A REPORT

ON

HUMAN DETECTION AND SEGMENTATION IN IMAGES AND SURVEILLANCE VIDEOS

UNDERTAKEN AT

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI

APRIL, 2020

HUMAN DETECTION AND SEGMENTATION IN IMAGES AND SURVEILLANCE VIDEOS

This report has been submitted by:

Name: Rohit Jain

ID: 2017A7PS0122P

In partial fulfilment of the course CS F266 Study Oriented Project, Second Semester 2019-20 under Prof. J. Jennifer Ranjani, CSIS Department, BITS Pilani.

ABSTRACT

With technological advancement in society, security systems have become an important aid for us. CCTV cameras are becoming more and more popular and being deployed in high security and regulatory areas. While security systems at public places of utmost importance like airports and railway stations have proven themselves to be robust with high detection rates for potentially harmful objects in the luggage via thermal and infrared imaging, they perform poorly on the concealed objects. In recent years, deep learning models have worked excellently in providing efficient and faster solutions to real-time detection of objects in image and video data with high accuracy rates. We aim to develop an end-to-end deep learning model that takes in real-time data from the surveillance videos (CCTV) and helps in the detection and segmentation of humans with high sensitivity which could be used further in the detection of concealed weapons. We also provide a review of various state-of-the-art algorithms in the broad domain of Computer Vision and compare them on the basis of their results on the COCO dataset for images and TownCentre and VIRAT dataset for pedestrian detection in videos.

ACKNOWLEDGMENT

I wish to express my gratitude towards Prof. J. Jennifer Ranjani, CSIS Department, BITS Pilani for her enthusiastic support, cooperation, and help. Her constant mentorship and useful critiques have helped me progress in the project and provided me with a great learning experience.

I also wish to thank the CSIS Department of BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI for facilitating the course work CS F266 Study Oriented Project which gave me an opportunity to have an academic exposure in the domain of academic research.

TABLE OF CONTENTS

COMPUTER VISION

Computer Vision is a subfield of Artificial Intelligence that deals with training machines to make them interpret their surroundings. Think of what more can be done by a machine when they will be able to see as accurate as a human eye. The human eye is a complex structure and it goes through more complex phenomena of understanding the environment. In a similar fashion, making machines see things and make them capable enough to figure out what they are seeing and further categorize it, is still a pretty tough job. It can be further divided into the following domains:

1. Object Detection
2. Object Segmentation
3. Pose Estimation

and many more...

Fig 1. Computer Vision: Making Machines identify objects

In recent years, deep learning models have become very popular owing to their accuracy rates and faster training process.

HOW DO MACHINES SEE?

Computer Vision is all about converting a picture into pixels, and then try to make sense of what's in the picture through those pixels. However, the major challenge is how to extract information from those pixels and understand what they represent. This process of extracting useful information is called feature extraction.

Feature Extraction

The computer needs to extract useful information and encode them in a way that a computer can understand. An image is defined majorly with the following features:

1. Color Gradients
2. Edges
3. Textures

Each image is represented as a matrix of pixels with a color value where each color is represented as a HEX code in RGB format (Red, Blue, Green) Thus, in 6 digit HEX code black becomes 0x000000 and white becomes 0xffffff.

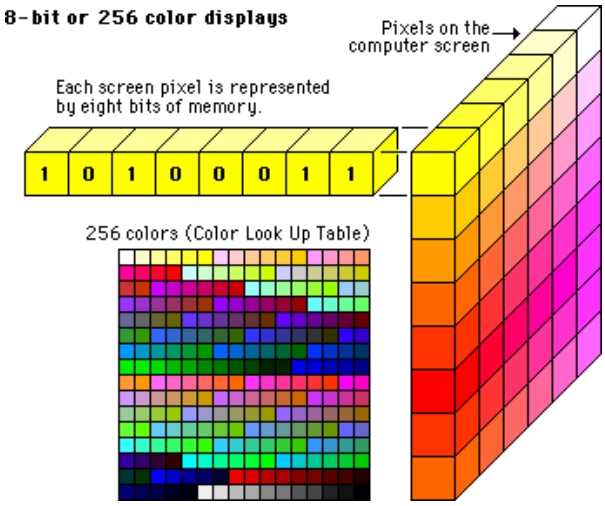


Fig 2. Representing Colors as Numbers

OBJECT DETECTION

We all know about the image classification problem. Given an image can you find out the class the image belongs to? This problem is solved using special kinds of neural networks called Convolution Networks.

But, what makes object detection different from image classification problems and computer vision problems? Fig 3. Below illustrates the difference.

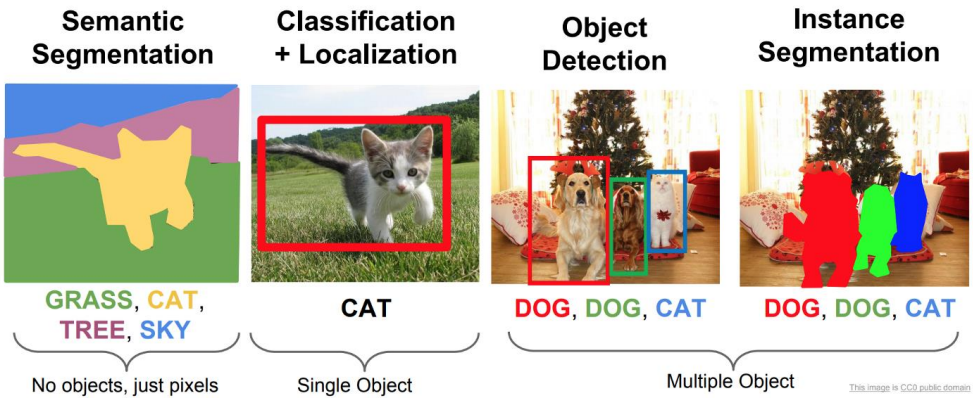


Fig 3. Segmentation, Localization, and Detection

1. **Semantic Segmentation:**

   Given an image, can we classify each pixel as belonging to a particular class?

2. **Classification and Localization:**

   Given an image of an animal can we classify the animal as a cat or dog and build a bounding box around the classified animal?

3. **Object Detection:**

   Given an image, can we detect all the objects in the image and draw bounding boxes around them?

4. **Instance Segmentation:** Can we create masks for each individual object in the image?

OBJECT DETECTION TECHNIQUES

Sliding Window Detectors

One brute force approach for object detection is to slide windows from left and right, and from up to down to iterate through the input image and identify objects using classification. We use windows of all sizes and aspect ratios.

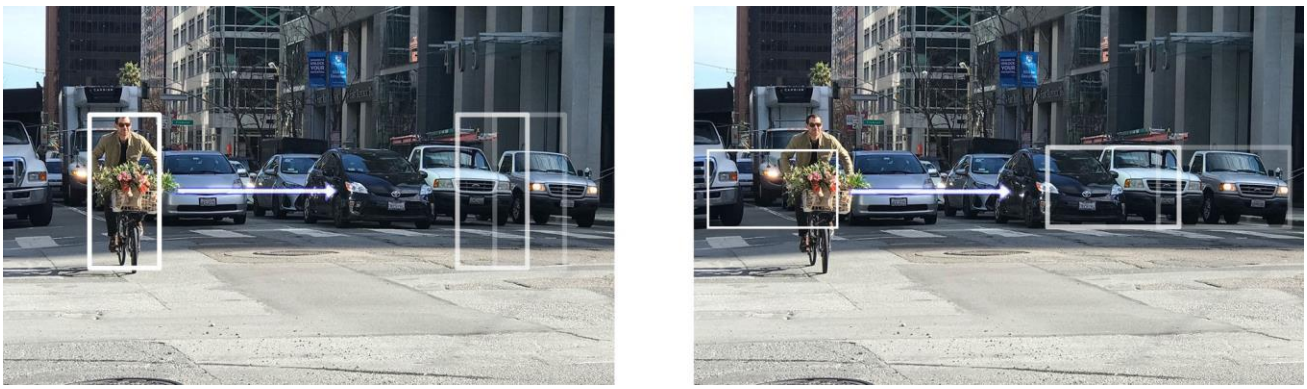


Fig 4. Sliding Window Detector

However, since there are a large number of possible sizes and aspect ratios, this method is not used in practice.

Selective Search Algorithm

To improve performance, one obvious solution is to reduce the number of windows. Thus, instead of a brute force approach, we use a region proposal method to create regions of interest (ROIs) for object detection that tries to group similar pixels together and then propose them as potential object windows or regions of interest. The algorithm can be described in three major steps:

1. Start with each individual pixel as its own group
2. Calculate the texture for each group and combine two that are the closest. Prefer grouping smaller groups first.
3. Continue merging regions until everything is combined together.

Figure 5 below illustrates the above algorithm.

Fig 5. Selective Search Algorithm

R-CNN

R-CNN uses the selective algorithm and uses region proposals to create about 2000 ROIs. The regions are then warped into fixed-size images and fed into a CNN network individually. It is then followed by fully connected layers to classify the object and to refine the boundary box.

With far fewer but higher quality ROIs, R-CNN runs faster and more accurately than the sliding windows. Figure 6 on the next page illustrates the architecture for the R-CNN model.

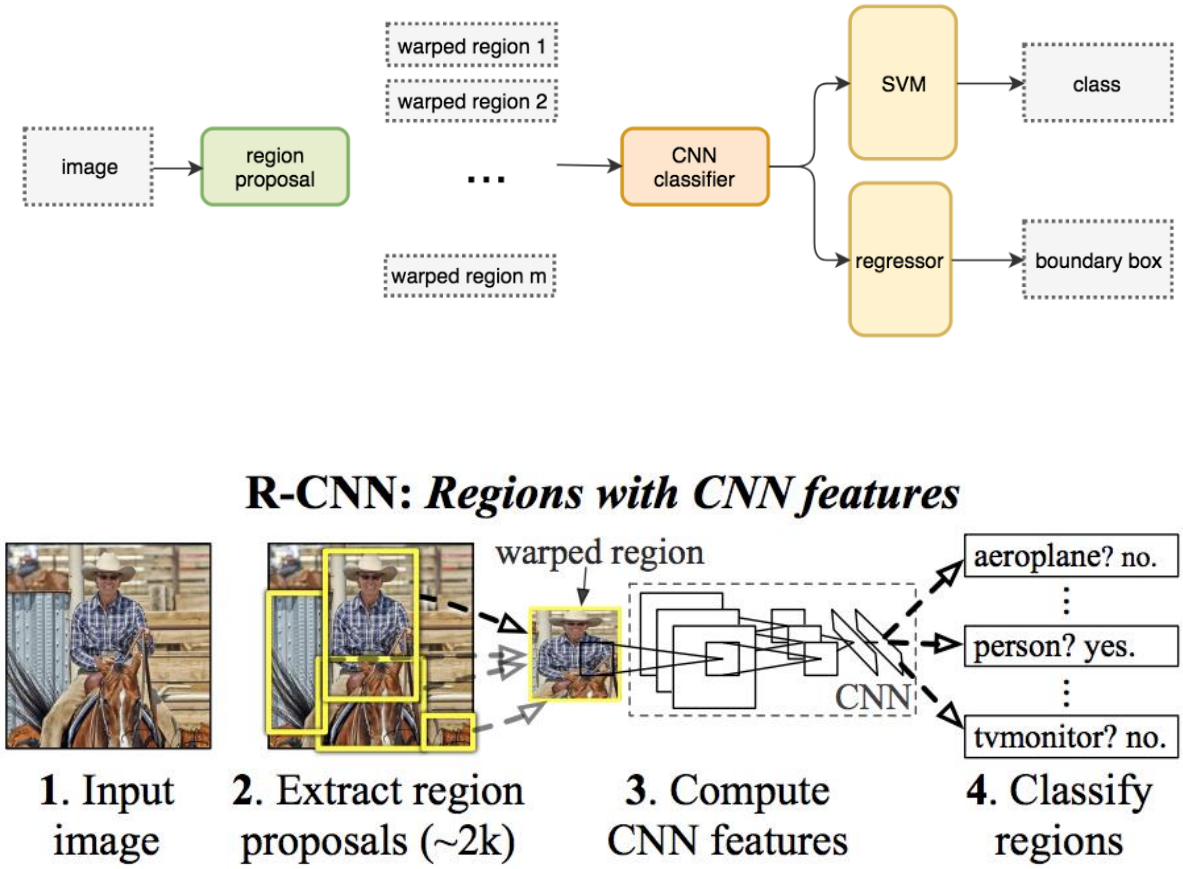


Fig 6. RCNN Architecture

As illustrated in figure 6, the model uses a boundary box regressor because region proposal methods are computation intense. To speed up the process, we often pick a less expensive region proposal method to create ROIs followed by a linear regressor (using fully connected layers) to refine the boundary box further.

While R-CNN provides a major improvement over sliding window detectors they still suffer from some problems. R-CNN is slow in training and inference. If we have 2,000 proposals, each of them is processed by a CNN separately, i.e. we repeat feature extractions 2,000 times for different ROIs. Thus, we need to make it faster.

Fast R-CNN

Fast R-CNN builds upon the R-CNN model by optimizing the feature extraction process. Instead of extracting features for each image patch from scratch, we use a feature extractor (a CNN) to extract features for the whole image first. We then warp the patches to a fixed size using ROI pooling and feed them to fully connected layers for classification and localization. Figure 7 below illustrates the whole architecture. Fast R-CNN is 10x faster than R-CNN in training and 150x faster in inferencing.

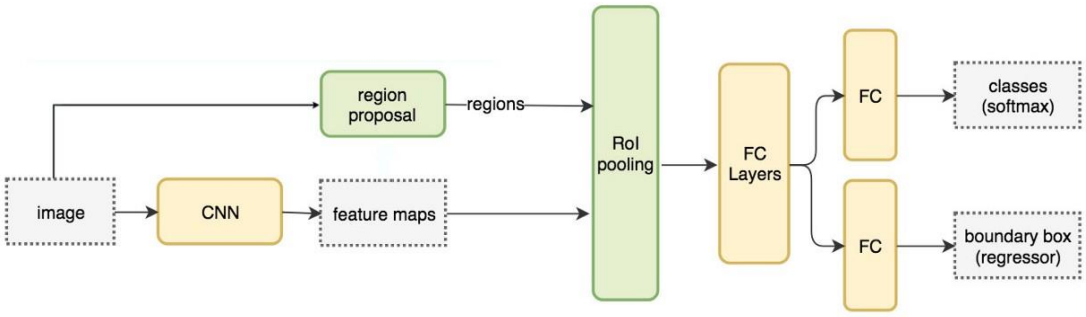
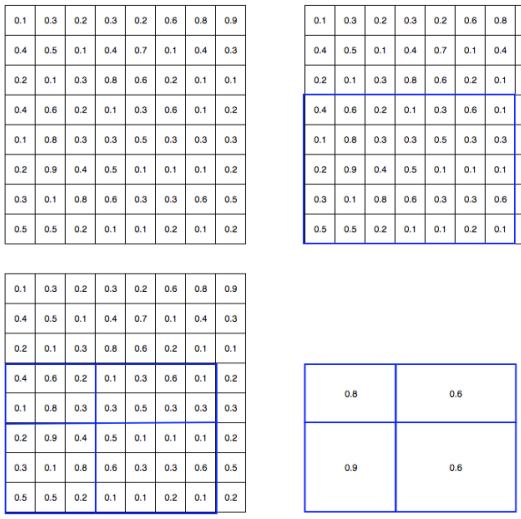


Fig 7. Fast R-CNN Architecture

What is RoI Pooling?



We apply ROI pooling to warp the variable size ROIs into a predefined size shape. Let's simplify the discussion by transforming 8 × 8 feature maps into a predefined 2 × 2 shape. With our 2×2 target, we split the ROIs into 4 sections with similar or equal sizes and find the maximum for each section and the result is our warped feature maps.

Fig 8. RoI Pooling

FASTER R-CNN

Fast R-CNN depends on an external region proposal method like selective search. However, those algorithms run on the CPU and they are slow. In testing, Fast R-CNN takes 2.3 seconds to make a prediction in which 2 seconds are for generating 2000 ROIs.

Faster R-CNN adopts a similar design as the Fast R-CNN except it replaces the region proposal method by an internal deep network and the ROIs are derived from the feature maps instead. The new region proposal network (RPN) is more efficient and runs at 10 ms per image in generating ROIs.

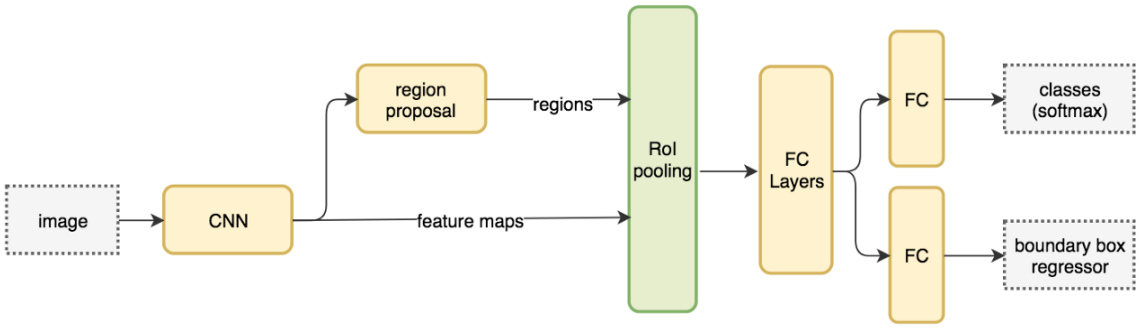


Fig 9. Faster R-CNN

Region Proposal Network

The region proposal network (RPN) takes the output feature maps from the first convolutional network as input. It slides 3 × 3 filters over the feature maps to make class-agnostic region proposals using a convolutional network like the ZF network. The ZF network outputs 256 values, which are fed into 2 separate fully connected layers to predict a boundary box and 2 objectness scores. The objectness measures whether the box contains an object. Faster R-CNN uses a classifier with 2 possible classes: one for the “have an object” category and one without (i.e. the

background class).

The Faster R-CNN Algorithm can be defined as follows:

1. Make an input image and pass it to the ConvNet which returns feature maps for the image.
2. Apply Region Proposal Network (RPN) on these feature maps and get object proposals.
3. Apply ROI pooling layer to bring down all the proposals to the same size.
4. Finally, pass these proposals to a fully connected layer in order to classify any predicted bounding boxes for the image.
5. Generating proposal targets for each location suggested in 4
6. Using 2 and 3 to calculate rpn_cls_loss and rpn_reg_loss.
7. Using 5 and 6 to calculate roi_cls_loss and roi_reg_loss

Anchor Boxes

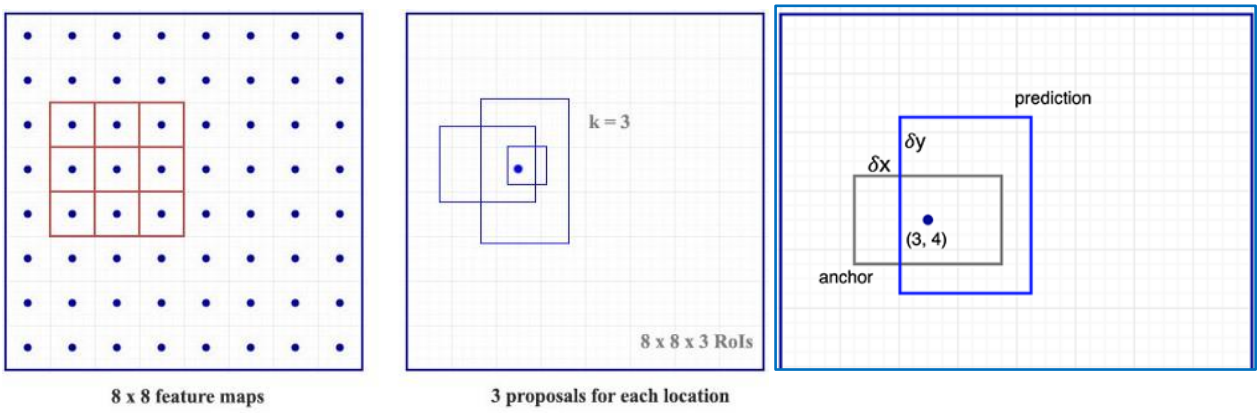


Fig 10. Anchor Boxes

For each location in the feature maps, RPN makes k guesses. Therefore RPN outputs 4×k coordinates and 2×k scores per location. The diagram below shows the 8 × 8 feature maps with a 3× 3 filter, and it outputs a total of 8 × 8 × 3 ROIs (for k = 3). The right side diagram demonstrates the 3 proposals made by a single location.

Faster R-CNN does not make random boundary box proposals. Instead, it predicts offsets like ?x, ?y that are relative to the top left corner of some reference boxes called anchors. We constrain the value of those offsets so our guesses still resemble the anchors.

Performance Comparison

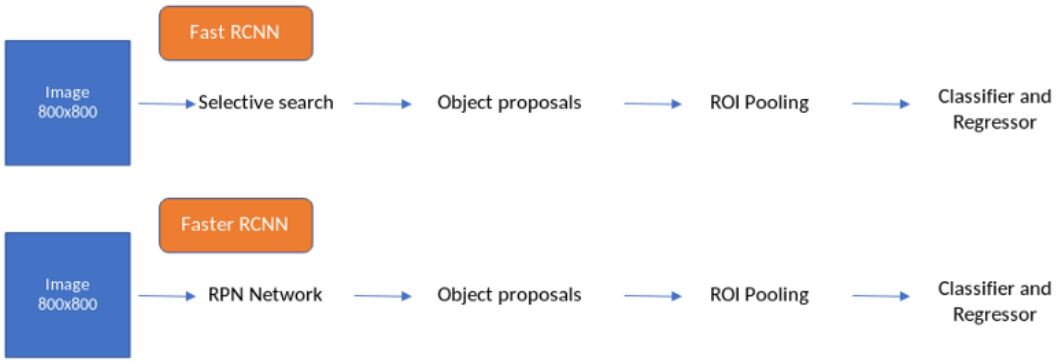


Fig 11. Fast R-CNN vs Faster R-CNN

Figure 12 below shows the testing performance of the three algorithms. Faster R-CNN is approximately 250 times faster than R-CNN.

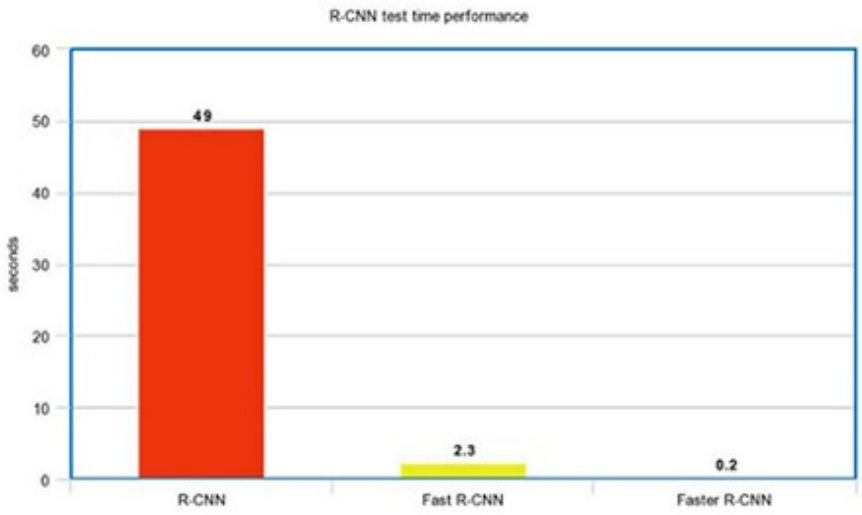
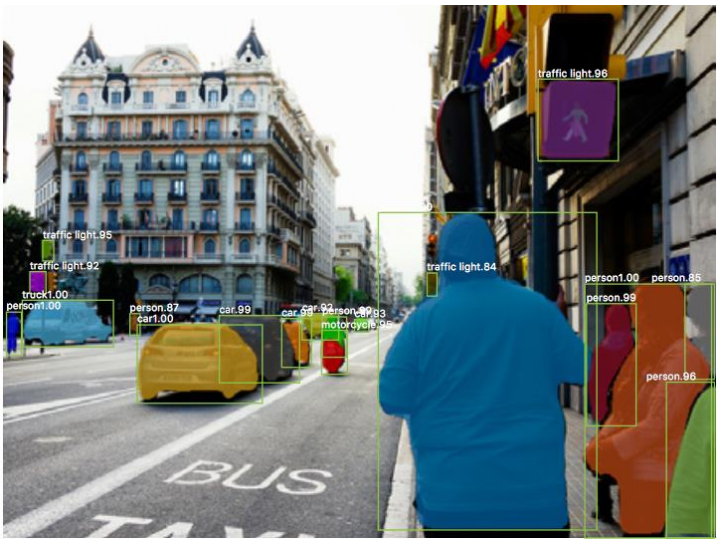


Fig 12. Performance: R-CNN, Fast R-CNN, Faster R-CNN

MASK R-CNN



The main idea behind this algorithm is if we can extend the above techniques to go one step further and locate exact pixels of each object instead of just bounding boxes? That is can we create a boundary around the objects?

Fig 13. Segmentation using Mask R-CNN

Mask R-CNN does this by adding a branch to Faster R-CNN that outputs a binary mask that says whether or not a given pixel is part of an object. The branch, as before, is just a Fully Convolutional Network on top of a CNN based feature map. Once these masks are generated, Mask R-CNN combines them with the classifications and bounding boxes from Faster R-CNN to generate such wonderfully precise segmentation

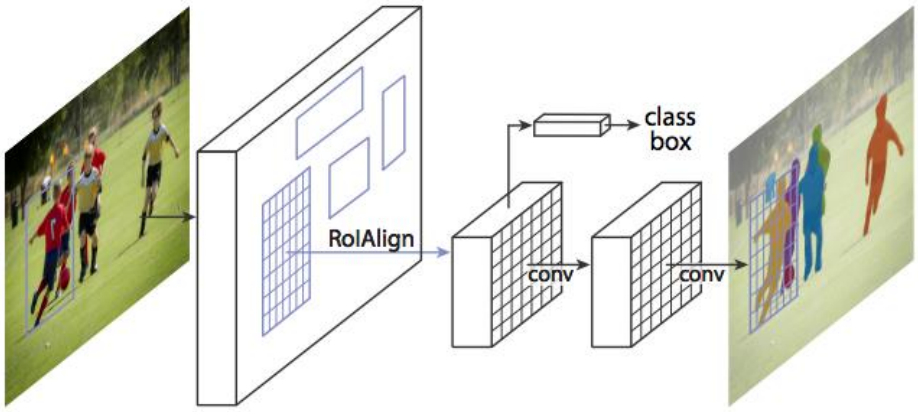


Fig 14. Mask R-CNN architecture

METHODOLOGY

Human pose estimation and segmentation are important information to have a better understanding of the human activity. There is a lot of research focused on this topic. One of the most popular deep learning methods is Mask R-CNN which is a simple and general framework for object instance segmentation.

Problem Statement

Given the real-time data from CCTV and other security systems at public places, detect and segment all the humans in the video data with high sensitivity. We consider video as a series of frames and model this as a Human Detection and Segmentation Problem in Images.

Experiments

We use Faster R-CNN and Mask R-CNN for human detection in both images (COCO Dataset) and video data (TownCentre and VIRAT Dataset).

FASTER R-CNN

Instead of using the ZF network as used by the authors, in our experiments, we use the VGG16 network as a feature extraction module. This acts as a backbone for both the RPN network and the Fast_R-CNN network. We need to make a few changes to the VGG network in order to make this work. We need to check here the VGG16 module is achieving this feature map size and trim the network. Thus,

Output image size = Input Image size/sub-sample

At each of the anchor locations, we take a combination of scales 0.5x, 1x,

and 2x with the aspect ratios being one of 1:2, 2:1, or 1:1. RPN produces object proposals wrt to each anchor box along with their objectness score. An anchor box is assigned +ve if it has max_iou with the ground truth object or iou greater than 0.7. An anchor box is assigned negative if it has iou < 0.4. All the anchor boxes with iou [0.4, 0.7] are ignored. Anchor boxes that fall outside the image are also ignored.

Smooth L1 loss and cross-entropy loss were used for regression and classification.

Results

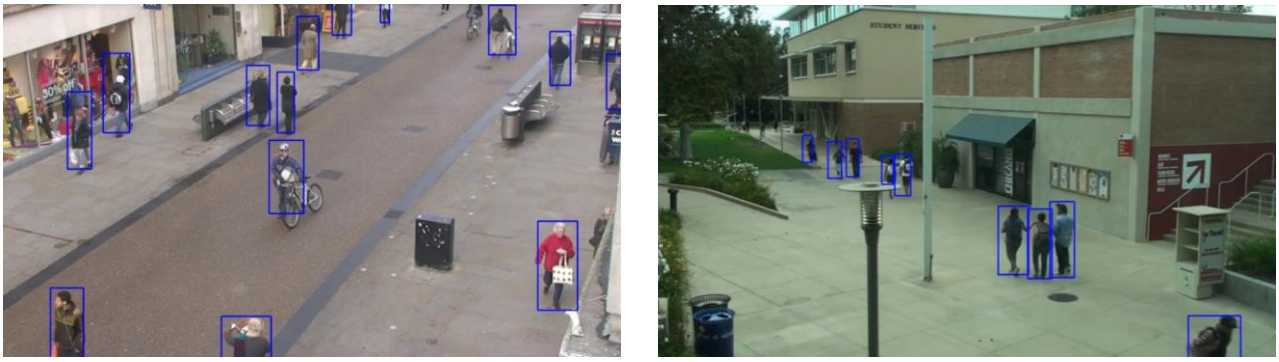


Fig 15. Faster R-CNN Result: Town Centre and VIRAT Video Dataset

While Faster R-CNN worked efficiently on the video datasets it followed from two major problems:

1. Since a video is a series of frames, Faster R-CNN detects humans in each frame individually and combines all the generated frames into one video. Due to this, the boundary boxes suffer from a jittering effect in the output video.
2. In frames with high occlusion between two humans, the bounding box combines into one.
3. Faster R-CNN gives poor results on humans on a small scale like the ones far away from the camera.

MASK R-CNN

Mask R-CNN authors realized that the regions of the feature map selected by RoIPool were slightly misaligned from the regions of the original image. Since image segmentation requires pixel-level specificity, unlike bounding boxes, this naturally led to inaccuracies. This problem by cleverly adjusting RoIPool to be more precisely aligned using RoIAlign. In RoIPool, we would round this down and select 2 pixels causing a slight misalignment. However, in RoIAlign, we avoid such rounding. Instead, we use bilinear interpolation to get a precise idea of what would be at pixel 2.93. This, at a high level, is what allows us to avoid the misalignments caused by RoIPool.

Once these masks are generated, Mask R-CNN combines them with the classifications and bounding boxes from Faster R-CNN to generate such wonderfully precise segmentation thus, solving all the above three problems.

Results

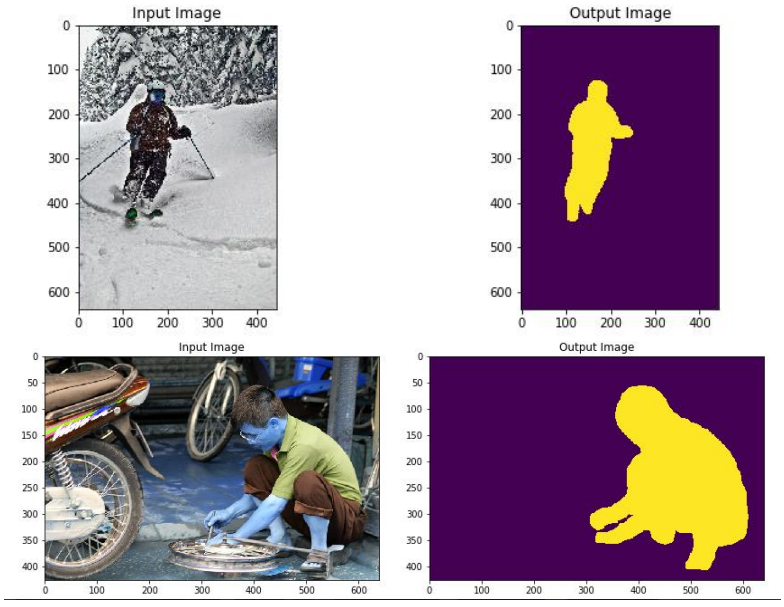


Fig 16. Mask R-CNN Results: Coco Dataset

Pose2Seg Model

Problems with Mask R-CNN

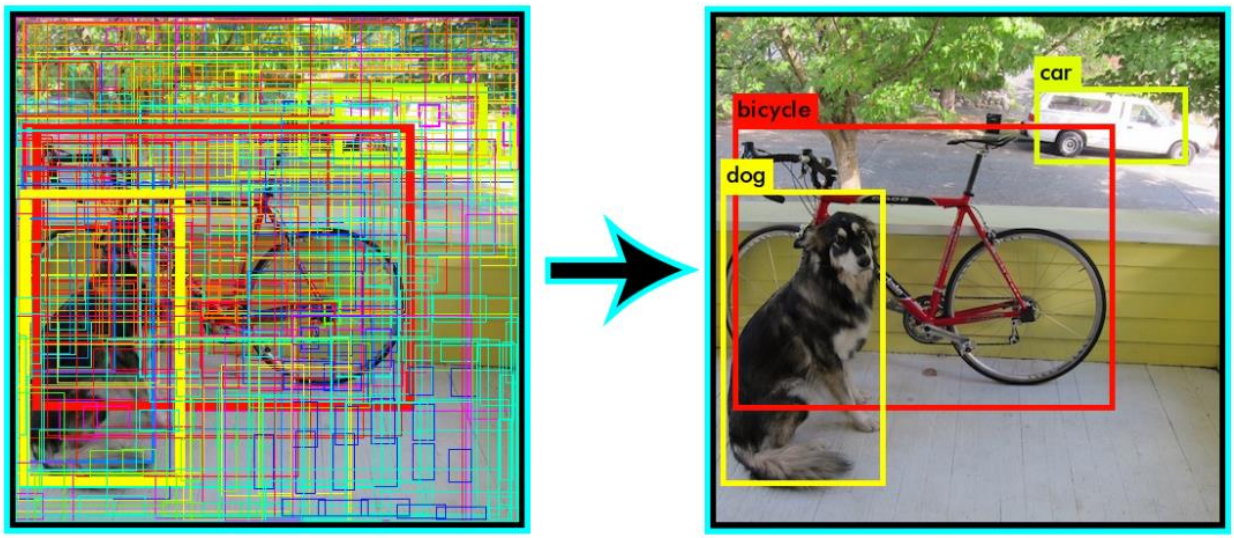


Fig 17. NMS Suppression in Mask R-CNN: A drawback

All the algorithms discussed till now follow a top-down strategy that is they perform the object detection first, then remove the redundant regions using Non-maximum Suppression (NMS) and segment the object from the detection bounding-box. When two objects of the same category have a large overlap, NSM will treat one of them as a redundant proposal region and eliminate it. However, “Human” is a special category and can be defined based on the pose skeleton. The current methods like Mask-RCNN don’t take advantage of the pose information for segmentation.

Using Pose Estimation for Human Segmentation

The main idea that this model argues is to use bottom-up approaches which are based on the pose information rather than the bounding box detection. Since each human has its own pose, this solves the problem of high occlusion in our video datasets.

We would first detect key points for each body part for all the people and then group or connect those parts to form several instances of human pose, which makes it possible to separate two intertwined human instances with a large overlap.

POSE2SEG: ALGORITHM

1. The model takes both the image and human pose as input. The human pose can be the output of other methods such as OpenPose or the ground truth of the dataset.
2. The entire image is passed through a base network to extract features of the image.
3. An align module, Affine-Align, is used to align Regions of Interest to a uniform size. You can imagine that this module will extract multiple fixed-size regions from the large image. Each fixed-size region corresponds to each human in the image.
4. The Affine-Align region will perform affine transformations to align each pose to one of the pose templates.
5. The aligned output of the Affine-Align will be concatenated with Skeleton Features and feed toward a SegModule to generate the segmentation mask.
6. Skeleton features: are simply the affinity fields which is a 2-channel vector field map for each skeleton. This is the output of OpenPose.
7. SegModule is a CNN network that starts with a 7 x 7 stride-2 Conv layer and is followed by several standard residual units. Then, a bilinear upsampling layer is used to recover the resolution and 1 x 1 Conv layer is used to predict the mask result.
8. Finally, the segmentation masks for each person are combined into one final segmentation mask using the reverse of an affine transformation.

Figure 18 below illustrates the pose2seg algorithm.

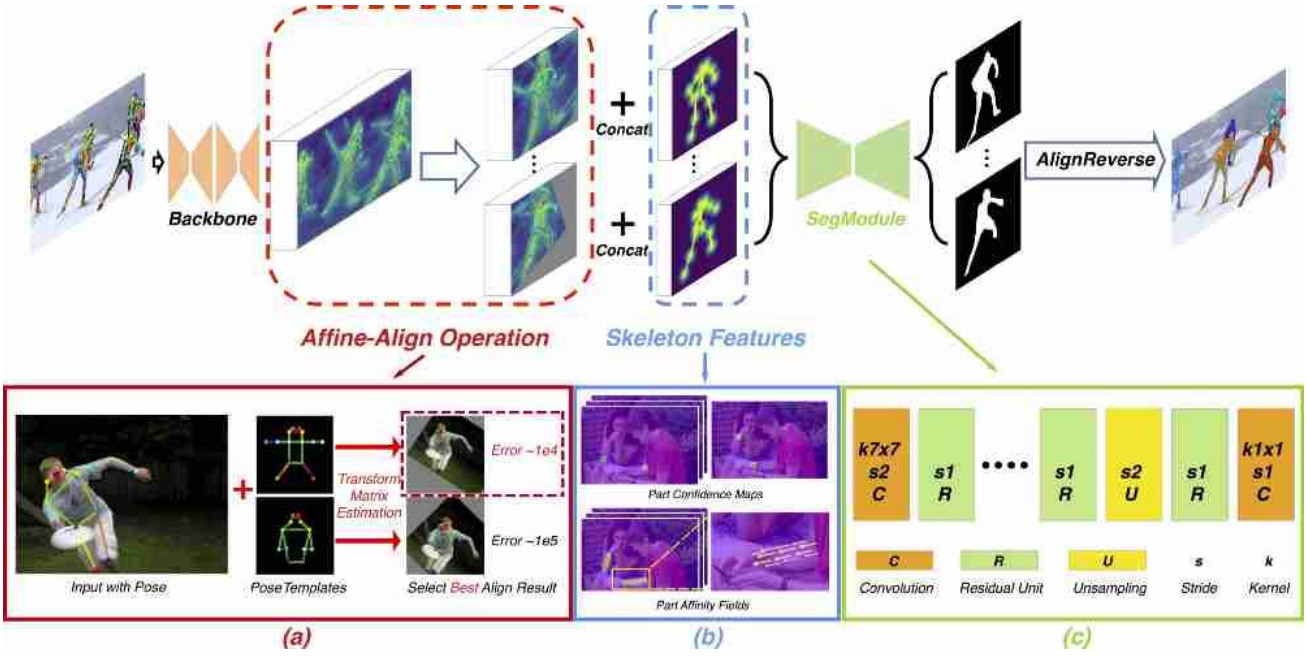


Fig 18. Pose2Seg Architecture

Affine Align Operation

The Affine-Align operation is mainly inspired by the RoI-Pooling presented in Faster R-CNN and RoI-Align in Mask R-CNN. But, while those align humans according to their bounding-boxes, Affine-Align is used to align based on the human pose.

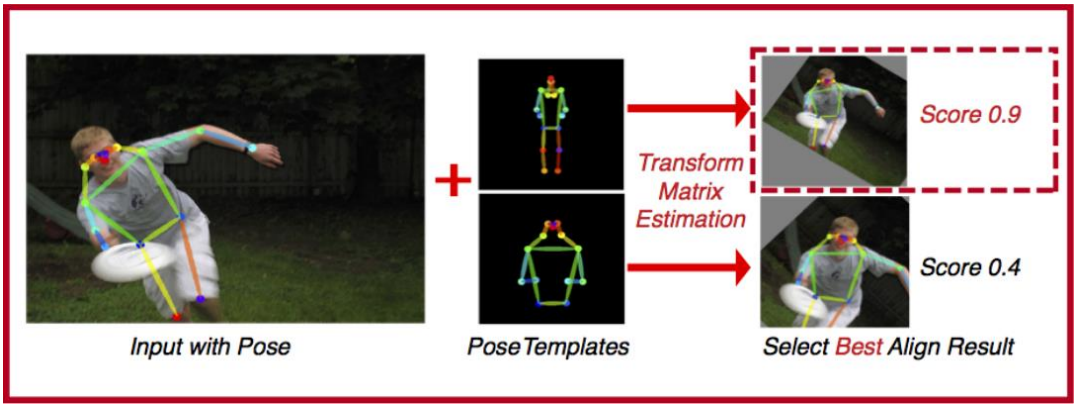


Fig 19. Affine Align Operation

As illustrated in figure 19, the most frequent human poses are stored offline, later to be compared with every input pose at training/inference. The idea is to choose the best template for each estimated pose. This is accomplished by estimating an affine transformation matrix H between input pose and the templates and choosing the one yielding the best score.

$$H^* = \arg\min_H \|H \cdot P - P_\mu\|.$$
$$score = \exp(-\|H^* \cdot P - P_\mu)\|)$$

Here P_u represents a pose template and P represents a single person pose estimation. The matrix H* is the affine transform chosen for the best suited per pose template. Finally, The transformation H* that results with the best score is applied to the image or features and transforms it to the desired resolution.

Pat Affinity Fields

A PAF is a set of 2D vector fields that encode the location and orientation of the limbs. In essence, a PAF is a set of vectors that encodes the direction from one part of the limb to the other; each limb is considered as an affinity field between body parts.

Fig 20. Part Affinity Fields

Skeleton Features

Figure 20 below shows the skeleton features. For this task, part affinity fields (PAFs) are adopted. The output from PAF is a 2-channel vector field map for each skeleton. PAF is used to represent the skeleton structure of a human pose along with a part confidence map for body parts to emphasize the importance of those regions around the body part key points.

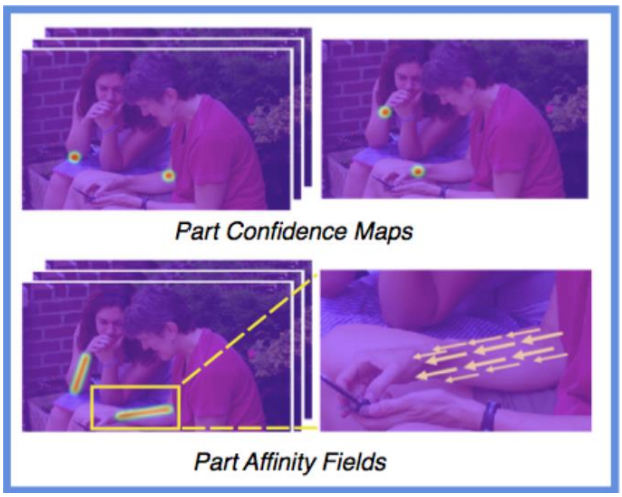


Fig 20. Open Pose and PAF Algorithm

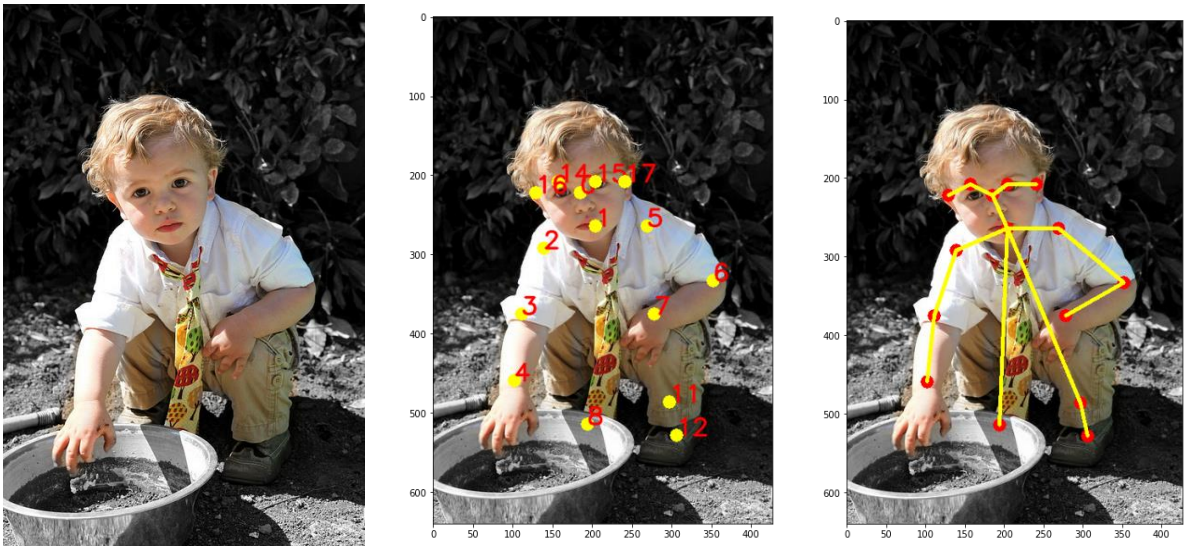
Segmentation Module

The network starts with a 7 × 7, stride-2 convolution layer, and is followed by several standard residual units to achieve a large enough receptive field for the ROIs. After that, a bilinear up-sampling layer is used to restore the resolution, and another residual unit, along with a 1 × 1 convolution layer is used to predict the final result. Such a structure with 10 residual units can achieve about 50 pixels of the receptive field, corresponding to the alignment size of 64 × 64. Fewer units will make the network less capable of learning, and more units enable little improvement in the learning ability.

RESULTS

We obtained the following results on COCO Dataset for the Human Pose Estimation using the OpenPose Algorithm.

Single Person Pose Estimation



(a)Input Image (b) Keypoint Detection (c) Pose Structure

Fig. 21 Results: Single Person Pose Estimation

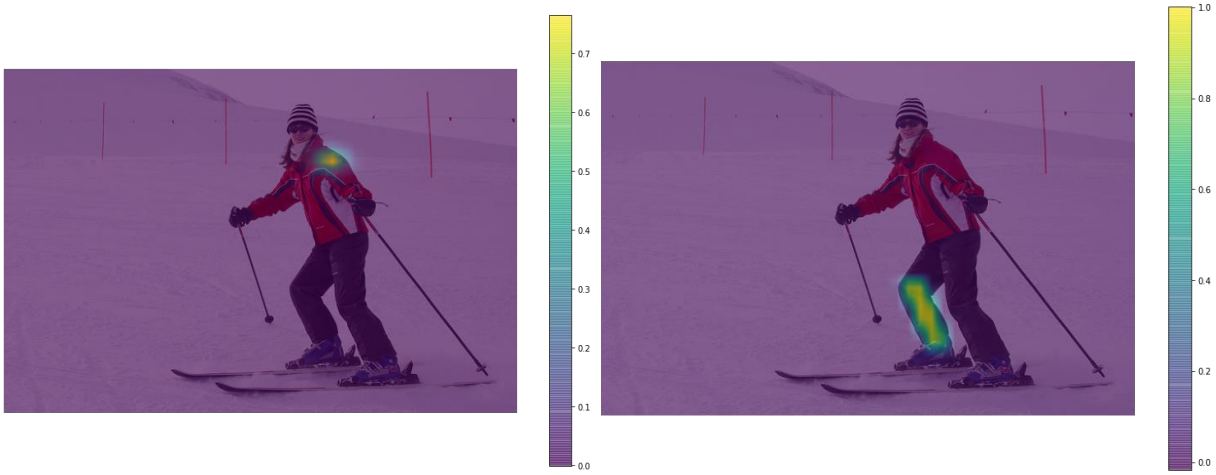


Fig 22. Process of Part Affinity Fields Detection

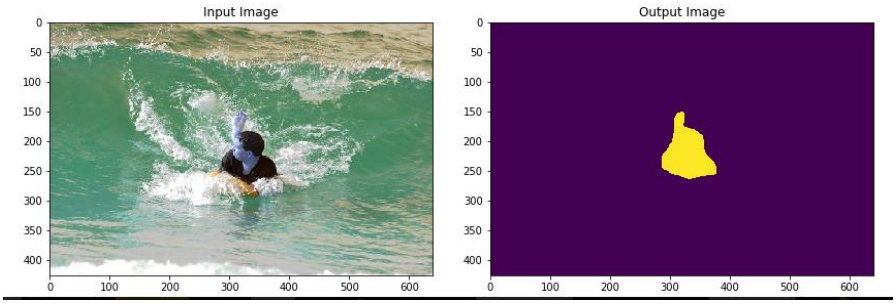
Multi-Person Pose Estimation

(a) Input Image (b) Part Affinity Fields

(c) Keypoint Detection (d) Pose Estimation

Fig 23. Pose Estimation using OpenPose

Human Detection using Pose2Seg on COCO Dataset



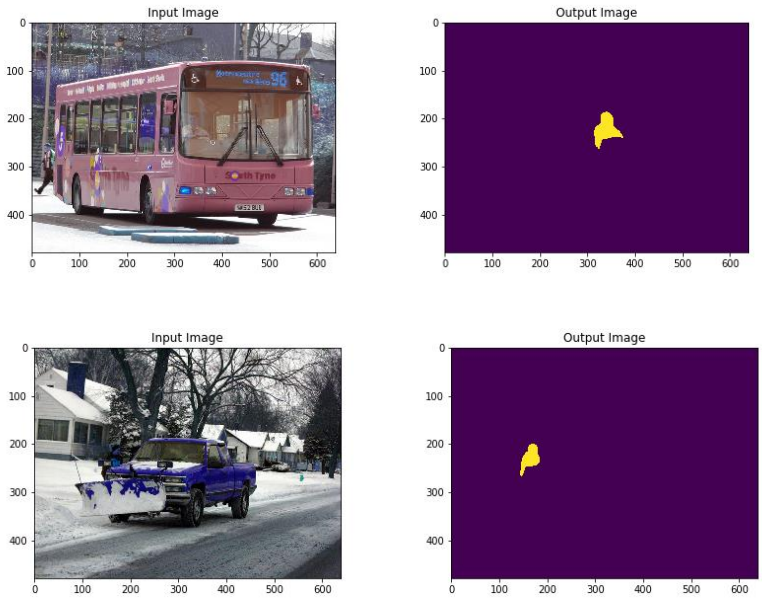


Fig 24. Human Detection: Pose2seg

As can be seen in Fig 24, Pose2Seg works excellently on the images where only a part of a human is visible. Since Pose2Seg is based on part affinity fields and pose estimation, the model accurately detects the humans even if only a few limbs might be present.

Conclusion

With COCO, TownCentre, and VIRAT dataset, we analyzed three popular algorithms in the domain of object detection. We achieved high accuracy using Faster R-CNN, however, the output suffers from jittering and merger of two very close windows. Mask R-CNN proves to be a robust algorithm taking an average of 0.2s testing time. However, it works poorly on highly occluded data. Pose2Seg takes a reverse bottom-up strategy for human detection via pose estimation. While Pose2Seg works best with the data, it takes a comparatively longer time (on the order of 2s). Thus, in cases of high occlusion as in our problem, Pose2Seg proves to be the best model.

References

1. A Large-scale Benchmark Dataset for Event Recognition in Surveillance Video" by Sangmin Oh, Anthony Hoogs, Amitha Perera, Naresh Cuntoor, Chia-Chih Chen, Jong Taek Lee, Saurajit Mukherjee, J.K. Aggarwal, Hyungtae Lee, Larry Davis, Eran Swears, Xiaoyang Wang, Qiang Ji, Kishore Reddy, Mubarak Shah, Carl Vondrick, Hamed Pirsiavash, Deva Ramanan, Jenny Yuen, Antonio Torralba, Bi Song, Anesco Fong, Amit Roy-Chowdhury, and Mita Desai, in Proceedings of IEEE Computer Vision and Pattern Recognition (CVPR), 2011.
2. Benfold, Ben, and Ian Reid. "Stable multi-target tracking in real-time surveillance video." CVPR 2011. IEEE, 2011.
3. Cao, Zhe, et al. "OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields." arXiv preprint arXiv:1812.08008 (2018).
4. Forsyth, David A., and Jean Ponce. Computer vision: a modern approach. Prentice-Hall Professional Technical Reference, 2002.
5. Girshick, Ross. "Fast r-cnn." Proceedings of the IEEE international conference on computer vision. 2015.
6. Uijlings, Jasper RR, et al. "Selective search for object recognition." International journal of computer vision 104.2 (2013): 154-171.
7. He, Kaiming, et al. "Mask r-cnn." Proceedings of the IEEE international conference on computer vision. 2017.
8. Lin, Tsung-Yi, et al. "Microsoft coco: Common objects in context." European conference on computer vision. Springer, Cham, 2014.
9. Ren, Shaoqing, et al. "Faster r-cnn: Towards real-time object detection with region proposal networks." Advances in neural information processing systems. 2015.
10. Zhang, Song-Hai, et al. "Pose2seg: detection free human instance segmentation." Proceedings of the IEEE conference on computer vision and pattern recognition. 2019.