# *OBJECT DETECTION*

# APPAREL WAREHOUSE DETECTION

Introduction

* Object Detection is one of the most famous and extensively researched topics in the field of Machine Vision.
* To understand Object Detection in simplistic terms, it deals with identifying and localizing some of the classes such as person, car, bus, spoon, etc. from the image. This can be achieved by drawing a bounding box around the given specific target class.
* In this article, let’s have a proper understanding of these trends which are followed in Deep and other Traditional Learning approaches. We will also be looking at the upsides and downside of both the approaches in a modularized fashion — Object Detection.

*So tighten your seatbelt and get ready for the ride :)*.

Literature Survey

* Problem statement of Object Detection can be stated as: to determine the locations of the objects and the classes to which it belongs to. To accomplish this task, any object detection method viz. Deep and “*not so Deep*” Learning can be stated in three given steps:

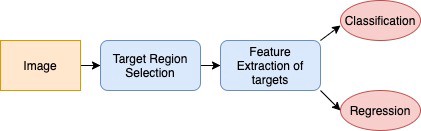


Fig 1: Sufficient conditions to complete an object detection algorithm

* Let’s understand the above-given pipeline by following an *old school*method, a.k.a combination of traditional computer vision and machine learning classification algorithms.

— Target Region Selection

* Region selection in the traditional method is mainly done by brute force sliding window technique on an image. This sliding window has a fixed size and shape. It slides over an image and gets the crops.

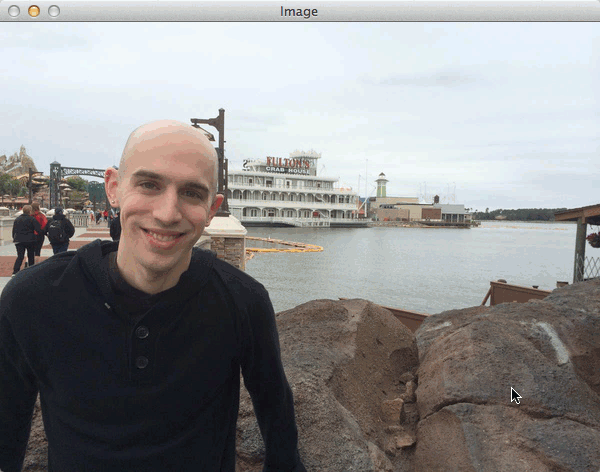


Fig 2: Source: [[Link](https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.pyimagesearch.com%2F2015%2F03%2F23%2Fsliding-windows-for-object-detection-with-python-and-opencv%2F&psig=AOvVaw3-vJuNA6s4haNb5kyY_LEA&ust=1587914801907000&source=images&cd=vfe&ved=0CAIQjRxqFwoTCLjc59_xg-kCFQAAAAAdAAAAABAe)]

* Sounds simple right…Problem solved. Okay, but it’s not that straight forward for the generic scenarios. There are many objects which need to be classified and have different aspect ratios, sizes, and positions in an image. Finding a perfect window for every object in an image is very computationally exhaustive and produces too many redundant windows, which can further slow-down the blocks in our given pipeline. So now, what if we take a fixed number of sizes and template windows and slide over the image. Yeah, it will decrease the time constraint, but it will not take into account the same object at different scales. Being said that, we will run in the same problematic loop again and again.

— Feature Extraction of Targets

* Feature extraction is the brain of the given pipeline. After getting the crops from the above step, now we need to analyze and learn the sematic and visual representation of every object in an image. This can be accomplished by good old feature descriptors (Local and Global) such as SIFT**[1]**, HoG**[2]** and Harr-Like**[3]** features descriptors.
* These descriptors can be tweaked for the different objects in general and can give some very promising results. But due to the variabilities in the appearance of an object due to noise, scale, illumination, occlusion, it becomes very cumbersome to manually design and tweak the feature descriptors of each object.

— Classification/Regression

* The final step of our pipeline is to classify the existing crops using the obtained feature descriptor values and assign a crop to a given class. Simultaneously, drawing a bounding box around an object in a given image. Most of the well-known classification techniques used are Support Vector Machines**[4]**, AdaBoost**[5]**, Random Forest**[6]**, etc.
* These models need far more information about a class and so, tedious tweaking is needed to get good results. For example, SVM generally does not support class probability discrimination. So, it becomes very tedious in multi-class classification. These methods also fail in generalizing the data i.e. SVM generally performs very bad on the data containing noise and have overlapping data points.

But after the advent of theCNNs and Deep Neural Network architecture, it has become more convenient and reliable to fill the gaps which are present in the traditional object detection algorithms.

With the availability of petabytes of data and the “*deeper*” architecture of the neural networks, more and more complex features are learned automatically which helps us to fill the gap which we were facing *Feature Extraction of Targets* module.

Also, thanks to extensive training approaches that help to learn more informative object representations, removing the problem to learn features per-object manually. Let’s have a look at some famous architectures and end-to-end method of object detection. Let’s go “**DEEPER**”.

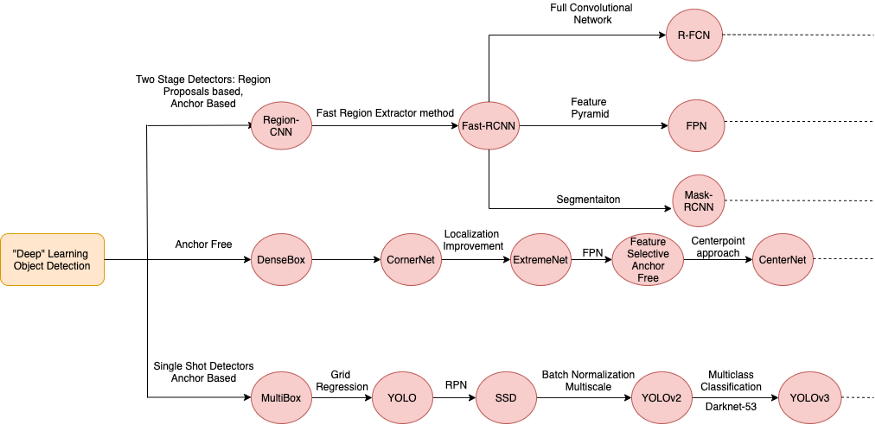


Fig 3: Different types of Object Detection Architectures/Methods.

As shown, there are as of now, two types of object detection methods available(*Actually three, thanks to science, but let’s wait for that ;)*).

i) Two-Stage Detectors: Region Proposal Based.

ii) One-Shot Detectors: Regression-Based.

So, let’s understand the pipeline for each of these types.

**— Target Region Selection**

1. **Two-Stage Detectors**

* **Selective Search:**As we discussed in traditional methods for the region selection, instead of using the redundant window slides, we take a pixel-based approach. This method deals with the merging of similar pixels based on texture information using Merge-Set Data Structure. We can see from the given below figure how different pixels are combined to form different similar regions. This is also known as Super Pixel Segmentation and can be done using the Graph-Cut algorithm**[7]**.
* Okay but let’s get to the downside of this method. After getting proposals, these are fed into CNN for feature extraction. If 500 proposals are obtained, each of these is then fed into a simple Convolutional Neural Network for further feature extraction. These make training and inference very slow because of overlapping regions and redundant feature extraction for all the proposals. This method was first used in R-CNN**[8]**.

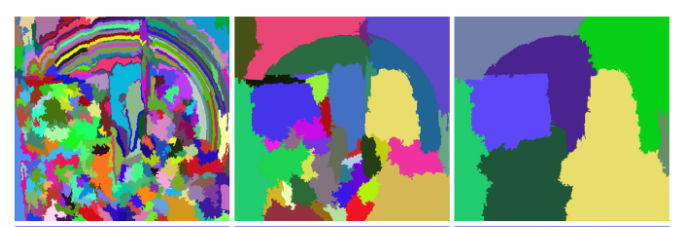


Fig 4: Amalgamation of the same region pixels(Source: [[Link](https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.researchgate.net%2Ffigure%2FSelective-search-example-The-first-row-are-the-combined-regions-the-blue-and-green_fig2_331716356&psig=AOvVaw0lTT_YhGtsOMf2eficV5--&ust=1587917899924000&source=images&cd=vfe&ved=0CAIQjRxqFwoTCMiSsqb9g-kCFQAAAAAdAAAAABAD)]).

* **Fast RCNN[9]**(Removing redundant forward passes in CNN): To solve the above method, instead of passing an ROI patch to CNN every time, we use feature extractor for the whole image first. We then use the region extractor method such as *selective search* and extract the patches from the feature maps generated. The process can be seen from below given figure. This method helps to cut down the redundant forward passes of every patch and helps in a drastic cutdown of the processing time.

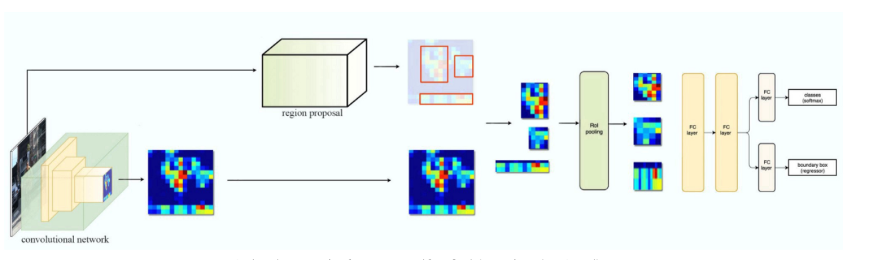


Fig 5: Source: [[Link](https://miro.medium.com/max/2000/1*Wmw21tBUez37bj-1ws7XEw.jpeg)]

* **Region Proposal Networks**: Training/Inferencing using a *selective search* algorithm is very time consuming as it runs on CPU. So it is not feasible to run this algorithm in real-time. To solve this con, Region Proposal Networks comes into action. These networks are trained end-to-end using a lightweight CNN to generate ROIs from feature maps instead of using raw high dimensional images. Due to the trainable feature of this network and it’s tweaking of hyper-parameters, it can generate more number of ROIs in very less time. This was first introduced in Faster RCNN**[10]**.

2. **One-Shot Detectors**

***Get yourself lucky***. Fortunately, this step is skipped in single-shot detectors. As these detectors do not depend on the region proposals, it predicts the limited fixed amount of proposals at a given time from an image and directly undergoes global regression/classification, mapping straightly from image pixels to bounding box coordinates and class probabilities. These types of models/networks are kind of tremendously fast but at a cost of decreasing accuracy.

— Feature Extraction from Targets

1. **Two-Stage/One-Shot Detectors**:

* Feature extraction is a method to extract the low-level latent representation of the image. This information is helpful because of its small size and contains only useful information which helps in decreasing our search space. Sometimes, this module is used beforehand in deep networks.
* The latent map obtained from these backbones is further used in *Target Region Selection*module. Each of these backbones is designed to pursue specific tasks and some of them are the advanced versions of the latter. Some of the feature extractor modules are VGG16**[11]**, GoogleNet**[12]**, ResNet**[13]**, DarkNet 53**[14]**, different variations of FCN**[15]** etc.

— Regression/Classification

1. **Two-Stage/One-Shot Detectors**:

* The final step to object detection is the Classification and Bounding Box Localization. This step is generally based and modified by using different combinations of loss functions (*including regression loss and classification loss. Different Methods/Networks have different variations of the final loss used, but the main functions used in loss function are mentioned below*).
* The final output of the feature extractor is then used to calculate the loss which is backpropagated to adjust the localized values and class probabilities. These modules in generic terms are also known as ***Classifier Head*** and ***Regressor Head*.**

Some of the loss functions used in *Regressor Head* are:

1. Mean Squared Error Loss/ L2 Norm Loss: MSE loss is one of the most commonly used loss function. It is the sum of the squared distance between the target variable and the predicted variable.

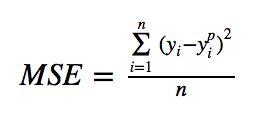


Fig 6: Formula of MSE Loss

2. Mean Absolute Error: MAE is another loss function used for Regression Head. It is the sum of the difference between the absolute values of the target and the predicted variable.

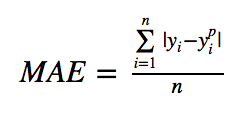


Fig 7: Formula of MAE.

3. Huber Loss: Huber loss is less sensitive to outliers in data than the squared error loss. It’s also differentiable at 0. It’s absolute error, which becomes quadratic when an error is small. How small that error has to be to make it quadratic depends on a hyperparameter, 𝛿 (delta), which can be tuned. Huber loss approaches MAE when 𝛿 ~ 0 and MSE when 𝛿 ~ ∞ (large numbers.)

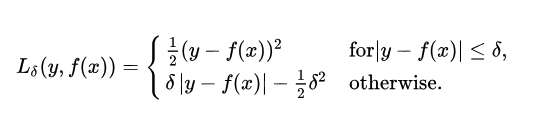
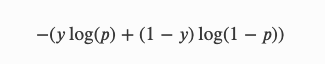


Fig 8: Formula of Huber Loss

The most common loss function used in *Classifier Head*is Cross Entropy Loss

1. Cross-entropy loss, or log loss, measures the performance of a classification model whose output is a probability value between 0 and 1. Cross-entropy loss increases as the predicted probability diverge from the actual label. So predicting a probability of .012 when the actual observation label is 1 would be bad and result in a high loss value. A perfect model would have a log loss of 0.

  
Fig 9: Formula of Cross-Entropy Log Loss

This ends the bird-view of some of the famous methods which are ongoing in the field of Deep Networks and Object Detections. Going further in this series, we will be explaining some of the famous papers on Object Detection. So stay tuned.

Challenges

Object detection is widely used in the field of surveillance, security, forensics, automated vehicle systems. Due to this kind of sensitive use-cases, it is of utmost importance that, detectors work with great speed and at the same time give very good accuracy. Some of the challenges faced in object detection are as follows:-

* **Detection at different scales**: One of the most common problems faced is the object that is detected at one scale may or may not be detected on other smaller/bigger scale.
* So it becomes important for the feature extractor to generalize the features which can be used for any scale. For this **FPN** a.k.a. Feature Pyramidal Networks are used which helps to extract the features at every scale(small, medium, and large). These type of feature extractors are highly used in most of the object detectors.
* **Training for different image resolutions**: Another point for generic object detection is to train at every input sized image. Most Regressor and Classifier Heads are Fully Connected Layers. So resizing at run time is not possible.
* The network trained on one resolution may not give good results on the other. Fully Convolutional Networks are the solution to solve this problem. Instead of FC Layers, FCN**[15]** follows 1X1 convolutional layers of the *Regressor* and *Classifier* head.
* **Speed/Accuracy**: One of the biggest challenges which many people are facing in industries is a speed factor. Deploying these heavy object detectors on a cheap embedded device is of a major concern, which can balance both the aspects of speed and accuracy.
* So, open-research is going on making of a network, which has a decent speed to run in real-time such as YOLOv3**[14]**, and accuracy equally compared to different state-of-the-art detectors, such as Mask RCNN**[16].**
* **Class Imbalance:**Class imbalance makes the network biased towards learning more background information and affects accuracy. To overcome this problem, some of the combinations of oversampling and undersampling are done on datasets to generate an equal ratio of positive(objects) and negative(background) samples.
* **Anchor Free Detection:**Most of the Single Shot Detectors are based on the fixed anchor sizes. Due to this, it becomes very hard to generalize on a particular type of learning. We may have to finetune/train a given pre-trained architecture for different datasets containing a particular object.
* To solve this problem, rigorous research is going on the Anchor Free Approaches. CornerNet**[17]**, ExtremeNet**[18]**, Fully Convolutional One Stage**[19]**, CenterNet**[20]**, etc are some of the papers which follow anchor free paradigm.

Below are some of the chosen object detection papers you may find useful.

*CenterNet: Objects as Points: [*[*paper*](https://arxiv.org/abs/1904.07850)*] [*[*code*](https://github.com/xingyizhou/CenterNet)*]*

*MaskRCNN: [*[*paper*](https://arxiv.org/abs/1703.06870)*] [*[*code*](https://github.com/matterport/Mask_RCNN)*]*

*FCOS: Fully Convolutional One-Stage Object Detection: [*[*paper*](https://arxiv.org/abs/1904.01355)*] [*[*code*](https://github.com/tianzhi0549/FCOS)*]*

*Faster RCNN: [*[*paper*](https://arxiv.org/abs/1506.01497)*][*[*code*](https://github.com/rbgirshick/py-faster-rcnn)*]*

*YOLOv3: An Incremental Improvement [*[*paper*](https://arxiv.org/abs/1804.02767)*] [*[*code*](https://github.com/pjreddie/darknet)*]*

*SSD: Single Shot Multi-Box Detector [*[*paper*](https://arxiv.org/abs/1512.02325)*] [*[*code*](https://github.com/amdegroot/ssd.pytorch)*]*

*CBNet: A Novel Composite Backbone Network Architecture for Object Detection [*[*paper*](https://arxiv.org/pdf/1909.03625v1.pdf)*] [*[*code*](https://github.com/PKUbahuangliuhe/CBNet)*]*

If you have managed to reach here, then I believe you are a part of an elite group who have a thorough understanding to get started in the captivating problem of object detection.

Please feel free to share your thoughts and ideas in the comment section below.

If you think that article was helpful, please do share it and also clap(s) would hurt no one.

References

[0] [Object Detection with Deep Learning: A Review: Zhong-Qiu Zhao, Member, IEEE, Peng Zheng, Shou-tao Xu, and Xindong Wu, Fellow, IEEE](https://arxiv.org/pdf/1807.05511.pdf)

[1][Distinctive Image Features from Scale-Invariant Keypoints, David G. Lowe](https://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf)

[2] [Histogram of Oriented Gradients for Human Detection, Navneet Dalal, Bill Triggs](https://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf)

[3] [Rapid Object Detection using a Boosted Cascade of Simple Features, Paul Viola, Michael Jones](https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-cvpr-01.pdf)

[4] <https://monkeylearn.com/blog/introduction-to-support-vector-machines-svm/>

[5] [A Short Introduction to Boosting, Yoav Freund, Robert E. Schapire](https://cseweb.ucsd.edu/~yfreund/papers/IntroToBoosting.pdf)

[6] [Random Forests, Leo Breiman](https://www.stat.berkeley.edu/~breiman/randomforest2001.pdf)

[7] [Efficient Graph-Based Image Segmentation, Pedro F. Felzenszwalb, Daniel P. Huttenlocher](http://people.cs.uchicago.edu/~pff/papers/seg-ijcv.pdf)

[8] [Rich feature hierarchies for accurate object detection and semantic segmentation, Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik](https://arxiv.org/pdf/1311.2524.pdf)

[9] [Fast R-CNN, Ross Girshick](https://arxiv.org/pdf/1504.08083.pdf)

[10] [Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun](https://arxiv.org/pdf/1506.01497.pdf)

[11] [Very Deep Convolutional Networks for Large-Scale Image Recognition, Karen Simonyan, Andrew Zisserman](https://arxiv.org/abs/1409.1556)

[12] [Going Deeper with Convolutions, Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich](https://arxiv.org/abs/1409.4842)

[13] [Deep Residual Learning for Image Recognition, Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun](https://arxiv.org/abs/1512.03385)

[14] [YOLOv3: An Incremental Improvement, Joseph Redmon Ali Farhadi](https://arxiv.org/pdf/1804.02767.pdf)

[15] [Fully Convolutional Networks for Semantic Segmentation, Evan Shelhamer, Jonathan Long, Trevor Darrell](https://arxiv.org/abs/1605.06211)

[16] [Mask R-CNN, Kaiming He, Georgia Gkioxari, Piotr Dollár, Ross Girshick](https://arxiv.org/abs/1703.06870)

[17] [CornerNet: Detecting Objects as Paired Keypoints, Hei Law, Jia Deng](https://arxiv.org/abs/1808.01244)

[18] [Bottom-up Object Detection by Grouping Extreme and Center Points, Xingyi Zhou, Jiacheng Zhuo, Philipp Krähenbühl](https://arxiv.org/abs/1901.08043)

[19] [FCOS: Fully Convolutional One-Stage Object Detection, Zhi Tian, Chunhua Shen, Hao Chen, Tong He](https://arxiv.org/abs/1904.01355)

[20] [Objects as Points, Xingyi Zhou, Dequan Wang, Philipp Krähenbühl](https://arxiv.org/abs/1904.07850)