Report on Car Damage Detection

Problem Description:

In this problem statement, we will take different images of damaged parts of a car to identify the damage using computer vision technique such as Fast RCNN and Mask RCNN, since Mask RCNN is the extension version of Fast RCNN we will use Mask RCNN for detecting the damage in the car

Literature Survey:

Ren et al. [1], proposed a faster RCNN, which is a hybrid of fast RCNN and Region Proposal Network (RPN). RPN is a fully convolutional network capable of predicting bounding boxes and calculating predictive scores. Faster RCNN is made up of two modules: (I) A dense convolutional neural network that proposes areas, and (II) A Fast RCNN detector that employs the proposed regions. Faster RCNN takes ten times longer to compute than Fast RCNN because the RPNs tell the Fast RCNN detector to look at specific portions of an image rather than the entire image. In addition, the feature map is shared to speed up the computation.

Mask RCNN by He et al. [2], an extension of Faster RCNN, is a framework that can detect and segment an object instance at the same time. It additionally adds colored masks to the class objects in addition to the bounding box, category, and confidence. Mask R-CNN is similar to Faster RCNN in that it uses a two-stage network, using RPN for bounding box prediction and ROI Pool for recognition. They added a third output called segmentation mask to Mask RCNN, which is completely independent to classification prediction. In Faster RCNN, it was included as a third branch running parallel to the second stage. As a result, it was able to achieve high-quality instance segmentation and classification with only a minor overhead over its predecessor.

Dataset Information:

1. Initial data set is taken from the Kaggle coco car damage detection. (Open-source data available) [3], for training purpose.
2. The open-source code is taken from the GitHub DeepLearning_MaskRCNN [4]

MRCNN Application-Detecting Car Exterior Damage

Convolutional neural networks (CNN) are used in the automated detection and quantification of car exterior defects (damage severity)

1. Extracting Regions of Interest (ROI)
    a. The image is fed into a Convolutional Network(ConvNet), which uses methods like selective search (RCNN) or RPN (Region Proposal N/W for Faster RCNN) to return the region of interest.
    b. After that, apply a RoI pooling layer to the extracted ROI to ensure that all of the regions have the same size.

2. Classification

a. Regions are sent to a fully connected network, which sorts them into several image classifications. Scratch(‘damage') or background, for example (car body without damage).

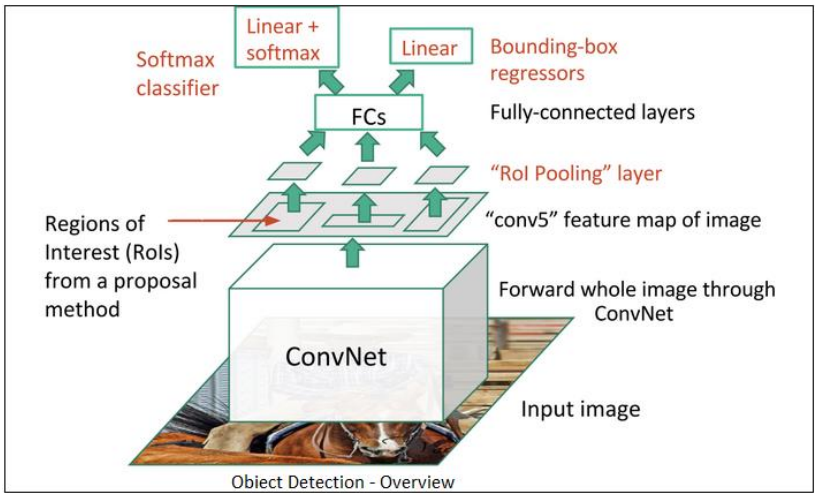

Fig.1 Object detection overview

3. Regression
    a. For strengthening the bounding boxes, bounding box (BB) regression is utilised to predict the bounding boxes for each recognised region (getting exact BB defining relative coordinates)
    b. to go backwards instead of forwards from either region proposals or fixed anchor boxes to neighbouring bounding boxes of pre-defined target object classes
4. To precisely identify the position and quantify the damage, the exact pixel location in the bounding box that match to the class(damage) must be identified.

5. Masked Region based CNN (Mask R-CNN)

a. Identifying pixel-wise delineation for object class of our interest
i. Object detection based on BB (uses RoI align to allow the pixel to pixel preserve of RoI’s and prevent information loss)
ii. Semantic segmentation - pixel-by-pixel segmentation of specific items within an image, regardless of shape (pixel-wise categorization) (pixel-wise shading of the class of interest)
iii. Instance Segmentation – This goes a step farther than semantic segmentation by identifying items inside designated groups. For example, in the street picture, you would draw different limits for each group and identify them separately - Humans (Adult, Kid), Automobiles (Cars, Bus, Motor Bikes...), and so on.

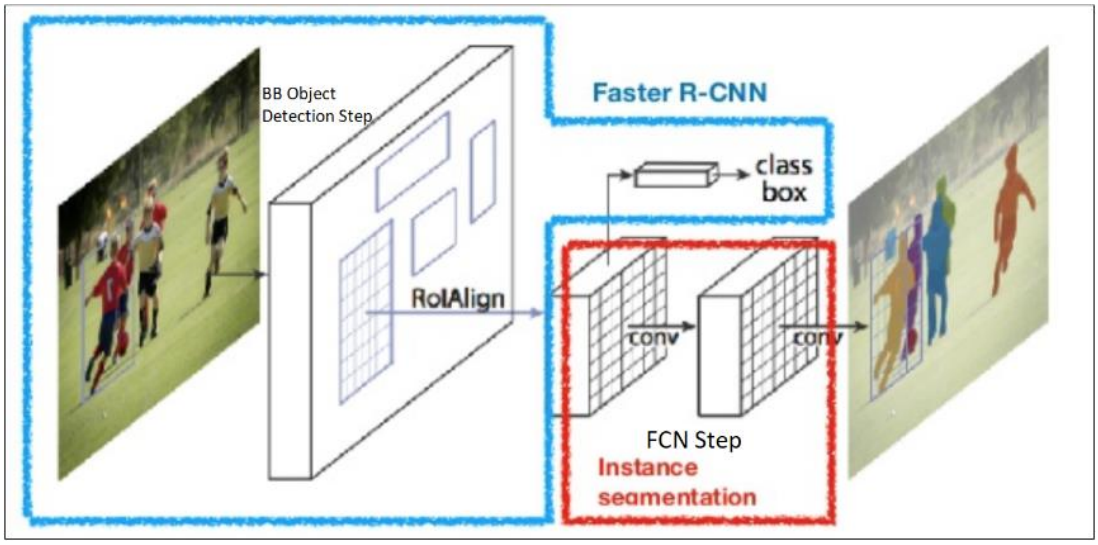

Fig.2 Mask RCNN

6. Loses at each phase [6]

$$L_{Overall} = L_{cls} + L_{box} + L_{mask} \quad (1)$$

a. Classification Loss: In practice the gradients are computed over a simple cross entropy loss function commonly used in multi-class classification.

$$L_{cls} = -\sum_{u=0}^{K} \log(p_u) \quad (2)$$

P is a (k + 1)-dimensional vector representing the probability of a pixel belonging to the k class or background for each ROI, P = (P0, P1, . . ., Pk), and Pu represents the probability corresponding to class u

b. Regression Loss - The regression loss is defined as Loss = R (t -t*) where t and t* is the vector representing the four vertices of our predicted and ground truth bounding box, and R is the robust loss function defined here.

$$L_{box} = \sum_{i=1}^{4} \text{smooth}_{L1}(t_i^u - v_i) \quad (3)$$

$t_u = t_x^u, t_y^u, t_w^u, t_h^u$ represents the predicted translation scaling parameter of class u, t_x^u, t_y^u refer to the translation with the same scale as the object proposal, and t_w^u, t_h^u refer to the height and width of the logarithmic space relative to the object proposal. $t_1, t_2, t_3,$ and t_4 in equation (3) represent u, t_x, t_y, t_w, t_h respectively. Moreover, vi represents the corresponding parameter of the ground-truth bounding box

Total network loss

$$L_{RPN} = L_{cls} + L_{box}$$

c. MaskRCNN loss: The mask branch has K dimensions, where K is the number of classes and is the mask resolution, resulting in K binary masks, i.e., 1's where the segmented foreground object is located and 0's elsewhere. The loss is then the average per-pixel sigmoid output's binary cross entropy loss over the K dimensions. As a result of the averaging, every mask dimension in the output can contribute to the final mask prediction without competing with the classes, decoupling mask and class predictions. Regular Fully Convolutional Networks for image segmentation, on the other hand, often use a per-pixel soft-max and multinomial cross entropy loss.

$$L_{mask} = -\frac{1}{k}\sum_{i=1}^{k}\sum_{j=1}^{m*m} \log P_{i,j}^{M}$$

L_{mask} is the average of the cross-entropy loss of each pixel:

Here, $P_{i,j}^{M}$ is the J^{th} pixel of the I^{th} generated mask

Architecture of MaskRCNN and Detectron 2

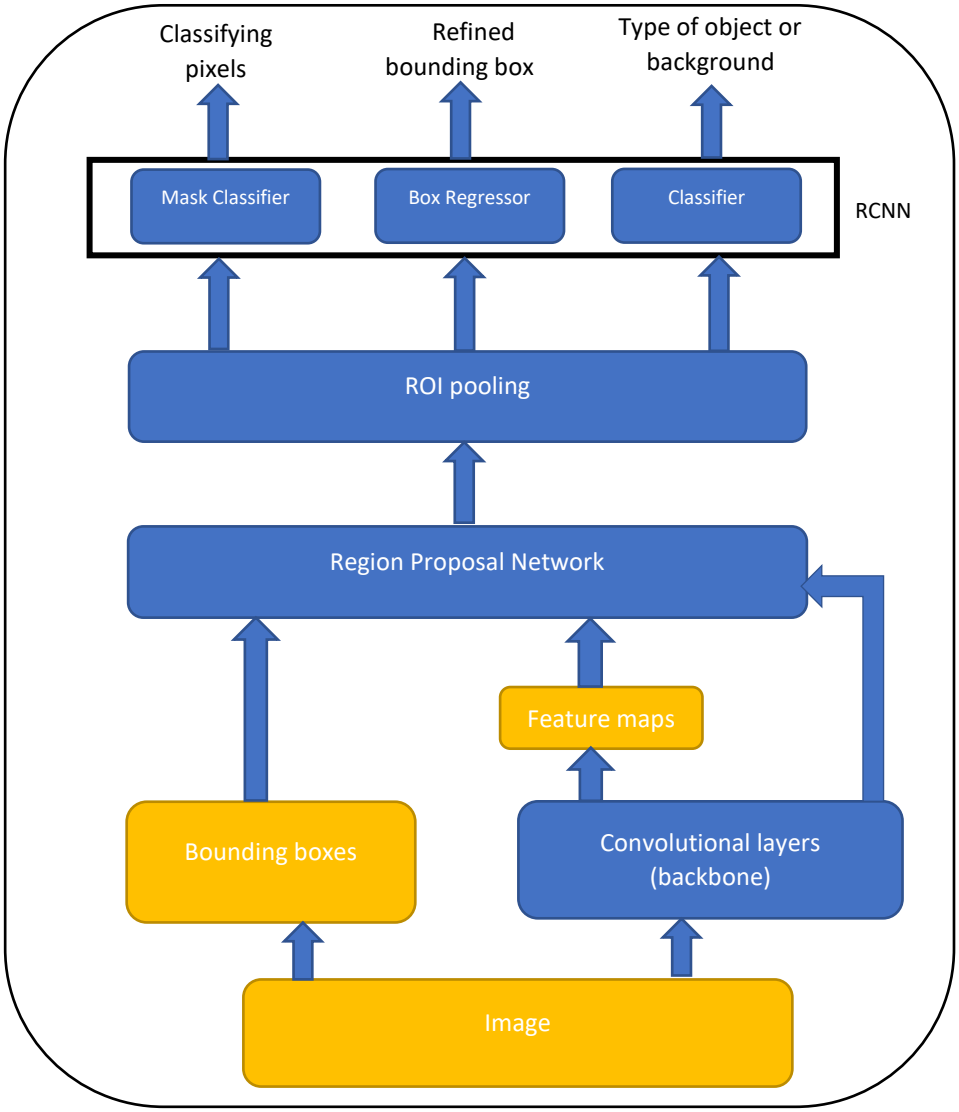

Fig.3 Architecture of Mask RCNN

We begin with an image, then send it via convolutional layers or a backbone to obtain feature maps on one side and anchor boxes on the other, all of which are sent to the region proposal network. The region proposal network goes into ROI pooling with the feature maps, and by this I mean the regions that were proposed by this RPN original proposal network, and the output of this with the feature maps will go to ROI Pooling, and finally we will get the classes using ROI pooling, so there is a classification sublayer that tells us the type of the object or if it's a background, It's a box regressor, and the other branch is for classifying pixels, which is what gives us the segmented parts, so all three branches are active at the same time.

Detectron2:

Facebook AI Research's Detectron 2 [5] is a next-generation open-source object detection system. You may use and train state-of-the-art models for detection tasks including bounding-box identification, instance and semantic segmentation, and person keypoint detection using the repo.

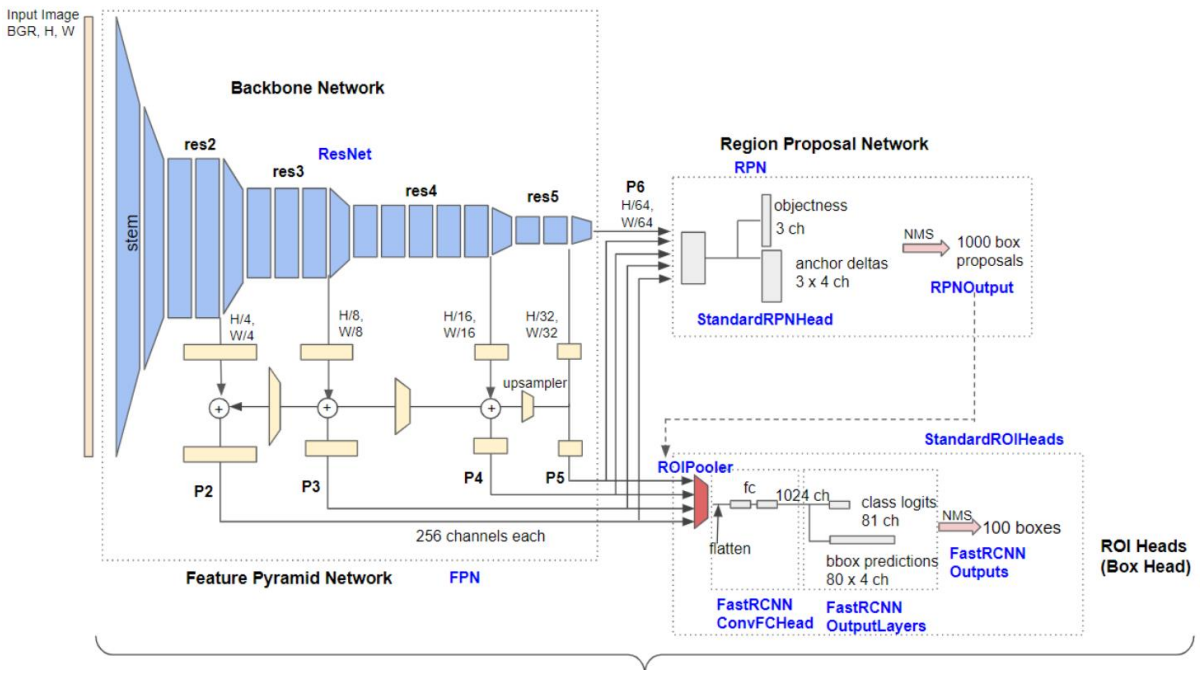

Fig.4 Architecture of Detectron 2 [9]

1) Backbone Network: extracts feature maps at various scales from the input image. P2 (1/4 scale), P3 (1/8), P4 (1/16), P5 (1/32) and P6 (1/64) are the output features of Base-RCNN-FPN. The output feature of non-FPN ('C4') architecture is only from the 1/16 scale.

2) The Region Proposal Network uses multi-scale characteristics to detect object regions. By default, 1000 box proposals with confidence scores are generated.

3) Box Head cuts and warps feature maps into several fixed-size features using proposed boxes, and achieves fine-tuned box positions and classification results using fully-connected layers. Finally, using non-maximum suppression, 100 boxes (by default) are filtered out (NMS). One of the sub-classes of ROI Heads is the box head. Mask R-CNN, for example, features additional ROI heads, such as a mask head.

Input:

1) Detectron2: In this problem statement we capture the open-source COCO image dataset. We initialize the instance annotations using COCO_train_annos.json and COCO_mul_train_annos.json. We receive the different categories and super-categories images in the form of damage parts. E.g., headlamp, rear bumper, door, hood, front bumper. They are then mapped using dictionary for category id and category name.
2) MaskRCNN: We specify image id, image, mask, class id, bbox. There are 2 classes: 1) BG (Background), 2) Damage. We also run via_region_data.json file to get the details of each image we want to train for detecting damages.

Output:

1) Detectron2: We get the plots for the annotations on different damaged parts in the form of bounding boxes and segmentation. When we run the detectron2 libraries for training the model, we receive results like total number of damage instances, Average Precision, Average Recall, IOU score for each prediction, class accuracy, total loss. We also run the validation images to get the desired results of the damages in the form of IOU scores above the bounding box.
2) MaskRCNN: We receive the results of semantic segmentation for the damaged parts separating foreground and background along with the bounding box for annotations. Visualization in the form of histogram for the model weight matrix is carried out in the form of descriptive statistics. We also run the training and validation images to get the desired results of the scratches in the form of IOU scores above the bounding box.

Comparative Results:

Sr. No.	Models Used for Car Parts Damage Detection	Kinds of Detection	Separating damaged part from the image	Segmentation Result	Bounding Box Result	Average Predicted IOU Score
1.	Detectron2	Damage	Not Supported	100%	100%	94.46%
2.	MaskRCNN	Damage, Scratch	100%	100%	100%	94.48%

Table: Comparison of results of Damage Detection with Detectron2 and MaskRCNN

Support from the organization:

Requirements:

1. Data: We can request for the real-time data of car damaged parts and assembly line segments.
2. Camera Mount: PTZ Camera with high resolution
3. GPU: NVIDIA GeForce RTX 3080/3090 series

Conclusion: From the above results, we can say that Detectron2 is used for detecting the damages in the car where as MaskRCNN is used for detecting the scratches in the car.

Advantages of using Mask RCNN

I. Mask R-CNN is a simple to train neural network.

II. Performance: On every challenge, Mask R-CNN surpasses all existing single-model entries.
III. Faster R-CNN adds only a little overhead to the approach, which is incredibly efficient.
IV. Mask R-CNN is versatile and can be applied to a variety of problems. In the same framework, Mask R-CNN may be used to estimate human pose, for example.

Future scope: We can use the MaskRCNN algorithm for other applications as well such as pot hole detection and car parts detection and many more

References:

1. Ren, Shaoqing, Kaiming He, Ross Girshick, and Jian Sun. "Faster r-cnn: Towards real-time object detection with region proposal networks." In Advances in neural information processing systems, pp. 91-99. 2015
2. He, Kaiming, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. "Mask r-cnn." In Proceedings of the IEEE international conference on computer vision, pp. 2961-2969. 2017
3. Dataset: https://www.kaggle.com/lplenka/coco-car-damage-detection-dataset
4. Reference Code: https://github.com/nasirtrekker/DeepLearning_MaskRCNN
5. https://medium.com/@hirotoschwert/digging-into-detectron-2-47b2e794fabd
6. Jian-Hua Shu, Fu-Dong Nian, Ming-Hui Yu, and Xu Li, "An Improved Mask R-CNN Model for Multiorgan Segmentation", Mathematical Problems in Engineering, Hindawi, https://doi.org/10.1155/2020/8351725.
7. https://towardsdatascience.com/
8. https://viso.ai/deep-learning/mask-r-cnn/
9. https://medium.com/@hirotoschwert/digging-into-detectron-2-47b2e794fabd