Semantic Image Segmentation

A Human Detection Application using Mask R-CNN

[1]Tapan Kamath, [2]Shaaz Merchant, [3]Nikhil Gunale, [4]Ketan Dudhe, and [5]Ankush Hutke

[1]BEIT Student, [2]BEIT Student, [3]BEIT Student, [4]BEIT Student, [5]Assistant Professor, Rajiv Gandhi Institute of Technology
[1]Bachelors of Engineering in Information Technology (BEIT),
[1]Rajiv Gandhi Institute of Technology,Mumbai,India

Abstract : Semantic image segmentation is one of the key problems in the field of computer vision, particularly for core computer vision problem such as scene understanding. The importance of scene understanding is that an increasing number of applications benefit from inferring knowledge from images. This is especially needed when it comes to applications such as virtual reality, human and computer interactions, object detection, and driverless cars.
A typical semantic image segmentation process involves two main parts, the encoder and decoder. The encoder is a pre-trained classifier, in this case, VGG and ResNet, followed by a decoder network. The task of the decoder, meanwhile, is to semantically project discriminative features learnt from the encoder on to the pixel space to a dense classification.
Mask R-CNN (Regional Convolutional Neural Network) is one of the most popular implementations of image segmentations. We will develop a human detection application, which applies bounding boxes on detected humans and indicates the confidence factor along with the object detected, in our case, humans.

Keywords : Mask Regional-Convolutional Neural Network (MRCNN), Image Segmentation, Human Detection

I. INTRODUCTION

Semantic Image Segmentation is very important in the field of computer vision as it helps us understand the image. In a broader sense, semantic segmentation is one of the high-level task that helps in complete scene understanding. Scene understanding is especially important when it comes to important computer vision applications such as virtual reality, human and computer interactions, object detection, driverless cars, etc.

The task includes labelling each pixel of the image a specific category label i.e. person, car, etc. Recently great progress has been made in this field due to Deep Convolutional Neural Networks (CNNs). A semantic segmentation architecture can be seen as an encoder network followed by a decoder network. The encoder is a pre-trained classifier, in this case VGG and ResNet followed by a decoder network. The decoder semantically projects the discriminative features (lower resolution) acquired by the encoder onto the pixel space (higher resolution) to get a dense classification of objects. One of the various approaches to semantic image segmentation is Region-based Convolutional Neural Network (R-CNN), which creates bounding boxes, or regions, using a process called Selective Search. Selective Search finds windows of different sizes, and for each size, it tries to group together, adjacent pixels by intensity, texture or color to identify different objects. In Mask R-CNN, a Fully Convolutional Network (FCN) is added on top of the CNN features of Faster R-CNN to generate a mask (segmentation output). Hence, we chose Mask-RCNN as a base to develop our application.

In this paper, we focus on using Mask R-CNN algorithm and extend its application to work on a Real Time basis. We created an application that uses a real time feed, using devices such as webcam, and applies object detection using Mask R-CNN in real time on the incoming video steam, frame by frame, to give us a live video feed with object detection. Our application weights have been explicitly trained to only detect humans, thus no other objects will be detected.

Another application we developed along with Real Time detection, was to apply a splash effect on static images or videos using Mask R-CNN. This splash effect applies a grayscale effect on the background, thus highlighting the objects in the foreground.

1.1 Advantages

- Objects can be detected in real time, using cameras on dedicated systems or on webcams of laptops/desktops.
- A ‘splash’ effect can be applied to separate objects from their backgrounds
- The application is trained to solely detect humans, and avoids any detection of other objects in and around the main object.
- A mask is added over the object, i.e. over the human detected, for pixel-wise detection.
- Live detection can be useful in multiple applications where real time data is crucial to the functioning, for example, security cameras, self-driving cars, etc.

1.2 Limitations

- It still takes a huge amount of time to train the network as you would have to classify multiple region proposals per image.
- Training is very resource intensive which is very time consuming.
- Running live detection on consumer grade hardware often results in latency which can be improved using more powerful hardware and dedicated graphics, but at the expense of increased
- cost of the project.
- Since selective search algorithm is a fixed algorithm, no learning takes place at this stage. This could lead to the errors in generation of region proposals.

II. RELATED WORK

To decide which algorithm would be best suited for our application, we needed to first understand various algorithms that had already been implemented

1. **Fully Convolutional Network for semantic segmentation [1]**

   As per our understanding, Long et al. first proposed an end-to-end Fully Convolution Network. This paper shows how FCNs for semantic segmentation dramatically improve accuracy by performing end-to-end and pixel-to-pixel operation on images, which would speed up the learning process.

2. **Learning Deconvolution Network for semantic segmentation [2]**

   This paper proposed a semantic segmentation algorithm by learning a deep deconvolutional network. The proposed algorithm reduces the limitations of the existing models that are based on fully convolutional networks by combining proposal-level prediction and deep deconvolution network.

3. **DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs [3]**

   It uses upsampled filters for rich feature extraction called 'atrous convolution'. It uses atrous spatial pyramid pooling which encodes objects and image context at multiple scales. Ideas from deep convolutional neural networks and fully-connected conditional random fields are combined. It advances over state-of-the-art methods in several challenging datasets.

4. **Conditional Random Fields as Recurrent Neural Networks [4]**

   A major issue in deep learning techniques this methodology is the limited capacity of deep learning techniques to delineate or separate visual objects, in our case that would be differentiating between humans and their background. CRF-RNN tries to formulate Conditional Random Fields with Gaussian pairwise potentials and mean-field approximate inference as Recurrent Neural Networks. CRF-RNN, is then provided as a part of a CNN to obtain a deep network that has required properties of both CNNs and CRFs. Importantly, the system fully integrates CRF modelling with CNNs, making it possible to train the hole deep network end-to-end with the usual back-propagation algorithm, avoiding offline post-processing methods for object delineation. It integrates CRF-based probabilistic graphical modelling with emerging deep learning techniques.

5. **Convolutional Random Walk Networks for Semantic Image Segmentation (RWN) [5]**

   Currently many of the semantic segmentation methods rely on Fully Convolutional Networks (FCNs). However, the spatial resolution inside the deep layers is lowered due to large receptive fields and a large number of pooling layers. This leads to predictions with poor localization around the boundaries, which could be a major issue for application such as ours. Random Walk Networks (RWN), jointly optimizes the objectives of pixel-wise affinity and semantic segmentation.

   This leads to the following advantages :
   - produce improved accuracy for the same model complexity.
   - Addresses the issues of
     - poor localization around the segmentation boundaries and
     - spatially disjoint segmentations.
   - presents a better alternative to the denseCRF based approaches.

6. **BoxSup: Exploiting Bounding Boxes to Supervise Convolutional Networks for Semantic Segmentation [6]**

   Pixel-level masks are time consuming and expensive as they require expensive labelling efforts which improves with the amount of data.
   In this paper, bounding box is used in addition to supervision to train the CNN for semantic segmentation.
   The CNN is trained iteratively with approximate masks which helps improve the estimated masks from training.
   This method is less expensive then pixel-level labelling.
   It provides competitive results with state-of-the-art semantic segmentation methods.

III. PROPOSED SYSTEM

3.1 Overview of the System

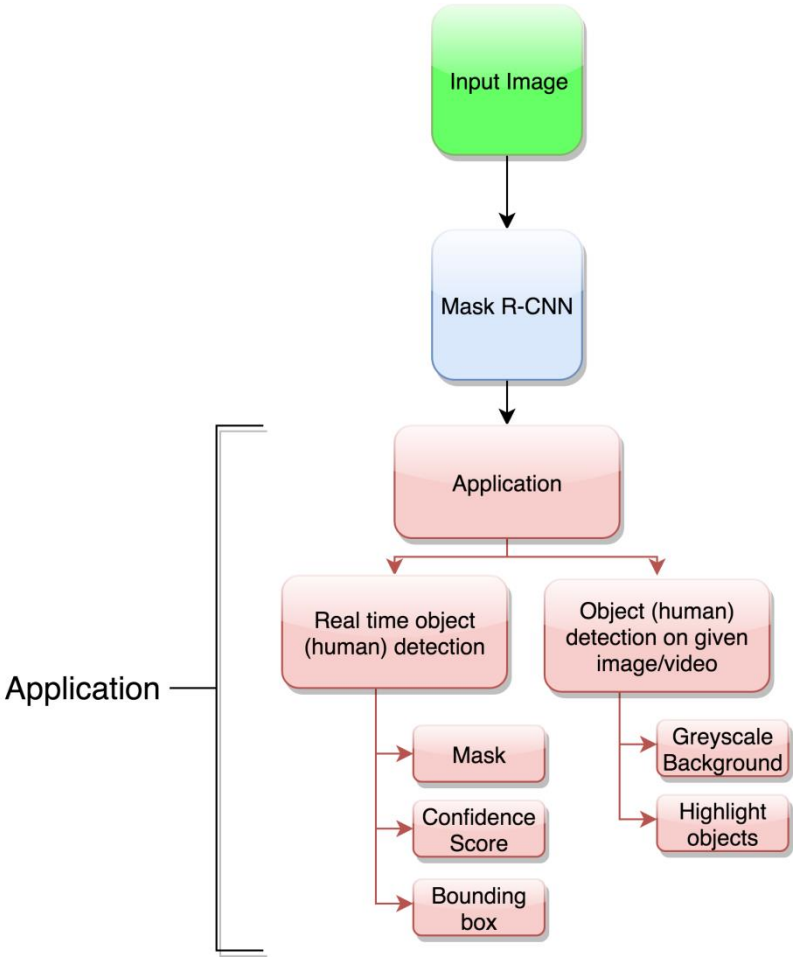

Fig. 1 : Human Detection Application using Mask R-CNN

We use Mast R-CNN [7] to train our datasets, which is further used for detecting humans in foreground and background of our image. This is then extended for Real Time detection of humans using devices such as webcams, CCTVs, self-driving cars etc, or to Object(human) detection on static images/videos.

While using Real time detection, we use OpenCV to visualize the object (human) detection by Mask-RCNN, which will result it real-time bounding boxes, masks, and confidence scores being displayed over the input live feed.

When we use static images or videos as input, the images will be go through Mask RCNN and we will grayscale the background, thus highlighting the main objects (humans), which is called a Splash effect.

For the videos, each frame in a video would be individually segmented through Mask-RCNN, resulting in an output video where just the human subjects in the video will be highlighted with a Splash effect. Unlike real time detection, the video will go segmentation on a frame by frame basis making the overall procedure quite long (varies based on computing power available).

3.2 Description of Components

Mask R-CNN algorithm :

Region-based Convolutional Neural Network (R-CNN) is a convolution algorithm which creates bounding boxes, or regions, using a process called Selective Search. Selective Search finds windows of different sizes, and for each size, it tries to group together, adjacent pixels by intensity, texture or color to identify different objects. In Mask R-CNN, a Fully Convolutional Network (FCN) is added on top of the CNN features of Faster R-CNN to generate a mask (segmentation output).

VGG Image Annotator:

VGG Image Annotator (VIA) is a tool used to annotate images. It is used for specifying region of interest in an image. To train our weights to exclusively detect humans as objects, we used VIA to annotate over 200 images for training purposes, which included outlining humans in a given image to form a polygon which is then exported into JSON file format, and then used for training our weight.

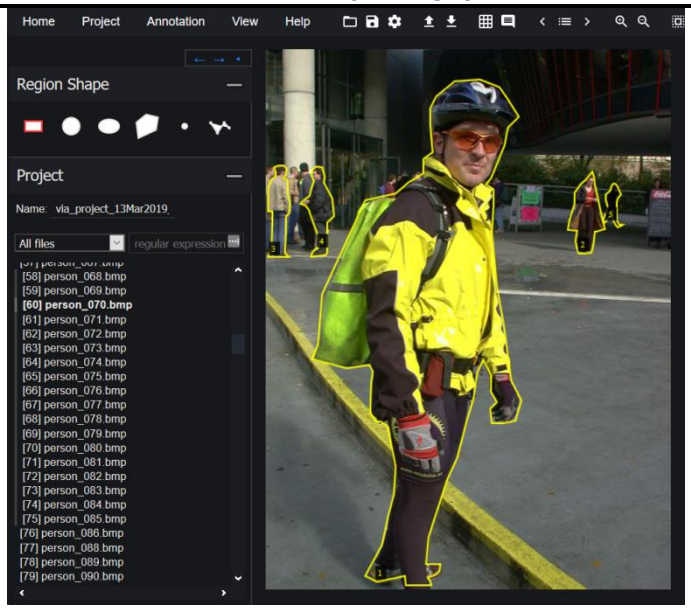

Fig. 2: VGG Image Annotator

Trained Weights :

Training is performed on the training data set, which is the actual dataset used to train the model for performing various actions. This is the actual data the ongoing development process models learn with various algorithms to train the machine to work automatically and accurately. We utilise train and val folders to hold the training and validation dataset, used to train the model.

- Training Dataset: It uses examples of data to fit the parameters of the model.
- Validation Dataset: It uses examples of data used to provide an unbiased evaluation of a model used for tuning the hyperparameters.

This helps us train the model and create weights as output result.

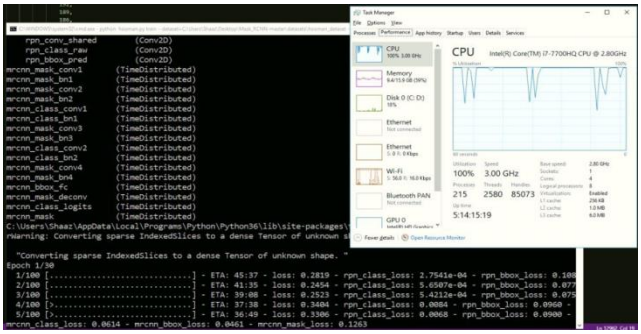

Fig. 3: Training our weights on annotated images

hooman.py:

hooman.py is a python script which is used for training our weights and applying the splash effect on static images and video files using R-CNN algorithm.

Fig. 4 : Splash effect of our secondary application

rt_detection.py:

rt_detection.py is another python script that is used for our main application of Human Detection. We use OpenCV to take in video feed, through devices such as webcams, and applies real-time object (human) detection.

Fig. 5 : Real Time Detection working on webcam feed.

IV. RESULT ANALYSIS

Tensorboard

To visualize the performance our weights, we used Tensorboard. We trained 5 sets of weights over varying number of images, the total images trained were just over 200. Each weight was trained on its preceding weights, i.e. Weight 2 was trained on Weight 1, Weight 3 on Weight 2, etc.

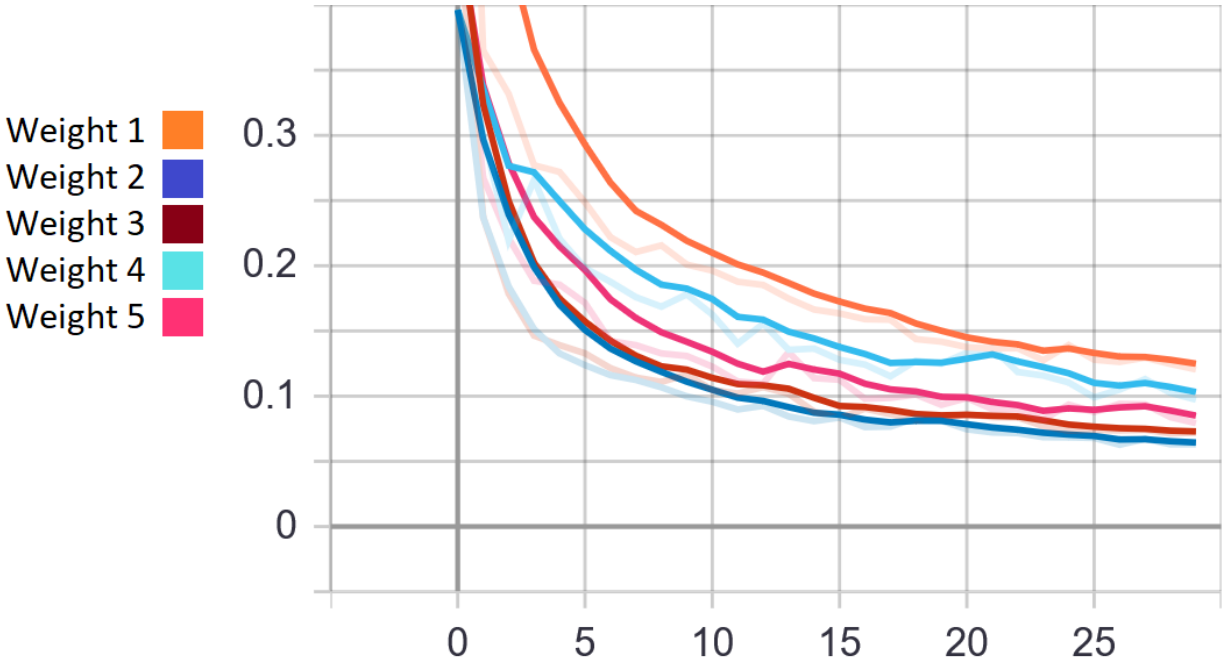

Fig. 6 : main_loss

Every new weight is trained on the previous weight using transferred learning on an mscoco weight as the initial weight. Each training session has an approximate of 40 new images in the train folder with JSON annotated data, keeping the val folder the same. Utilising Tensorboard, we can observe the improvements in our weight as we train more images. We observe that for

Weights 1 and 2, we see a significant improvement in our weights, by looking at a decrease in main_loss (Fig No. 6). Thereafter, Weight 3 shows a decrease, signifying that overfitting has taken place. Overfitting takes place when a model learns the detail and noise in the training data in such a way reduces the performance of the model on new data. Thereafter, we see a slight improvement in weights 4 and 5. From this, we infer that weight 2 is out best weight from training.

Output Images :

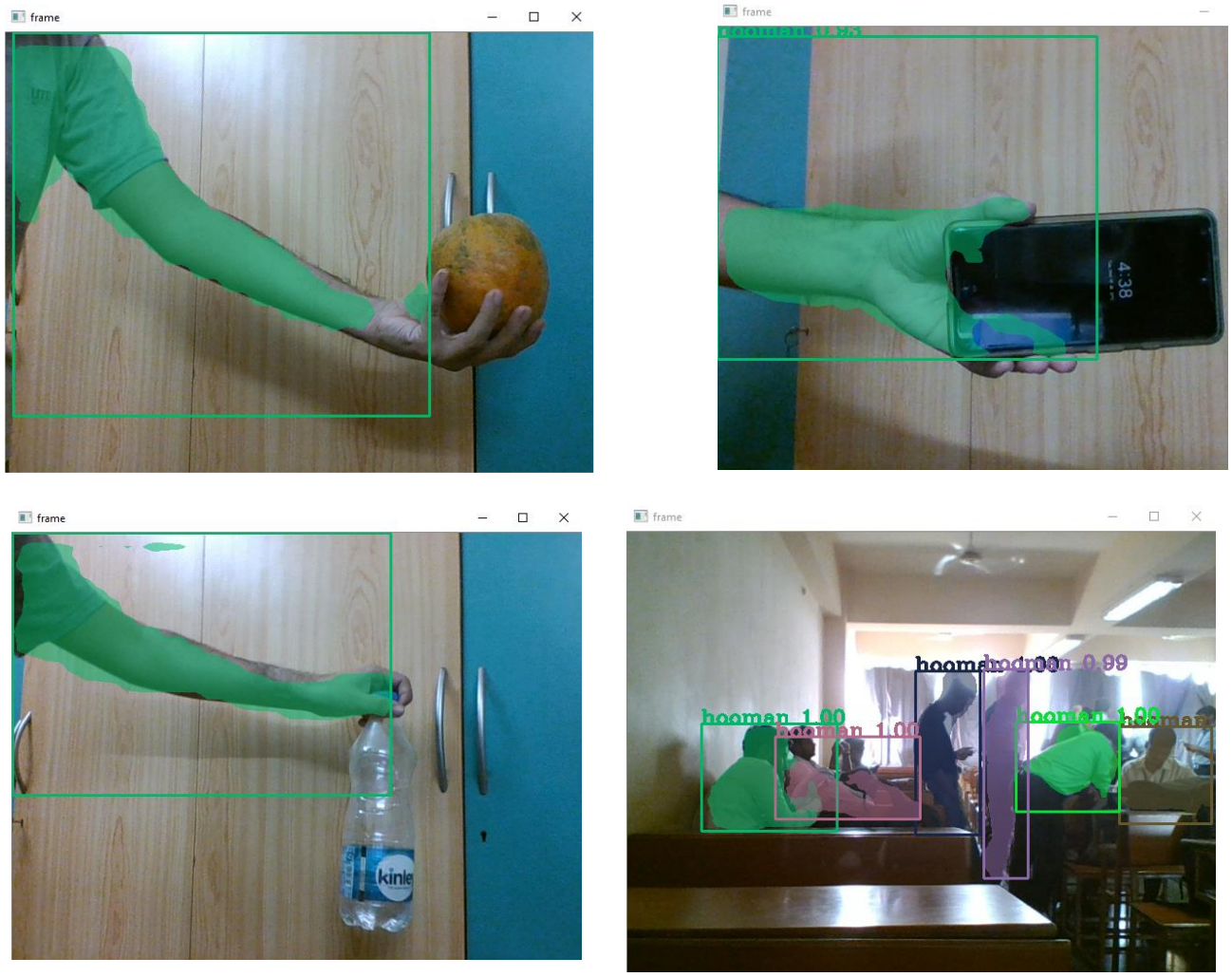

Fig. 7 : Screenshots of Real-Time Detection

As seen in the above images, when we try to detect other objects, the application fails. But it clearly detects the hand holding the other objects. Meanwhile when the application is put in front of a group of people, the application accurately points out all the objects (humans).

Real Time Improvement From CPU To GPU :

Initially we worked with Tensorflow CPU for our project, but as we approached the testing phase, we realized that we had a major performance hit when testing on consumer hardware, like laptops. We saw that the real-time feed was anywhere between 6-8 seconds per frame (or 0.125 FPS). This was not acceptable in our case.

To overcome this issue we had switched to Tensorflow GPU. We installed cuda_10.0.130_411.31_win10 (uses GPU for parallel programming which provides better performance for solving complex problems)for our Nvidia Geforce GPU (GTX 1050 2GB). We then unzipped cudnn-10.0-windows10-x64-v7.4.2.24 (Nvidia CUDA Deep Learning Library) and added the various environment variables. We then installed tensorflow-gpu and ran our program on our GPU.

This resulted in an increased real-time performance of 112.5% to 1 FPS. Scene understanding does not require very high FPS since significant changes do not take place in a very short period of time for our application input, 1 FPS is, thus, enough to recognize the objects (human) from our live input video.

V. CONCLUSION

Various companies, especially those involved in developing self driving cars, try to develop their own application for object detection. It's quite crucial to get it right, since a wrong judgement by a autonomous vehicle can be lethal. Our application uses one of, if not the best, algorithms available to detect human beings in the environment. The exclusivity of detecting only humans, because of the weight training, prevents accidental detections and potential false positives.

The Human Detection Application works just as we planned to, detecting only humans, with no false positives. The future scope, is to improve the weights by training over even more images, on the best weights available, and moving the weight training from consumer grade hardware to server grade hardware, which is highly advantageous when it comes to the time taken to do the same.

REFERENCES

[1] J. Long, E. Shelhamer, T. Darrell, "Fully convolutional networks for semantic segmentation," IEEE Conference on Computer Vision and Pattern Recognition, 2015, 39(4): 640-651.
[2]H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," IEEE International Conference on Computer Vision, 2015.
[3]L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," arXiv preprint arXiv: 1606.00915, 2016.
[4]S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, P. H. S. Torr, "Conditional random fields as recurrent neural networks,"
[5]G. Bertasius, L. Torresani, S. X. Yu, J. Shi, "Convolutional random walk networks for semantic image segmentation," IEEE Conference on Computer Vision and Pattern Recognition, 2017.
[6]J. Dai, K. He, J. Sun, "BoxSup: exploiting bounding boxes to supervise convolutional networks for semantic segmentation," IEEE International Conference on Computer Vision, 2015. IEEE International Conference on Computer Vision, 2015.
[7]Kaiming He, Georgia Gkioxari, Piotr Dollár, Ross Girshick, "Mask R-CNN", arXiv: 1703.06870