A Review of Object Detection

Xiya Lv

2017, Nov. 24

Abstract

Object detection as a core role of computer vision, ideals with detecting instances of semantic objects of a certain class in digital images and videos, involving not only recognizing and classifying every object, but localizing each one by drawing the appropriate bounding box around it. With the development of deep Learning, which has recently shown outstanding performance on many fields, many successful approaches to object detection are proposed, making the object detection systems faster and more accurate.

In this paper, I review some basic knowledges of object detection, such as the datasets using by most of the papers, the criteria for determing performance, non-maximal suppression for fixing multiple detections. And review the main successful methods in the recent years: R-CNN, Fast R-CNN, Faster R-CNN, R-FCN, YOLO and SSD. By the end of this review, we will hopefully have gained an understanding of how deep learning is applied to object detection, and how these object detection models inspire and diverge from others.

Keywords: Object Detection, Deep Learning, CNN, SSD, YOLO

1 Introduction

Humans glance at an image and instantly know what the objects are, where they are and how they interact. Many researchers are committed to make computers to do the same things as human, which leads to many subdomains of computer vision, such as object recognition to konw what the object are, object detection to know what they are and where they are, semantic segmentation to know both above and to know how the images interact, and so on. As we move towrads more complete image undestanding, having more precise and detailed object recognition becomes crucial. Detecing instances of semantic objects of a certain class in digital images, asks us not only to classify images, but also precisely estimatine the classes and locations of the objects, a problem known as objetc detection.

The last decade of progress on object detection has been based on the use of SIFT[9] and HOG[2], which is considered as a traditional method. But it is generally acknowledged that the gains of performance has been small during 2010-2012, on the canonical visual recognition task, PASCAL VOC object detection[3]. Most advance is obtained by building ensemble system and emplying minor fariants of successful methods. Before the advent of convolutional nerual network, the state of the art for those two approaches - Deformable Part Model(DOM)[4] and Selective Search[12], had comparable performance. In recent years, with the development of deep learning, object detection develop rapidly since convolution neural networks(CNNs) are heavy used to get big processes. Especially R-CNN[6] are proposaled in 2014 achieving a dramaic improvement, which combines selective search region proposals and convolution neural network based post-claddification. From then on, a series of improvement measures are proposed to get better preformances and faster speed on object detection.

Fast R-CNN[5] trains networks using a multi-task loss in a single training stage based on the original R-CNN. Faster R-CNN[11] replace the slow selective search algorithm with a fast neural net based on Fast R-CNN. R-FCN[1] shares 100% of the computations across every single output based on Faster R-CNN, getting a faster speed. YOLO[10], a new approach to object detection presented in CVPR 2016, framing object detection as a regression problem to spatially separated bounding boxes and associated class probabilities, is very different form the family of R-CNN. SSD[8] combines the core ideas of Faster R-CNN and YOLO, as one-stage method, getting a faster speed than YOLO and almost the same accuracy as Faster R-CNN. All above algorithms are introduced in detail in chapter 3.

Current state-of-the-art object detection systems are variants of two strategies: one-stage method and two-stage method. The representatives of two-stage method are R-CNN family, SPPnet[7] et al. The approaches use region proposal methods to first generate potential bounding boxes in an image and then run a high-quality classifier or a convolutional neural network on those proposed boxes. The first step generating potential bounding boxes is generally implemented in two ways, one based on sliding windows and the other based on region proposal network(RPN). Selective search as a traditional method is fading away since RPN proposaled in Faster R-CNN, but still an useful method. While accurate, these approaches have been too computationally intensive for embedded systems and, even with high-end hardware, too slow for real-time applications. YOLO and SSD speed up the progress of object detection significantly, known as the one-stage method. YOLO make up for this shortcoming using a single nerual network to predict bounding boxes and class probabilities directly from full images in one evaluation, while SSD uses different scales from feature maps of different scales and considers multiple aspect ratios .

2 Basic knowledge

In this chapeter, I introduce some basic knowledge of object detection in brief, which appearing in many papers for convenience of explanation, including data set, intersection-over-union(IoU), mean average precision(mAP) and non-maximum suppression(NMS).

Data Set: Most of papers train and test their models based on PASCAL VOC challenge. The PASCAL VOC project provides standardised image data sets for object class recognition. Organised annually from 2005 to 2012, the PASCAL VOC challenge and its associated dataset has become accepted as the benchmark for object detection. VOC 2007 having 9,963 images, containing 24,640 annotated objects is the most popular data set. It has 20 classes: person, bird, cat, cow, dog, horse, sheep, aeroplane, bicycle, boat, bus, car, motorbike, train, bottle, chair, dining table, potted plant, sofa, tv/monitor. The later dataset 2010 and 2012 are similar with 2007. In recent years, the dataset COCO is also used to train detectors.COCO is a large-scale object detection, segmentation, and captioning dataset, having several features described as the website: Object segmentation, Recognition in context, Superpixel stuff segmentation, 330K images (>200K labeled), 1.5 million object instances, 80 object categories, 91 stuff categories, 5 captions per image and 250,000 people with keypoints.

IoU: Intersection over Union is an evaluation metric used to measure the accuracy of an object detector on a particular dataset.We can typically find it used to evaluate the performance of object detectors, such as R-CNN, R-FCN, YOLO, etc. More formally, in order to use the IoU, we need to know two sets of values. One is the ground-truth bounding boxes, and the other is the predicted bounding boxes outputed from an objetc detector. A simple IoU is simply a ratio, can be determined as figure 1.In the numerator we compute the area of overlap between the predicted bounding box and the ground-truth bounding box. The denominator is the area of union, or more simply, the area encompassed by both the predicted bounding box and the ground-truth bounding box. Dividing the area of overlap by the area of union yields gets the final score – IoU.

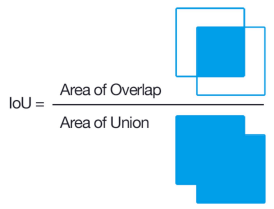

Figure 1: Computing the IoU is as simple as dividing the area of overlap between the bounding boxes by the area of union.

mAP: Mean average precision is a criterion for judging the performance of a objetc detector. We have to understand average precision before introducing mean average precision. Precision and recall are single-value metrics based on the whole list of documents returned by the system. For systems that return a ranked sequence of documents, it is desirable to also consider the order in which the returned documents are presented. By computing a precision and recall at every position in the ranked sequence of documents, one can plot a precision-recall curve, plotting precision $p(\gamma)$ over the interval from $\gamma = 0$ to $\gamma = 1$:

$$AP = \int_0^1 p(\gamma) d\gamma$$

That is the area under the precision-recall curve. Normally, to reduce the impact of ”wiggles” in the curve in the PASCAL Visual Object Classes challenge, researchers computes average precision by averaging the precision over a set of evenly spaced recall levels 0, 0.1, 0.2, ... 1.0:

$$AP = \frac{1}{11} \sum_{\gamma \in \{0,0.1,\ldots,1.0\}} p_{interp}(\gamma)$$

where $p_{interp}(\gamma)$ is an interpolated precision that takes the maximum precision over all recalls greater than γ: $p_{interp}(\gamma) = max_{\tilde{\gamma}:\tilde{\gamma}\geq\gamma} p(\tilde{\gamma})$.

Mean average precision for a set of classes is the mean of the average precision scores for each class.

$$mAP = \frac{\sum_{q=1}^{Q} AP(q)}{Q}$$

where Q is the number of queries.

NMS: The application of non-maximum supression in the field of object detection is very wide. The main purpose is to remove the redundant frames and, produce the final and hope the best location of detections, since no matter what object detection method you choose, you will detect multiple bounding boxes surrounding the object in the image. NMS finds the bounding box of highest confidence, according to the score matrixes and the coordinate information of regions. The main steps are following: First of all, NMS calculate the class-specific confidence score of each bounding box, sort them accroading to the scores, and then take the one with highest score as a candidate. Go through the rest of the boxes, and if the IoU with the candidate is greater than a certain threshold, delete it. Selecting a candidate with the highest score from the untreated boxes, Repeat the above steps until there is no other box.

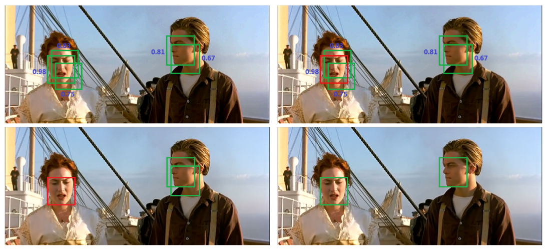

Figure 2: NMS removes the redundant frames and produces the final location of detections.

3 Approaches

As we all know, the evolution of object detection algorithm is divided into two stages, one is based on the traditional features, the other is based on deep learning. Traditional feature-based optimization methods are still the mainstream until 2013. However, after 2013, the whole academia and industry are increasingly using the deep learning, because compared with traditional method, the deep learning improve the performance a lot. It is worth noting that the traditional detection method will become saturated with the increase of data volume, which means the detection performance will gradually increase as the data volume increases, but to a certain extent, the perforance gains little with the data volume increasing. However, the methods of deep learning are different. When the data conforming to the actual scene distribution is more and more, the detection performance will be better and better.

3.1 R-CNN

The R-CNN, or regions with proposals with CNNs, was presented by Ross Girshick, etc in CVPR 2014.

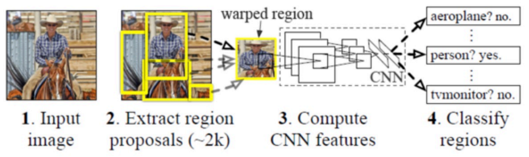

Figure 3: Object detection system overview. The system (1) takes an input image, (2) extracts around 2000 bottom-up region proposals, (3) computes features for each proposal using a large convolutional neural network (CNN), and then (4) classifies each region using class-specific linear SVMs.

The object detection system consists of three modules. The first generates category-independent region proposals using selective search. These proposals define the set of candidate detections available to the detector. The second module is a large convolutional nerual network that extracts a fixed-length feature vector from each region. The third module is a set of class-specific linear SVMs.The three modules can be also summarized in three steps illustrated by figure 4: (1) region proposals generate. Scan the input image for possible objects using an algorithm called Selective Search, generating 2000 region proposals. (2)feature extraction. Run a convolutional neural net (CNN) on top of each of these region proposals. (3)object classification and localization detection. Take the output of each CNN and feed it into a) an SVM to classify the region and b) a linear regressor to tighten the bounding box of the object, if such an object exists.

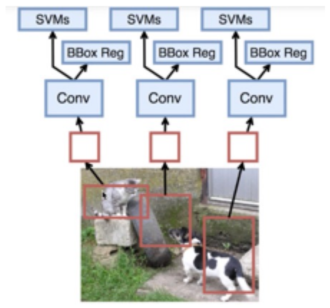

Figure 4: An example of R-CNN detector.

In the first step, each region proposal's size is different, while the CNN requires fixed size(227 x 227 pixel size) images.We must first convert the image data in thar region into a form that is compatible with the CNN. Of the many possible transformations, the authors opt for the simplest, which regardless of the size or aspect ratio of the candidate region, warp all pixels in a tight bounding box around it to the required size.

The CNN architecture is Alexnet. To adapt the CNN to the new task and the new domain, the paper continues stochastic gradient descent training of the parameters using only warped region proposals form VOC, with treating all region proposals with $\geq$ 0.5 IoU overlap with a ground-truth box as positices for the box's class adn the rest as negatives. The learning rate of stochastic gradient desecnt is 0.001, which allows fine-tuning to make progress while not clobbering the initialization. Once features are extracted and training labels are applied, optiminzing one linear SVM per class.

The R-CNN achieves a performance of 53.3%mAP on VOC 2012 test, which is proves mAP by more than 30% relative to the best result at that time. The speed is 13s/image on a GPU or 53s/image on a CPU.

As a groundwork of the later researches, R-CNN has excellent object detection accuracy. However, it also has notable drawbacks.(1) Training is a multi-stage pipelines. one first time fine-tunes a ConvNet for detection using cross-entropy loss. Then linear SVMs are fit to ConvNet features computed on warped object proposals. In the third training stage, bouding-box regressors are learned. (2) Traing is expensive in space and time. For SVM and regressor training, features are extracted from each warped object proposal in each image and written to disk. These features require hundreds of gigabytes of storage. (3) Test-time detection is slow. At test-time, features are extract from each warped object proposal in each test image. R-CNN is slow because it warps and then processes each object proposal indepently. There is much repetitive work, the selective search, convolutional neural network for per region proposal progress. Even for one region, we must classify it by using 21 SVMs. Fortunately, the later work improves every aspect to get a faster speed and a more accurate performance.

3.2 Fast R-CNN

R-CNN's immediate descendant is Fast R-CNN. Fast R-CNN resembled the original in many ways, but improved on its detection speed through two main augmentations: (1) Performing feature extraction over the image before proposing regions, thus only running one CNN over the entire image instead of 2000 CNN's over 2000 overlapping regions. (2) Replacing the SVM with a softmax layer, thus extending the neural network for predictions instead of creating a new model.

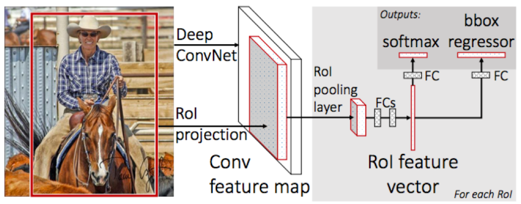

Figure 5: Fast R-CNN architecture. An input image and multi- ple regions of interest (RoIs) are input into a fully-convolutional network. Each RoI is pooled into a fixed-size feature map and then mapped to a feature vector by fully-connected layers (FCs). The network has two output vectors per RoI: softmax probabilities and per-class bounding-box regression offsets. The architecture is trained end-to-end with a multi-task loss.

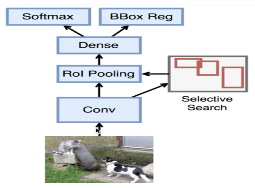

Figure 6: An example of Fast R-CNN detector.

As we can see from the figure 6, we are now generating region proposals based on the last feature map of the network, not from the original image itself. As a result, we can train just one CNN for the entire image. In addition, instead of training many different SVM's to classify each object class, there is a single softmax layer that outputs the class probabilities directly. Now we only have one neural net to train, as opposed to one neural net and many SVM's.

The Fast R-CNN achieves a performance of 66%mAP on VOC 2012 test. The speed is 0.3s/image excluding object proposals time, 9x faster at training VGG16 for detection, 213x faster at test-time. However, generating the object proposals on CPU takes about 2s of time, which means the Fast R-CNN has reduced the running time of thes detection network, exposing region proposal computation at a bottleneck.

3.3 Faster R-CNN

Faster R-CNN is now a canonical model for deep learning-based object detection. It helped inspire many detection and segmentation models that came after it. Faster R-CNN introduce a Region Proposal Network(RPN) that shares full-image convolutional features with the detection network, enabling nearly cost-free region proposals.

A RPN is a fully-convolutional network that simulatneously predicts object bounds and objectness scores at each position. RPN is trained end-to-end to generate high-quality region proposals, which are used by Fast R-CNN for detection. With simple alternating optimization, RPN and Fast R-CNN can be trained to share convolutional features. In a sense, Faster R-CNN = RPN + Fast R-CNN.

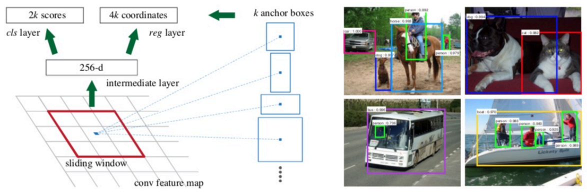

Figure 7: Left: Region Proposal Network (RPN). Right: Example detections using RPN proposals on PASCAL VOC 2007 test. Our method detects objects in a wide range of scales and aspect ratios.

Selective search as one of the most popular methods, greedily merges superpixels based on engineerred low-level features. Yet when compare to efficient detection networks, selective search is an order of magnitude slower. One may note that fast region-based CNNs take advantage of GPUs, while the traditional region proposal methods used in research are implemented on the CPU, makeing such runtime comparisons inequitable.

A RPN takes an image(of any size) as input and output a set of rectangular object proposals, each with an objectness score. To genereate regional proposals, slide a small network over the conv feature map output by the last shared conv layer. This network is fully connected to an

3×3 spatial window of the input conv feature map. Each sliding window is mapped to a lower-dimensional vector. This vector is fed into two sibling fully-connected layers—a box-regression layer and a box-classification layer.

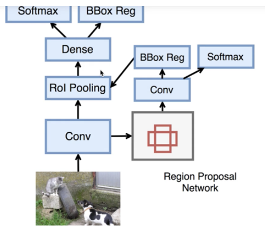

Figure 8: An example of Faster R-CNN detector.

The paper including 4 main parts, including translation-invariant anchors, a loss function for learning region proposals, optimization, and sharing convolution features for RPN and object detection.

anchor: At each sliding-window location, we simultaneously predict k region proposals. Each region proposal is called anchor. So the reg layer has 4k outputs encoding the coordinates of k boxes. The cls layer outputs 2k scores that estimate probability of object/not-object for each proposal. Each anchor is centered at the sliding window in question, and is associated with a scale and aspect ratio. Use 3 scales and 3 aspect ratios, yielding k = 9 anchors at each sliding position. For a conv feature map of a size $W \times H$ (typically 2,400), there are $W \times H \times k$ anchors in total. An important property of the approach is that it is translation invariant, both in terms of the anchors and the functions that compute proposals relative to the anchors.

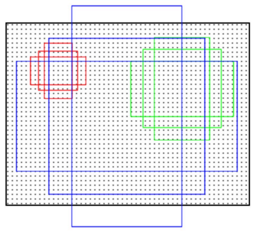

Figure 9: The example of anchors at each sliding-window location.

The loss function is defined as:

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{rcg}} \sum_i p_i^* L_{reg}(t_i, t_i^*)$$

Here, i is the index of an the index of an anchor in a mini-batch and p_i is the predicted probability of anchor i being anobject.The ground-truth label p_i^* is 1 if the anchor is positive, and is 0 if the anchor is negative. t_i is a vector representing the 4 parameterized coordinates of the predicted bounding box, and t_i^* is that of the ground-truth box associated with a positive anchor. The classification loss L_{cls} is log loss over two classes (object vs. not object).

The network is trained by adopting the "image-centric" sampling strategy. Each mini-batch arises from a single image that contains many positive and negative anchors. It is possible to optimize for the loss functions of all anchors, but this will bias towards negative samples as they are dominate. Instead, the paper randomly sample 256 anchors in an image to compute the loss function of a mini-batch, where the sampled positive and negative anchors have a ratio of up to 1:1. If there are fewer than 128 positive samples in an image, pad the mini-batch with negative ones.

To share convolutional features for RPN and Fast R-CNN is not as easy as simply defining a single network that include both RPN and Fast R-CNN, and then optimizing it jointly with back-propagaton, because Fast R-CNN training depends on fixed object proposals and it is not clear a priori if learning Fast R-CNN while simultaneously changing the proposal mechanism will converge. The paper develop a 4-step training algorithm to learn shared features via alternating optimization, illustrated by figure 10.

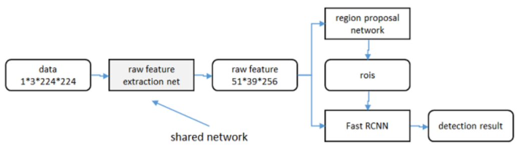

Figure 10: 4-step algorithm to learn shared features: (1) init the network with an ImageNet-pretrained model and fine-tuned end-to-end for the region proposal task. (2) train a separate detection network by Fast R-CNN using the proposals generated by the step-1 RPN. (3) use the detector network to initialize RPN training, but fix the shared conv layers and only fine-tune the layers unique to RPN. (4) keeping the shared conv layers fixed, fine-tune the fc layers of the Fast R-CNN

The Faster R-CNN achieves a performance of 73.2%mAP on VOC 2007 test and 70.4% on VOC 2012 test. The speed is 5 frames per second. It is worth noting that although future models did a lot to increase detection speeds, few models managed to outperform Faster R-CNN by a significant margin. In other words, Faster R-CNN may not be the simplest or fastest method for object detection, but it is still one of the best performing. At the end of the day, Faster R-CNN may look complicated, but its core design is the same as the original R-CNN: hypothesize object regions and then classify them.

3.4 R-FCN

Fast R-CNN improved on the original's detection speed by sharing a single CNN computation across all region proposals. That kind of thinking was also the motivation behind R-FCN: increase speed by maximizing shared computation.

R-FCN, or Region-based Fully Convolutional Net, shares 100% of the computations across every single output. Being fully convolutional, it ran into a unique problem in model design.

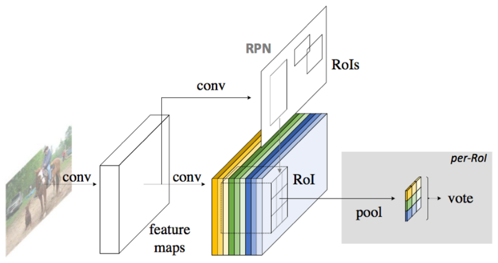

Figure 11: Key idea of R-FCN for object detection.

On the one hand, when performing classification of an object, we want to learn location invariance in a model: regardless of where the cat appears in the image, we want to classify it as a cat. On the other hand, when performing detection of the object, we want to learn location variance: if the cat is in the top left-hand corner, we want to draw a box in the top left-hand corner. So if we're trying to share convolutional computations across 100% of the net, how do we compromise between location invariance and location variance?

R-FCN's solution is position-sensitive score maps. Each position-sensitive score map represents one relative position of one object class. For example, one score map might activate wherever it detects the top-right of a cat. Another score map might activate where it sees the bottom-leftof a car. You get the point. Essentially, these score maps are convolutional feature maps that have been trained to recognize certain parts of each object.

R-FCN works as follows: (1) Run a CNN (in this case, ResNet) over the input image. (2) Add a fully convolutional layer to generate a score bank of the aforementioned "position-sensitive score maps." There should be $k^2(C+1)$ score maps, with k^2 representing the number of relative positions to divide an object (e.g. 3^2 for a 3 by 3 grid) and C+1 representing the number of classes plus the background. (3) Run a fully convolutional region proposal network (RPN) to generate regions of interest (RoI's). (4) For each RoI, divide it into the same k^2 "bins" or subregions as the score maps. (5) For each bin, check the score bank to see if that bin matches the corresponding position of some object. For example, if I'm on the "upper-left" bin, I will grab the score maps that correspond to the "upper-left" corner of an object and average those values in the RoI region. This process is repeated for each class. (6) Once each of the k^2 bins has an "object match" value for each class, average the bins to get a single score per class. (7)Classify the RoI with a softmax over the remaining C+1 dimensional vector.

With this setup, R-FCN is able to simultaneously address location variance by proposing different object regions, and location invariance by having each region proposal refer back to the same bank of score maps. These score maps should learn to classify a cat as a cat, regardless of where the cat appears. Best of all, it is fully convolutional, meaning all of the computation is shared throughout the network.

The R-FCN achieves a performance of 83.6%mAP on VOC 2007 test and 82.0% on VOC 2012 test. The speed is 170 milliseconds per image. R-FCN is several times faster than Faster R-CNN, and achieves comparable accuracy.

3.5 YOLO

YOLO, or you only look once, is a totally new approach to object detection, framing object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. A single neural network pre- dicts bounding boxes and class probabilities directly from full images in one evaluation.

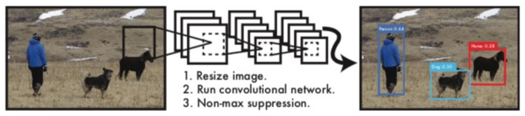

Figure 12: The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448×448, (2) runs a single convolutional net- work on the image, and (3) thresholds the resulting detections by the model's confidence.

The system model is discribed as figure 13.

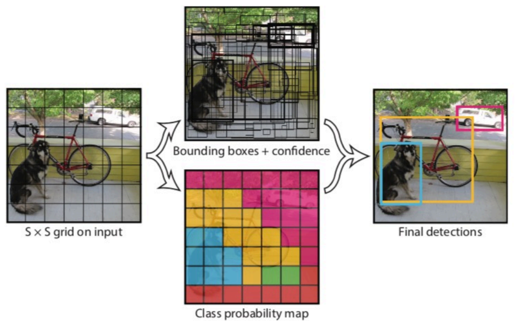

Figure 13: The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448×448, (2) runs a single convolutional net- work on the image, and (3) thresholds the resulting detections by the model's confidence.

The network architecture is inspired by the GoogleNet model for image classification, having 24 convolutional layers followed by 2 fully connected layers.

Instead of the inception modules used by GoogLeNet, the paper simply use 1×1 reduction layers followed by 3×3 convolutional layers. The full network is shown in Figure 14. The paper also train a fast version of YOLO designed to push the boundaries of fast object detection. Fast YOLO uses a neural network with fewer convolutional layers (9 instead of 24) and fewer filters in those layers. Other than the size of the network, all training and testing parameters are the same between YOLO and Fast YOLO.

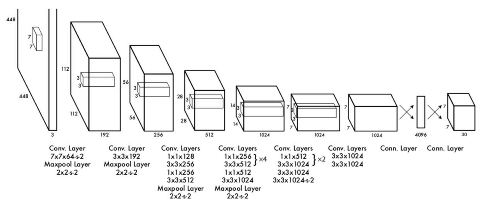

Figure 14: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. Pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

YOLO achieves a performance of 63.4%mAP on VOC 2007 test and 57.9% on VOC 2012 test. The base YOLO model processes images in real-time at 45 frames per second. The faster one processes an astounding 155 frames per second.

The unified model has several benefits over traditional methods of objecct detection:(1) YOLO is extremely fast. (2) YOLO reasons globally about the image when making predictions. (3) YOLO learns generalizable representations of objects. But the drawback is also obvious.YOLO still lags behind state-of-the-art detection systems in accuracy. While it can quickly idedtify objects in images, it struggles to precisely localize some objects, especiallt small one.

3.6 SSD

The final model is SSD, which stands for Single-Shot Detector. Like R-FCN, it provides enormous speed gains over Faster R-CNN, but does so in a markedly different manner, similar to YOLO. It can be thought a combination of Faster R-CNN and YOLO.

SSD like YOLO, simultaneously predicting the bounding box and the class as it processes the image. Concretely, given an input image and a set of ground truth labels, SSD does the following: (1) Pass the image through a series of convolutional layers, yielding several sets of feature maps at different scales (e.g. 10x10, then 6x6, then 3x3, etc.). (2)For each location in each of these feature maps, use a 3x3 convolutional filter to evaluate a small set of default bounding boxes. These default bounding boxes are essentially equivalent to Faster R-CNN's anchor boxes. (3) For each box, simultaneously predict a) the bounding box offset and b) the class probabilities. (4)During training, match the ground truth box with these predicted boxes based on IoU. The best predicted box will be labeled a "positive," along with all other boxes that have an IoU with the truth >0.5.

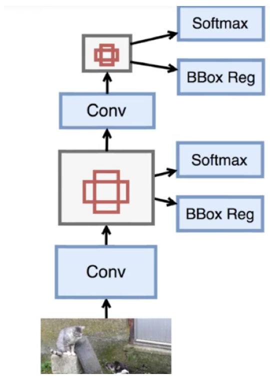

Figure 15: An example of SSD detector.

The SSD approach is based on a feed-forward convolutional network that produces a fixed-size collection of bounding boxes and scores for the presence of object class instances in those boxes, followed by a non-maximum suppression step to produce the final detections.

We classify and draw bounding boxes from every single position in the image, using multiple different shapes, at several different scales. As a result, we generate a much greater number of bounding boxes than the other models, and nearly all of the them are negative examples.To fix this imbalance, SSD does two things. Firstly, it uses non-maximum suppression to group together highly-overlapping boxes into a single box. In other words, if four boxes of similar shapes, sizes, etc. contain the same dog, NMS would keep the one with the highest confidence and discard the rest. Secondly, the model uses a technique called hard negative mining to balance classes during training. In hard negative mining, only a subset of the negative examples with the highest training loss (i.e. false positives) are used at each iteration of training. SSD keeps a 3:1 ratio of negatives to positives.

There are "extra feature layers" at the end that scale down in size. These varying-size feature maps help capture objects of different sizes. For example, here is SSD in action: In smaller feature maps (e.g. 4x4), each cell covers a larger region of the image, enabling them to detect larger objects. Region proposal and classification are performed simultaneously: given p object classes, each bounding box is associated with a (4+p)-dimensional vector that outputs 4 box offset coordinates and p class probabilities. In the last step, softmax is again used to classify the object.

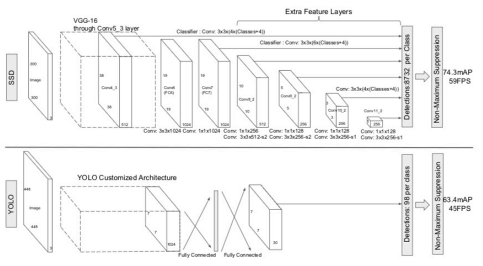

Figure 16: A comparison between two single shot detection models: SSD and YOLO. Our SSD model adds several feature layers to the end of a base network, which predict the offsets to default boxes of different scales and aspect ratios and their associated confidences. SSD with a 300 × 300 input size significantly outperforms its 448 × 448 YOLO counterpart in accuracy on VOC2007 test while also improving the speed.

Ultimately, SSD is not so different from the first two models. It simply skips the "region proposal" step, instead considering every single bounding box in every location of the image simultaneously with its classification. Because SSD does everything in one shot, it is faster than the base YOLO at speend of 59 fps, and still performs quite comparably achieving 74.3%mAP on VOC 2007 test.

3.7 Contradistinction and Prospect

To visually compare each algorithm, I use a table to perform the speed and the accuracy.

Table 1: algorithm's performance

Approach	speed(fps)	accuracy(mAP)
R-CNN	0.02	53.3%
Fast R-CNN	3	66%
Faster R-CNN	5	73.2%
R-FCN	6	83.6%
YOLO	45/155	63.4%
SSD	59	74.3%

The evalution has been discussed in chapter3. Although the R-FCN, YOLO and SSD have made great progress, the deep learning still has a lot potential. Considering the SSD combines the core ideas of YOLO and Faster R-CNN, and the R-FCN is a improvement on Faster R-CNN, I think it is also a good way to combine these. while the recurrent nerual network and the reinforcement learning do well in other fields, it can be consider a new way to detect objects in computer vision. Because of the limited time, I can not prove my idea. But I will continua the research to optimize the review.

4 Conclusion

This review discribes R-CNN, Fast R-CNN , Faster R-CNN, R-FCN, YOLO and SSD in detail, introduces the evalution of Object detection in brief and finally compares the methods. Faster R-CNN, R-FCN and SSD are three of the best and most widely used object detection models out there right now. YOLO version2 has been published, achieving a higher accuracy. Object detection is growing rapidly.

5 Acknowledgments

It is a project assignment for the course Ubiquitous Computing teached by professor Li. I take this opportunity to learn a lot of new knowledge in this field.Thanks for professor Li and his team. Many reasearchs have push the field forward. I would like to thank them, and I wish I can be one of them one day.

References

[1] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems*, pages 379–387, 2016.

[2] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[3] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[4] Pedro Felzenszwalb, David McAllester, and Deva Ramanan. A discriminatively trained, multiscale, deformable part model. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[5] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[6] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *European Conference on Computer Vision*, pages 346–361. Springer, 2014.

[8] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[9] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[10] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016.

[11] Shaoqing Ren, 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*, pages 91–99, 2015.

[12] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.