Mask R-CNN

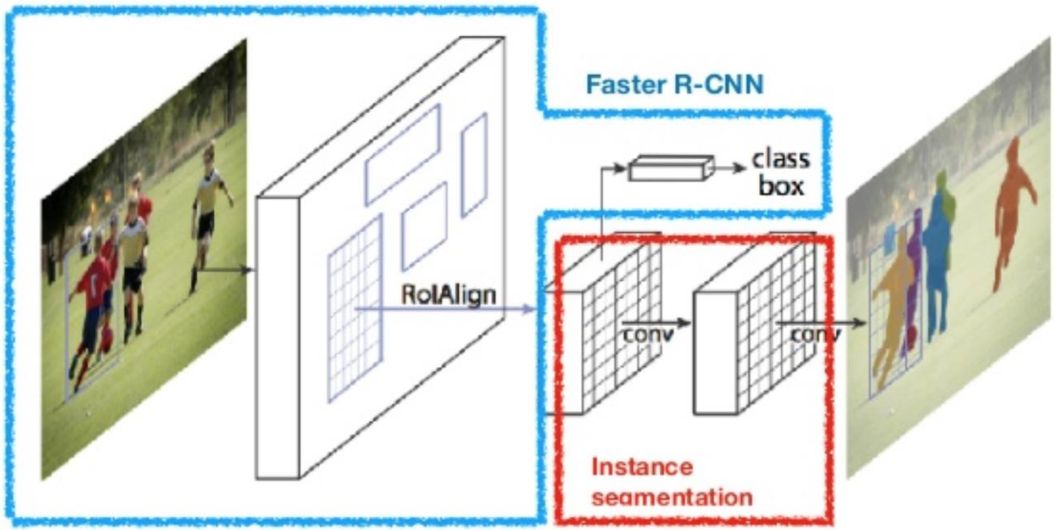
Mask Region-based
Convolutional Neural Network
for Instance Segmentation

Group 5

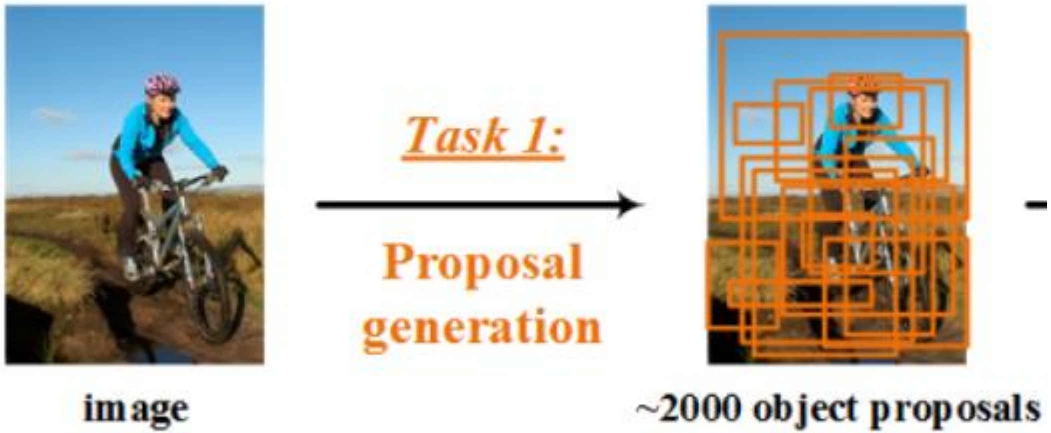
ID	Name	Tasks	Progress
19127191	**Ngô Văn Anh Kiệt**	Research paper: Backbone, Head Architecture; Presentation script	100%
19127005	Trần Phan Thanh Hải	Model coding: train, test, deploy (primary role)	100%
19127505	Triệu Nguyên Phát	Research paper: Mask RCNN results and comparisons.	100%
19127511	La Ngọc Hồng Phúc	Research paper: Mask RCNN for Human Pose Estimation Video editing	100%
19127575	Nguyễn Thái Tiến	Research paper: RoI Align, Mask branch	100%

1

Backbone Architecture



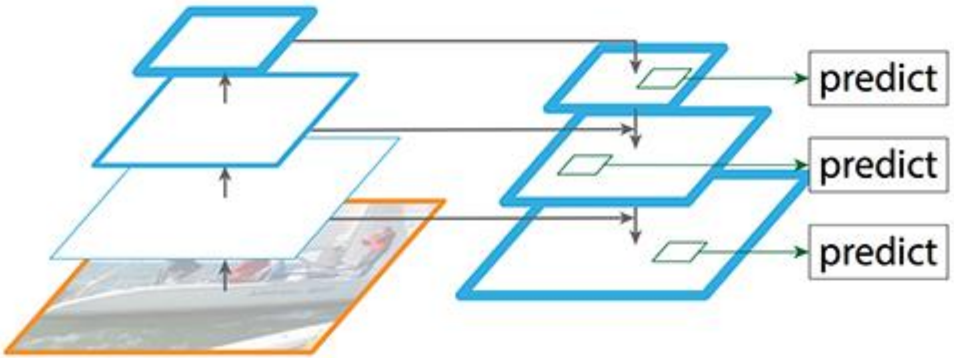
- A backbone architecture is for extracting regions of interest from an image.
- Mask RCNN uses a backbone similar to Faster RCNN: ResNet/ResNeXt
- It's recommended in the paper that Mask RCNN should be used with ResNet-FPN backbone for good accuracy and speed.



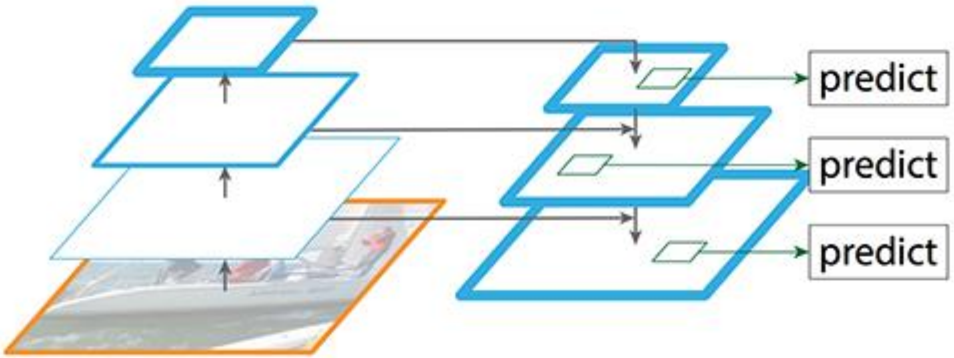
- The output of the backbone is a collection of possible regions in the image that can contain an object.
- These regions are the raw forms of the bounding boxes we see in the final output of Mask RCNN.
- Each of these region will be forwarded to the later layers.

2

Feature Pyramid Network (FPN)

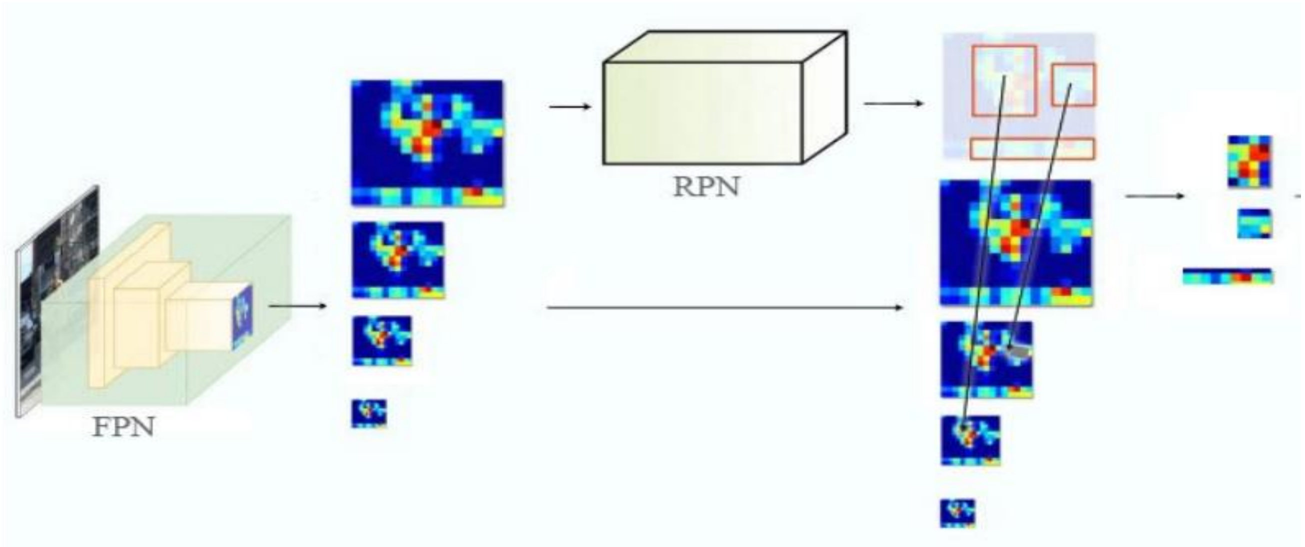


- An important modification of Mask RCNN to the backbone of Faster RCNN.
- The regions of interest generated by the backbone will be fed into the FPN for feature extraction.

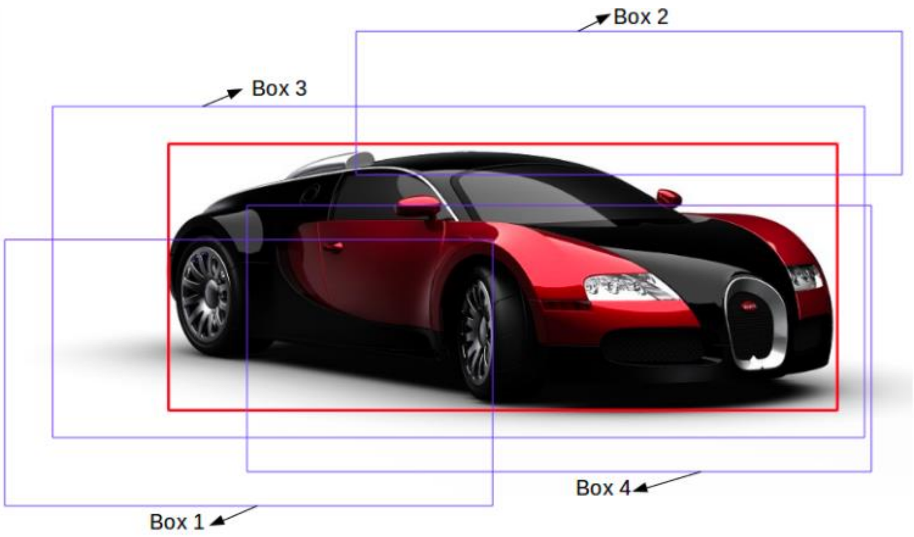


FPN composes of a bottom-up and a top-down pathway:

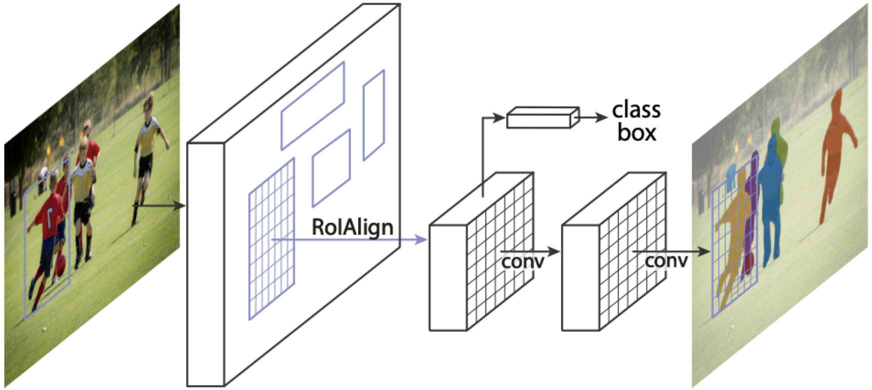
- The bottom-up pathway: convolutional network for feature extraction. As we go up, the spatial resolution decreases, higher-level structures detected, and the semantic value for each layer increases.
- The top-down pathway reconstructs higher resolution layers from a semantic rich layer.



- In the backbone architecture, FPN is used along with a Region Proposal Network (RPN).
- This helps extracts RoI features from different levels of the feature pyramid according to their scale.
- Then the RPN will also predict whether the extracted region contains an object.



- Regions that have been predicted to contain objects will be evaluated using a metric called Intersection Over Union (IoU).
- IoU = Area of Intersection / Area of Union
- Regions that have their bounding boxes satisfy IoU >= 0.5 will be forwarded to the next layer.



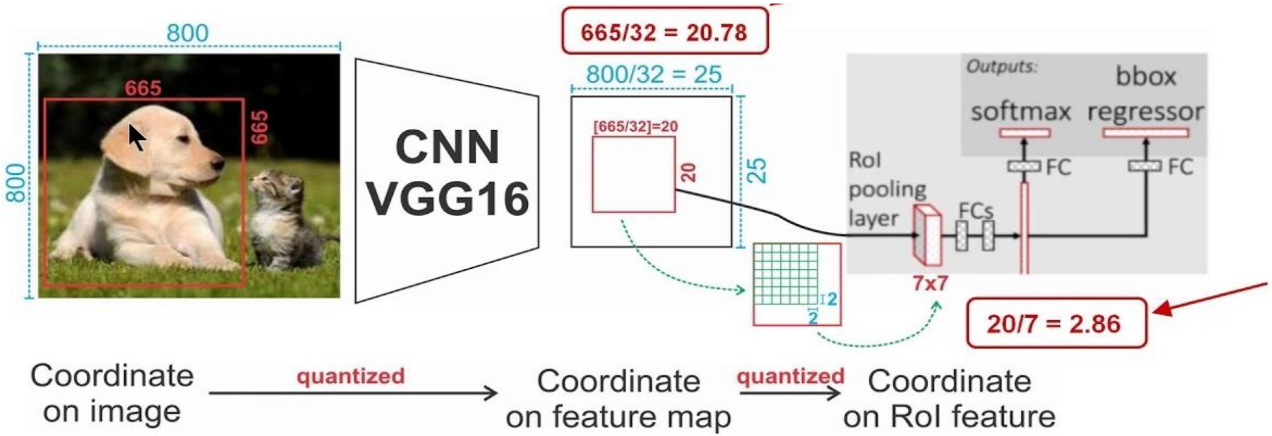
Mask R-CNN = Faster R-CNN + FCN

- To fix the misalignment between the RoI and the extracted features, we propose a simple, quantization-free layer, called RoIAlign

- Mask R-CNN, extends Faster R-CNN by adding a branch for predicting segmentation masks on each Region of Interest (RoI) . The mask branch is a small FCN

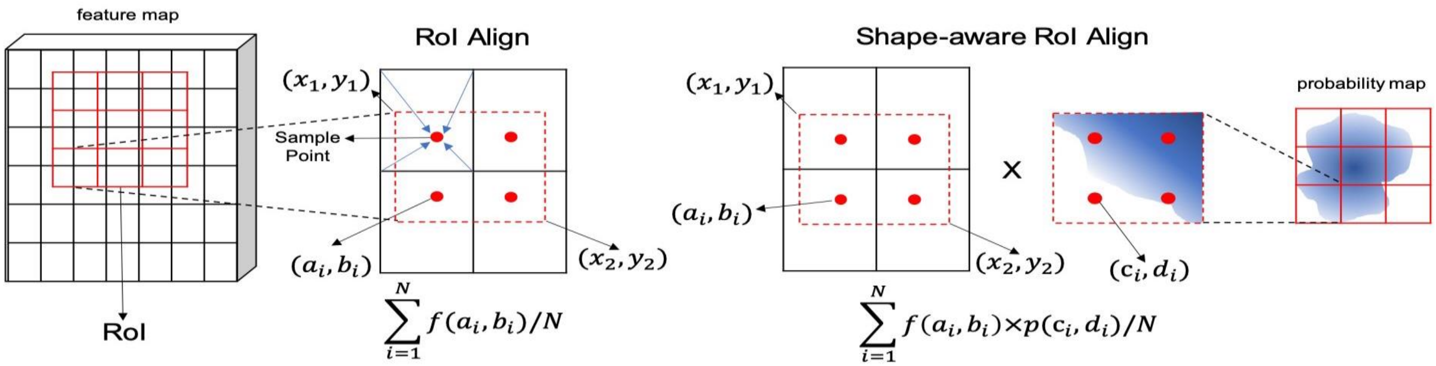
3

ROI ALIGN

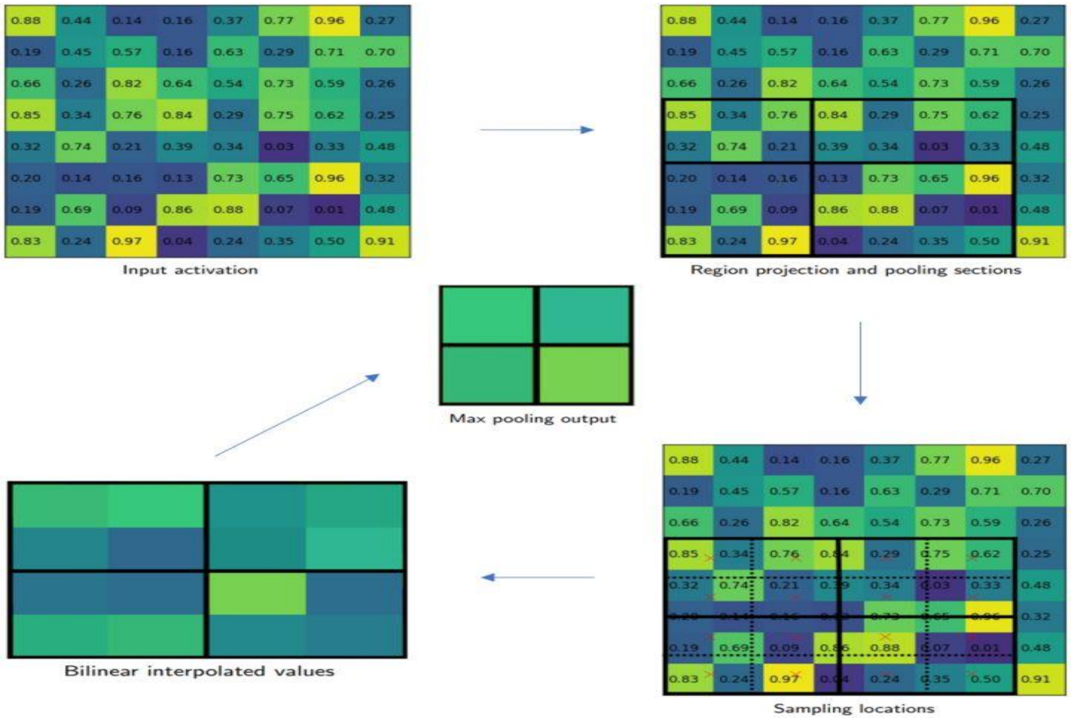


- RoIAlign layer that removes the harsh quantization of RoIPool , properly aligning the extracted features with the input .

- RoI Align is not using quantization for data pooling.

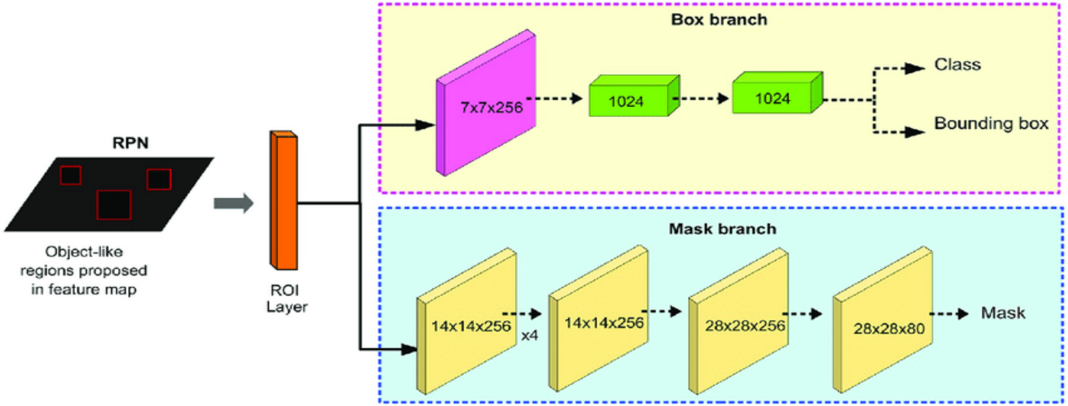


- ROI align divides each coordinate by *k*: **x / k** and does NOT round it to integer.
- Nevertheless, cropped part is also divided into **grid**, but for defining concrete values in these bins ROI align choose regularly **4 points** in each bin using **bilinear interpolation** (as shown in picture above). And from these 4 points maximum or average value from each bin is taken. are used to reduce the dimensions of the feature maps



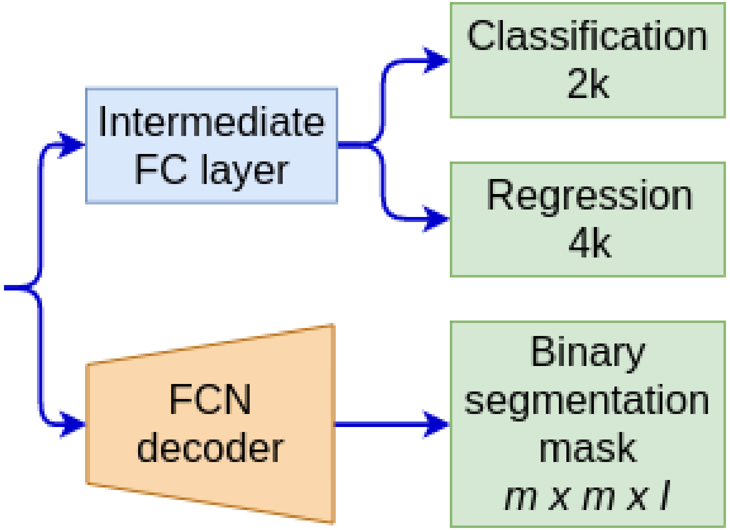
RoIAlign improves mask accuracy by relative 10% to 50% , show bigger gains under stricter localization metrics

4 Fully convolutional mask



- The mask branch is a small FCN applied to each RoI, predicting a segmentation mask in a pixel-to-pixel manner .

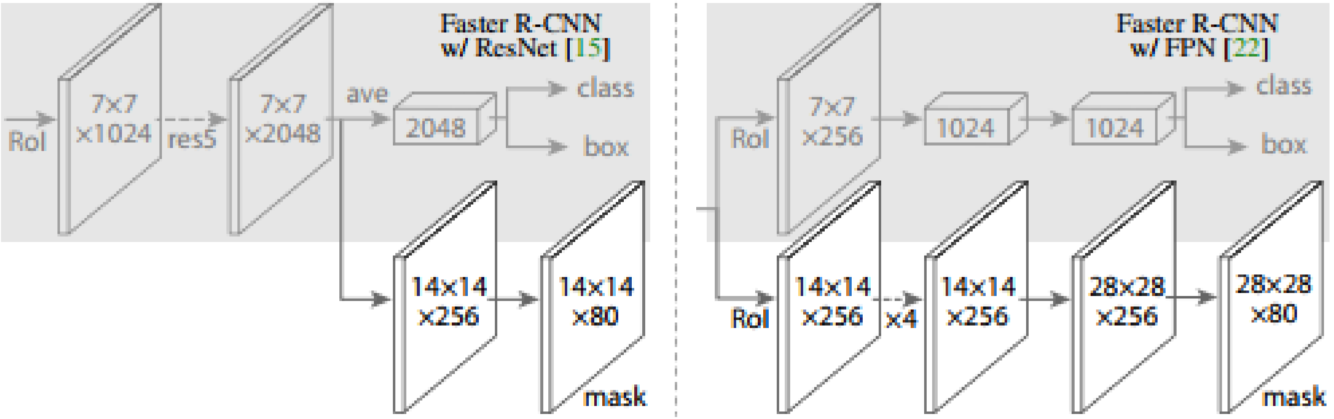
- Represent a mask as m x m matrix . Use binary loss to train the network .



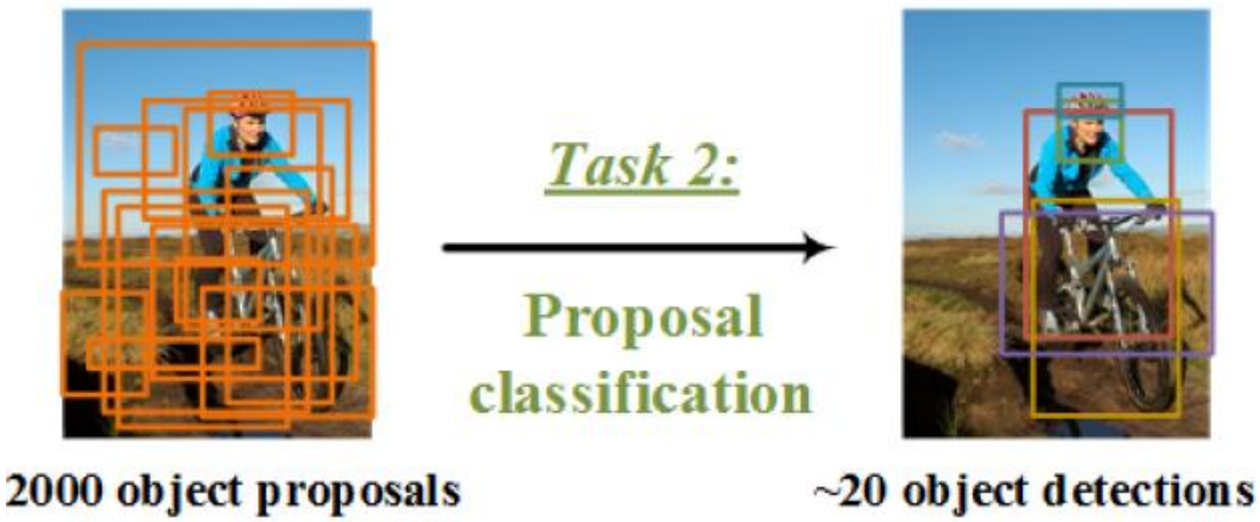
- Use sigmoid to predict probability for each pixel . L_{mask} mean binary cross-entropy

- We just simply need to use a few extra convolutional layers on each region of interest

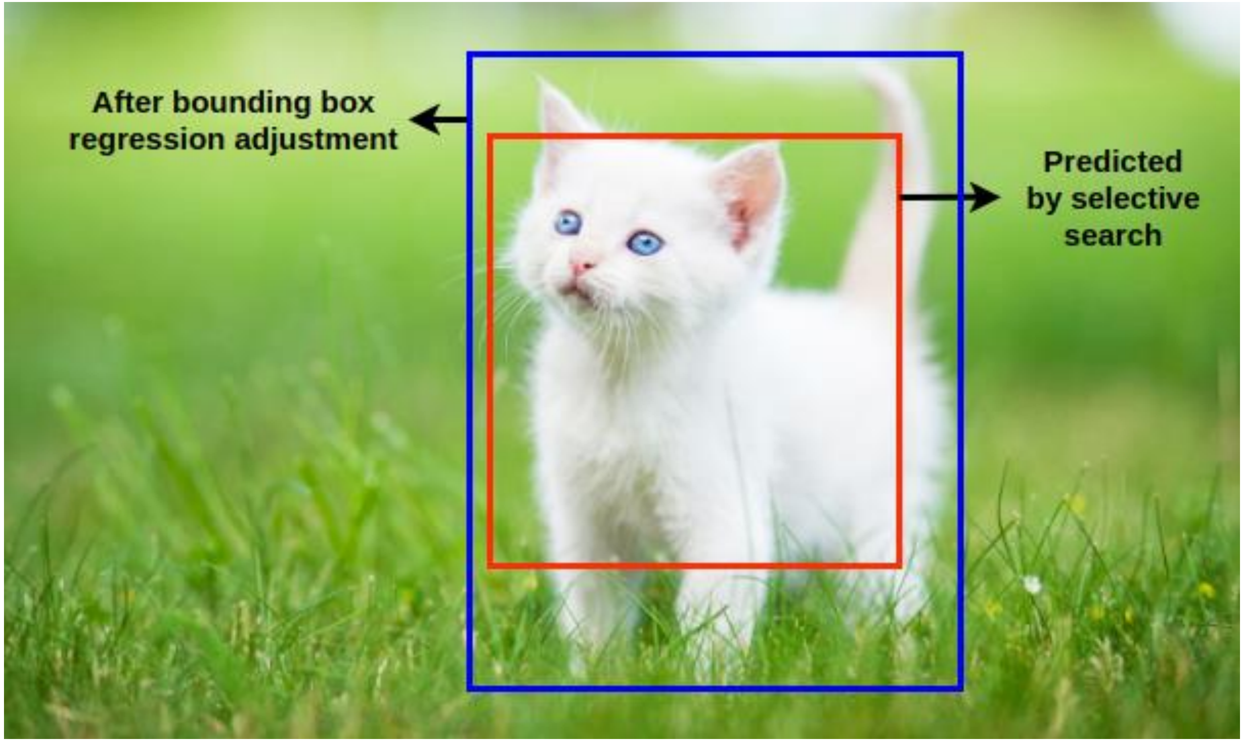
5 Head architecture



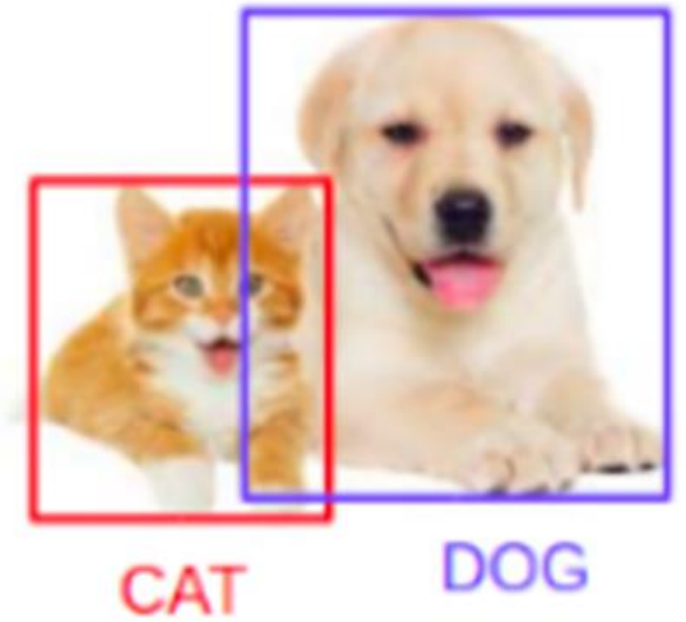
- With the additional of the fully convolutional mask branch, the head architecture has 3 branches:
  - The bounding box regression branch
  - The classification branch
  - The mask branch
- So how do these branches work together to make the final output?



- Multiple overlapping bounding boxes will be reduced into 1 bounding box that tries to cover all of the object.
- This is done by the Non-Maximum Suppression algorithm.



Then the bounding box regression head will readjust the
predicted bounding box so that it will fully cover the object.



For each of the region of interest, the classification branch will try to label that region with a class name.

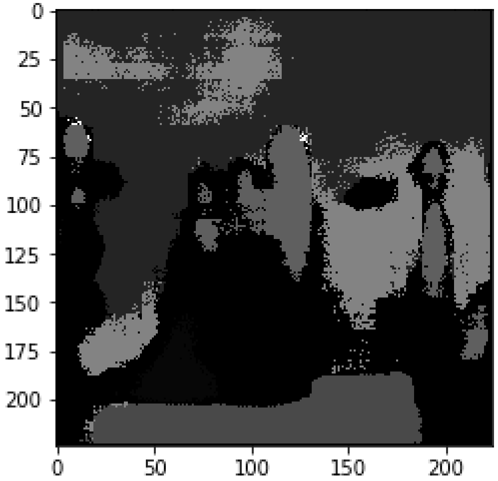
- The mask branch will predict the mask shapes separately for each class.
- Depending on the class label the classification branch predicts, the mask corresponding to that class will be apply on the image.

- At the end, we have 3 separated outputs: the bounding boxes, the labels and the masks.
- The final output is visualized by applying all 3 of the above on the original input image.

6 Achievements

Output comparison

Original



FCN prediction

Mask R-CNN Prediction

Output comparison

Mask R-CNN Prediction

Intersection over Union

$$IoU = \frac{\text{area of overlap}}{\text{area of union}}$$

IoU measures how much the predicted boundary overlaps with the ground truth (the real object boundary).

https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173

Average Precision

AP	$\text{IoU} \geq 0.5{:}\,0.05{:}\,0.95$ (primary challenge metric)
AP_{50}	$\text{IoU} \geq 0.5$ (PASCAL VOC metric)
AP_{75}	$\text{IoU} \geq 0.5$ (strict metric)
AP_{s}	For small objects: $area < 32^2$
AP_{m}	For medium objects: $32^2 < area < 96^2$
AP_{l}	For large objects: $area > 96^2$

Results on Cityscapes dataset

	Training data	AP[val]	AP	AP_{50}	person	rider	car	truck	bus	train	mcycle	bicycle
InstanceCut	fine + coarse	15.8	13.0	27.9	10.0	8.0	23.7	14.0	19.5	15.5	9.3	4.7
DWT	fine	19.8	15.6	30.0	15.1	11.7	32.9	20.4	20.4	15.0	7.9	4.9
SAIS	fine	-	17.4	36.7	14.6	12.9	35.7	23.2	23.2	19.0	10.3	7.8
DIN	fine + coarse	-	20.0	38.8	16.5	16.7	25.7	30.0	30.0	23.4	17.1	10.1
SGN	fine + coarse	29.2	25.0	44.9	21.8	20.1	39.4	33.2	33.2	30.8	17.7	12.4
Mask R-CNN	fine	31.5	26.2	49.9	30.5	23.7	46.9	32.2	32.2	18.6	19.1	16.0
Mask R-CNN	fine + COCO	**36.4**	**32.0**	**58.1**	**34.8**	**27.0**	**49.1**	**40.9**	**40.9**	**30.9**	**24.1**	**18.7**

Cityscapes dataset

Results on Cityscapes dataset

	Training data	AP[val]	AP	AP_{50}	person	rider	car	truck	bus	train	mcycle	bicycle
InstanceCut	fine + coarse	15.8	13.0	27.9	10.0	8.0	23.7	14.0	19.5	15.5	9.3	4.7
DWT	fine	19.8	15.6	30.0	15.1	11.7	32.9	20.4	20.4	15.0	7.9	4.9
SAIS	fine	-	17.4	36.7	14.6	12.9	35.7	23.2	23.2	19.0	10.3	7.8
DIN	fine + coarse	-	20.0	38.8	16.5	16.7	25.7	30.0	30.0	23.4	17.1	10.1
SGN	fine + coarse	29.2	25.0	44.9	21.8	20.1	39.4	33.2	33.2	30.8	17.7	12.4
Mask R-CNN	fine	31.5	26.2	49.9	30.5	23.7	46.9	32.2	32.2	18.6	19.1	16.0
Mask R-CNN	fine + COCO	**36.4**	**32.0**	**58.1**	**34.8**	**27.0**	**49.1**	**40.9**	**40.9**	**30.9**	**24.1**	**18.7**

Comparison in Internal features

	AP^{kp}	AP_{50}^{kp}	AP_{75}^{kp}	AP_{M}^{kp}	AP_{L}^{kp}
RoIPool	59.8	86.2	66.7	55.1	67.4
RoIAlign	**64.2**	**86.6**	**69.7**	**58.7**	**73.0**

RoIAlign *vs.* **RoIPool** for keypoint detection.

	AP_{person}^{bb}	AP_{person}^{mask}	AP^{kp}
Faster R-CNN	52.5	-	-
Mask R-CNN, mask-only	**53.6**	**45.8**	-
Mask R-CNN, keypoint-only	50.7	-	64.2
Mask R-CNN, keypoint & mask	52.0	45.1	**64.7**

Multi-task learning of box, mask, and keypoint about the *person* category

7

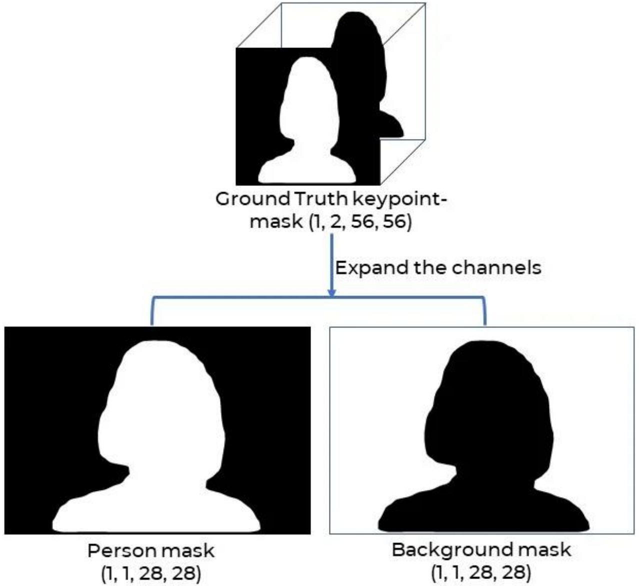
Mask R-CNN for Human Pose Estimation

Keypoint R-CNN

In human pose estimation, a **keypoint represent a body part** (nose, left eye, right elbow, ...) location in the image.
Treat individual keypoint as a one-hot $m \times m$ binary mask. For COCO dataset, 1 person can have up to 17 masks

Keypoint example

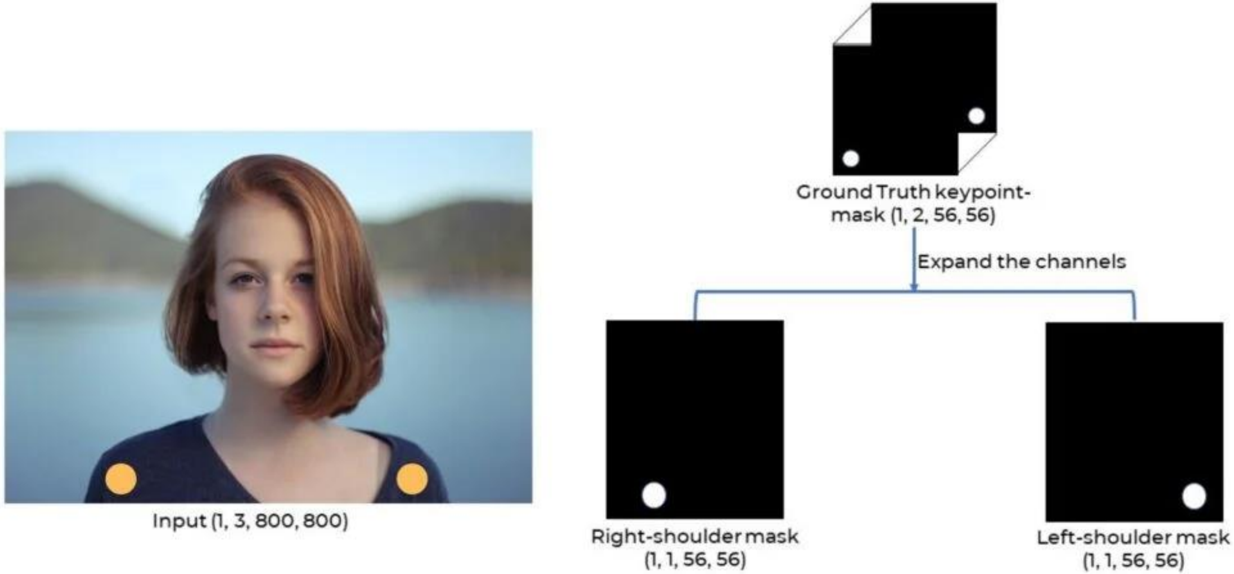
Input (1, 3, 800, 800)



Example output with binary classification

https://learnopencv.com/human-pose-estimation-using-keypoint-rcnn-in-pytorch/

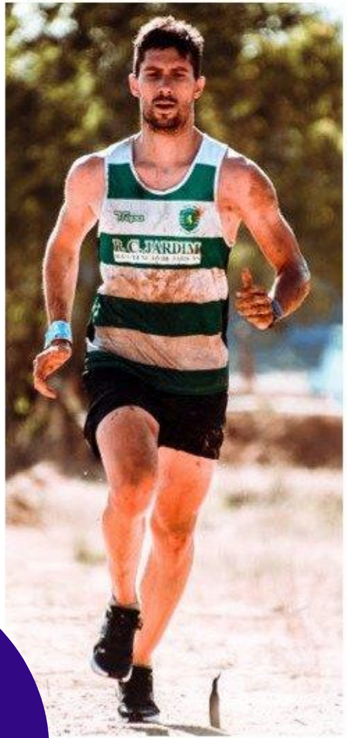
Keypoint example



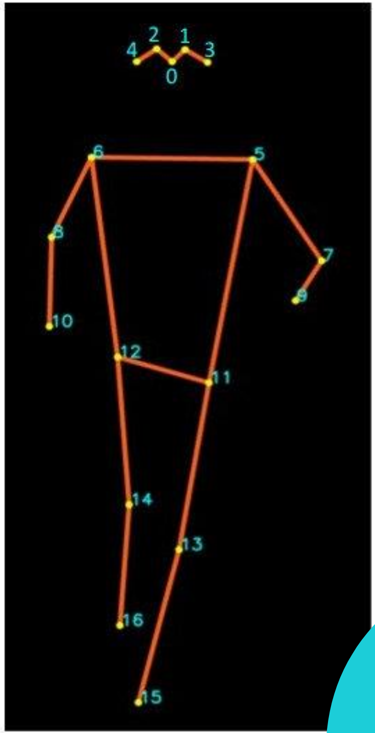
Example output with 2 keypoints

https://learnopencv.com/human-pose-estimation-using-keypoint-rcnn-in-pytorch/

COCO Keypoint



Index	Key point
0	Nose
1	Left-eye
2	Right-eye
3	Left-ear
4	Right-ear
5	Left-shoulder
6	Right-shoulder
7	Left-elbow
8	Right-elbow
9	Left-wrist
10	Right-wrist
11	Left-hip
12	Right-hip
13	Left-knee
14	Right-knee
15	Left-ankle
16	Right-ankle



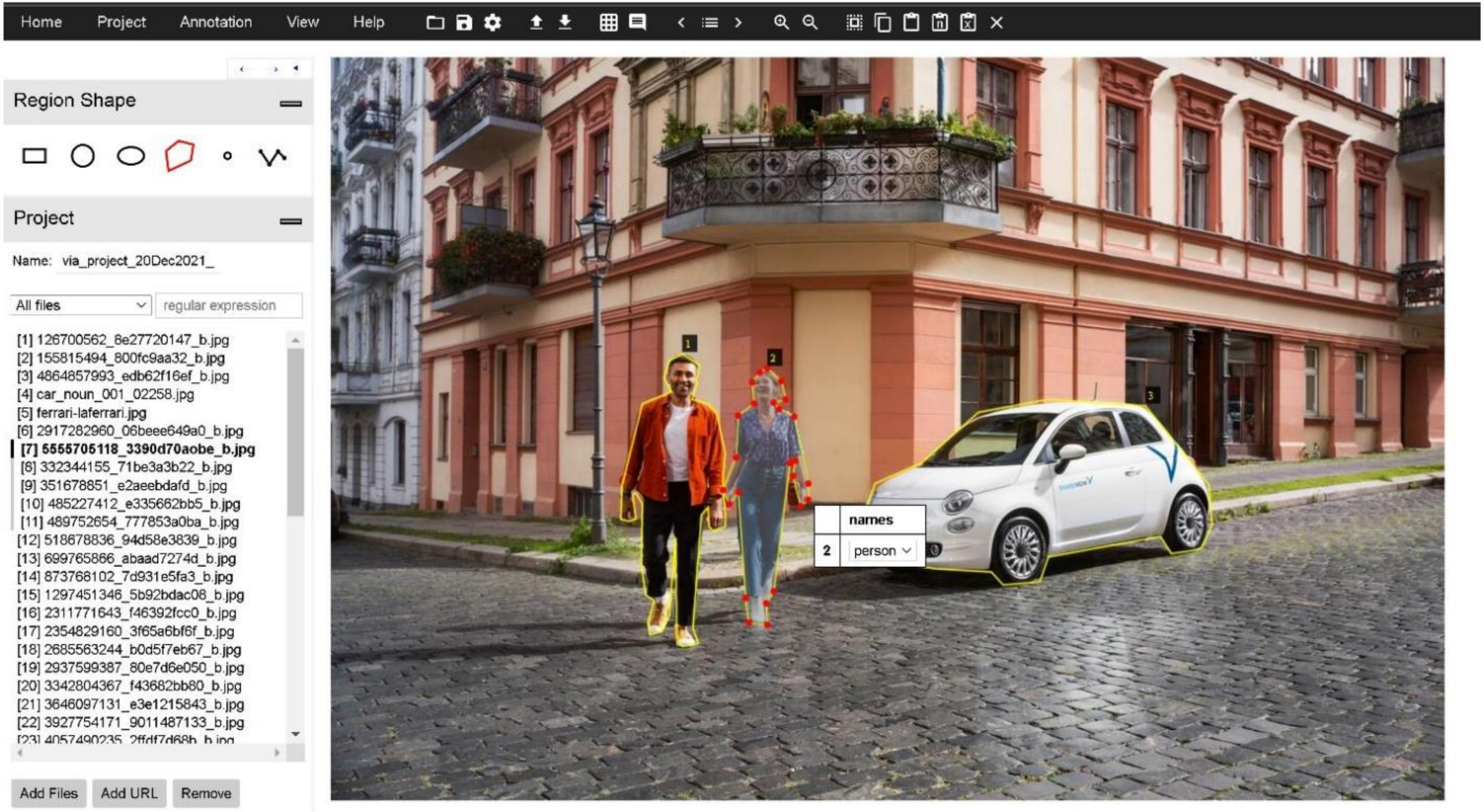
https://learnopencv.com/human-pose-estimation-using-keypoint-rcnn-in-pytorch/

Experiment results

	AP^{kp}	AP^{kp}_{50}	AP^{kp}_{75}	AP^{kp}_{M}	AP^{kp}_{L}
CMU-Pose+++	61.8	84.9	67.5	57.1	68.2
G-RMI	62.4	84.0	68.5	**59.1**	68.1
Mask R-CNN, keypoint-only	62.7	87.0	68.4	57.4	71.1
Mask R-CNN, keypoint & mask	**63.1**	**87.3**	**68.7**	57.8	**71.4**

Mask R-CNN (ResNet-50-FPN) with COCO test-dev

8 Mask RCNN Demonstration


Home
Project
Annotation
View
Help
Region Shape
Project
Name: via_project_20Dec2021_
All files
regular expression
[1] 126700562_8e27720147_b.jpg
[2] 155815494_800fc9aa32_b.jpg
[3] 4864857993_edb62f16ef_b.jpg
[4] car_noun_001_02258.jpg
[5] ferrari-laferrari.jpg
[6] 2917282960_06beee649a0_b.jpg
[7] 5555705118_3390d70aobe_b.jpg
[8] 332344155_71be3a3b22_b.jpg
[9] 351678851_e2aeebdafd_b.jpg
[10] 485227412_e335662bb5_b.jpg
[11] 489752654_777853a0ba_b.jpg
[12] 518678836_94d58e3839_b.jpg
[13] 699765866_abaad7274d_b.jpg
[14] 873768102_7d931e5fa3_b.jpg
[15] 1297451346_5b92bdac08_b.jpg
[16] 2311771643_f46392fcc0_b.jpg
[17] 2354829160_3f65a6bf6f_b.jpg
[18] 2685563244_b0d5f7eb67_b.jpg
[19] 2937599387_80e7d6e050_b.jpg
[20] 3342804367_f43682bb80_b.jpg
[21] 3646097131_e3e1215843_b.jpg
[22] 3927754171_9011487133_b.jpg
Add Files
Add URL
Remove
1
2
3
names
2
person

▶ ferrari-laferrari.jpg48867 {4}
▶ 2917282960_06beee649a0_b.jpg81894 {4}
▼ 5555705118_3390d70aobe_b.jpg265602 {4}
filename : 5555705118_3390d70aobe_b.jpg
size : 265602
▼ regions [3]
▼ 0 {2}
▼ shape_attributes {3}
name : polygon
▶ all_points_x [40]
▶ all_points_y [40]
▼ region_attributes {1}
names : person
▼ 1 {2}
▶ shape_attributes {3}
▶ region_attributes {1}
▼ 2 {2}
▶ shape_attributes {3}
▶ region_attributes {1}

```
config = InferenceConfig()
config.display()
```

```
Configurations:
BACKBONE_SHAPES                [[256 256]
 [128 128]
 [ 64  64]
 [ 32  32]
 [ 16  16]]
BACKBONE_STRIDES               [4, 8, 16, 32, 64]
BATCH_SIZE                     1
BBOX_STD_DEV                   [ 0.1  0.1  0.2  0.2]
DETECTION_MAX_INSTANCES        100
DETECTION_MIN_CONFIDENCE       0.5
DETECTION_NMS_THRESHOLD        0.3
GPU_COUNT                      1
IMAGES_PER_GPU                 1
IMAGE_MAX_DIM                  1024
IMAGE_MIN_DIM                  800
IMAGE_PADDING                  True
IMAGE_SHAPE                    [1024 1024    3]
LEARNING_MOMENTUM              0.9
LEARNING_RATE                  0.002
MASK_POOL_SIZE                 14
MASK_SHAPE                     [28, 28]
MAX_GT_INSTANCES               100
MEAN_PIXEL                     [ 123.7  116.8  103.9]
MINI_MASK_SHAPE                (56, 56)
NAME                           coco
NUM_CLASSES                    81
POOL_SIZE                      7
POST_NMS_ROIS_INFERENCE        1000
POST_NMS_ROIS_TRAINING         2000
ROI_POSITIVE_RATIO             0.33
RPN_ANCHOR_RATIOS              [0.5, 1, 2]
RPN_ANCHOR_SCALES              (32, 64, 128, 256, 512)
RPN_ANCHOR_STRIDE              2
RPN_BBOX_STD_DEV               [ 0.1  0.1  0.2  0.2]
RPN_TRAIN_ANCHORS_PER_IMAGE    256
STEPS_PER_EPOCH                1000
TRAIN_ROIS_PER_IMAGE           128
USE_MINI_MASK                  True
USE_RPN_ROIS                   True
VALIDATION_STEPS               50
WEIGHT_DECAY                   0.0001
```

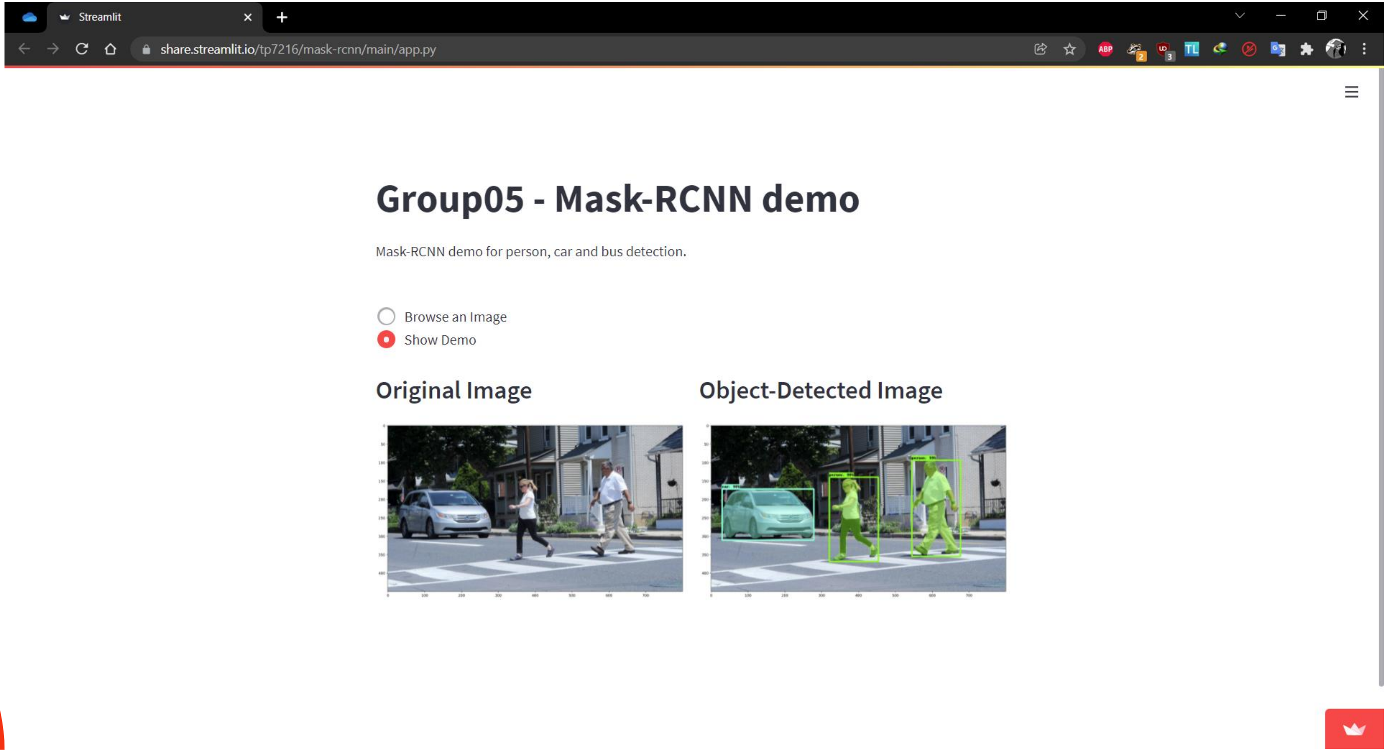
```
Epoch 17/20
10/10 [==============================] - 852s 91s/step - batch: 4.5000 - size: 2.0000 - loss: 0.6003 - rpn_class_loss: 0.0049
- rpn_bbox_loss: 0.2226 - mrcnn_class_loss: 0.0275 - mrcnn_bbox_loss: 0.2090 - mrcnn_mask_loss: 0.1362 - val_loss: 0.6343 - v
al_rpn_class_loss: 0.0057 - val_rpn_bbox_loss: 0.3623 - val_mrcnn_class_loss: 0.0270 - val_mrcnn_bbox_loss: 0.1211 - val_mrcn
n_mask_loss: 0.1181
Epoch 18/20
10/10 [==============================] - 1034s 109s/step - batch: 4.5000 - size: 2.0000 - loss: 0.5002 - rpn_class_loss: 0.00
64 - rpn_bbox_loss: 0.1961 - mrcnn_class_loss: 0.0289 - mrcnn_bbox_loss: 0.1459 - mrcnn_mask_loss: 0.1230 - val_loss: 0.6648
- val_rpn_class_loss: 0.0075 - val_rpn_bbox_loss: 0.2538 - val_mrcnn_class_loss: 0.0336 - val_mrcnn_bbox_loss: 0.2349 - val_m
rcnn_mask_loss: 0.1349
Epoch 19/20
10/10 [==============================] - 913s 96s/step - batch: 4.5000 - size: 2.0000 - loss: 0.6730 - rpn_class_loss: 0.0072
- rpn_bbox_loss: 0.3128 - mrcnn_class_loss: 0.0306 - mrcnn_bbox_loss: 0.2000 - mrcnn_mask_loss: 0.1224 - val_loss: 0.7293 - v
al_rpn_class_loss: 0.0071 - val_rpn_bbox_loss: 0.2751 - val_mrcnn_class_loss: 0.0273 - val_mrcnn_bbox_loss: 0.3166 - val_mrcn
n_mask_loss: 0.1031
Epoch 20/20
10/10 [==============================] - 1018s 108s/step - batch: 4.5000 - size: 2.0000 - loss: 0.5399 - rpn_class_loss: 0.00
60 - rpn_bbox_loss: 0.2215 - mrcnn_class_loss: 0.0237 - mrcnn_bbox_loss: 0.1731 - mrcnn_mask_loss: 0.1156 - val_loss: 0.4462
- val_rpn_class_loss: 0.0055 - val_rpn_bbox_loss: 0.1366 - val_mrcnn_class_loss: 0.0293 - val_mrcnn_bbox_loss: 0.1364 - val_m
rcnn_mask_loss: 0.1384
```

IIBA CANVIO BASICS 500GB (E:) › ml › logs › object20211219T1623

Name	Date modified	Type	Size
plugins	12/19/2021 4:25 PM	File folder	
events.out.tfevents.1639905824.DESKTO...	12/19/2021 8:58 PM	DESKTOP-4UIIT3P...	19,996 KB
events.out.tfevents.1639905905.DESKTO...	12/19/2021 4:25 PM	PROFILE-EMPTY F...	1 KB
mask_rcnn_object_0001.h5	12/19/2021 4:37 PM	H5 File	258,132 KB
mask_rcnn_object_0002.h5	12/19/2021 4:51 PM	H5 File	258,132 KB
mask_rcnn_object_0003.h5	12/19/2021 5:05 PM	H5 File	258,132 KB
mask_rcnn_object_0004.h5	12/19/2021 5:18 PM	H5 File	258,132 KB
mask_rcnn_object_0005.h5	12/19/2021 5:31 PM	H5 File	258,132 KB
mask_rcnn_object_0006.h5	12/19/2021 5:44 PM	H5 File	258,132 KB
mask_rcnn_object_0007.h5	12/19/2021 5:57 PM	H5 File	258,132 KB
mask_rcnn_object_0008.h5	12/19/2021 6:10 PM	H5 File	258,132 KB
mask_rcnn_object_0009.h5	12/19/2021 6:23 PM	H5 File	258,132 KB
mask_rcnn_object_0010.h5	12/19/2021 6:36 PM	H5 File	258,132 KB
mask_rcnn_object_0011.h5	12/19/2021 6:49 PM	H5 File	258,132 KB
mask_rcnn_object_0012.h5	12/19/2021 7:02 PM	H5 File	258,132 KB
mask_rcnn_object_0013.h5	12/19/2021 7:15 PM	H5 File	258,132 KB
mask_rcnn_object_0014.h5	12/19/2021 7:28 PM	H5 File	258,132 KB
mask_rcnn_object_0015.h5	12/19/2021 7:41 PM	H5 File	258,132 KB
mask_rcnn_object_0016.h5	12/19/2021 7:54 PM	H5 File	258,132 KB
mask_rcnn_object_0017.h5	12/19/2021 8:08 PM	H5 File	258,132 KB
mask_rcnn_object_0018.h5	12/19/2021 8:26 PM	H5 File	258,132 KB
mask_rcnn_object_0019.h5	12/19/2021 8:41 PM	H5 File	258,132 KB
mask_rcnn_object_0020.h5	12/19/2021 8:58 PM	H5 File	258,132 KB

```
Processing 1 images
image                    shape: (1024, 1024, 3)       min:    0.00000  max:  255.00000  uint8
molded_images            shape: (1, 1024, 1024, 3)    min: -123.70000  max:  151.10000  float64
image_metas              shape: (1, 15)               min:    0.00000  max: 1024.00000  int32
anchors                  shape: (1, 261888, 4)        min:   -0.35390  max:    1.29134  float32
the actual length of the ground truth vect is :  10
the actual length of the predicted vect is :  10
Average precision of this image :  0.5
The actual mean average precision for the whole images 0.875
Processing 1 images
image                    shape: (1024, 1024, 3)       min:    0.00000  max:  255.00000  uint8
molded_images            shape: (1, 1024, 1024, 3)    min: -123.70000  max:  148.10000  float64
image_metas              shape: (1, 15)               min:    0.00000  max: 1024.00000  int32
anchors                  shape: (1, 261888, 4)        min:   -0.35390  max:    1.29134  float32
the actual length of the ground truth vect is :  12
the actual length of the predicted vect is :  12
Average precision of this image :  0.25
The actual mean average precision for the whole images 0.7857142857142857
Processing 1 images
image                    shape: (1024, 1024, 3)       min:    0.00000  max:  255.00000  uint8
molded_images            shape: (1, 1024, 1024, 3)    min: -123.70000  max:  151.10000  float64
image_metas              shape: (1, 15)               min:    0.00000  max: 1024.00000  int32
anchors                  shape: (1, 261888, 4)        min:   -0.35390  max:    1.29134  float32
the actual length of the ground truth vect is :  15
the actual length of the predicted vect is :  15
Average precision of this image :  0.25
The actual mean average precision for the whole images 0.71875
Processing 1 images
```

Deploy model on Streamlit



References

- Mask R-CNN on Keras and TensorFlow (github.com) [https://github.com/matterport/Mask_RCNN]
- facebookresearch/Detectron (github.com) [https://github.com/facebookresearch/Detectron]
- Mask R-CNN Explained | Papers With Code [https://paperswithcode.com/method/mask-r-cnn]
- Human Pose Estimation using Keypoint RCNN [https://learnopencv.com/human-pose-estimation-using-keypoint-rcnn-in-pytorch/]
- Human Pose Detection – DebuggerCafe [https://debuggercafe.com/human-pose-detection-using-pytorch-keypoint-rcnn/]