Real-Time Object Detection and Recognition

Neural-Networks - Assignment 3

May 16, 2018

Abstract

Computer Vision, object detection and recognition is a growing field within the realm of Artifical Intelligence. Neural networks can be used to detect and recognize objects in images, and since video is just a sequence of images, we can also use this technology to extract information from video. Artificial neural networks can be trained to detect and recognize objects in images with increasing accuracy and efficiency. This paper aims to give an overview of various recent groundbreaking developments in neural network architectures as well as important datasets. We conclude with results from our benchmarking experiments.

1 Introduction

There is a lot of data on the Internet. This data can be used with suitable processes such as big data analysis which is the process of collecting, organizing and analyzing large amounts of data to discover data patterns and useful information. There are many types of data, and one of them is images, pictures. Image data contains a lot of information which can be used in various systems such as license plate recognition systems, environmental awareness for self-driving vehicles, augmented reality in smartphones and the Google reverse image search.

The race and progress in recent years towards robust and efficient object recognition and detection in images was enabled by the creation of large-scale, publicly available data sets. These data sets established challenging benchmarks to evaluate new methods for visual object recognition that have substantially improved the state-of-the-art across a broad range of computer vision tasks. Object recognition in images is an interesting topic, in which we attempt to model and mimic one of the human processes which enable an understanding of our environments. In other words, object recognition is one of the building blocks of Artificial Intelligence.

Deep learning and artificial neural networks are commonly used to help solve big data problems. In this paper we will discuss notable datasets and recent groundbreaking neural network architectures. We will also look into how to make use of these advancements with the help of tools such as OpenCV, Keras, TensorFlow and pre-trained models [25]. Neural networks come in many different shapes and sizes. We will discuss some notable architectures commonly used in the field of computer vision, such as Deep Convolutional Neural Networks (CNN) and their variants (R-CNN, Fast R-CNN, Faster R-CNN, VGG) as well as Residual Networks (ResNet) and Single Shot Multibox Detectors (SSD)[3].

2 Related work

Significant progress has been made in the last couple of years in this field. Over the past couple of years significant progress has been made and researches that have tired for archiving object recognition in images or video live feed. Several papers have proposed way of using Scale Invariant Feature Transform (SIFT) algorithm [17, 10]. Paper [17] suggested an image recognition system with pyramidal descriptor adapted SIFT algorithm and paper [10] proposed image recognition system for colorectal polyp histology with SIFT. Research [1] suggested persimmon growing monitoring system with analyzing image and paper [23] proposed image recognition system with three dimensional Speed Up Robust Feature (SURF) algorithm.

We also studied several recent papers which have proposed ways of using deep networks for locating class-specific or class agnostic bounding boxes [23, 18, 4, 22]. In the OverFeat method [18], a fully-connected (fc) layer is trained to predict the box coordinates for the localization task that assumes a single object. The MultiBox methods [4, 22] generate region proposals from a network whose last fc layer simultaneously predicts multiple (e.g., 800) boxes, which are used for R-CNN [6] object detection. However, their proposal network is applied on a single image or multiple large image crops (e.g., 224×224) [22]. We will briefly discuss this subject later in context within our methods and solutions section. Other shared computation of convolutions [18, 7, 2, 5] has been attracting increasing attention for efficient, yet accurate, visual recognition. The OverFeat paper [18] computes conv features from an image pyramid for classification, localization, and detection.

Adaptively-sized pooling (SPP) [7] on shared conv feature maps is proposed for efficient region-based object detection [7, 15] and semantic segmentation [2]. Fast R-CNN [5] enables end-to-end detector training on shared conv features and shows compelling accuracy and speed.

3 Methods and Solutions

3.1 Image classification

Image classification takes an image and predicts the object in an image. For example, when we built a cat-dog classifier, we took images of cat or dog and predicted their class:

Figure 1: Finding the difference between 2 objects

The main question is what will actually happen when, let's say, dog and a cat present in the picture. What will the model predict? One of the obvious solutions to solve this problem is that we can train a multi-label classifier which will predict both the classes(dog as well as cat). However, we still won't know the location of cat or dog. The problem of identifying the location of an object(given the class) in an image is called localization. However, if the object class is not known, we have to not only determine the location but also predict the class of each object.

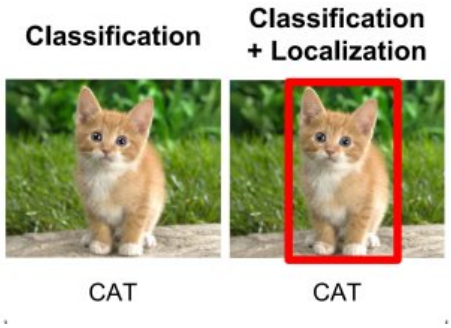


Figure 2: Classification vs Classification plus Localization

Predicting the location of the object along with the class is called object Detection. In place of predicting the class of object from an image, we now have to predict the class as well as a rectangle(called bounding box) containing that object. It takes 4 variables to uniquely identify a rectangle. So, for each instance of the object in the image, we shall predict following variables:

1. class_name
2. bounding_box_top_left_x_coordinate

3. bounding_box_top_left_y_coordinate
4. bounding_box_width
5. bounding_box_height

Just like multi-label image classification problems, we can have multi-class object detection problem where we detect multiple kinds of objects in a single image.
In the following subsections we will cover some of the popular methodologies to train object detectors. Historically, there have been many approaches to object detection starting from Haar Cascades proposed by Viola and Jones in 2001. However, we will be focusing on state-of-the-art methods, all of which use neural networks and Deep Learning.

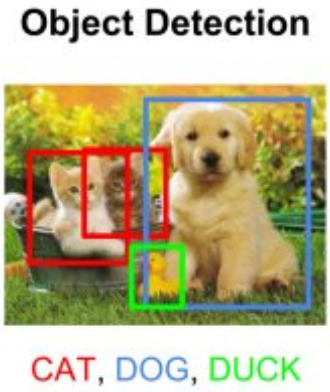


Figure 3: Multiple objects detection in a single image

Object Detection is modeled as a classification problem where we take windows of fixed sizes from input image at all the possible locations and feed these patches to an image classifier. On an image each rectangle window is fed to the classifier which predicts the class of the object in the window (or background if none is present). Hence, we know both the class and location of the objects in the image. But a few problems can arise, such as making sure a window always contains an image [19].

To solve this problem an image pyramid is created by scaling the image. The idea is that we resize the image at multiple scales and we count on the fact that our chosen window size will completely contain the object in one of these resized images. Most commonly, the image is downsampled (it's size is reduced) until a certain condition, typically a minimum size is reached. On each of these images, a fixed size window detector is run. It's common to have as many as 64 levels on such pyramids. Now, all these windows are fed to a classifier to detect the object of interest. This will help us solve the problem of size and location.

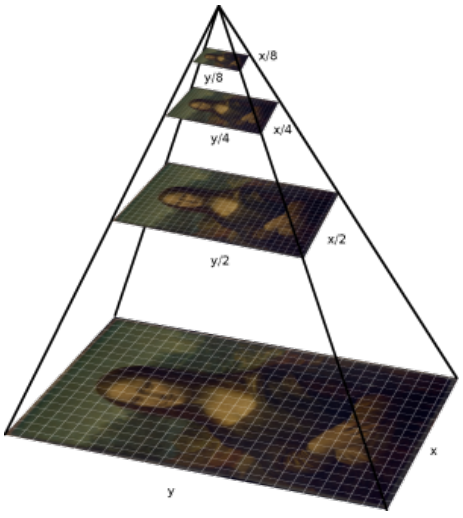


Figure 4: Classifier to detect the object of interest in multiple scales

There is one more problem, aspect ratio. A lot of objects can be present in various shapes like a sitting person will have a different aspect ratio than a standing person or sleeping person. There are various methods for object detection like RCNN, Faster-RCNN, SSD. Lets explore.

Object Detection using Hog Features

One of the most groundbreaking papers ever published in computer vision world is by Navneet Dalal and Bill Triggs introduced Histogram of Oriented Gradients(HOG) features in 2005. Hog features are computationally inexpensive and are good for many real-world problems. On each window obtained from running the sliding window on the pyramid, we calculate Hog Features which are fed to an SVM(Support vector machine) to create classifiers. This can run in real time

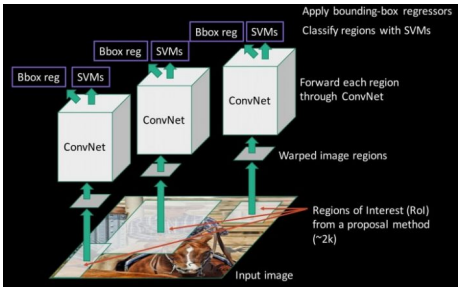
r 

Figure 5: Region-based Convolutional Neural Networks(R-CNN) Architecture

on videos, for example for pedestrian detection, face detection, and many other object detection use-cases.

Region-based Convolutional Neural Networks(R-CNN)

Since object detection has been modeled as a classification problem, success depends on the accuracy of classifications. After the rise of deep learning, it must have been an obvious choice to replace HOG based classifiers with more accurate onvolutional neural network based classifiers. However, there was one problem. CNNs can be slow and computationally expensive when working with very large datasets. R-CNN solves this problem by using an object proposal algorithm called [Selective Search] which reduces the number of bounding boxes that are fed to the classifier to close to 2000 region proposals. Selective search uses local cues like texture, intensity, color and other measures to generate all the possible locations of an object. Now, we can feed these boxes to our CNN based classifier. It is also worth noting that the fully connected classification part of CNNs take a fixed sized input so we resize(without preserving aspect ratio) all the generated boxes to a fixed size (224×224 for VGG) and feed to the CNN part. Hence, there are 3 important parts of R-CNN:

1. Run Selective Search to generate probable objects.
2. Feed these patches to CNN, followed by SVM to predict the class of each patch.
3. Optimize patches by training bounding box regression separately.

Spatial Pyramid Pooling(SPP-net)

There was still room for improvement, RCNNs can be slow, mainly because running CNN on 2000 region proposals generated by Selective search takes a lot of time. SPP-Net tried to alleviate performance bottlenecks by calculating the CNN representation for the entire image only once and can use that to calculate the CNN representation for each patch generated by Selective Search. This can be done by performing a pooling type of operation on JUST that section of the feature maps of last conv layer that corresponds to the region. The rectangular section of conv layer corresponding to a region can be calculated by projecting the region on the conv layer by taking into account the down-sampling happening in the intermediate layers(simply dividing the coordinates by 16 in case of VGG).

Another challenge is that we need to generate a fixed size of input for the fully connected layers of the CNN so, SPP introduces one more trick. It uses spatial pooling after the last convolutional layer as opposed to the traditionally used max-pooling. The SPP layers divide a region of any arbitrary size into a constant number of bins and max pool is performed on each of the bins. Since the number of bins remains the same, a constant size vector is produced as demonstrated in the figure below.

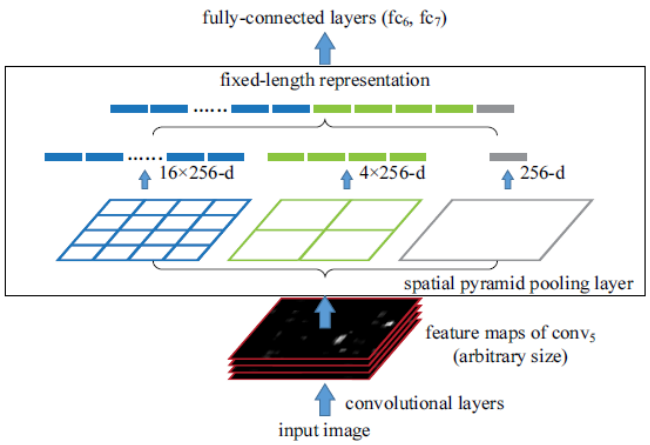


Figure 6: The architecture and the idea behind Spatial Pyramid Pooling(SPP-net)

One of the big drawbacks with SPP-Net is that it isn't trivial to perform back-propagation through a spatial pooling layer. Hence, the network only fine-tuned the fully connected part of the network. SPP-Net paved the way for more popular Fast RCNN..

Fast R-CNN

Fast RCNN uses the ideas from SPP-Net and RCNN and fixes the key problem in SPP-Net i.e. they made it possible to train end-to-end. To propagate the gradients through spatial pooling, It uses a simple back-propagation calculation which is very similar to the max-pooling gradient calculation with the exception that pooling regions overlap and therefore a cell can have gradients pumping in from multiple regions.
One more thing that Fast RCNN did was adding the bounding box regression to the neural network training itself. Now the network had two heads, a classification head and a bounding box regression head. This multitasking objective is a salient feature of Fast-RCNN as it no longer requires training of the network independently for classification and localization. These two changes reduce the overall training time and increase the accuracy in comparison to SPP-Net because of the end to end learning of CNN.

Faster R-CNN

Faster R-CNN is an object detection algorithm proposed by Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun in 2015. The research paper is titled 'Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks'[12]. Faster R-CNN builds on previous work to efficiently classify object proposals using deep convolutional networks.

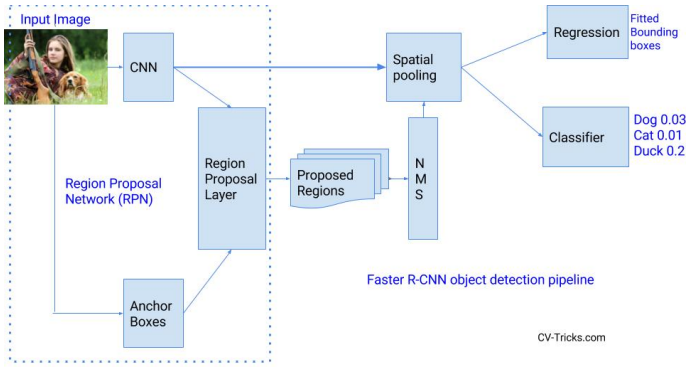


Figure 7: Faster R-CNN: Faster RCNN replaces selective search with a convolutional network called RPN to generate regions of Interests

The bottleneck in Fast R-CCN is Selective Search or Edge boxes. Compared to previous work, Faster R-CNN employs a region proposal network (explained a bit later in this report) and does not require an external method for candidate region proposals. To handle the variations in aspect ratio and scale of objects, Faster R-CNN introduces the idea of anchor boxes. At each location, the original paper uses 3 kinds of anchor boxes for scale 128x 128, 256×256 and 512×512. Similarly, for aspect ratio, it uses three aspect ratios 1:1, 2:1 and 1:2. So, In total at each location, we have 9 boxes on which RPN predicts the probability of it being background or foreground. The varying sizes of bounding boxes can be passed further by apply Spatial Pooling just like Fast-RCNN. The remaining network is similar to Fast-RCNN. Faster-RCNN is 10 times faster than Fast-RCNN with similar accuracy of datasets like VOC-2007. That's why Faster-RCNN has been one of the most accurate object detection algorithms. Here is a quick comparison between various versions of RCNN [19].

	R-CNN	Fast R-CNN	Faster R-CNN
Test Time per Image	50 Seconds	2 Seconds	0.2 Seconds
Speed Up	1x	25x	250x

Figure 8: Comparison between various versions of RCNN

3.2 Object recognition using Tensorflow Object detection API

Last year google released a object detection API using tensorflow. The very first release contains some pre-trained models (especially with a focus on light-weight models, so that they can run on mobile devices) which one could use this in conjunction with open-cv and recognize object in real-time using either your webcam or live videos. The codebase is an open-source framework built on top of TensorFlow that makes it easy to construct, train and deploy object detection models. The goals of this designing this system was to support state-of-the-art models while allowing for rapid exploration and research. The very release contains the following:
Note: Some of the subjects below will be explained briefly in later sections in this paper.
A selection of trainable detection models, including:

1. - Single Shot Multibox Detector (SSD) with MobileNets
2. - SSD with Inception V2
3. - Region-Based Fully Convolutional Networks (R-FCN) with Resnet 101
4. - Faster RCNN with Resnet 101
5. - Faster RCNN with Inception Resnet v2

Frozen weights (trained on the COCO dataset) for each of the above models to be used for out-of-the-box inference purposes.

1. - A Jupyter notebook for performing out-of-the-box inference with google's released models.
2. - Convenient local training scripts as well as distributed training and evaluation pipelines via Google Cloud

The SSD models that use MobileNet are lightweight, so that they can be comfortably run in real time on mobile devices. A winning COCO submission in 2016 used an ensemble of the Faster RCNN models, which are more computationally intensive but significantly more accurate.[16]

3.2.1 VGG16

VGG16, proposed by Simonyan et al. in 2014 [21] is a Deep Convolutional Neural Network. The goal was to study the effects of the depth of a CCN on it's accuracy. They were competing in the ImageNet Challenge in 2014 and landed the first two places in the competition. Simonyan et al. subsequently publicized their CNN in order to facilitate further research. VGG is a popular choice due to it's relatively simple, uniform architecture, as seen in figure 9. It's a fairly standard CNN. VGG16 is composed of 16 layers, but there are other variants, such as VGG19 with 19 layers.

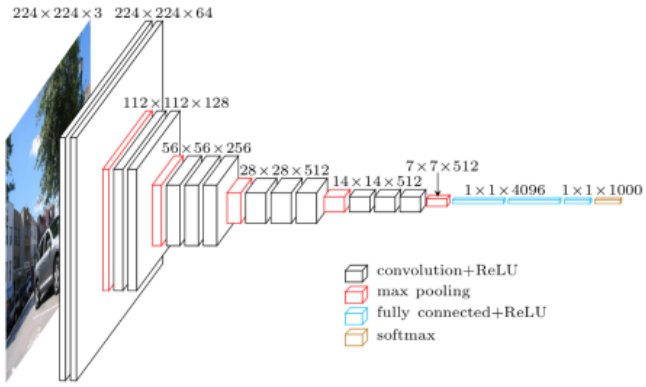


Figure 9: The VGG16 CNN

3.2.2 ResNet50

In their paper, Deep Residual Learning for Image Recognition, He et al. [8] describe a novel network structure, the Residual Network. They argued that increasingly deep CNN networks are getting harder to train, so they proposed a new approach. They argue that deep CNNs commonly suffer from the vanishing gradient problem, which in summary causes the early layers of a network to learn more slowly when layers are added. Residual Networks are composed of residual blocks, each of which has an identity shortcut. Shortcut connections allow gradients to skip one ore more layers through these shortcuts.

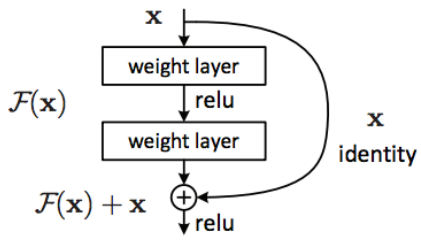


Figure 10: A residual block, the fundamental building block of a Residual Network

So far, all the methods discussed handled detection as a classification problem by building a pipeline where first object proposals are generated and then these proposals are send to classification/regression heads. However, there are a few methods that pose detection as a regression problem. Two of the most popular ones are YOLO and SSD. These detectors are also called single shot detectors.

3.3 YOLO (You only Look Once)

A simple detection is a regression problem which takes an input image and learns the class probabilities and bounding box coordinates. The idea behind YOLO is that it divides each image into a grid of S x S and each grid predicts N bounding boxes and confidence. The confidence reflects the accuracy of the bounding box and whether the bounding box actually contains an object(regardless of class). YOLO also predicts the classification score for each box for every class in training. You can combine both the classes to calculate the probability of each class being present in a predicted box. A total of SxSxN boxes are predicted. However, most of these boxes have low confidence scores and if we set a threshold, say 30 % confidence, we can remove most of them as shown in the example below.

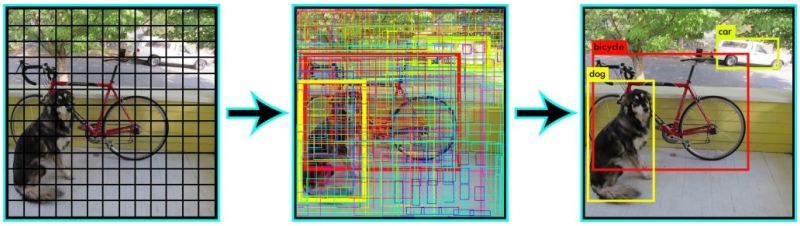


Figure 11: YOLO(You only Look Once)

Notice that at runtime, we have run our image on CNN only once. Hence, YOLO is super fast and can be run real time. Another key difference is that YOLO sees the complete image at once as opposed to looking at only a generated region proposals in the previous methods. So, this contextual information helps in avoiding false positives. However, one limitation for YOLO is that it only predicts 1 type of class in one grid hence, it struggles with very small objects.

3.4 Single Shot MultiBox Detector

Single Shot MultiBox Detector (SSD) is a method proposed by Liu et al. in 2016 [13]. It's based on VGG16, combined with carefully engineered classification layers. It can efficiently and accurately classify multiple objects in a single frame, with bounding boxes. Liu et al. used the previously mentioned COCO dataset for training.

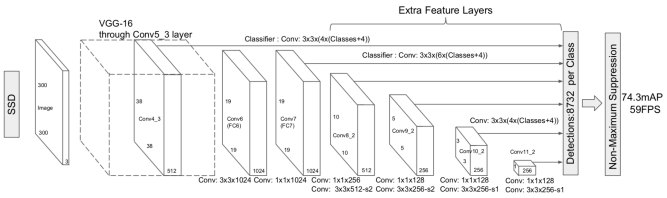


Figure 12: Single Shot MultiBox Detector

Single Shot Detector achieves a good balance between speed and accuracy. SSD runs a convolutional network on input image only once and calculates a feature map. Now, we run a small 3×3 sized convolutional kernel on this feature map to predict the bounding boxes and classification probability. SSD also uses anchor boxes at various aspect ratio similar to Faster-RCNN and learns the off-set rather than learning the box. In order to handle the scale, SSD predicts bounding boxes after multiple convolutional layers. Since each convolutional layer operates at a different scale, it is able to detect objects of various scales.

4 Experiments

4.1 Datasets

4.1.1 ImageNet

ImageNet, as described by Krizhevsky et. al [11], is an image database organized according to the WordNet [14] hierarchy. It has advanced the field of computer vision and object recognition significantly by providing an unprecedented amount of images, with a vast array of categories. The list below shows some general statistics about this dataset. Synsets are categories, including high-level categories and their subcategories. SIFT (Scale-Invariant feature transform) are a sort of feature descriptions, obtained with an algorithm that can detect edges of objects within an image, for example by finding high contrast keypoints.

- Total number of non-empty synsets: 21841

- Total number of images: 14,197,122
- Number of images with bounding box annotations: 1,034,908
- Number of synsets with SIFT features: 1000
- Number of images with SIFT features: 1.2 million

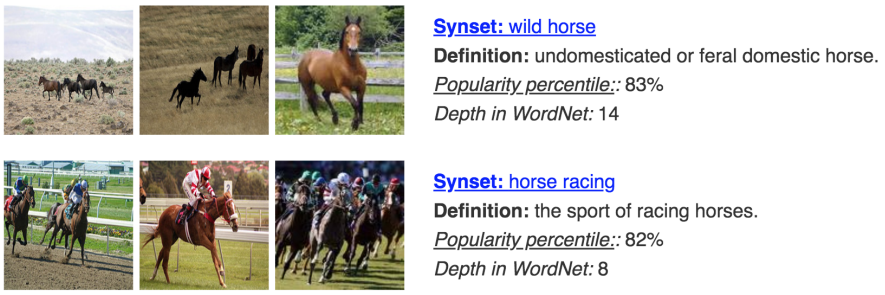


Figure 13: Example Synsets in ImageNet's database

Keras provides a number of pre-trained models which were trained on the ImageNet database. They are Xception, VGG16, VGG19, ResNet50, InceptionV3, InceptionResNetV2, MobileNet, DenseNet and NASNet [9].

4.1.2 COCO

Microsoft's COCO (Common Objects in Context) [12] is a more recent image database. It contains photos of 91 objects with a total of 2.5 million labeled instances in 328k images. Compared to ImageNet, COCO contains fewer categories, but a larger amount of labels. Objects within images have semantic pixel-level segmentation. Semantic labeling requires each individual pixel of an object to be labeled as belonging to a category. This allows extraction of the shape of an object, as can be seen in 14. TensorFlow's Object Recognition API provides a number of object recognition models, which were all trained with the COCO dataset. See [24] for a complete list.

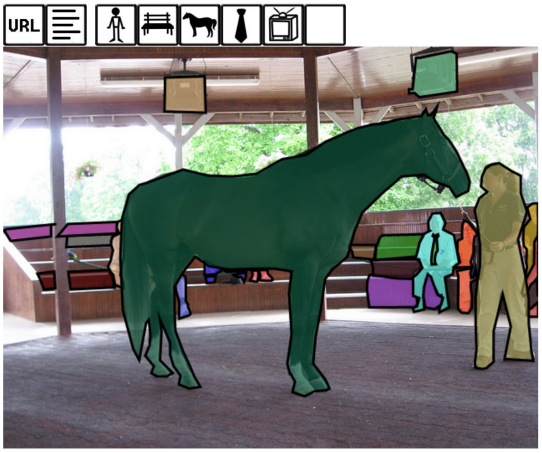


Figure 14: Example of a labeled image in the COCO dataset

4.2 Results and Discussions

Keras and pre-trained Resnet50

We compared various approaches for real time object recognition which we will summarize here. First, we tried a very simple approach. We used Keras and the ResNet50 network, pre-trained using the ImageNet Database. The ImageNet database is described in section 4.1.1. We used OpenCV to capture a video feed, in our case a webcamera. The video source could be anything, and is easily swappable. We split our program into 2 threads, one dedicated to frame classification, using the ResNet50 model to predict frame object labels. The other thread was used to output the video feed, and overlay the label and prediction accuracy on top. This approach considers the whole frame as a single object, which isn't ideal. The background and surroundings of the object interfere with the classification, giving us imprecise results.

Multi-object bounding boxes with TensorFlow, SSD and COCO

The second solution we explored was more complex, but more robust and performant. It can handle identifying multiple objects in a single frame. Each detected object gets a bounding box, which

isolates it from the rest of the image. We used TensorFlow, and a Single-Shot MultiBox Detector (3.4), pre-trained on the COCO dataset described in section 4.1.2. This solution uses multiprocessing instead of threading. We are also using a utility for improved webcam feed performance [16] along with various TensorFlow visualization utilities. This solution can very accurately detect people in a frame since the COCO dataset contains 250,000 instances of people, with keypoints. Our previous ResNet50 and Keras approach really struggled with person classification. Common objects such as smartphones, laptops, pens, coffee cups and cars are also very easily and accurately detected.

4.2.1 Architectural Differences Between SSD and Faster-RCNN

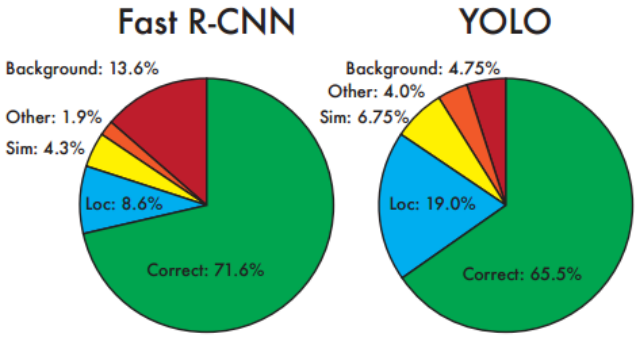


Figure 15: Architectural Differences Between SSD and Faster-RCNN.

Proposal based networks, like faster-RCNN, propose a number of bounding boxes (regions of interest) using one neural network and then send each proposal to two neural nets: one that does image classification on the proposed region. For example, is it a car or a pedestrian or nothing in particular? The other net performs bounding box regression, for instance how should we modify the corners of the proposed region to match the ground truth? This process is quite slow because a huge number of regions are proposed and then evaluated for every image. It is more accurate than the alternatives, however, because you get to use a network whose sole purpose is image classification.

Single shot detectors, like the SSD Mobilenet we experimented with or the bit more accurate YOLO, go straight from image pixels to bounding boxes by making the same default region proposals for every image, and then predicting bounding box coordinates and class probabilities for each proposal in one neural network.

Recent improvements to the original R-CNN architecture are focused on cutting down the computational cost of region proposals in order to bridge the speed gap with the SSDs. Fast R-CNN (2015), for example, improved inference speed from 47s per image to 2s per image by computing the feature volume once per image, and then passing those features, and computes the feature volume used by the proposal network once and then passes convolutional features and proposals (generated by the selective search algorithm) to an RoI pooling layer, which crops the big convolutional feature map to the region included in the proposal, and then resizes the cropped features so that they are consistently shaped.

Faster R-CNN (2016), the architecture we also experimented with, supposedly achieves a further 10x speedup by introducing a region proposal network that takes the place of selective search. The training scheme alternates between training RPN to generate boxes and objectness scores (starting from shared convolutional feature volume) that the image detector (which is unchanged from Fast-RCNN) and bounding box regressor can predict accurately.

The following chart explains the differences between RCNN and its successors, as well as the newer R-FCN, much better which, from what we heard is doing much more better in terms of speed and accuracy.

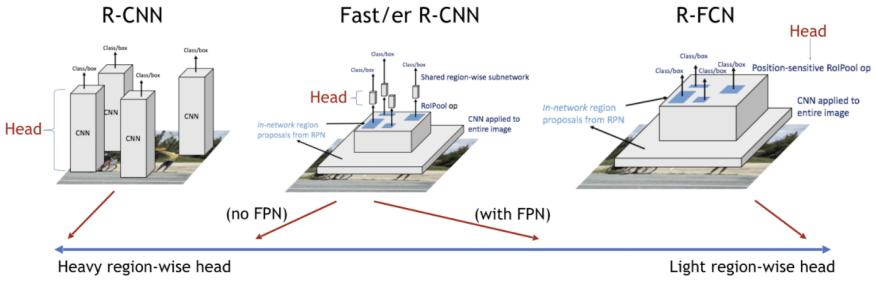


Figure 16: Transition to less region-wise computation

We have worked on Tensorflow object detection API, trained SSD Mobilenet V1 and Faster-RCNN-inception-resnet-v2, which were the least accurate (but fastest) and most accurate (but slowest) pre-trained models available from the model zoo. Unsurprisingly, faster-RCNN was much more accurate and much slower than SSD Mobilenet [20].

Caveats:

- Some mAP scores (such as mean Average Precision, an accuracy metric, higher is better) we were able to capture are below.
- Hardware: Experiments were mostly executed on DSLab (duranium & tritanium) computers using Centos OS.

Final examples we have in which we used Faster R-CNN in which we had the most trouble with.

Field	Faster R-CNN	Mobilenet
final mAP	0.81	0.5
Hours Trained	12	6
Last TotalLoss	0.47	2.11
Min TotalLoss	0.02	1.12
Train Steps per Second	1.05	2.6
Coco MAP	0.16	0.11
Inference Time	1.6	.40

Category	Faster R-CNN	Mobilenet
vehicles	0.722	0.85
cyclist	0.47	2.11
dontcare	0.02	1.12
misc	0.41	0.661
pedestrian	0.257	0.193
overall	0.104	0.208

Table 1: Left: Experiment Results, Right: Faster-RCNN Predictions

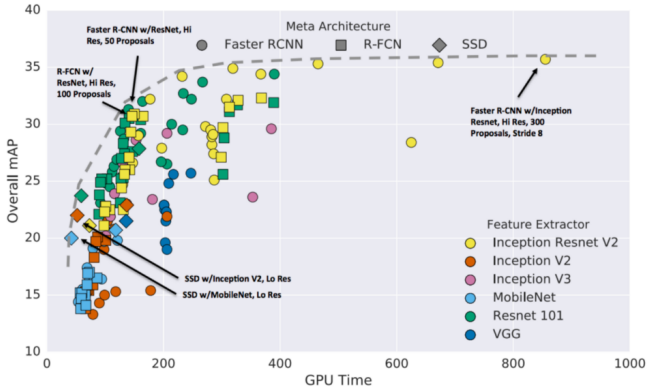


Figure 17: Huang et al. Speed/Accuracy Tradeoffs for Modern Convolutional Object Detectors. CVPR 2017

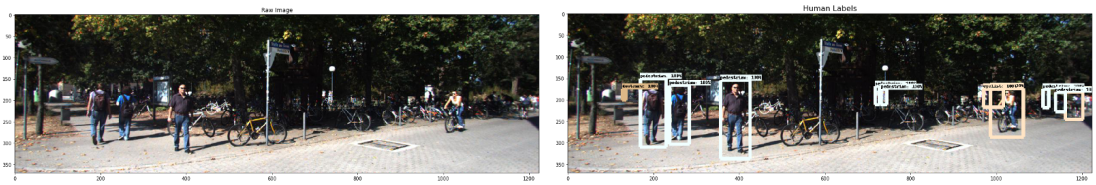


Figure 18: Left: RAW image, Right: Human Labels (provided by Kitti as ground truth

Note that the pedestrians in the center of the image are very hard to see!

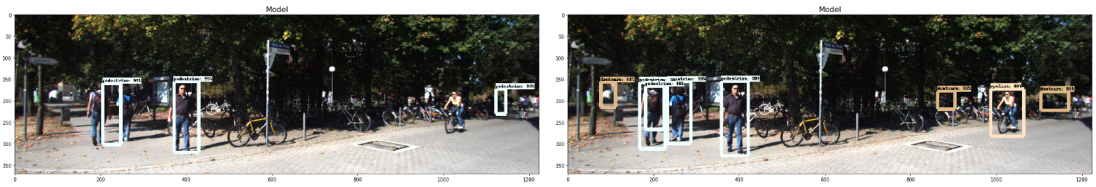


Figure 19: Left: SSD Mobilenet Predictions, Right: Faster-RCNN Predictions

5 Conclusions and Future Work

We discussed a lot of different algorithms in the previous sections. Currently, Faster-RCNN is the choice if one is fanatic about the accuracy. However, if you are strapped for computation, SSD is a better, more resource efficient recommendation. Finally, if accuracy is not too much of a concern but you want to go super fast, YOLO will be the way to go. First of all a visual understanding of speed vs accuracy trade-off:

Figure 20: speed vs accuracy between networks

SSD seems to be a good choice as we are able to run it on a video and the accuracy trade-off is very little. However, it may not be that simple, look at this chart that compares the performance of SSD, YOLO, and Faster-RCNN on various sized objects. At large sizes, SSD seems to perform similarly to Faster-RCNN. However, look at the accuracy numbers when the object size is small, the gap widens.

Choosing the right object detection method is crucial and depends on the problem you are trying to solve and your set-up. Object Detection is the backbone of many practical applications of computer vision such as autonomous cars, security and surveillance, and many industrial applications. Hopefully, this post gave you an intuition and understanding behind each of the popular algorithms for object detection.

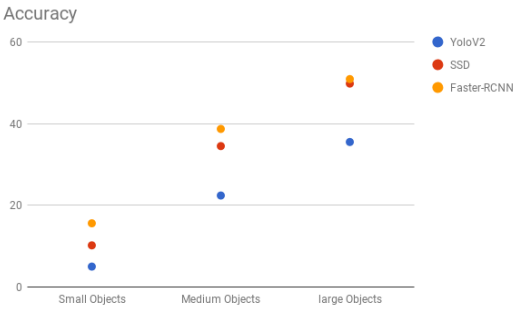


Figure 21: Accuracy trade-off

Most of the mistakes that faster-RCNN makes involve not detecting small objects in the background, many of which we can't see without squinting. None of the architectures make mistakes on cars, unless there's a huge number of cars in the image. According to our research we also understood that Faster-RCNN is 9X slower at time-inference. Perhaps more focus on time-inference for future purposes. We certainly cannot do

as much object detection as these architectures in the same amount of time, at least not consciously. One of the next steps would be to try different models (there are new 6 tensorflow models) and try to replicate the mAP vs inference time plot. One of the tips would be to try to track mAP during eval, rather than after freezing and also time inference on many images, not just one.

References

[1] K. C. Chang, P. K. Liu, Z. W. Kuo, and S. H. Liao. Design of persimmon growing stage monitoring system using image recognition technique. In *2016 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW)*, pages 1–2, May 2016.

[2] Jifeng Dai, Kaiming He, and Jian Sun. Convolutional feature masking for joint object and stuff segmentation. *CoRR*, abs/1412.1283, 2014.

[3] Adit Deshpande. The 9 deep learning papers you need to know about (understanding cnns part 3), 2016. https://adeshpande3.github.io/The-9-Deep-Learning-Papers-You-Need-To-Know-About.html, accessed 2018-05-10.

[4] Dumitru Erhan, Christian Szegedy, Alexander Toshev, and Dragomir Anguelov. Scalable object detection using deep neural networks. *CoRR*, abs/1312.2249, 2013.

[5] Ross B. Girshick. Fast R-CNN. *CoRR*, abs/1504.08083, 2015.

[6] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *CoRR*, abs/1406.4729, 2014.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[9] Keras. Keras applications, 2018.

[10] Yoko Kominami, Shigeto Yoshida, Shinji Tanaka, Yoji Sanomura, Tsubasa Hirakawa, Bisser Raytchev, Toru Tamaki, Tetsushi Koide, Kazufumi Kaneda, and Kazuaki Chayama. Computer-aided diagnosis of colorectal polyp histology by using a real-time image recognition system and narrow-band imaging magnifying colonoscopy. 83, 08 2015.

[11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[12] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[13] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: single shot multibox detector. *CoRR*, abs/1512.02325, 2015.

[14] George A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41, November 1995.

[15] Shaoqing Ren, Kaiming He, Ross B. Girshick, Xiangyu Zhang, and Jian Sun. Object detection networks on convolutional feature maps. *CoRR*, abs/1504.06066, 2015.

[16] Adrian Rosebrock. PyImageSearch increasing webcam fps with python and opencv, 2015. https://www.pyimagesearch.com/2015/12/21/increasing-webcam-fps-with-python-and-opencv/, accessed 2018-05-14.

[17] L. Seidenari, G. Serra, A. D. Bagdanov, and A. Del Bimbo. Local pyramidal descriptors for image recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(5):1033–1040, May 2014.

[18] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *CoRR*, abs/1312.6229, 2013.

[19] Sam Shleifer. Guide to object detection using deep learning: Faster r-cnn,yolo,ssd, 2017. http://cv-tricks.com/object-detection/faster-r-cnn-yolo-ssd/, accessed 2018-05-11.

[20] Sam Shleifer. How to finetune tensorflow's object detection models on kitti self-driving dataset, 2017. https://medium.com/@sshleifer/how-to-finetune-tensorflows-object-detection-models-on-kitti-self-driving-dataset-c8fcfe3258e accessed 2018-05-11.

[21] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[22] Christian Szegedy, Scott E. Reed, Dumitru Erhan, and Dragomir Anguelov. Scalable, high-quality object detection. *CoRR*, abs/1412.1441, 2014.

[23] Christian Szegedy, Alexander Toshev, and Dumitru Erhan. Deep neural networks for object detection. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2553–2561. Curran Associates, Inc., 2013.

[24] TensorFlow. Tensorflow detection model zoo, 2018. https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md, accessed 2018-05-14.

[25] Dat Tran. Building a real-time object recognition app with tensorflow and opencv, 2017. https://towardsdatascience.com/building-a-real-time-object-recognition-app-with-tensorflow-and-opencv-b7a2b4ebdc32, accessed 2018-05-11.