YOLO Based Instance Segmentation

Siddarth Reddy IMT2016037
Srujan Swaroop IMT2016033
Rohith Yogi IMT2016072
Durga Yasasvi IMT2016060

Abstract: We present a YOLO based instance segmentation, a simple, flexible and scalable approach for object detection with instance segmentation. This approach requires a sequential based framework where the first component of the architecture is designed to detect and classify objects using YOLO framework, which predicts the bounding boxes and class probabilities in single evaluation, followed by a network to assign a high quality mask for each instance of the class in the image based on prior classification. Our method extends YOLO by adding a branch for predicting an object mask and a CNN for obtaining the feature map which is required to generate localized features for mask network.

1. Introduction

To develop a semantic understanding of an image the primary level of actions include:-

- Object Detection and localization of the detected image.
- Associating a particular class to the detected objects(Classification of the object).
- Semantic segmentation (Process of linking each pixel in an image to it's class label).

These actions can provide information of the image on a higher level and in an abstract sense, But if we want to know where exactly a particular instance of any object is located in an image and to know its exact geographical structure and physicality we need to perceive the image with granular level of understanding and the pixel level structure of the image with specifications should be considered,which is where the concept of **instance segmentation** arises.

So here we propose a paradigm for instance segmentation, which by design also performs object detection and classification. Instance segmentation is definitely not a very easy problem to tackle when compared to actions like object detection or classification,because this it in turn demands some of these actions to be performed as intermediate steps in the process.
Some of them are :-

- Every object present in the image should be detected and localized in the form of boundary boxes.
- Categorization of the object classes as the instances should be done prior, as segmentation takes it to consideration.

Once the classes of each object are identified and localized using bounding box, then the task simply reduces to pixel level classification. Each pixel in the boundary box should be classified whether it belongs to the object or not, so the precise locations of the object in the boundary box should be known.

The approach we came up with is a simple and straight forward formulation which solves the above mentioned issues in a sequential manner . The choices adapted by us are as follows:-

- We decided to use YOLO framework for object detection, localization and classification of the detected objects. This is an end to end model which performs this actions at real time with 155 frames per second and 52.7mAP.
- We designed a neural network which generates high quality binary segmentation masks for

each object detected from YOLO. The masks here refer to a binary matrix with 1^s and 0^s,where 1 represents that corresponding pixel in the image belongs to the object and 0 represent that the corresponding pixel doesn't belong to the object. This is referred as Mask segmentation Network.

The Mask segmentation Network is built on top of YOLO framework because, this architectural decision was made to incorporate the fact that the output generated from YOLO is a potential and primary input for binary mask segmentation network. So to capture this high level of dependency on YOLO this structural design was implemented.

Since structure of binary mask segmentation network is standardized it is trained to take input of a fixed size and outputs a fixed size matrix. Hence the pixel correspondence between the matrix and actual region should be maintained. Since their sizes are not always integral multiples of each other the size of corresponding feature map shouldn't be discretized as pixel level accuracy should be preserved, because discretization introduces pixel misalignment which obliges to compromise on exact correspondence on scaling. Hence we adapt a method called ROI-align which preserves exact spatial correspondence between the region of the image and the binary segmentation mask.

2. YOLO

YOLO stands for **Y**ou **O**nly **L**ook **O**nce. YOLO is a real time object detection and classification framework. YOLO solves object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. Unlike complex pipelines like RCNN this is a single network and can be optimized as an end to end directly on detection performance. When compared to state of the art networks for object detection YOLO is less likely to predict false positives in the background, but it makes more localization errors.

Since YOLO is framed as a regression problem it does not have all the complex pipelines so it is extremely fast, it processes images at 45 fps. It trains on full images and optimizes the object detection performance. So it is used in real time applications as we simply run the network on the new image during test time and predict the results using image pixels, bounding box coordinates, and class probabilities. Unlike RCNN, YOLO sees the entire image during training and testing. YOLO makes fewer background errors than RCNN. When applied to new domains or unexpected inputs, YOLO is highly generalizable it is less likely to break down.

YOLO uses features from the entire image to predict each bounding box. It divides the input image into an $S * S$ grid. If the centre of an object falls into a grid cell, then that grid cell is responsible for detecting that object. Each grid cell predicts B number of bounding boxes and respective confidence scores. The confidence score of a bounding box indicate the chance of presence of an object in that box. It is defined as $Pr(Object) * IoU^{truth}_{pred}$. And, each bounding box B_i consists of 5 prediction parameters: $x, y, h, wand$ confidence. Confidence prediction shows the presence of an object in that bounding box. It also represents the IoU between predicted bounding box B_i and the ground truth. Each grid cell is also responsible for predicting conditional class probabilities(C), which is given an object find the probability that the object belongs to the class i, $Pr(Classi/Object)$. These class probabilities shows that the chance that the grid cell belong to a particular class. In order to get the probability of a class appearing in that bounding box, it can be obtained by

$$Pr(Classi/Object) * Pr(Object) * IoU$$

. This score also encodes how well the predicted box fits the object. $Fig-1$ shows the high level picture of what predictions does the network output. Class probability map tells that each cell was given a color based on its class probabilities predicted. It is obvious that, neighbouring pixels have a more chance of similar color (here color represents the class label). At this stage for each grid cell we have B bounding boxes and its class probabilities, and each of these bounding boxes

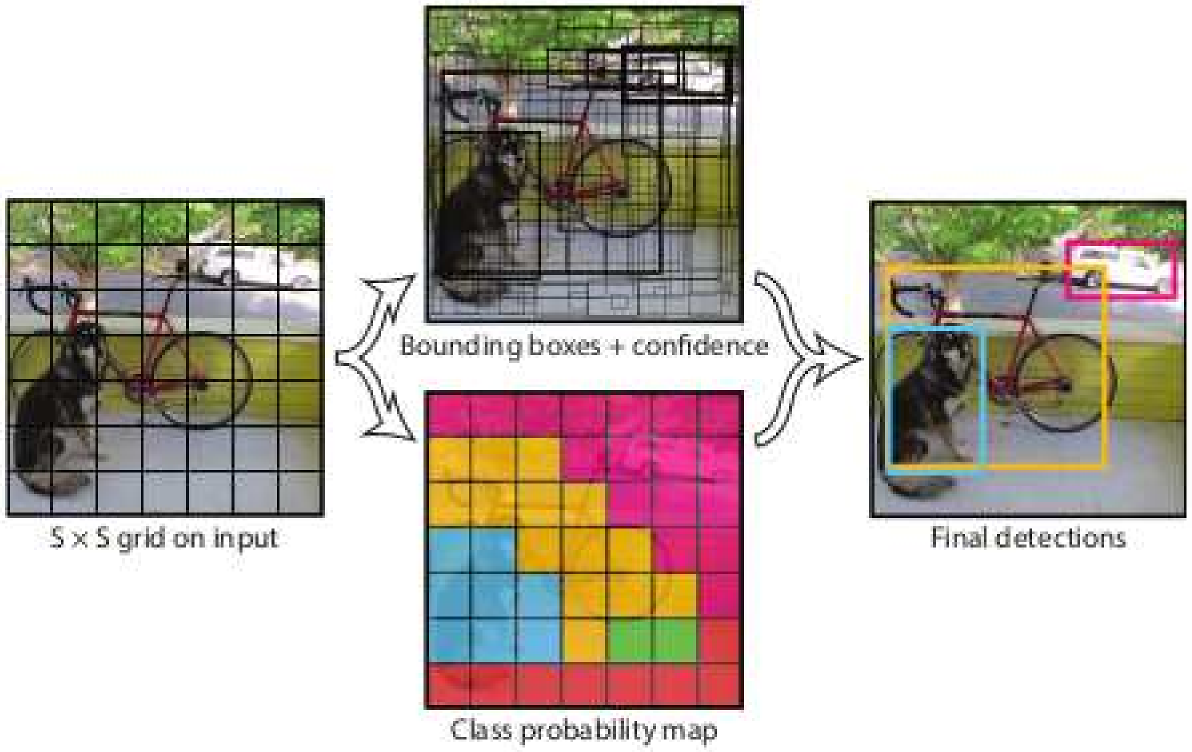

Fig. 1. Predictions are encoded as an $S*S*(B5+C)$ tensor.

carries 5 parameters x, y, h, w and confidence score. All these parameters are sent to a CNN and trained as a regression problem. For ex: An object occupies K number of grid cells on an input image, and each grid cell now proposes K*B number of bounding boxes and each bounding box having those 5 parameters. Now all these are together are responsible for tightening the box and later outputs the coordinates which are encoded in a tensor format. Later this is passed into non maximum suppression technique which is part of the post processing and finally outputs the final detection.

All the predictions are encoded as $S*S*(B*5+C)$ tensor. On PASCAL VOC dataset, used $S=7, B=2$ and $C=20$ classes. Hence, the output tensor would be $7*7*30$ tensor.

The network design is inspired from GoogleNet. Has 24 convolutional layers, followed by 2 fully connected layers. First 24 layers are responsible for extracting features and last 2 layers predict the output probabilities. First 20 convolutional layers are pre-trained on the Imagenet classification task at a resolution of $224*224$ input image. For detection, the input size is doubled. The final output of the network is $7*7*30$ tensor.

For training and inference, used Darknet. Normalized the variables h, w, x, y to 0 and 1. Linear activation function for final and all other layers used $LeakyReLU$. Error: Sum squared. This error weighs localization error equally with classification error. Grid cells may or may not contain an object in it. Which causes the confidence score to be zero. This causes the model to be unstable and may diverge quickly.So they increased the loss from bounding box coordinate predictions and decreased the loss from confidence predictions for boxes that dont contain objects.For this they introduced two parameters, λ_{coord} and λ_{noobj}. And set them to 5 and 0.5 respectively. This makes the model acquire a little more loss for a bounding box without an object containing in it.

YOLO predicts multiple bounding boxes per grid cell. But we only want one bounding box to

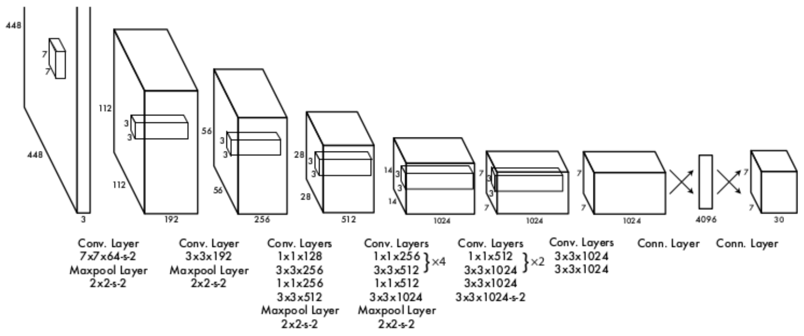

Fig. 2. The YOLO Architecture

be assigned per object.So they Assigned a predictor which is responsible for predicting an object based on the prediction that has the highest IoU.

Model is optimized using the Loss Function:

$$\lambda_{coord}\sum_{i=0}^{S^2}\sum_{j=0}^{B}1_{ij}^{obj}[(x_i-\hat{x}_i)^2+(y_i-\hat{y}_i)^2]+\lambda_{coord}\sum_{i=0}^{S^2}\sum_{j=0}^{B}1_{ij}^{obj}[(\sqrt{w_i}-\sqrt{\hat{w}_i})^2+\sqrt{h_i}-\sqrt{\hat{h}_i})^2]$$

$$+\sum_{i=0}^{S^2}\sum_{j=0}^{B}1_{ij}^{obj}[(C_i-\hat{C}_i)^2+\lambda_{noobj}\sum_{i=0}^{S^2}\sum_{j=0}^{B}1_{ij}^{obj}[(C_i-\hat{C}_i)^2+\sum_{i=0}^{S^2}\sum_{j=0}^{B}1_{i}^{obj}[(p_i(c)-\hat{p}_i(c))^2$$

Loss function penalizes classification error only if an object is present in that grid cell. And it also only penalizes bounding box coordinate error if that predictor is **responsible** for the ground truth box (highest IoU in that grid cell).

The grid is designed to enforce spatial diversity in the bounding box predictions. Non-maximal suppression (described in the design section-4) is used to fix the multiple detections. This also adds 2-3% mAP.

There are few limitations for YOLO. As each grid cell only predicts two bounding boxes and can only have one class, it imposes spatial constraints on bounding box predictions. It also struggles with small objects.The model also has multiple down sampling layers so it uses relatively coarse features for predicting bounding boxes. The training on a loss function treats the errors the same in small bounding boxes and also the same case with the large bounding boxes as well. A small error in a small box has a much greater effect on IoU, and a small error in a large box is generally fine. The main source of error lies in incorrect localizations.

But YOLO is fast and detects objects accurately which is ideal for computer vision applications.

3. Mask R-CNN

This model is used as a general framework for instance segmentation. It efficiently detects the object and generates a high quality mask for each instance. Basically, this method extends Faster R-CNN by adding a branch parallel to bounding box regression and object classification for predicting an object mask. The added branch(mask branch) is a small fully convolutional layer which when applied to each ROI obtained from RPN network, predicts a segmentation mask in pixel to pixel manner. In Faster R-CNN we have ROI-Pooling to convert the corresponding feature map of each ROI to a fixed size so that we can send them to the FCs. The ROI-Pooling basically performs a non-uniform quantization where some pixel details are lost. But, for instance segmentation we need to preserve exact spatial locations to achieve it. To fix this misalignment, they came up with a method called ROI-Align.

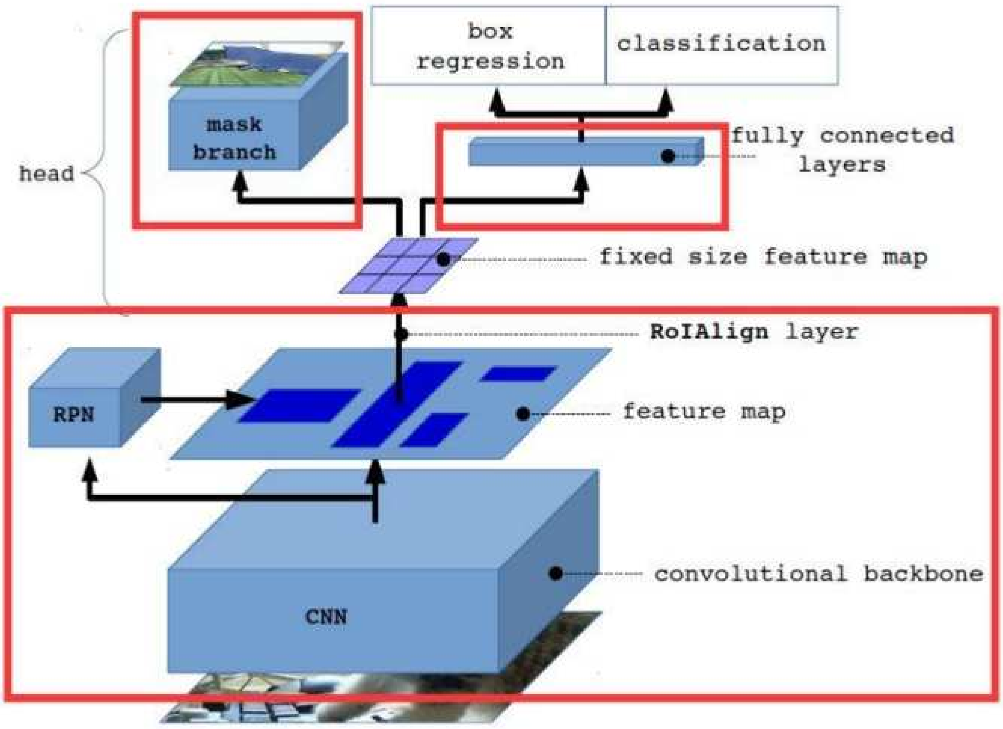

Fig. 3. Architecture of Mask R-CNN

In this Mask RCNN, there is no dependency between masking an ROI and class prediction. The method predicts the binary mask independently for each class without competition with other class. Then from ROI classification branch, we get the class of the object in the corresponding ROI. By experiments, it was observed that FCNs(Fully Convolutional Network) which usually perform per-pixel multi-class categorization which couples segmentation and classification performs poorly for instance segmentation.

3.1. Architecture

- Stage 1
  - Regional Proposal Network
- Stage 2
  - ROI-Align
  - Binary Mask Prediction

Regional Proposal Network (RPN):

First we have to send the feature map we obtained from pre-trained convolutional network as input to this RPN. Feature map will be of size $M \times N \times C$ where M,N are with respect to image(that is resized because of pooling layers) and C is number of kernels used.

The RPN is usually a simple network with a 3 convolutional layers. There is one common layer which feeds into a two layers, one for classification and the other for bounding box regression. We use a 3 $\times$ 3 sliding window to traverse the feature map. While traversing we consider k anchor boxes at center of each sliding window.

But, why use k anchor boxes??

- Because objects can in different shapes, and thus by using different anchor boxes we can find a better bounding box match to objects.

In this paper k is considered 9. For anchors, we use 3 scales with box areas of 128^2, 256^2, 12^2 pixels and 3 aspect ratios of 1:1, 1:2 and 2:1. These anchor co-ordinates will be with respect to the image. Classifier in RPN network determines the probability of object and not object for each proposal and Regression adjusts the coordinates of the proposals. Thus, the size of bounding box regression will be 4k(4 values x,y,h,w with respect to the image) and the size of class prediction will be 2k(probabilities of anchor containing object and not containing object). For a convolutional feature map of a size $W \times H$ (typically 2,400),there are $W \times H \times k$ anchors in total.
These anchors are assigned label based on these factors:
Positive label(that is containing the object)

- The anchors with highest Intersection-over-union (IoU) overlap with a ground truth box.
- The anchors with Intersection-Over-Union Overlap higher than 0. 7.

Negative label(that is not containing the object)

- We assign a negative label to a non-positive anchor if its IoU ratio is lower than 0. 3 for all ground-truth boxes.

Some RPN proposals highly overlap with each other. To reduce redundancy, we adopt non-maximum suppression (NMS) on the proposal regions based on their cls scores. We fix the IoU threshold for NMS at 0. 7, which leaves us about 2000 proposal regions per image.

ROI-Align:

- After, we get the bounding boxes from RPN network we have to send these as an input to FCs. But FCs require fixed size inputs but our proposed regions can be of different sizes, so we have to resize all of them to a fixed size. Here we cannot use ROI-Pooling as it uses quantization where we loose some information about spatial alignment(that is we loose per-pixel spatial correspondance. Therefore they came up with a method to prevent it called ROI-Align

First while extracting corresponding feature map of each bounding box we do not round off like [x/16] but we actually consider (x/16) that is floating number like shown in below figure 4. So ,our question is what is pixel value at 2.93? For this we look at image given in the Mask-RCNN paper. let's assume that for an ROI we got corresponding feature map size as $f \times j$ where f and j are float. We overlap our co-ordinates on original feature map we get our boundaries at middle of grids. We want a fixed size output from ROI-Align so let's say that as $k \times k$. So divide our $f \times j$ in to $k \times k$ grids. From each grid lets take four sampling points and we find the value at these by using bi-linear interpolation from nearby grid points on the feature map. Then we take maximum or average of four values and represent as value in our $k \times k$ matrix. Thus by this method we find the fixed size input for FCs.No quantization is performed on any coordinates involved in the ROI, its bins, or the sampling points. Thus, there is no misalignment in pixel to pixel correspondance.

Binary mask prediction:

Here mask RCNN outputs a binary mask for each ROI. In this matrix '0' represent not belonging to the object and '1' represents that pixel belongs to the object present in that ROI. During training, they defined multi-task loss on each sampled ROI as
$L = L_{cls} + L_{box} + L_{mask}$

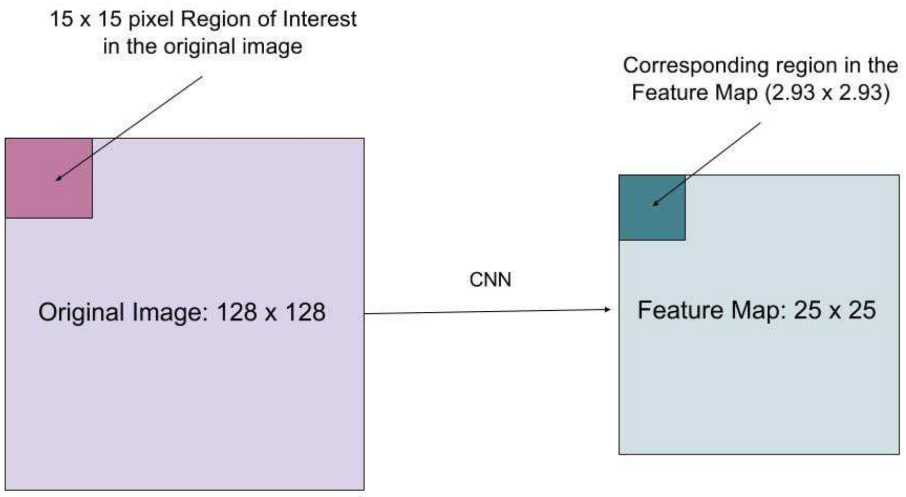
15 x 15 pixel Region of Interest in the original image
Corresponding region in the Feature Map (2.93 x 2.93)
Original Image: 128 x 128
CNN
Feature Map: 25 x 25

Fig. 4. How do we accurately map a region of interest from the original image onto the feature map?

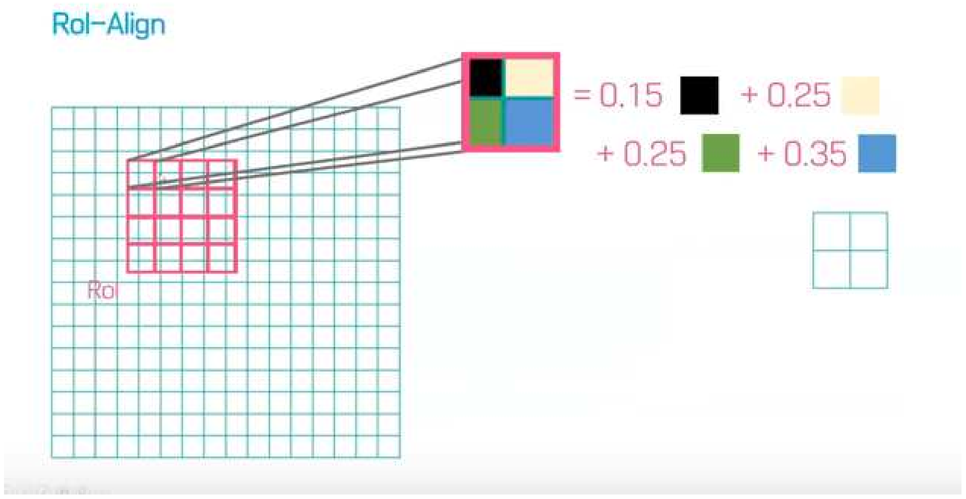

Fig. 5. How do we accurately map a region of interest from the original image onto the feature map?

The classification and regression loss are same as in Faster-RCNN but here we also add loss related to the masking which is calculated per pixel by applying sigmoid on each pixel. The mask branch outputs Km^2 matrix which is binary matrix for each of the K classes. But while considering loss for mask we only consider matrix of the class which is ground truth(other masks do not contribute to loss). By this definition L_{mask} allows the network to to generate masks for every class without competition among classes.

4. Model workflow and Design

Our proposed framework is a two-stage pipeline, where the two stages broadly structured as follows;-

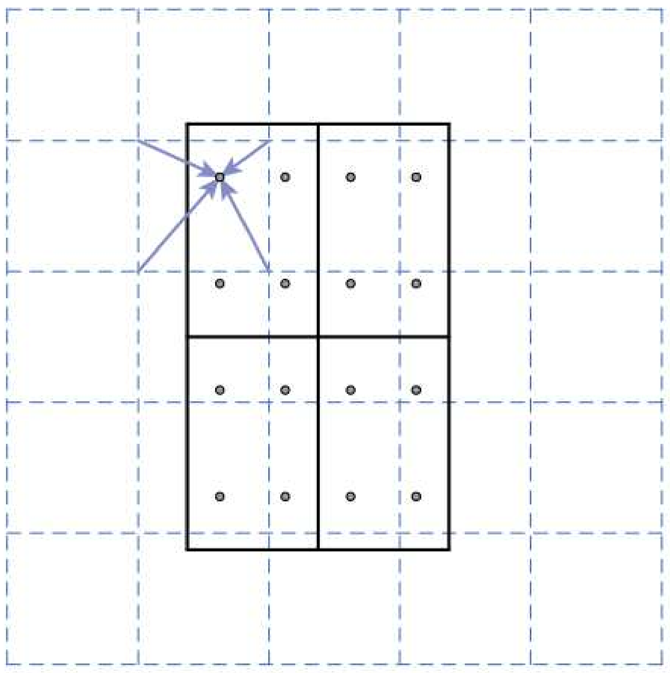

Fig. 6. bi-linear interpolation

- **Stage 1**:-
The first stage consists of two parallel computing CNN branches. On a given input image, the first branch uses YOLO architecture which detects and classifies each object in the full image. So it find the predicts the bounding boxes. On the same input image, the second branch is a deep convolutional network that gives the feature map of that input image. Later, using the output obtained from first branch i. e. YOLO which are bounding box coordinates of the objects in the input image which are then applied on the feature map obtained through the second branch. Now, we have object's bounding box coordinates on the feature map of that particular object. Later this feature map with bounding boxes on it is passed into the Stage-2 which then segments out the image.

- **Stage 2**:-
As stated in the Stage-1, YOLO detects bounding boxes for each object present in the image. So for each detected bounding box of an object ROI-align is applied on the corresponding feature map to extract a fixed size feature grid which is sequentially fed into mask segmentation network. This mask segmentation network is a fully convolutional network(FCN) which generates a binary matrix of the fixed size. Binary matrix represents a pixel level structure which essentially provides the information about the pixel whether the pixel belongs to the object or not. So 1 indicates that the corresponding pixel belongs to the object and similarly 0 indicates that the pixel doesn't belong to the object.

4.1. Object detection and Classification

The objective of YOLO here is to detect the objects and associate class probabilities to them. On an input image, YOLO divides the image into $S * S$ sized grid. If the centre of an object falls into a grid cell, then that cell is "responsible" for detecting that object.

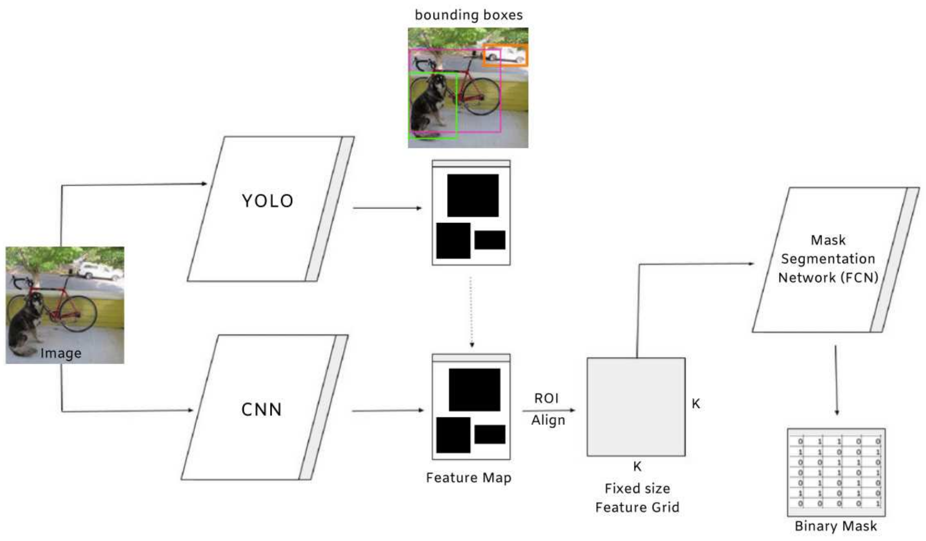

Fig. 7. Architecture of YOLO based instance segmentation

Each grid cell then predicts B bounding boxes and its respective confidence scores. Each bounding box consists of 5 parameters which are its coordinates - x, y, h, w and confidence scores.

Also, each grid cell predicts the C number of class probabilities. All these predictions S(grid size), B(number of bounding boxes) and C(class probabilities) are encoded as $S * S * (B * 5 + C)$ tensor. Hence when an input image is passed into the YOLO architecture, it returns an $7 * 7 * 30$ tensor (on PASCAL VOC dataset).

Later these tensor is sent for post processing. Here using the technique non maximal suppression it gives the best bounding box fit for the objects present in the test input image.

After obtaining the bounding boxes coordinates from YOLO, these bounding box coordinates are used to extract the corresponding feature map using ROI-align subsequently.

4.1.1. Non-maximum Suppression

The main purpose of non maximum suppression is to pick one of the multiple candidates for object. So it eliminates some candidates that are different detections of the same object. It makes sure that the object is identified only once. For ex: On an input image, a bicycle occupies some particular grid cells, then Non maximum suppression ensures we identify the optimal cell among all other candidates where this car belongs. Below is the way non maximum suppression works:

- 1. Discard all those grid cells where there is chance of presence of an object in that bounding box in (confidence score) $\leq 0.\ 6$.
- 2. While there are any remaining boxes, do the following:
  - Select the box with the largest confidence score of that bounding box. And output that as a prediction.
  - Then discard any remaining box with IoU $\geq 0.\ 5$ with box output in the previous step.

4.2. Instance Segmentation

After input image is processed with several convolutional layers and max-pooling layers to generate feature map,it receives N bounding boxes information from YOLO in the from of (x,y,w,h) and c ,where N denotes number of objects in the image. So the dimensions of the corner(x,y) ,height(h) and width(w) for each bounding box of objects present in the image are given to receiving end of the generated feature map.

Since convolutional feature map represents features for an entire image. But we have to find the feature map corresponding to the bounding box detected for the image. The corresponding feature map is reduced to fixed size feature grid using ROI-align.

The reason we are using ROI-align is to preserve the exact spatial correspondence between the feature map grid and the actual bounding box in the image. Instead if we use ROI-pooling layer it does coarse quantization or rounding-off approximations while extracting corresponding feature grids. The level of approximations should be completely cut off as the the accuracy of detecting pixel based masks is highly sensitive towards them and induces the possibility of pixel misalignment.

Hence for the pixel-level segmentation fine grained alignment is required therefore rounding-off is not performed(i. e x/n is considered instead of [x/16]). Then pooling is applied independently to each grid cell using bi-linear-interpolation. This method helps in achieving the precise mapping of binary mask to corresponding actual regions in the image.

The corresponding feature grid is fed into mask segmentation network. This network takes the fixed size feature grid and generates a $K\times$ matrix as output. Detailed application and design of mask segmentation network is discussed in the following section.

5. Mask Segmentation Network

The primary objective of the framework is to perform object instance segmentation,i.e to distinctively differentiate the various instances of a object class present in the image.

But to differentiate the instances we need to detect and the categorize the objects present in the image to different classes. After which the objects associated with the same class are considered to be different instances of that respective class. This particular work is essentially carried out by the YOLO model in the Stage-1 of the pipeline.

So Mask segmentation network is designed considering the fact that objects are pre-classified, Since the objects are pre-classified the task is essentially reduced to semantic segmentation. Hence this network is formulated just to perform pixel classification in the bounding boxes generated by YOLO without considering the class information of the object present in that bounding box. The detailed design is mentioned in the following sub-section-:

5.1. Design

- As stated in the model workflow ,a fixed size feature grid is fed into mask segmentation network. This network is a fully convolutional network(FCN) applied on feature grid corresponding to each bounding box to generate a binary segmentation mask in a pixel to pixel manner.
- Hence Fully convolutional network is being used as the binary mask is predicted with pixel level correspondence, Therefore sparsity in the convolutional layers is removed.
- Each convolutional layer in Full convolutional network tries to preserve the fixed spatial dimensional layout and the layout is maintained across the layers.
- By design, fully connected layers(FC's) are not incorporated as they dissolve the spatial structure of grid into to a single dimension vector and preserving spatial layout throughout is an integral part of formulation to maintain the alignment between input feature grid and output binary segmentation mask.
- Final convolutional layer generates a segmentation mask of fixed size. This binary mask is resized to the size of bounding box and binarize with a defined threshold.

5.2. Loss Function

For training the mask segmentation network learning should happen at pixel level. Since detection and classification are happening at a different stage of pipeline independently,this simplifies to binary pixel classification.Also the objective of the network is a single streamed task hence we use a dedicated loss function designed only to solve this particular problem of binary pixel classification,
Hence we apply loss function to each grid cell of the feature map individually. So the loss function we use is binary cross-entropy loss and the objective function used for pixel classification is per-pixel sigmoid.
The loss function designed for the defined purpose and above principles is as follows:-
Let $K \times K$ be the size of feature grid and f be the feature grid matrix then Loss function *L* is:-

$$L = \frac{-1}{K^2} \sum_{0 \le i \le K} \sum_{0 \le j \le K} (y_{ij} log(h_{ij}(f)) + (1 - y_{ij})(1 - log(h_{ij}(f)))$$

Where is h_{ij} is the sigmoid function i.e

$$h_{ij}(f) = \sigma(f(i,j))$$

$$h_{ij}(f) = \frac{1}{\exp^{-W*f(i,j)} + 1}$$

and W are the weights are learned by mask segmentation network.

5.3. Instance segmentation using Mask Segmentation Network and YOLO

Mask Segmentation Network produces a binary mask for each predicted boundary box which comprises of information pixel-specificity. The 0's in binary mask state that particular pixel is not part of object,if it is 1 then the pixel belongs to the object. Using the class probabilities (from YOLO) each boundary box is associated with a class,which directs to conclude that each object of same class is considered as an instance of that corresponding class. Therefore now different instances of the object can be uniquely differentiated with distinct color intensities using the binary mask produced by mask segmentation network.

6. Architecture

Architectural decisions :-

- YOLO architecture is pre-trained by ImageNeT 1000 class dataset with first 20 convolutional layers followed by average pooling layer and a fully connected layer.
- ResNet with 101 layers residual architecture is used for pre training Convolutional Neural Network(CNN) used for extracting feature map for an entire Image.
- The last pooling layer used for CNN (used for extracting feature map) is replaced by ROI-align layer.
- The ROI-align generated a fixed size feature grid from feature map corresponding to the bounding box region in the image.
- Mask segmentation network outputs a similar fixed size binary matrix of size $K \times K$.

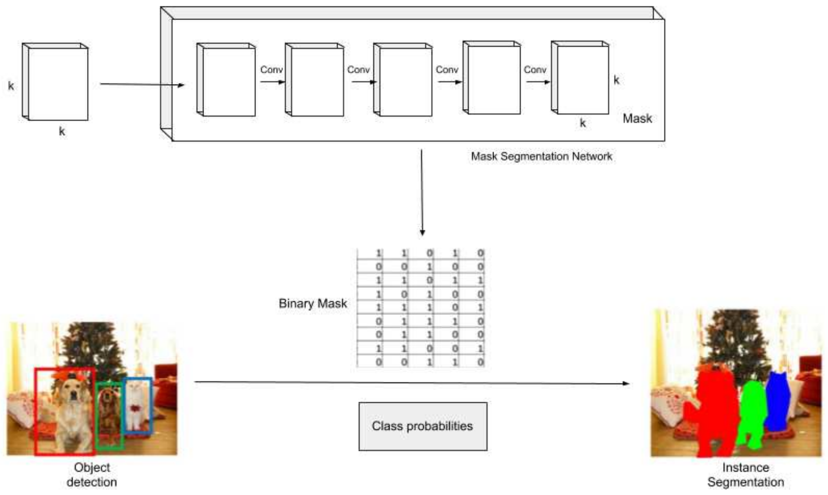

Fig. 8. Instance segmentation using Mask Segmentation Network and YOLO

7. Comparisons

MASK-RCNN is one of the paper published for object instance segmentation in recent times, which has performance over speed trade off. Some of the structural comparisons drawn between our framework and Mask-RCNN:-

- Mask-RCNN uses RPN for generating regional proposals which are in the order of $\approx$ 2000. But, in our model we only get exact number of regions corresponding to objects present in an image. But at this aspect our frameworks seems to be slightly better-off.
- In Mask-RCNN to train the model they use multi task loss which incorporates three different losses corresponding to bounding box regression,object classification and segmentation. But, in our model we used dedicated loss designed especially to solve one single problem of instance segmentation.

8. Future work

- Our network is single unified sequential network. Rather we could improve this network by introducing cascade way to it. This can be extended to cascades that have multiple stages.
- Instead of using a single loss function, we can use the loss function independently for different branches at different stages.
- Able to define a loss function that tackles problem of localization error in bounding boxes in YOLO (as YOLO treats error the same for small and large boxes).
- The framework proposed by us cannot do the instance segmentation in real time as the Mask segmentation network cannot be proceed until YOLO detects bounding boxes for the objects. Hence work in this direction could be extended to introduce the concurrency between the detection and segmentation.

9. Conclusion

We present a YOLO based instance segmentation network, which is mainly formulated to perform object instance segmentation, which by design also includes object detection and classification. Hence by studying various functional and structural characteristics of research work done on this domain. Considering all these observations we have concluded to design a framework which incorporates the effective and efficient aspects of all the designs without inducing complexities and trying to preserve the model simplicity. To demonstrate this fact, object detection and classification is a task where performance matters than speed hence YOLO framework is adapted which performs actions at real time, whereas to predict high quality segmentation masks, performance and efficiency matters the most, hence it derives its inspiration from performance oriented model MASK-RCNN. This makes it a simple, functional and highly qualitative model to perform object instance segmentation.

10. References

Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi. **You Only Look Once**: Unified, Real-Time Object Detection, University of Washington, Allen Institute for AI, Facebook AI Research.

Kaiming He,Georgia Gkioxari,Piotr Dollr,Ross Girshick.**Mask R-CNN**.Facebook AI Research (FAIR)

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun.**Faster R-CNN**: Towards Real-Time Object Detection with Region Proposal Networks.

Ross Girshick.**Fast R-CNN**, Microsoft Research.

Ross Girshick Jeff Donahue Trevor Darrell Jitendra Malik.**RCNN**.Rich feature hierarchies for accurate object detection and semantic segmentation, UC Berkeley.

https://www.groundai.com/project/yolact-real-time-instance-segmentation

https://everitt257.github.io/blog/2019/02/07/RoI-Explained

https://blog.athelas.com/a-brief-history-of-cnns-in-image-segmentation-from-r-cnn-to-mask-r-cnn

https://medium.com/@smallfishbigsea/faster-r-cnn-explained

https://lilianweng.github.io/lil-log/2017/12/31/object-recognition-for-dummies-part-3