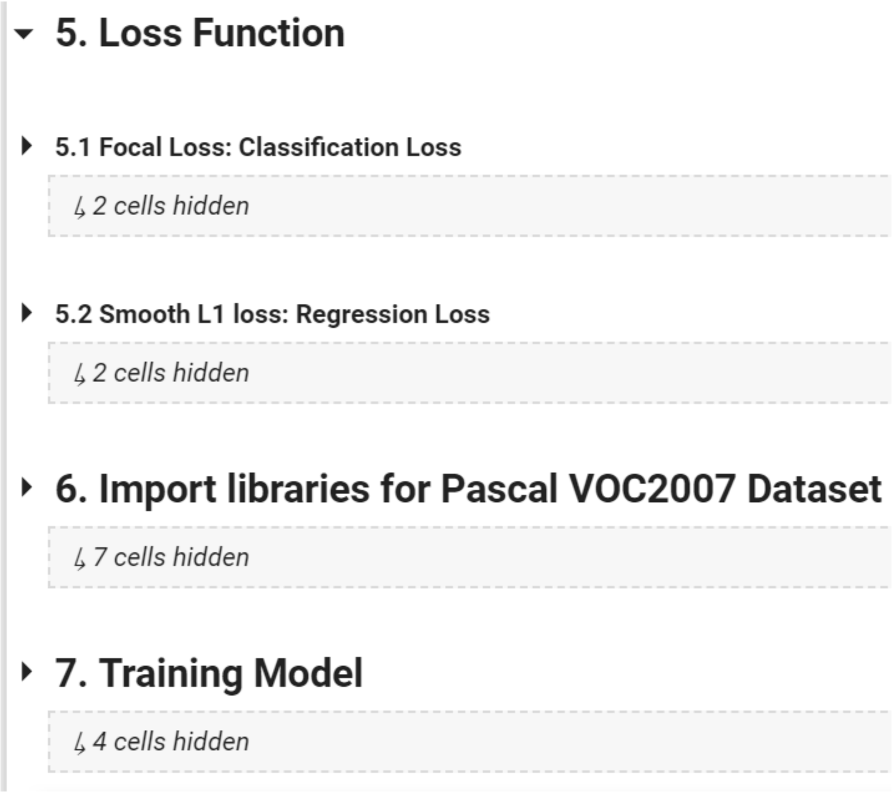
Figure: Retinanet_box code hierarchy

3. Training RetinaNet Model

Now, we start training RetinaNet Model

Training contains mainly 3 logic blocks

1. **Loss Function:**
    - Classification Loss: Focal Loss is used, which resolve class imbalance problem
    - Regression Loss: Smooth L1 Loss
2. **Dataset:** import Pascal VOC2007 Dataset & train_generator logic
3. **Training:** Adam optimizer is used & Learning rate = 1e-5



5. Loss Function

5.1 Focal Loss: Classification Loss

↳ 2 cells hidden

5.2 Smooth L1 loss: Regression Loss

↳ 2 cells hidden

6. Import libraries for Pascal VOC2007 Dataset

↳ 7 cells hidden

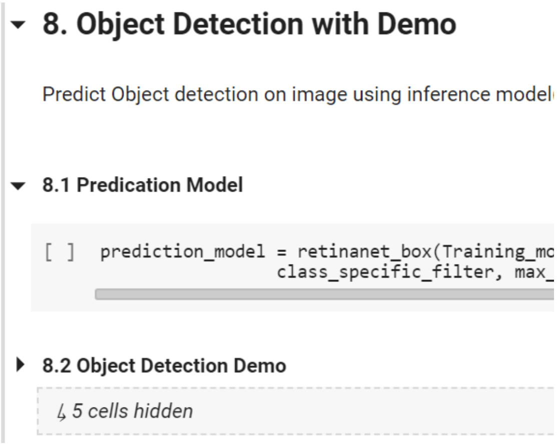
7. Training Model

↳ 4 cells hidden

Figure: Training code hierarchy

4. Object Detection using Retinanet_box (Inference Model)

Now we can perform object detection for any image using Inference model- retinanet_box



8. Object Detection with Demo

Predict Object detection on image using inference model

8.1 Predication Model

```
[ ]  prediction_model = retinanet_box(Training_mo
                          class_specific_filter, max_
```

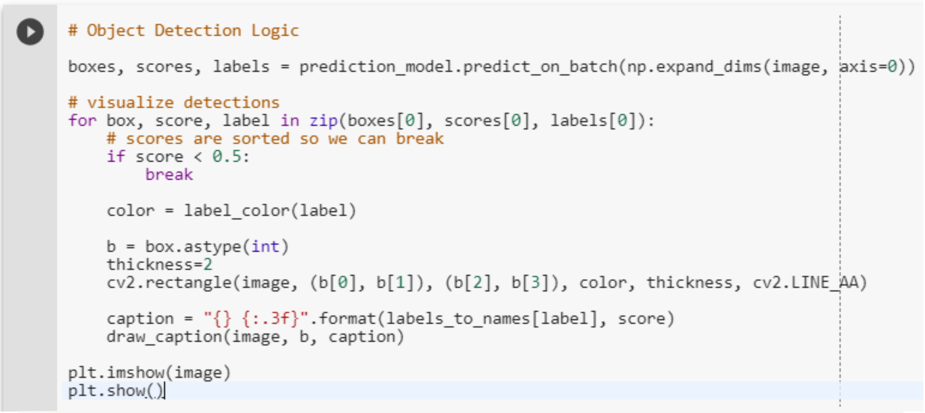
8.2 Object Detection Demo

↳ 5 cells hidden

Figure: Object Detection code hierarchy

Mainly 2 Steps:

1. Predict boxes, score, labels values using retinanet_box (Inference model)
2. Using OpenCV, we can represent image with detected box



```
# Object Detection Logic

boxes, scores, labels = prediction_model.predict_on_batch(np.expand_dims(image, axis=0))

# visualize detections
for box, score, label in zip(boxes[0], scores[0], labels[0]):
    # scores are sorted so we can break
    if score < 0.5:
        break

    color = label_color(label)

    b = box.astype(int)
    thickness=2
    cv2.rectangle(image, (b[0], b[1]), (b[2], b[3]), color, thickness, cv2.LINE_AA)

    caption = "{} {:.3f}".format(labels_to_names[label], score)
    draw_caption(image, b, caption)

plt.imshow(image)
plt.show()
```

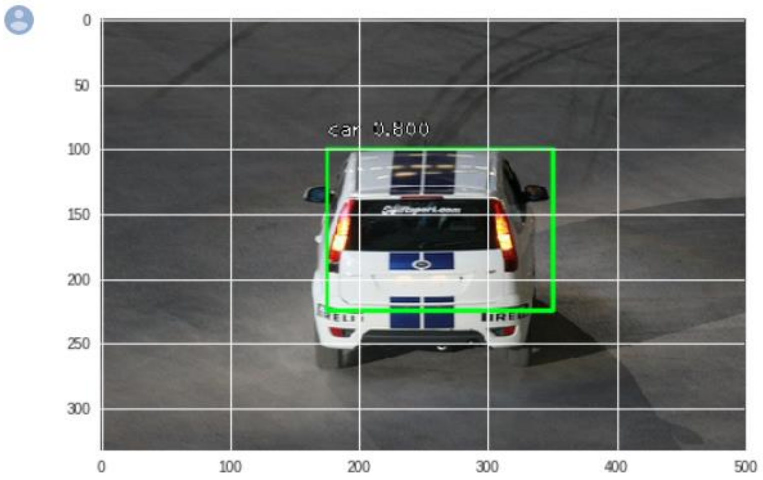


Figure: Detection Demo