YOLACT

Real-time Instance Segmentation

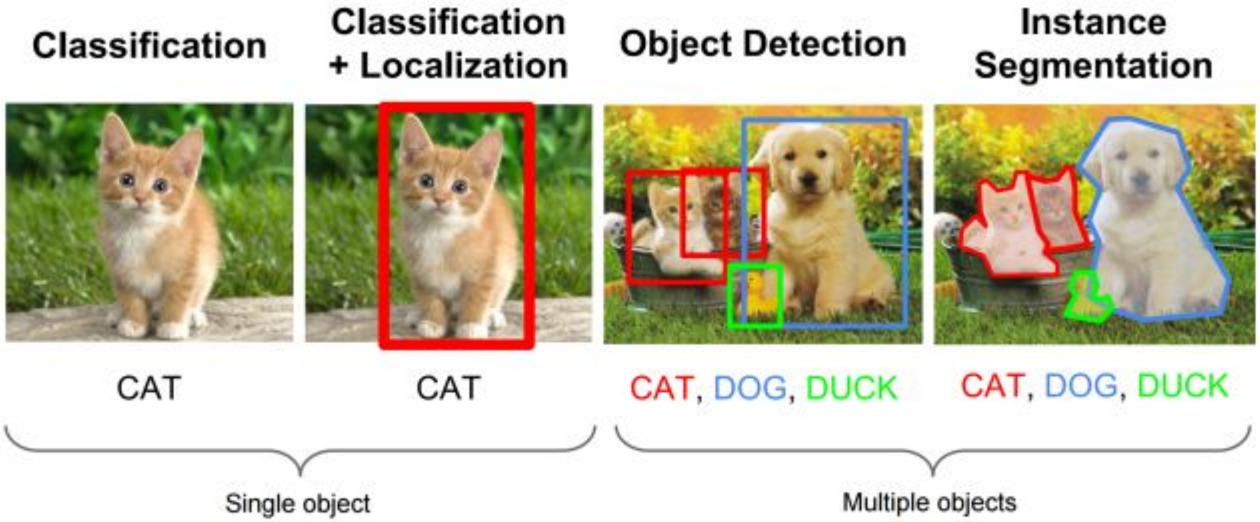
Daniel Bolya, Chong Zhou, Fanyi Xiao, Yong Jae Lee

Outline

- Background
- Instance Segmentation
- Real-time instance segmentation and its complexity
- Introduction to YOLACT
- Architecture of YOLACT
- Improvements
- Results
- Summary

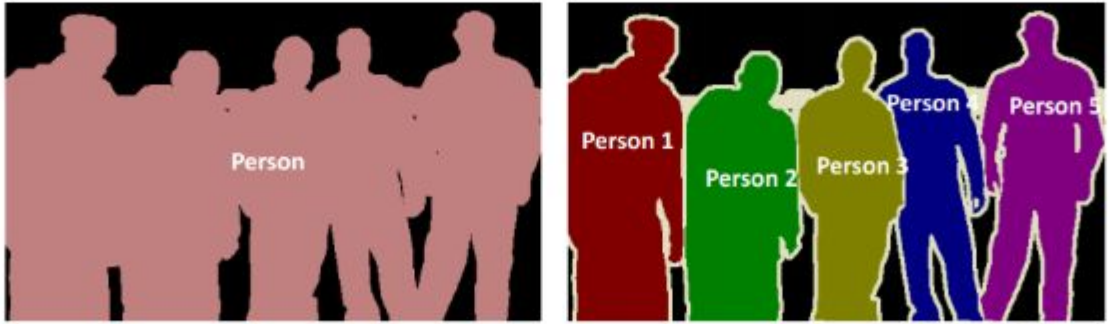
Types of Visual Recognition Tasks

1. Classification Task
2. Object Detection Task
3. Segmentation Task



What makes it complex?

- The task is to identify the different classes of instances in the image as well as the different instances belonging to each class.
- Way more complex than segmentation task which only needs to identify what pixels belong to what classes.



Semantic Segmentation   Instance Segmentation

Instance Segmentation

Classify every pixel in the image to a class such that each pixel is assigned to an instance.

Instance Segmentation in Real-time

- Steady improvement in field of instance segmentation over the years.
- Previous models aim for accuracy over speed.
- No practically usable real time model until YOLACT.

2-Stage Models

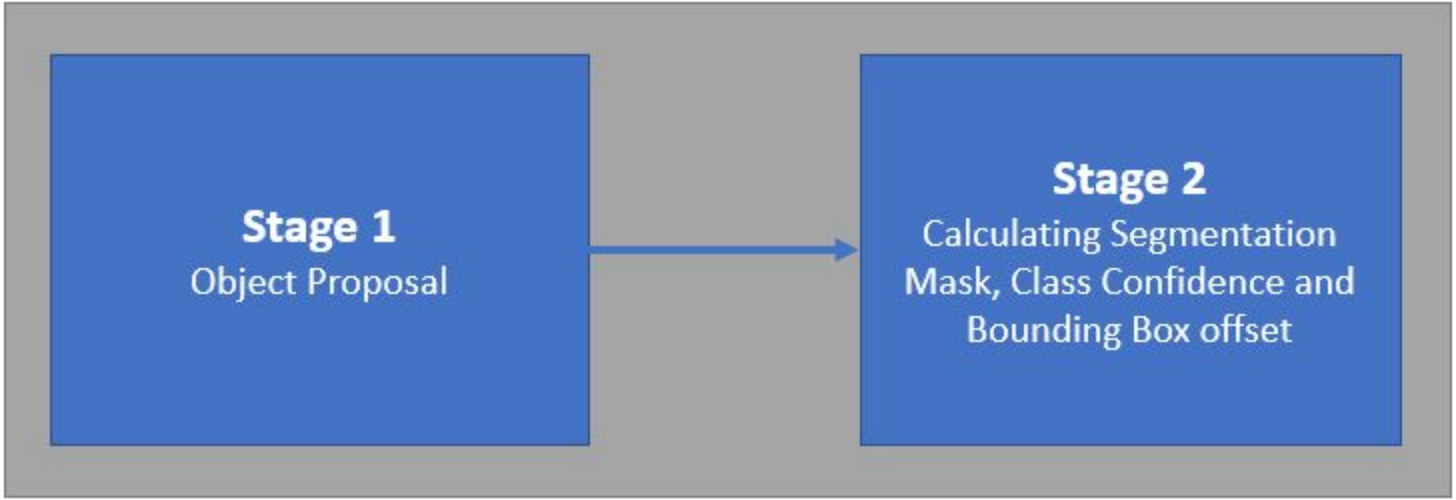


Fig. Working of Mask R-CNN

YOLACT

- Simple, fully-convolutional model

- 29.8 *mAP* accuracy with 33 *fps* on MS COCO dataset.

- Breaks down instance segmentation into parallel subtasks for fast performance
    - Generating a set of prototype masks
    - Predicting per-instance mask coefficients

Prototype Masks

- Think of prototypes as parts of a whole mask.
- Each prototype has a unique behavior.
    - It may localize instances.
    - It may find edges and contours.
    - It may do a combination of these tasks.

Why is YOLACT faster ?

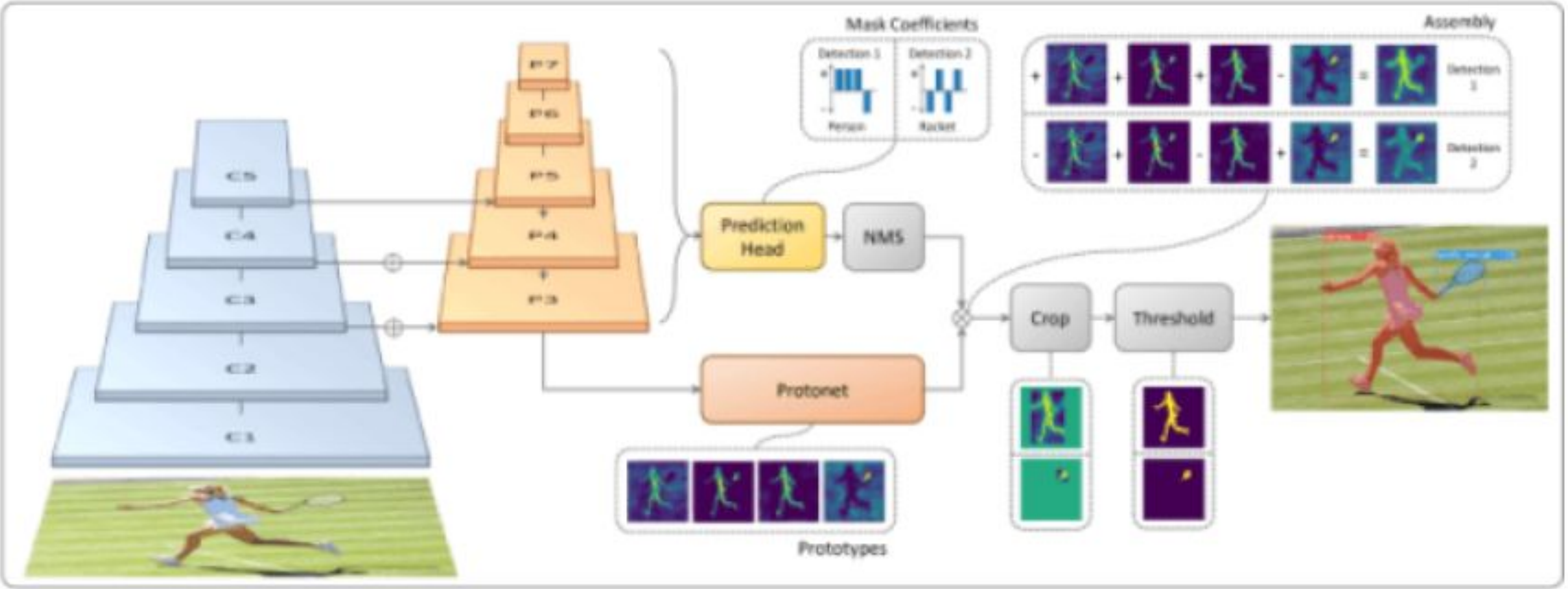
- Single Stage Model

- Prototype Masks and Mask Coefficients are calculated parallelly and independently.

- Other methods have an explicit localization step (Ex : ROIAlign in Mask R-CNN)

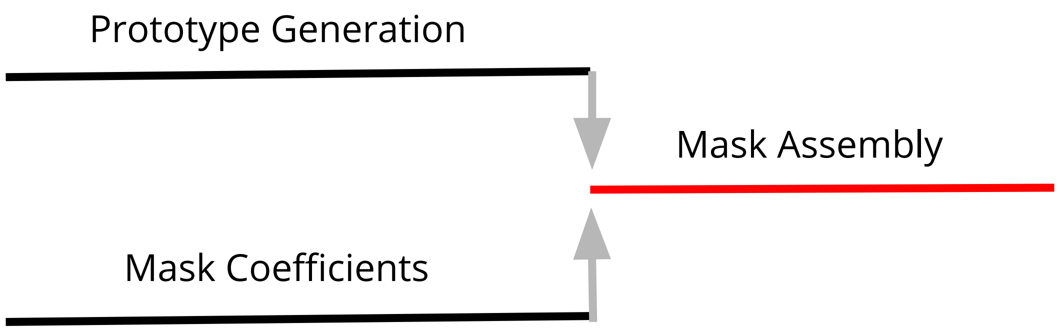
- YOLACT learns about localizing instances by itself and bypasses the explicit localization step.

Backbone Detector



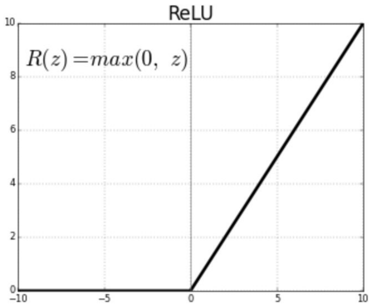
Backbone Detector

- We can use any type of backbone detector
- Here we have used ResNet-101 with FPN (Feature Pyramid Network) as backbone
- Base size image is 550 x 550

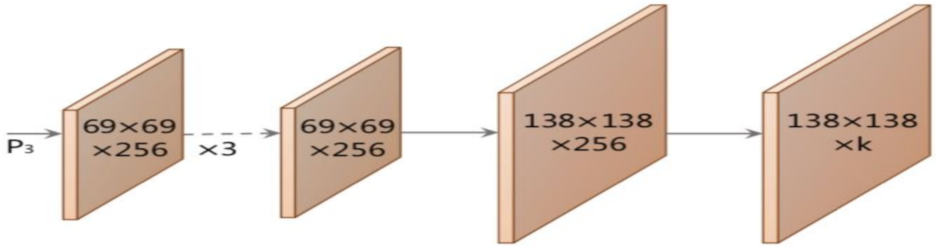

Prototype Generation
Mask Assembly
Mask Coefficients

Step 1 : Prototype Generation

- Also known as Pronet
- It is a Fully Convolutional Network
- Last layer produces k prototypes
- No explicit loss
- We leave the prototypes output being unbounded



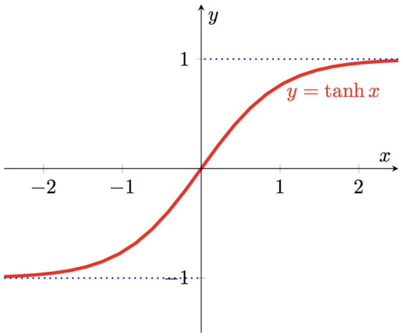
ReLU Activation Function

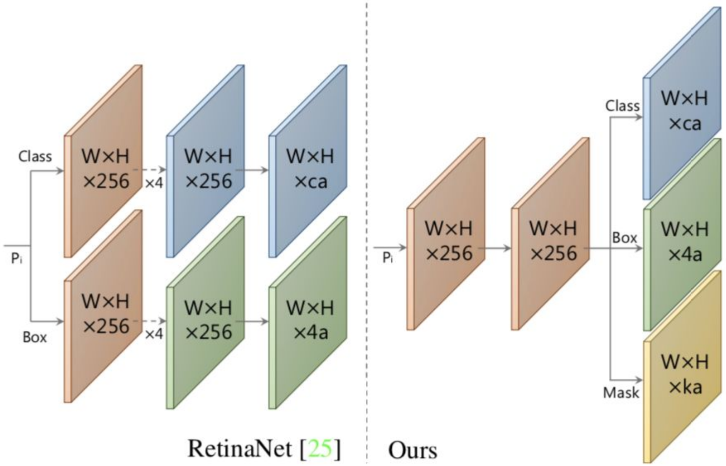


Step 1 : Mask Coefficients (In Parallel)

- Produces c + 4 + k coefficients

- Use tanh on k mask coefficients





Step 2 : Mask Assembly

- M => Mask
- P => Prototype (h x w x k)
- C => Mask Coefficients (n x k)
- Sigmoid

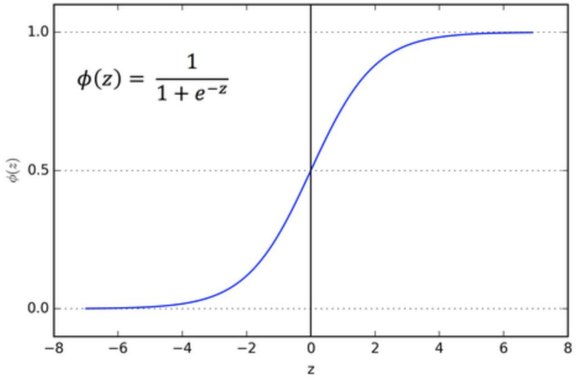


Fig: Sigmoid Function

$$M = \sigma(PC^T)$$

Losses

Three types of losses:

- Classification Loss

$$L_{conf}(x,c) = -\sum_{i\in Pos}^{N} x_{ij}^{p} log(\hat{c}_i^p) - \sum_{i\in Neg} log(\hat{c}_i^0) \quad \text{where} \quad \hat{c}_i^p = \frac{\exp(c_i^p)}{\sum_p \exp(c_i^p)}$$

Confidence Loss

- Box Regression Loss

$$L_{1;smooth} = \begin{cases} |x| & \text{if } |x| > \alpha; \\ \frac{1}{|\alpha|}x^2 & \text{if } |x| \leq \alpha \end{cases}$$

$$L_{loc}(x,l,g) = \sum_{i\in Pos}^{N} \sum_{m\in\{cx,cy,w,h\}} x_{ij}^{k} \mathrm{smooth}_{\mathrm{L1}}(l_i^m - \hat{g}_j^m)$$

$$\hat{g}_j^{cx} = (g_j^{cx} - d_i^{cx})/d_i^w \qquad \hat{g}_j^{cy} = (g_j^{cy} - d_i^{cy})/d_i^h$$

$$\hat{g}_j^w = \log\left(\frac{g_j^w}{d_i^w}\right) \qquad \hat{g}_j^h = \log\left(\frac{g_j^h}{d_i^h}\right)$$

Localization Loss

Losses (Continue...)

- Mask Loss - Pixel-wise Binary Cross Entropy

  Mgt: Lmask = BCE(M, Mgt).

Cropping Masks

During Evaluation :

- We crop final masks with the predicted bounding box

During Training:

- We crop with the ground truth bounding box and divide Lmask by the ground truth bounding box area

Improvements

- Increase Speed with little effect on performance
- Increase Performance with no speed penalty

NMS (Non Maximum Suppression)

- Most Object Detectors uses traditional NMS or Sequential NMS
  - Makes sure each object is detected only once
  - Sort the detected boxes in descending order by Confidence
  - Discard values less than a certain threshold
  - Discard the values greater than threshold IoU values

- Fast-NMS
  - A new version of NMS
  - Decides to either discard or keep parallely
  - Allows already removed detections to suppress other detections

Fast-NMS

- First we compute a pairwise IoU matrix (X) using,
    - c*n*n
    - c = classes
    - n = top n detections sorted in descending order

- Second, remove detections for
    - Confidence values less than a threshold ‘t’
    - IoU values greater than a threshold ‘t’

Fast-NMS contd..

- Second step implemented using:
  - Setting lower triangle and diagonal of X to be 0.

$$X_{kij} = 0 \qquad \forall k, j, i \geq j$$

    - Where
      - X = pairwise IoU matrix
  - Taking the column-wise max
    - Where K = Matrix of maximum IoU values

$$K_{kj} = \max_i (X_{kij}) \qquad \forall k, j$$

- Detections to keep given by threshold matrix t (K<t)

Semantic Segmentation Loss

- Increases performance with no speed penalty
- Authors attach a 1x1 conv layer with c output channels to the largest feature map P3 in the backbone
- Use of a sigmoid and c channels instead of a softmax
- Training with this loss resulted in +0.4 mAP boost

Results

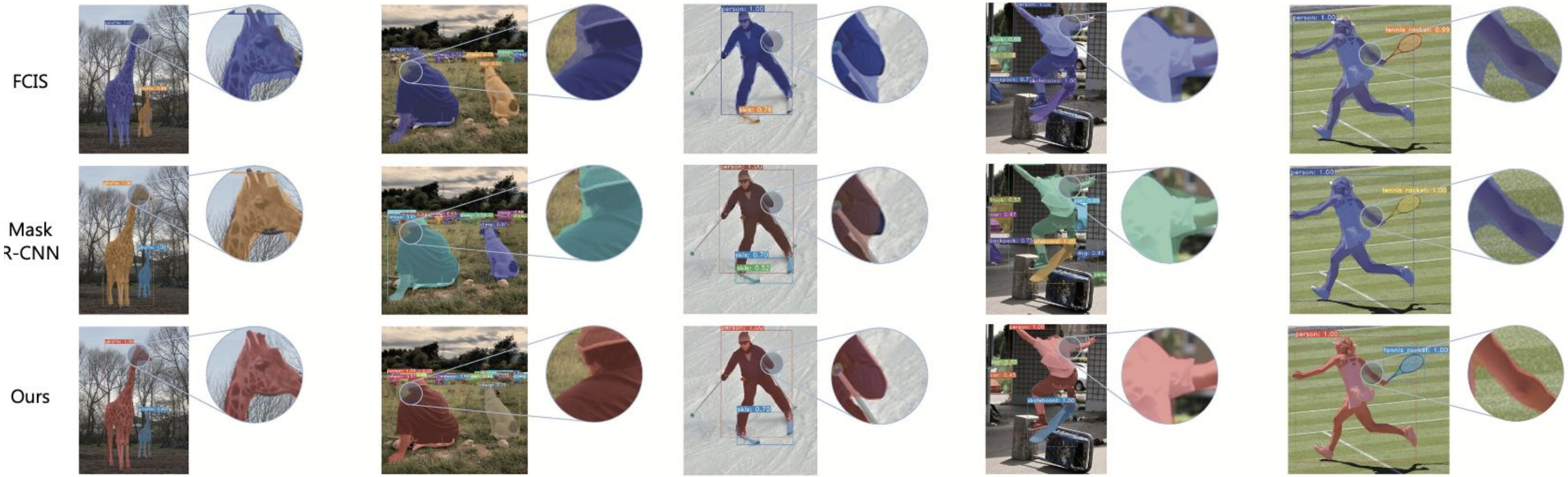
- Results were reported on MS COCO instance segmentation task

- Training was done on train2017

- Evaluation on val2017 and test-dev

Method	Backbone	FPS	Time	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
PA-Net [27]	R-50-FPN	4.7	212.8	36.6	58.0	39.3	16.3	38.1	53.1
RetinaMask [12]	R-101-FPN	6.0	166.7	34.7	55.4	36.9	14.3	36.7	50.5
FCIS [22]	R-101-C5	6.6	151.5	29.5	51.5	30.2	8.0	31.0	49.7
Mask R-CNN [16]	R-101-FPN	8.6	116.3	35.7	58.0	37.8	15.5	38.1	52.4
MS R-CNN [18]	R-101-FPN	8.6	116.3	**38.3**	58.8	41.5	17.8	40.4	54.4
YOLACT-550	R-101-FPN	**33.0**	**30.3**	29.8	48.5	31.2	9.9	31.3	47.7
YOLACT-400	R-101-FPN	44.0	22.7	24.9	42.0	25.4	5.0	25.3	45.0
YOLACT-550	R-50-FPN	42.5	23.5	28.2	46.6	29.2	9.2	29.3	44.8
YOLACT-550	D-53-FPN	40.0	25.0	28.7	46.8	30.0	9.5	29.6	45.5
YOLACT-700	R-101-FPN	23.6	42.4	31.2	50.6	32.8	12.1	33.3	47.1

Mask Quality



Trade-off

- Lowering of image size :
  - Decreases the performance
  - Increases the speed

- Increasing the image size:
  - Increases the performance
  - Decreases the speed

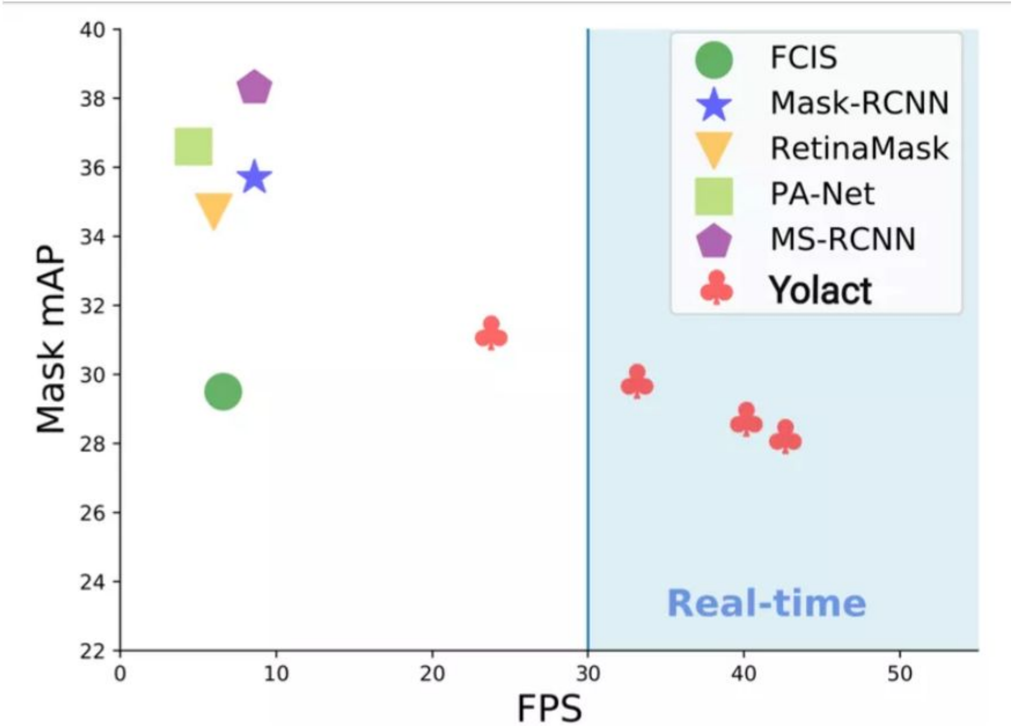
- Paper suggests using ResNet- 50 or DarkNet-53 to increase speeds.

Limitations

- Localization Failure
  - Too many objects cause the network to fail to localize each object
  - Network outputs something closer to the foreground mask

- Leakage
  - Network does not suppress noise outside boundary box
  - Inaccuracy of boundary box causes leakage
  - Also happen when two instances are away from each other

Summing it up



Strengths

- Fast, Real-time model.
- Achieves real-time performance while having comparable accuracies.
- Produces high quality masks.
- Shows that the model can learn about inherent behavior without being aimed at doing so.

Weaknesses

- Performance vs Speed Tradeoff.
- Might fail to segment images with too many instances in one spot.
- Leakage issue leading to extra noise.

Questions?