Extension of Faster R-CNN for Instance Segmentation

Adithyakrishna Venkatesh Hanasoge, Joshua Ernest P
University of Pennsylvania
{adikvh, josherne}@seas.upenn.edu

Abstract

Our project presents a straightforward and versatile approach for instance segmentation, which involves identifying and separating objects in an image. This method, called Mask R-CNN, extends Faster R-CNN by adding a branch for predicting an object mask in addition to the existing branch for bounding box recognition. While Faster R-CNN has 2 outputs for each candidate object, a class label and a bounding box, Mask R-CNN is the addition of another branch that outputs the object mask. Mask R-CNN adds only a small overhead to Faster R-CNN making it simple to train.

1. Introduction

Image segmentation is an essential component in many visual understanding systems and can be formulated as a classification problem of pixels with semantic labels or partitioning of individual objects. Semantic segmentation performs pixel-level labeling with a set of object categories for all image pixels, thus it is generally a harder undertaking than image classification. Instance Segmentation extends semantic segmentation scope further by detecting and delineating each object of interest in the image. Thus instance segmentation is challenging because it requires the correct detection of objects as well as segmenting each instance of the object.

Mask R-CNN [1] being an extension of Faster R-CNN, adds a branch for prediction of segmentation masks on each Region of Interest(ROI) in parallel with the already existing branch for classification and bounding box prediciton. For the mask branch, a small FCN is applied to each ROI, predicting a segmentation mask in a pixel-to-pixel manner. In order to achieve successful results with Mask R-CNN, it is crucial to properly design the mask branch, which extends Faster R-CNN. One key challenge is that Faster R-CNN was not designed for pixel-to-pixel alignment between its inputs and outputs. To address this issue, we are using Multiscale RoIAlign, which accurately preserves spatial locations without using quantization. This small change significantly improves mask accuracy by 10-50%, with the greatest improvements seen under stricter localization metrics. We also found it essential to separate mask and class prediction, by predicting a binary mask for each class independently without competition among classes, and using the RoI classification branch to predict the category. This differs from the approach used by Fully Convolutional Networks (FCNs), which perform per-pixel multi-class categorization, combining segmentation and classification, and which we found to be less effective for instance segmentation in our experiments.

2. Related Works

2.1. R-CNN

The Region-based Convolutional Network (RCNN) method was an early object detection algorithm that achieved high accuracy by using a deep convolutional network to classify object proposals. However, its multi-stage pipeline made it difficult to train. R-CNN first used a deep ConvNet to generate object proposals using log loss, then applied an SVM classifier to the ConvNet output instead of a softmax classifier to classify the proposals. Finally, it used regression to learn the bounding boxes. This made training spatially and temporally expensive, and the resulting model was slow to predict objects due to the lack of computation sharing for each object proposal. An optimization over the R-CNN is the Faster R-CNN that uses "Region Proposal Network" compared to R-CNN's selective search for generating Regions of Interest.

2.2. Fast R-CNN

The R-CNN method trains ConvNets to classify proposed regions into object categories or the background, but does not predict bounding box coordinates. Its accuracy depends on the performance of the region proposal module. The OverFeat method uses a fully connected layer to predict box coordinates as a localization task, but assumes the presence of a single object in the image. For multiple object categories, it converts the linear layers to convolutional layers. The MultiBox method uses a fully connected layer to simultaneously predict multiple class-agnostic boxes, gen-

eralizing the "singlebox" approach of OverFeat. However, this method lacks feature sharing between the proposal and detection networks, making it a difficult to train multistage method. Fast R-CNN, on the other hand, uses shared convolutional features for end-to-end detector training, improving the accuracy and speed of training significantly.

2.3. Faster R-CNN

The R-CNN approach used a simple ConvNet to generate proposal boxes on each region of interest (RoI), which were then extended to feature maps using RoIPool to improve training speed and accuracy. Faster R-CNN introduced a learned attention mechanism called the Region Proposal Network (RPN) to obtain a rough but cheap initial estimate of the bounding boxes of objects for object detection. The initial anchor boxes are classified into object or background classes, which are then used to refine and tighten the instance segmentation and class prediction. The use of RPN in Fast R-CNN makes it a fast and accurate algorithm, as it eliminates expensive greedy algorithms like Selective Search.

2.4. Instance Segmentation

There have been a number of approaches to instance segmentation in the field of computer vision. Some examples include:

Mask R-CNN: This method combines object detection and instance segmentation in a single end-to-end framework. It uses a deep convolutional network to predict object bounding boxes and a separate branch to predict segmentation masks for each instance.

Pixel-wise Masking: This method involves learning a pixel-wise mask for each instance in the image. It can be implemented using techniques such as Fully Convolutional Networks (FCN) or U-Net.

Panoptic Segmentation: This method aims to simultaneously segment both instance-level and stuff-level (e.g. background) regions in an image. It can be implemented using a combination of instance and semantic segmentation techniques.

Interactive Segmentation: This method involves allowing the user to manually specify regions or provide additional input to guide the segmentation process.

Instance Cut: This method involves separating the foreground and background regions in an image and then using graph cuts or other optimization techniques to refine the instance-level segmentation.

3. Dataset

The project will make use of a portion of the Microsoft COCO [2] dataset. This collection contains 3265 example RGB photos and their labels. The dataset is divided into three categories: vehicles, people, and animals. The data also includes ground truth masks and object bounding boxes. The flattened mask list, which comprises spatial information about the item, is allocated to the matching picture depending on the order of the labels.

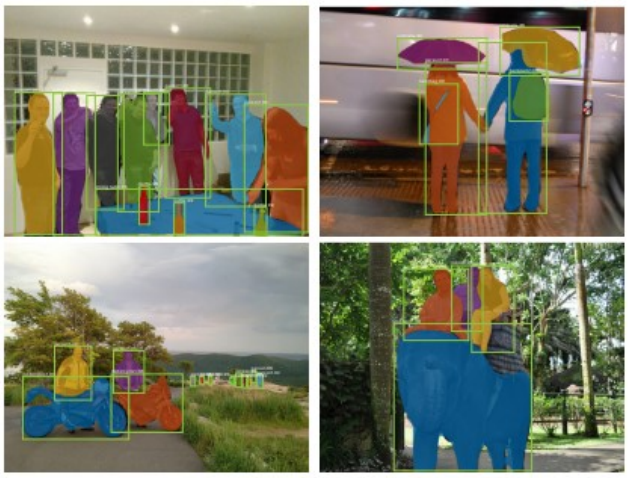

Figure 1. Dataset

3.1. Dataset Preparation

This tasks consists of the following subtasks

- The masks have to be assigned to their respective images
- Images have to be normalized between [0, 1]
- Images have to be rescaled to $(800, 1066)$
- Normalize each channel with mean: [0.485,0.456,0.406], std:[0.229,0.224,0.225]
- Adding zero padding to get an image of size 800×1088
- Rescaling bounding boxes and masks to match the image dimensions

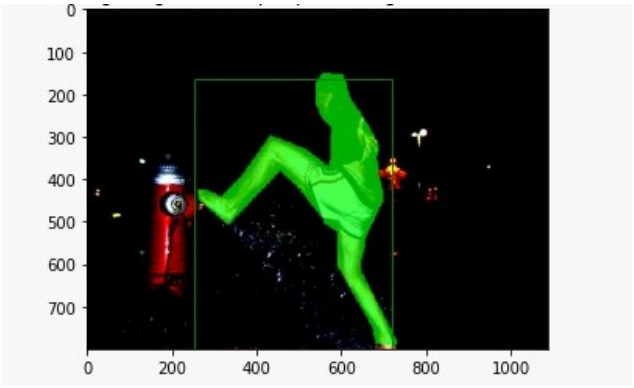

Figure 3. Verification of Dataset Preparation

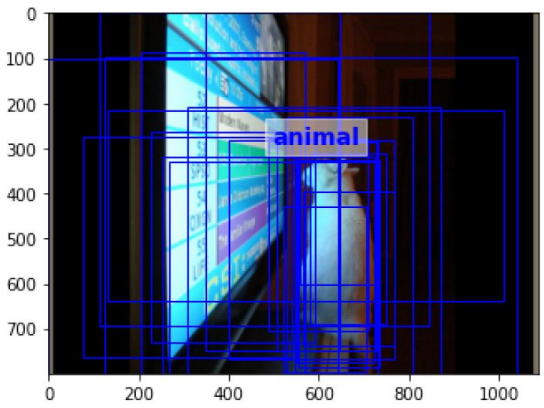

Figure 2. Verification of Dataset Preparation

4. Methodology

The Mask R-CNN works in two stages like the Fast R-CNN.

4.1. Stage 1

The first stage consists of two networks, the FPN backbone which is the Resnet 50 in our case and the Region Proposal Network. [3]. These networks run once per image to give a set of unrefined region proposals. The FPN serves as a shared backbone. The output from the FPN serves as the input to the RPN. Region Proposal Networks (RPNs) are a type of attention mechanism used in the task of object detection. They quickly and inexpensively estimate the locations of bounding boxes around objects in an image by classifying initial anchor boxes as either containing an object or being part of the background. The coordinates of the boxes containing objects are then refined by RPNs. These refined boxes are later further refined and classified into specific object classes by instance segmentation heads. The RPN has 3 heads, the intermediate layer, the classifier head and the regressor head to perform the above mentioned tasks. The architecture for the RPN and the later refinement of the proposals is shown below:

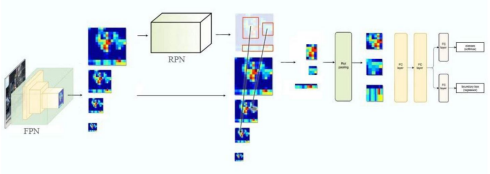

Figure 4. Architecture for Faster R-CNN

4.1.1 RPN Loss

Since the RPN has a classifier and a regressor head, we will be using two losses, the classifier loss and regressor loss. For the classifier's loss, we compute the binary cross entropy between the predicted objectness p and the ground truth objectness p*. For the regressor loss, we use the Smooth L1 loss between t,t*, which is defined as: and

$$L_{reg}(\mathbf{t}) = \frac{1}{N_{reg}} \sum_{i=1}^{N_{reg}} p_i^* \sum_{j=1}^{4} smooth_{L1}(\mathbf{t}_i[j] - \mathbf{t}_i^*[j])$$

where,

$$smooth_{L1}(x) = \begin{cases} 0.5x^2 & |x| \leq 1 \\ |x| - 0.5 & else \end{cases}$$

N_{reg} is the number of anchor locations in the sampled mini-batch. Note that the regressor loss is computed only on the positive anchors in the sampled mini-batch (where p*i is 1).

4.2. Stage 2 - Box Head

In the second stage, the outputs of the RPN have to be prepared for the Box Head. For this we use ROI Align to transform the region that each box surrounds into a fixed $P \times P$ matrix to train the network further. The RoI pool is replaced by RoI Align which helps to preserve spatial information which gets misaligned in case of RoI pool. RoI Align uses binary interpolation to create a feature map that is of fixed size for eg 7 X 7. The Box Head is responsible for refining and classifying the proposed regions(proposals) produced by the RPN in Stage 1. Similar to RPN it consists of an intermediate layer a regressor and classifier that predicts bounding boxes and object class for each of the proposed regions obtained in stage1.

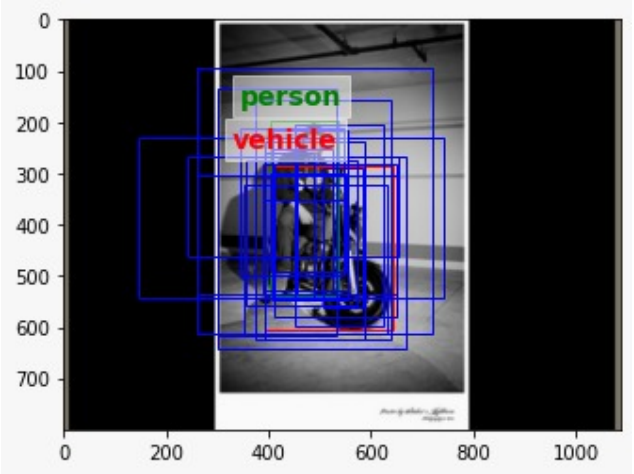

Figure 5. Box Head Results

4.2.1 Box Head Loss

For the loss of the classifier, we calculate the cross entropy loss between the predicted class probability vector p and the true class p*. This is similar to the loss calculation for the Region Proposal Network (RPN), with the exception that we now have C+1 classes, therefore we do not use Binary Cross Entropy Loss. For the regressor loss, similar to the RPN, we use the L1 smooth loss.

$$L_{reg}(t) = \frac{1}{N_{reg}} \sum_{i=1}^{N_{reg}} (\sum_{c=1}^{C} \mathbf{1}(p_i^* = c) \sum_{j=1}^{4} smooth_{L1}(t_{ij}^c - t_{ij}^*)) \tag{1}$$

4.3. Mask Head

The outputs of the box head are fed into the Mask Head network. Similar to the Box Head, we have to prepare the input for the Mask Head. This involves using the ROI Align similar to what we did previously. The network that was used for the Mask Head consists of 6 layers where the first four layers are convolutional layers with a kernel size of (3,3,256) and same padding, using a ReLU activation function. The final two layers are a transposed convolutional layer that doubles the region size and maintains 256 channels with a ReLU activation, followed by a convolutional layer with a kernel size of (1,1,number of classes). This mask head produces one mask for each class for each detection from the regressor, and the output is passed through a sigmoid activation function.

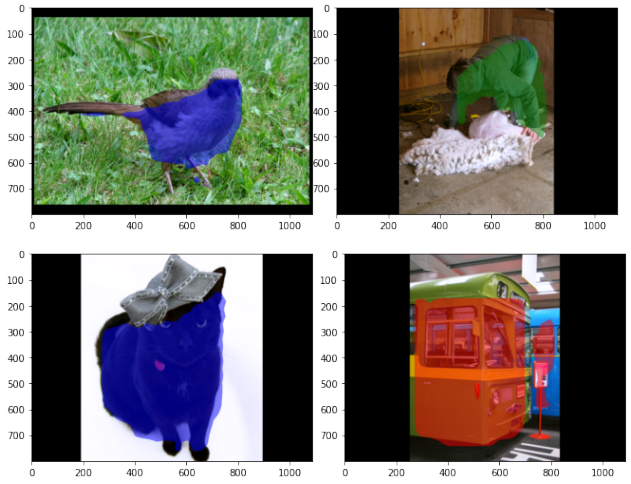

Figure 6. Mask Head Output

4.3.1 Mask Head Loss

We have used the mask predicted for a bounding box that belongs to the ground truth class to train the mask branch. The target for this training will be the intersection of the bounding box produced by the Box Head and the ground truth mask. The mask branch will be trained using binary cross entropy loss, which is calculated on a per-pixel basis. The equation for the Mask Head loss is as follows :

$$L_{mask} = -\frac{1}{m^2} \sum_{i,j} y_{ij} \log \hat{y}_{ij} + (1 - y_{ij}) \log(1 - \hat{y}_{ij}) \tag{2}$$

4.4. Overall Architecture

The overall architecture from FPN all the way to the outputs of MaskRCNN is shown below.

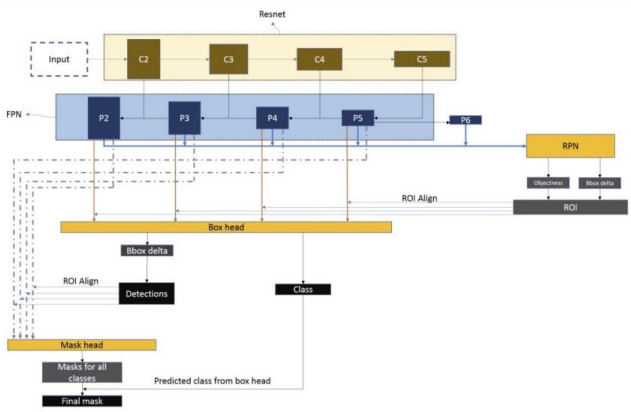

Figure 7. Overall Architecture

5. Hypotheses and Experiments

For the RPN head, the anchor boxes will be created for each level of the FPN and ground truth boxes will be made for each anchor. For the Box head, the ground truth boxes will further be assigned to classes as given by the output of the RPN. For the Mask head, we will be following the architecture as mentioned in the hand out. We will visualise the train loss after each postprocessing stage.

5.1. Postprocessing

For the RPN head and the Box head, we will be using Non Max Suppression as the post processor. For the mask head, we will rescale the predicted masks and use interpolation for the proposed bounding boxes.

5.2. Loss curves

The loss curves for the Region Proposal Networks are shown below. As can be seen from the figures 8, 9, 10. The perfomance curves show gradual improvements over 30 45 epochs.

Figure 8. RPN Classifier Loss

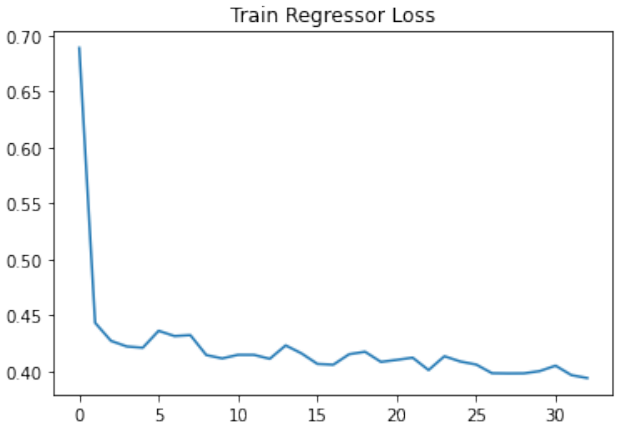

Figure 9. RPN Regressor Loss

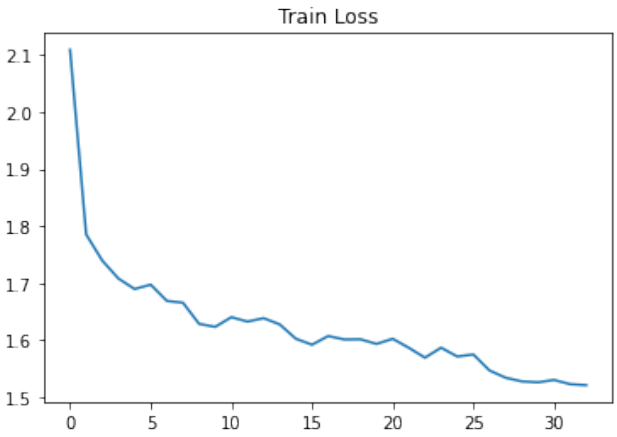

Figure 10. RPN Train Loss

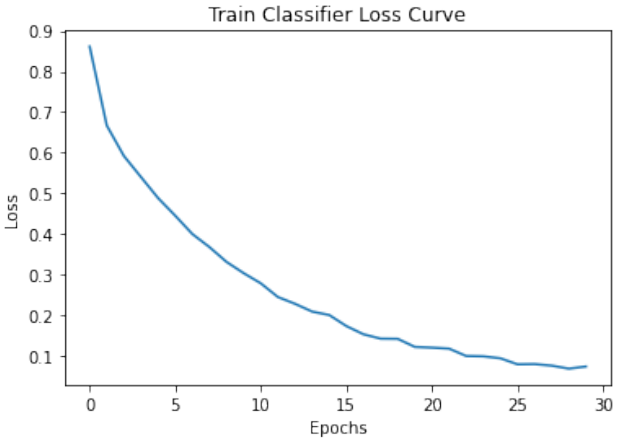

Figure 11. Box Head Classifier Loss

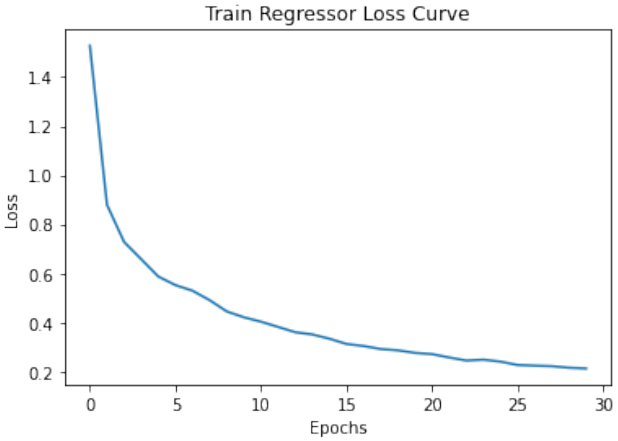

Figure 12. Box Head Regressor Loss

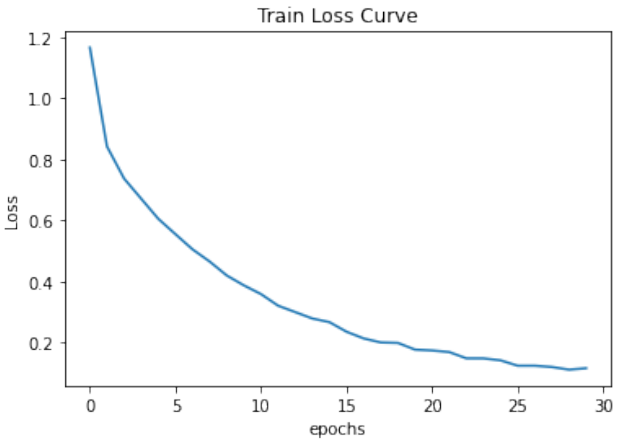

Figure 13. Box Head Train Loss

The loss for the Box and Mask head are shown above.

After extensive debugging and trying out different loss functions, we were able to obtain a reducing loss curve for both.

Please note, the results at every individual stage is show in the methodology section above. All the way from RPN to Mask Head outputs.

5.3. Evaluation Metric

As an evaluation metric, we used Mean Average Precision (mAP) to assess the performance of our Mask R-CNN algorithm on 2000 images, divided into three classes: vehicles, people, and animals. The mAP scores for each class was 0.408 for vehicles, 0.368 for people, and 0.6161 for animals. The higher mAP score for animals may be due to the greater number of images in this class in the dataset. The overall average mAP score for all three classes was 0.464.

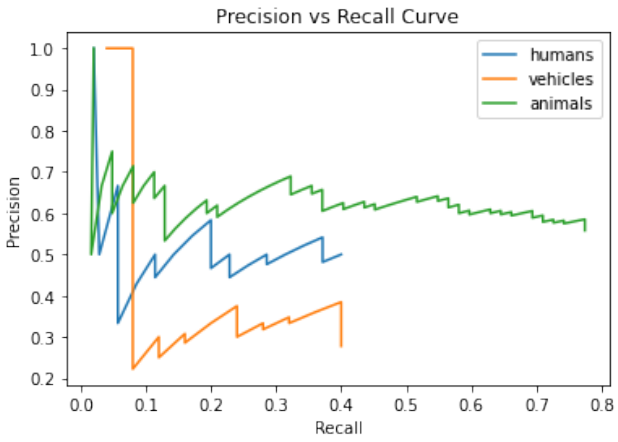

Figure 14. Precision Recall Curve

5.4. Experiments

During the course of this project, we experimented with different loss architectures, one being the Focal loss and another the Cross Entropy loss for 3D detection. The results for which were not so distinct and which we were not able to visualise. Our goal was to also implement 3D bounding boxes using stereo RCNN on the KITTI dataset. Stereo R-CNN focuses on accurate 3D object detection and estimation using image-only data in autonomous driving scenarios. It features simultaneous object detection and association for stereo images, 3D box estimation using 2D information, accurate dense alignment for 3D box refinement. But owing to the extensive debugging, lack of time and the computational expenses that we would require in training, we were not able to execute it in the current scenario.

6. Novelty

We also tried implementing ROI Pooling and ROI allign methods but only ROI allign gave us logical results. The issue with ROI pooling is that it takes hard boundaries and divides then divides them into regions and takes the average/max pool from those regions. But features are lost due to pooling from large sub spaces. Also, ROI allign does not result in any shifting. Its a combination of both ROI pooling and bilinear interpolation. Since Allign samples multiple points, it expands the area of image from where you are pooling and does not result in loss of features.

References

[1] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn, 2017. 1

[2] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2014. 2

[3] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. pages 1–10, 01 2016. 3