CNN based Model for Object Detection

JYOTI CHANDEL
MAHARISHI DYANAND UNIVERSITY

UMA SINGH
MAHARISHI DYANAND UNIVERSITY

Abstract

Object detection has considerable importance in areas, such as defense and military applications, urban studies, airport surveillance, vessel traffic monitoring, Marketing and transportation infrastructure determination. A lot of attention has been associated with Machine Learning, specifically neural networks such as the Convolutional Neural Network (CNN) winning object Detection. Need to focus on typical generic object detection architectures along with some modifications and useful tricks to improve detection performance further. This paper presents a model for providing the object detection on the given image and video. Object Detection is one of the core problems in Computer Vision field with a large variety of practical applications. The problem precision of object detection is to discover where objects are located in the provided image.

1 Introduction

The major concerns for today's computer systems are performance and energy . GPUs have become a very powerfulDue to object detection's close connection with video inspection and image mastery, it has engaged much research attentiveness in the last few years. Conventional object detection methods are built on handicraft features and slight trainable architectures. Their performance is simply motionless by establishing a complex circle which incorporates various low-level image features with high-level context from object detectors.

The problem with the standard CNN like R-CNN methods are two-stage detectors and was a slow and not end-to-end object detector. The biggest drawback with the R-CNN family of networks is their speed — they were implausibly slow, obtaining only 5 FPS on a GPU.

To improve the speed and with the high accuracy, trying to build an object detector library using modern deep learning algorithms like Retina Net and YOLO that will give an output faster and more accurate. Also, try to build a solution to provide a custom way for beginners to train models that requires deep inside knowledge of mathematics and expertise to train and build models.

Our study begins with a detailed introduction on the history of deep learning and its indicative tool, mainly Convolutional Neural Network (CNN). Then we aimed on classic generic object detection architectures along with few modifications and useful tricks to enhance detection performance further. As well defined specific detection tasks reveal different characteristics, we also shortly survey various specific tasks, including salient object detection, face detection. Experimental analyses are also given to compare various methods and draw some meaningful conclusions. The various promising ways and tasks are provided to function as guidelines for further work in object detection.

1.1 Object Detection:

Object Detection is a common Computer Vision issue which involves identifying and locating objects of certain groups in the image. The object localisation can be achieved in various methods, including creating a bounding box around the object or marking every pixel in the image which carries the object.

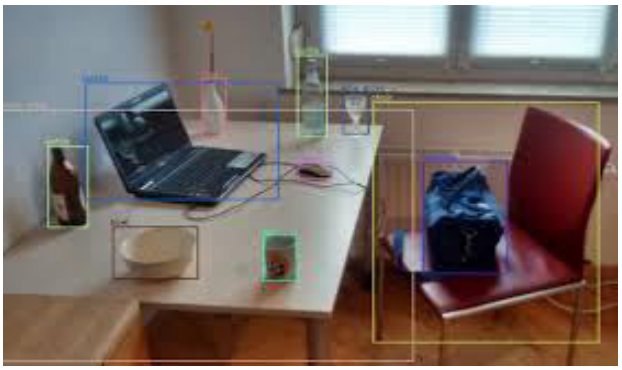


Figure1: Object Detection

The architecture of general object detection ways can be categorized into two types (see Figure 1.2). One follows the conventional object detection pipeline, generating area proposals at first and then classifying each proposal into different object categories. The second object detection method is a classification problem, adopting a unique architecture to achieve final results directly.

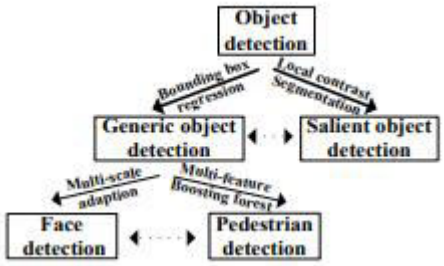


Figure 1.1: Two-Ways of Object Detection

1.1.1 Requirements and Specification:

Object classification: This technique Forecasts the probability of distinct object categories(car, turtle, rabbit, etc.) in an image, it essentially answers the question (What is in the picture?). It can only forecast one category for one image.

Object localization: This technique can forecast the probability of an object in the image along with its location in the image. This method basically answers (What is in the picture and where it is?).

Object detection: The above two techniques only care about one object and its location. In a real world scenario, we may have to find various objects from an image and its position.

1.2 The Basic Properties of Object Detection:

Data requirements:

In order to train a custom model, we need labelled data. Labelled data in the context of object detection are images with corresponding bounding box coordinates andit's labels. That is, the bottom left and top right (x,y) coordinates + the classog the object .

Feature extraction:

The motive of feature extraction is to decrease a variable sized image to get a fixed set of visual features. Image classification models are commonly constructed using strong visual feature extraction methods. Whether they are based on traditional computer vision approaches, such as for example, filter based approached, Pictorial methods, etc.

Non-maximum suppression:

The basic idea of non-maximum suppression is to decrease the number of detections in a frame to the actual number of objects present in it. If the object in the frame is fairly big and more than 2000 object plans have been created, it is quite likely that few of these will have significant overlap with each other and the object.

Evaluation metric:

The most repeated evaluation metric that is pre-owned in object recognition tasks is 'MAP', which stands for 'mean average precision'. It is a count from 0 to 100 and bigger values are typically superior, but it value is unlike from the accuracy metric in classification of data set.

1.3 Advantages and Disadvantages:

The advantages of CNN in case of traditional methods can be summarised as follows:

Hierarchical aspect representation, which is that the multilevel representations from pixel to high-level semantic features studied by a hierarchical multi-stage structure, are often gained from data automatically and hidden factors of input data can be disentangled through multi-level nonlinear mappings.

Compared with traditional shallow models, a deeper architecture gives an exponentially greater expressive capability.

The architecture of CNN gives a chance to mutually optimize various related tasks together (example: Fast RCNN combines classification and bounding box regression into a multi-task learning manner).

Benefitting from the bigger adaptive capacity of deep CNNs, few classical computer vision challenges can be recast as high-dimensional data transform issues and solved from different angles. Due to these benefits, CNN has been widely used into various research fields, like image super-resolution reconstruction, image classification, image retrieval and video analysis.

1.4 Need for Object Detection:

There are various applications which need object detection techniques. One of the famous applications is face detection and recognition. There are evencommercial software products accessible in the market for face recognition. Some of the important applications of object detection techniques are given in the following list.

Optical Character recognition:

OCR is the recognition of hand-written, printed, or typed characters from an image. These techniques are preferred for scanning printed books to a digital document. Other applications are data entry, traffic sign recognition, music symbols etc.

Self-driving cars:

These cars can drive by themselves. One of the major capabilities of self-driving cars is detecting pedestrians, cars, trucks, traffic signs, etc. These detections are essential for the proper working of self-driving cars.

Verification using face and IRIS code:

Face and IRIS verification and authentication are used in iPhone and Android phones. It does the device authorization if the exact face or IRIS match is detected.

Robotics:

There are a multiple applications in robotics using object detection. One of the usual applications is bin picking and classify of objects. Using object detection techniques, the robot is able to understand the location of objects. Using that information, the robot is able to pick the object and is able to sort it.

Object tracking and counting:

Using object detection techniques, you can track an object and can be used as an object counter. For example, how many cars have crossed in a junction, how people entered a shopping mall etc.

1.5 Scope of Present Work:

In order to gain a whole image understanding, we should not only focus on classifying different images, but also aim to precisely estimate the concepts and locations of objects contained in every image. This task is referred to as object detection, which usually consists of different subtasks such as face detection, pedestrian detection and skeleton detection. As one of the fundamental computer vision issues, object detection is able to give valuable information for semantic knowledge of images and videos, and is related to many applications, including image classification, human behaviour analysis, face recognition and autonomous driving. Meanwhile, Inheriting from neural networks and related learning systems, the progress in these areas will develop neural network algorithms, and will also have great impacts on object detection techniques which can be considered as learning systems.

1.6 Conclusion:

Every Object Detection option has a different way of working, but they all work on the same principle that we conclude in our study.

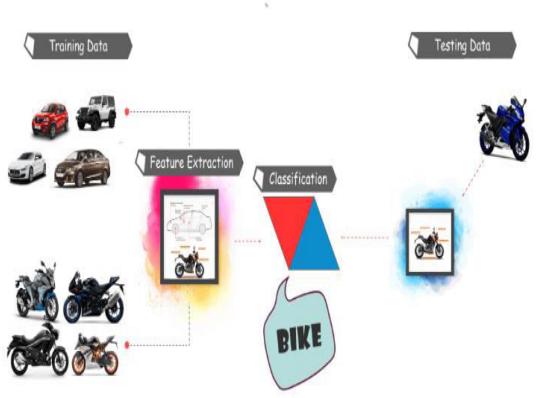


Figure 1.4: Flow of Object Detection

2 A Comparative Study of ODA

2.1 Introduction:

The traditional object detection algorithms models follow the below three stages:

Informative region selection:

As different objects may appear in any position of the image and have different aspect ratios or sizes, it's a natural option to scan the entire image with a multi-scale sliding window. Although this exhaustive strategy can determine all possible positions of the objects, its shortcomings are also are obvious. Due to a large number of candidate windows, it is computationally expensive and produces too many redundant windows. However, if only a fixed number of sliding window templates are applied, unsatisfactory regions may be produced.

Feature extraction:

To recognize different objects, we'd like to extract visual features which may provide a semantic and robust representation. This is due to the fact that these features can produce representations associated with complex cells in the human brain. However, due to the diversity of appearances, illumination conditions and backgrounds, it's difficult to manually design a robust feature descriptor to perfectly describe all kinds of objects.

Classification:

Besides, a classifier is required to differentiate a target object from all the opposite categories and to form the representations more hierarchical, semantic and informative for visual recognition. Usually, the Supported Vector Machine (SVM), AdaBoost and Deformable Part-based Model (DPM) are good choices. Among these classifiers, the DPM is a flexible model by combining object parts with deformation cost to handle severe

deformations. In DPM(Deformable Part-based Model), with the aid of a graphical model, carefully designed low-level features and kinematically inspired part decompositions are combined.

2.2 Basic Principles of ODA:

The object detection algorithm follows the basic principles mentioned below:

Data Acquisition:
Data acquisition may be a required step once you want to coach the model on the custom dataset. It will require more than 100 images with the object for which you want to train the model. Each image should have good quality, containing object/objects at least once (to be detected). No image should be repetitive within the dataset, the more the variability of images, the more extensive the training becomes.

Data Labelling:

For the custom dataset, we'd like to label objects from each of the pictures. For object detection, the model requires certain details of the objects that are to be detected from the image and the other people are the X-axis, y-axis, height, and width of the object within the image.

Data Preparation:

Each algorithm has its own way to prepare the data sets.

Model Training:

Each model needs a different time to train it according to the algorithm. It also depends on the dataset used for training the model.

Evaluation / Results analysis:

Once the model gets trained then we can test it on different image data sets and see the results. It helps us in analyzing the speed and performance of different ODA.

2.3 Variants of ODA:

The object detection algorithms using deep learning can be classified into two groups

Classification based algorithms:

There are mainly two stages in classification based algorithms. In the first stage, it'll select a bunch of Region of Interest (ROI)within the image where the probabilities of objects are high. In the second stage, it will apply a Convolution Neural Network to these regions to detect the presence of an object. One of the problems with this method is, we have to execute the detector in each of the ROI, and that makes is slow and computationally expensive. One example of this type of algorithm is R-CNN.

Regression-based algorithms:

In this algorithm, there's no selection of interesting ROI within the image, rather than that, it'll predict the classes and bounding boxes for the whole image directly. This makes detection faster than classification algorithms. One of the famous regression-based algorithms is YOLO ("You Only Look Once"). The YOLO detector is very fast so it is used in self-driving cars and other applications where real-time object detection is required.

2.4 Faster R-CNN:

With the objective of speeding up and simplifying the architecture of an R-CNN, a new type of network was created in order to improve test time.

The main difference regarding its predecessor is that the whole input image is fed into the CNN and a convolutional feature map is generated.

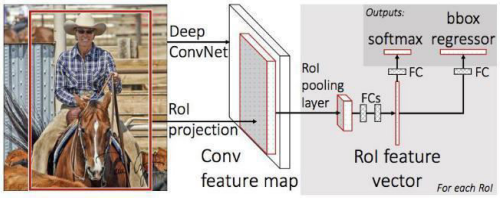
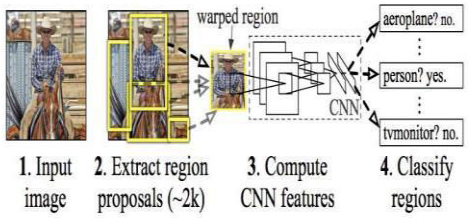


Figure 2.1. Fast R-CNN architecture

Region proposals are then inferred from this map and converted into squares through a Region of Interest (RoI) pooling layer. A RoI layer performs max pooling (down-sampling in order to reduce dimensionality) to the inputs (which are convolutional neural network feature maps) and produces a small feature map of a fixed size (i.e. 7x7) that is fed into the fully connected (FC) layer.

This network is faster than a standard R-CNN because region proposals are not fed to the CNN every time. Instead, a convolutional operation is done only once per image and a feature map is generated from it. Thanks to this, it only takes 2 seconds to test an image.

Figure 2.2: R-CNN Working Stages



However, the problems the first type of R-CNN presents go from a huge amount of training time (a fixed number of proposals have to be classified per each image), no real-time implementation because it takes more than 45 seconds to test an image and no learning in the selective search part since it is a fixed algorithm (which could trig bad region proposals).

2.6 SSD

SSD is developed for real-time object detection. SSD speeds up the flow by removing the need of the region proposal network. For recovering the drop in accuracy, SSD applies a few changes including multi-scale features and default boxes. These changes allow the SSD to match the Faster R-CNN's accuracy by utilizing the lower resolution images, which next pushes the speed higher. According to the following comparison, it achieves the real-time processing speed and even beats the accuracy of the Faster R-CNN.

Figure 2.3: SSD Object Detection

2.7 YOLO V-3

YOLOv3 is extremely fast and accurate.YOLO on the other way provides the object detection problem in a completely different way. It moves the whole image only once through the network. SSD is another object detection algorithm that forwards the image once though a deep learning network, but YOLOv3 is much faster than SSD while attaining very comparable accuracy. YOLOv3 gives faster than realtime results on a M40, TitanX or 1080 Ti GPUs.

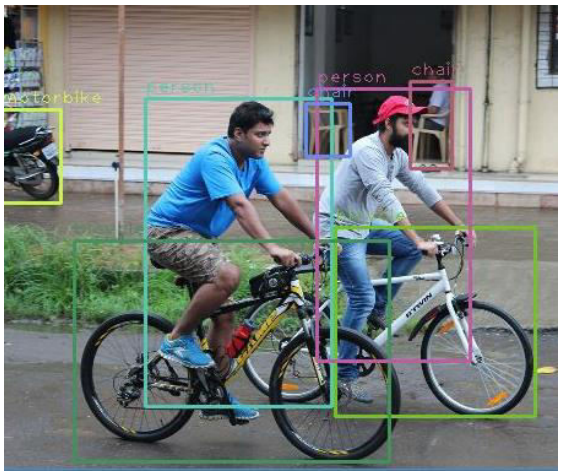


Figure 2.4: YOLO Object Detection

2.8 YOLO V-4

YOLO is a Fast as compared to other variants. It is good for real-time processing. The Predictions (object locations and classes/groups) are made from one single network. It can be trained end-to-end to improve accuracy.

YOLO easily accesses the whole image in predicting boundaries and creates the boundary boxes. With the extra context, YOLO demonstrates fewer false positives in background areas. YOLO detects one object per grid cell which makes it differentfrom others. It enforces spatial diversity in making predictions.

Figure 2.5: YOLO Bounding Box

2.9 Conclusion:

After studying thoroughly, we conclude that YOLO is a more generalized variant. It outperforms other methods when generalizing from natural images to other domains like artwork which makes it different from others. The Region proposal methods limit the classifier to the specific region. YOLO easily accesses the whole image in predicting boundaries and creates the boundary boxes.

3. Methods

3.1 Design of DataSet:

A data set is a collection of data. In other words, a knowledge set corresponds to the contents of 1database table, or one statistical data matrix, where every column

of the table represents a specific variable, and every row corresponds to a given member of the info set in question.

In Machine Learning projects, we need a training data set. It is the particular data set wont to train the model for performing various actions.

Why do we need a data set?
ML depends heavily on data, without data, it is impossible for an "AI" to learn. It is the foremost crucial aspect that makes algorithm training possible… regardless of how great your AI team is or the dimension of your data set, if your data set isn't ok, your entire AI project will fail! I have seen fantastic projects fail because we didn't have a good data set despite having the perfect use case and very skilled data scientists.

Gathering of data is enough but it is the opposite. In every AI project, classifying and labeling data sets takes most of our time , especially data sets accurate enough to reflect a realistic vision of the market/world. In our we have to collect lots of images in different classes to cover most of the objects in our training model.

3.2 Use of Ground Truth Fixation Data:

When analysing the fixation data that was collected during the experiment, it turned out that bottom-up effects had a big influence on the fixations: Independent of the task, the participants fixated for example faces and animals quite reliably. As a result, the fixation data includes a bias towards objects that are behaviourally relevant for the human observers. To make things worse, this bias is independent from the search task, so the fixation maps for the same image do not differ much for different target categories. It seems like the remaining differences between the fixations for the three tasks did not suffice for the model to learn task-specific weights. Visual inspection revealed almost no difference between the saliency maps produced by the models that should search for clocks, laptops or beds.

Removing the Bottom-up Bias:

The first option is to try to remove task-unrelated fixations from the fixation maps. It is of course not possible to precisely determine which fixations are task-related and which are not. A good heuristic is probably to reduce the fixations in areas that were fixated by participants from all three tasks. Modifying the fixations in this way could be considered bad scientific style: The reason why learning algorithms are used in the first place is that the underlying processes are not known. Removing fixations that are thought to not be relevant for the task has always the risk of excluding too many fixations and thereby losing information that the model could have used otherwise. Or in other words, by assuming that regions that were fixated under all tasks were not relevant for the search task, one introduces one's own assumptions about the process of human search into the data.

Adding Target Location Information:

The second possibility to make the ground truth data more task-specific is to add the knowledge about position and form of the target objects. The analysis of the fixation data showed that the targets usually drew no especially high amount of attention, so the actual location of the targets is not emphasised by the fixation data.

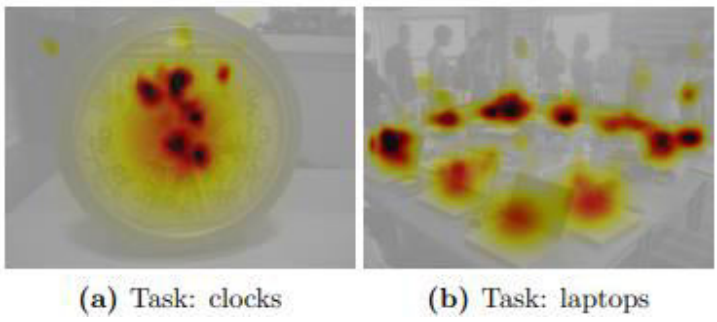


Figure 3.1: Ground Truth Fixation Data

4. Network Tuning

4.1 Introduction

One of the principal causes of poor data transfer performance is packet loss between the data transfer client and server hosts. There are many possible causes of packet loss, ranging from bad or failing hardware to misconfigured hosts or network equipment. Network design, including proper configuration of hosts, routers and switches, can eliminate packet loss and end in significantly improved data transfer performance.

4.2 Training and Validation

To get started, you need prepare your dataset in the Pascal VOC Format and organize it as detailed below:

– Decide the type of object(s) you want to detect and collect about 200-300 (minimum recommendation) or more picture of each of the object(s)

– Once you have collected the images, you need to annotate the object(s) in the images. You can use a tool like LabelIMG to generate the annotations for your images.

– Once you've got the annotations for all of your images, create a folder for your dataset (E.g headsets) and during this parent folder, create child folders train and validation.

– Within the train folder, create images and annotations sub-folders. Put about 70-80% of your dataset of each object's images in the images folder and put the corresponding annotations for these images in the annotations folder.

– Within the validation folder, create images and annotations sub-folders. Put the rest of your dataset images within the images folder and put the corresponding annotations for these images within the annotations folder.

– The structure of your picture dataset folder should look like:

>>train>> images >> img_1.jpg (shows Object_1)
>>images>> img_2.jpg (shows Object_2)
>>images>> img_3.jpg (shows Object_1, Object_3 and Object_n)
>>annotations >> img_1.xml (describes Object_1)
>>annotations >> img_2.xml (describes Object_2)
>>annotations >> img_3.xml (describes Object_1, Object_3 and Object_n)

>>validation>> images >> img_151.jpg (shows Object_1, Object_3 and Object_n)
>>images>> img_152.jpg (shows Object_2)
>>images>> img_153.jpg (shows Object_1)
>>annotations >> img_151.xml (describes Object_1, Object_3 and Object_n)
>>annotations >> img_152.xml (describes Object_2)
>>annotations >> img_153.xml (describes Object_1)

- Now we can train the custom detection model completely from scratch or use transfer learning (recommended for better accuracy) from a pre-trained YOLOv3 model.
- For the purpose of training your detection model, we have the Tensorflow-GPU

4.3 Comparison of Input features and Image Size

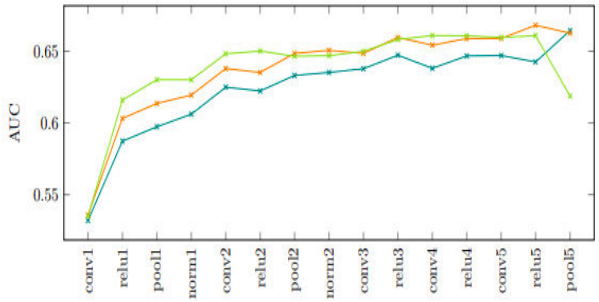
The most basic setup to determine which layers of the neural network provide the best input for the model and which size the input images should have for the feature extraction.

For this evaluation, three datasets (each split into training, validation and test set) had to be constructed, that included all output feature maps of the neural network for all its layers. For the set big, input images were used at full resolution (1024 × 768 pixels), for medium the resolution was halved (512 × 384 pixels) and for small halves again (256 × 192 pixels). As the feature maps of the different layers have different sizes, all were rescaled to 64 × 48 pixels, which is the size of the biggest output feature map for the smallest input image size.

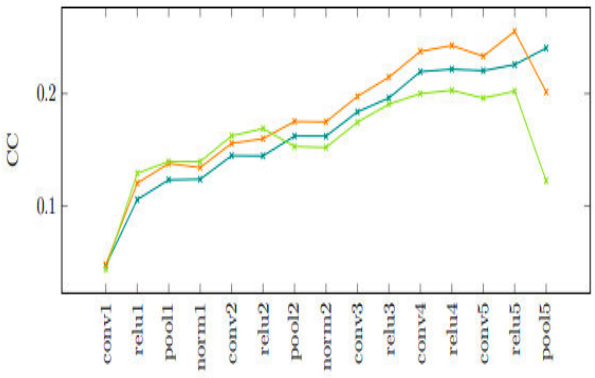
The network that is trained only consisted of the input layer that loads the data form the hdf5-files, the linear combination layer, a softmax calculation and the loss layer

Higher resolution improves object detection for little objects significantly while also helping large objects. When decreasing resolution by an element of two in both dimensions, accuracy is lowered by 15.88% on the average but the inference time is additionally reduced by an element of 27.4% on average.

Concerning the size of the input images, the medium input image size was chosen, as the overall best scores could be achieved with this configuration.



AUC scores



CC scores

4.4 Effect of Blurring

The benefit of applying a Gaussian blur to the linear combination of the input feature maps was evaluated by testing its effect with the selected input layers on the medium image size testset. After initializing a new test network, the weights were adapted according to a Gaussian function with a specific σ. Two different values for σ were evaluated: σ = 0.7 was found to fit best into a convolution kernel of size 5 × 5 and σ = 1.1 was used for a 7 × 7 kernel. While a stronger blurring (i.e. a bigger σ) might have been even more beneficial, this kernel size was found to be the biggest size for which the backpropagation step of the network training could still be computed without running out of memory.

The blurring can increase all scores, but by a smaller amount than expected. Blurring had the strongest effect on the CC score while the AUC and SIM scores only improved by less than 2 %.

5. Evaluation and Result

5.1 Evaluation on Different Data Set:

There are a number of open-source datasets available on the internet that we can easily get and use for our work. Some most common datasets are listed below that we can use.

1. MS-COCO

COCO is large-scale and rich for object detection, segmentation and captioning dataset. It has several features:

- 330K images (>200K labeled)
- 1.5 million object instances
- 80 object categories
- 5 captions per image

Reference Link: https://cocodataset.org/#home

2. Open Images Dataset

Open Images may be a dataset of just about 9 million URLs for images. These images are annotated with image-level labels bounding boxes spanning thousands of classes. The dataset accommodate a training set of 9,011,219 pictures, a validation set of 41,260 pictures and a test set of 125,436 images.

Reference Link:
https://storage.googleapis.com/openimages/web/index.html

3. CIFAR-10

The CIFAR-10 dataset cotain of 60000 32x32 colour pictures in 10 classes, with 6000 images per class. There endure 50000 training pictures and 10000 test pictures.

The dataset is split into five training batches and one test batch, each with 10000 pictures. The test group contains exactly 1000 randomly-selected pictures from each class. The training groups contain the remaining pictures in random order, but some training groupsmay contain more pictures from one class than another. Between them, the training groups contain exactly 5000 pictures from each class.

Reference link:
http://www.cs.toronto.edu/~kriz/cifar.html

4. YOLO Pre-Build Model

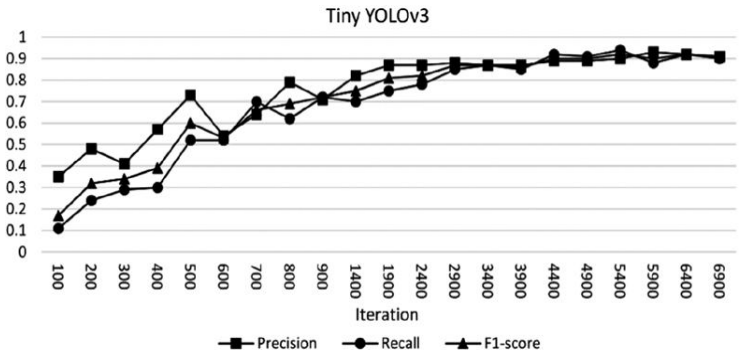
We can train our custom detection model completely from scratch or use transfer learning from a pre-trained YOLOv3 model for better accuracy. Also, You can train YOLO from scratch if you want to play with different training regimes, hyper-parameters, or datasets. YOu can use the COCO dataset for that. To train YOLO you'll need all of the COCO data and labels.

5.2 Experiments and Result:

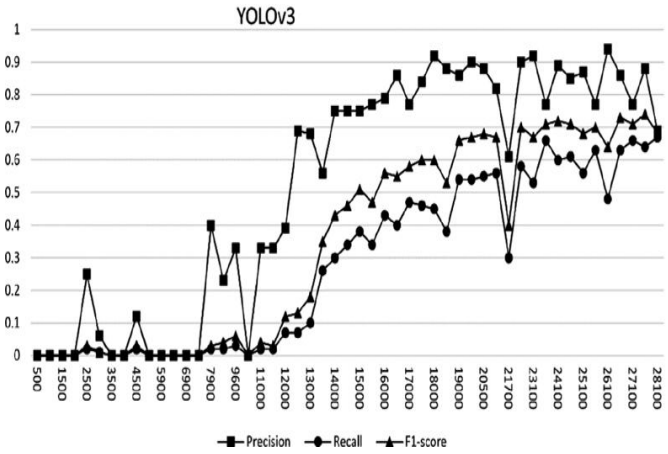
At starting, Tiny YOLO was trained on datasets of different sizes (for the same object) for the same number of steps. It is supposed that N is the size of a dataset. A small part of these experiments for different N.

To make a detector with a pretty good accuracy one needs only a few training examples while to make a very accurate detector many thousands of examples are required. The exact values depend on the complexity of the target object and its environment. It's worth noting that regardless of the dataset size, detector's recall usually has lower values compared to precision which is consistent with YOLO. Detection events can be regarded as more significant than the lack of it.

Since there are two main modern architectures of YOLO: full YOLOv3 and Tiny YOLOv3 we're going to compare their training process on small datasets.



Performance measures depending on the number of steps trained for the smaller version of YOLO—Tiny YOLOv3. From the graph it can be seen that the training process is rather stable and accuracy metrics grow almost monotonically. Additionally, only several thousands of steps are required to finish training.



The graph full network training is much less stable. At first, for about 7000 steps accuracy metrics are stuck at around zero. In comparison to that smaller network reaches 90% accuracy from the same number of steps. Next accuracy metrics do start to grow but slower than in the smaller version. One of the reasons could be the relatively small number of dataset images which produces asymmetry between positive and negative examples during training, and because of that the net tends to evaluate all bounding boxes as negative. Regardless of the reason Tiny YOLOv3 shows better results on our dataset.

Speed performance of two networks. Smaller version not only trains faster but requires less steps in total.

Network	Training speed	Training time	Inference time
YOLOv3	450 steps/hour	47 h	29 images/second
Tiny YOLOv3	3 700 steps/hour	8 min	67 images/second

Table 5.1 : Evaluation/Result

6. Conclusion and Future prospects

Due to its strong learning ability and benefits in dealing with occlusion, scale transformation and background switches, deep learning based object detection has been a research hotspot in the last few years. This provides a detailed review on object detection frameworks which handle different sub-problems, such as accuracy and performance, with different degrees of modifications on YOLO. The study starts on generic object detection pipelines which gives base architectures for other related tasks. Then, three other common tasks, namely object detection and image classification, are also briefly reviewed. Finally, we propose several promising future directions to gain a thorough understanding of the object detection landscape. This review is also useful for the developments in CNN and related learning systems, which provides valuable insights and guidelines for future progress.

REFERENCES

[1] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, “Object detection with discriminatively trained part-based models,” IEEE Trans. Pattern Anal. Mach. Intell., 2010.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in NIPS, 2012.

[3] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in CVPR, 2014.

[4] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards realtime object detection with region proposal networks,” in NIPS, 2015.

[5] Z. Yang and R. Nevatia, “A multi-scale cascade fully convolutional network face detector,” in ICPR, 2016.

[6] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in CVPR, 2016.

[7] D. Ribeiro, A. Mateus, J. C. Nascimento, and P. Miraldo, “A real-time pedestrian detector using deep learning for human-aware navigation,” arXiv:1607.04441, 2016.

[8] P. Druzhkov and V. Kustikova, “A survey of deep learning methods and software tools for image classification and object detection,” Pattern Recognition and Image Anal., vol. 26, no. 1, p. 9, 2016.

[9] Z. Cao, T. Simon, S.-E.Wei, and Y. Sheikh, “Realtime multi-person 2d pose estimation using part affinity fields,” in CVPR, 2017.

[10] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3d object detection network for autonomous driving," in CVPR, 2017.

[11] R. J. Cintra, S. Duffner, C. Garcia, and A. Leite, "Low-complexity approximate convolutional neural networks," IEEE Trans. Neural Netw. & Learning Syst, 2018.

[12] Kulik S.D., Shtanko A.N. (2020) "Experiments with Neural Net Object Detection System YOLO on Small Training Datasets", 2020.