**Review of models for Real time object detection**

**Abstract:** The goal of this paper is to serve as a guide for selecting a detection architecture that achieves real time architecture at given resources. With the rise of autonomous vehicles, smart video surveillance, facial detection and various people counting applications, fast and accurate object detection systems are rising in demand. These systems involve not only recognizing and classifying every object in an image, but localizing each one by drawing the appropriate bounding box around it. This makes object detection a significantly harder task than its traditional computer vision predecessor, image classification. Fortunately, however, the most successful approaches to object detection are currently extensions of image classification models. At the end of this we will come to conclusion of various ways of object detection using the different methods.

**Introduction:** All this years there are various types of paper proposing the perfect architecture, but due to various types of base feature extractors, different image resolutions and different hardware components point comparison is not possible. Keeping all this into consideration. I came up with the optimum solution for selecting the best model for real time object detection based on accuracy, hardware and use cases. Here we will discuss the two ends of line. Faster RCNN and SSD models. There are many papers which will discuss about the particular models, but the review of models based on accuracy and use-case keeping in back of mind.

Faster RCNN and SSD architecture used here is trained on coco datasets.

**1 EXPERIMENTAL SETUP**

**1.1 Feature Extractors**

We will be discussing a few feature extractors to get the gist of the entire architecture, where Meta architectures are combined with feature extractors like VGG, Resnet101 etc. The way it works is that we first apply convolutional neural networks via feature extractors to extract all the high level information from the images. The feature extractors must be wisely selected as types of layers, no of parameters directly affect the speed and memory performance of the object detectors. We will compare four different feature extractors to combine with our Meta architectures and test their performances. Out of the four chosen feature extractors, apart from Mobile Net, every other feature extractor has an open source Tensorflow implementation, which one can use via GitHub. We compare VGG-16, Resnet101, Inception Resnet(v2)[6] ,which combines the optimization quality of ResNet with computational efficiency of inception modules and Mobile Net which achieved accuracies similar to VGG-16 in Image net competition with 1/30 of its computational cost and model size. As the name suggests, Mobile Net because of its low computational cost has been at the forefront of various vision applications in mobile devices. Its building blocks are depth wise separable convolutions which factorize a standard convolution into a depth wise convolution and a 1x1 convolution, effectively reducing both computational cost and number of parameters. In R-FCN and R-CNN , where we must choose the layer that must be used for predicting region proposals, we have used ‘Conv5’ layer in VGG-16, and ‘Conv\_4\_x’ layers in ResNet-101, for other feature extractors we choose similar layers. In SSD, following previous methodologies, we have also selected the topmost convolutional feature map and a higher resolution feature map at a lower level, then adding a sequence of convolutional layers with spatial resolution.

**1.2 Model Details**

Here, we mention the intricate details of the different combination of Meta architectures with the feature extractors.

**1.2.1 Faster R-CNN**

Tensorflow’s standard ROI pooling is used, also batch normalization is used in all convolutional layers. Optimizer used is SGD with momentum set to 0.9. The learning rate was set as per the feature extractor.

**Inception Resnet:** The stride size is 8 in atrous mode and 16 otherwise. The features are extracted from the Mixed\_6a layers including the residual layers. Features maps are 17x17 size and learning rate is 1e-3.

**MobileNet:** The initial learning rate is 3e-3 and stride size is set to 16 pixels. We extract features from Conv2D\_11 layers and features maps of size 14x14 are used.

**1.2.2 SSD**

Batch normalization is used in all layers and the weights are initialized with a standard deviation of 0.03. The convolutional feature maps were added and all of them used for prediction, with the convolutional layers being added with a spatial resolution, decaying by a factor of 2.

**Inception Resnet:** The initial learning rate is set to 0.005, with a decaying factor of 0.8 after every 700k steps. The activation function ReLU is used. Mixed\_6a and Conv2d\_7b are used by appending with additional convolutional layers of depth 512,256,128 respectively.

**MobileNet:** The initial learning rate is set to 0.004 and we have used conv\_11 and conv\_13 layers with four additional layers with decaying spatial resolution with depths of 512,256,256 ,128.The activation function ReLU is used .

**2 Models**

**2.1 Faster-RCNN.**

Basically, Faster-RCNN is composed of 2 neural networks:

At the conceptual level, Faster-RCNN is composed of 3 neural networks — Feature Network, Region Proposal Network (RPN), Detection Network.

**2.1.1 Feature Network.**

The Feature Network is usually a well-known pre-trained image classification network such as VGG minus a few last/top layers. The function of this network is to generate good features from the images. The output of this network maintains the shape and Structure of the original image (i.e. still rectangular, pixels in the original image roughly gets mapped to corresponding feature “pixels”, etc.).

**2.1.2 Region Proposal Network (RPN)**

The RPN consists of simple network with a 3 convolutional layers. The first layer feeds the input into two other layers in network — one for classification and the other for regression of bounding box. The output of RPN is numerous bounding boxes known as **Region of Interests** (ROIs) which is supposed to be containing the detected object in particular or any object. The output is represented in 3 types viz, -1, 0, or 1, which represents box can be ignored, the particular box contains object and box can be ignored respectively.

**2.1.3 Detection Network**

It is also referred as RCNN network takes input from both the Feature Network and RPN, and generates the final class and bounding box. It is normally composed of 4 Fully Connected or Dense layers. There are 2 stacked common layers shared by a classification layer and a bounding box regression layer. To help it classify only the inside of the bounding boxes, the features are cropped according to the bounding boxes.

Both the RPN and Detection Network needs to be trained. This is where most of the complexities of Faster-RCNN lie.

**2.2 Steps involved in Faster RCNN:**

**2.2.1 Training the RPN**

For training the RPN, first a number of bounding boxes are generated by a mechanism called anchor boxes. Every ‘pixel’ of the feature image is considered an anchor. Each anchor corresponds to a larger set of squares of pixel in the original image (some reshaping is usually done on the original image before feature extraction). Anchors are positioned uniformly across both dimensions of the (reshaped) image. The input that is required from the feature generation layer to generate anchor boxes is the shape of the tensor, not the full feature tensor itself. A number of rectangular boxes of different shapes and sizes are generated centered on each anchor. Usually 9 boxes are generated per anchor (3 sizes x 3 shapes). Hence, there are 10s of thousands of anchor boxes per image. In the first step of reduction an operation called Non-Maximum Suppression (NMS) is used. NMS removes boxes that overlaps with other boxes that has higher scores (scores are non-normalized probabilities, e.g. before softmax is applied to normalize). About 2000 boxes are extracted during training phase (the number is lower, about 300 for testing phase). In the testing phase these boxes along with their scores go straight to the Detection Network. In the training phase the 2000 boxes are further reduced through sampling to about 256 before entering the Detection Network. To generate labels for RPN classification (e.g. foreground, background, and ignore), IOU of all the bounding boxes against all the ground truth boxes are taken. Then the IOUs are used to label the 256 ROIs as foreground and background, and ignored. These labels are then used to calculate the cross-entropy loss, after first removing the ignored (-1) class boxes. In addition to classification, the RPN also tries to tighten the center and the size of the anchor boxes around the target. This is called the bounding box regression. For this to happen, targets needs to be generated, and losses needs to be calculated for back propagation. The distance vector from the center of the ground truth box to the anchor box is taken and normalized to the size of the anchor box. That is the target delta vector for the center. The size target is the log of the ratio of size of each dimension of the ground truth over anchor box. The loss is calculated by using an expression called Smooth L1 Loss. The regular L1 loss (e.g. the norm or absolute value) is not differentiable at 0. Smooth L1 Loss overcomes this by using L2 loss near 0. The extent of L2 loss is tuned by a parameter called sigma. Mathematically the formula looks like the following pseudo code.

if abs(d) < 1/sigma\*\*2

loss = (d\*sigma)\*\*2 /2

else

loss = abs(d) — 1/(2\*sigma\*\*2)

The losses are back propagated the usual way to train RPN. The RPN can be trained by itself, or jointly with the Detection Network.

**2.2.2 Training the Detection Network**

The Detection Network can be considered the removed layers (top) of the classification network that is used for features generation. Hence the starting weights can be preloaded from that network before training. Training the Detection Network is similar to that of RPN. First, IOUs of all the 2000 or so ROIs generated by the NMS following RPN against each ground truth bounding box is calculated. Then the ROIs are labeled as foreground or background depending on the corresponding threshold values. Then a fixed number (e.g. 256) ROIs are selected from the foreground and background ones. If there are not enough foreground and/or background ROIs to fill the fixed number, then some ROIs are duplicated at random. The features are cropped (and scaled) to 14x14 (eventually max-pooled to 7x7 before entering the Detection Network) according to the size of the ROIs (for this, ROI width and heights are scaled to the feature size). The set of cropped features for each image are passed through the Detection Network as a batch. The final dense layers output for each cropped feature, the score and bounding box for each class (e.g. 256 x C, 256x4C in one-hot encoding form, where C is the number of classes).

To generate label for Detection Network classification, IOUs of all the ROIs and the ground truth boxes are calculated. Depending on IOU thresholds (e.g. foreground above 0.5, and background between 0.5 and 0.1), labels are generated for a subset of ROIs. The difference with RPN is that here there are more classes. Classes are encoded in sparse form, instead of one-hot encoding. Following a similar approach to the RPN target generation, bounding box targets are also generated.

However, these targets are in the compact form as mentioned previously, hence are expanded to the one-hot encoding for calculation of loss. The loss calculation is again similar to that of the RPN network. For classification sparse cross-entropy is used and for bounding boxes, Smooth L1 Loss is used. The difference with RPN loss is that there are more classes (say 20 including background) to consider instead of just 2 (foreground and background).

**3 SSD (single shot multibox detector)**

>SSD works by converting discrete output spaces for bounding boxes into sets of default boxes for different aspect ratio and for every feature map location.

> During predictions, the model generates the scores in each default box for every object detected and scales the default box to fit the object shape.

**Single Shot**: the tasks of object localization and classification are done in a single forward pass of the network.

**Multibox**: this is the name of the technique for bounding box regression.

**Detector**: this network is an object detector that also classifies those detected objects.

Compared to Faster Rcnn (7 frames per second), SSD is faster. This increase in frames per second comes from the cancellation of Bounding Box proposals and feature resampling stage

SSD uses the VGG16 model pre-trained on Imagenet as is the base model for extracting useful features of the image. In SSD, detection happens in every layer targeting at objects of various sizes.It generates the anchor boxes for every location of the feature map.Each anchor boxes has a fixed size and position relative to its corresponding cellSSD does the RPN and classification in a single step (predicting the bounding boxes and class as it processes the image)

**3.1 Steps in SSD**.

This given step is followed after providing images with respective annotations and annotation is formed using LabelIMG tool.

>Feed the image through a set of convolutional layers, producing several sets of feature maps at different sizes.

>For each location in each of the feature maps uses a 3\*3 convolutional filter to evaluate a small set of default bounding boxes. These bounding boxes are essentially equivalent to Faster RCNN anchor boxes.

>for each box parallel predicts the bounding boxes and class probabilities.

>During training math the ground truth box with these predicted boxes based on IOU (intersection of the union)

**3.2 Training**:

Training is a bit different in SSD. In faster RCNN, the rpn generated ensures that there is certain chances of the detected region contains the object. But that part is skipped here. Here we classify and draw bounding box for every single position in the image, using multiple different shapes at several different scales. This results in several bounding boxes and most of them are negative examples.

To overcome this disadvantage SSD does

>it uses the non-maxima suppression

>it does the hard negative mining to balance class during training.

|  |  |  |
| --- | --- | --- |
| **Model** | **Mobile SSD** | **Faster RCNN** |
| Test Loss  Number of epochs | 1.5  5114 | 0.1016  7167 |

**4 Analysis**

It is noted that usually SSD model is faster than its counterpart; also Faster R-CNN leads to much slower times, taking more than 100ms per image and giving the most accurate models of SSD. It is also found that the intuition of feature extractors’ accuracy correlating with the mAP scores on the inn-house dataset seems to be true only for Faster R-CNN as SSD models as less reliant on the classification accuracy of their feature extractors. The number of region proposals for Faster R-CNN can be reduced without affecting the mAP score by much, in turn saving lots of computation. We found that, in Faster R-CNN Inception Resnet, the mAP score with 300 proposals was 34.2, and it gave a mAP score of 28.7 with just 10 proposals, although the sweet spot is when we used 50 proposals as we are able to achieve more than 93% accuracy of those trained with 300 proposals by saving 3 times the running time. As far as memory utilization is considered, Faster R-CNN inception Resnet consumes the most memory whereas SSD MobileNet consumes the least, unsurprisingly; it is correlating with the speed of these models. The models performed much better on bigger images with more

|  |  |  |  |
| --- | --- | --- | --- |
| **Model Combination** | **mAP score** |  | **CPU time** |
|  |  |  | 40 |
| SSD MobileNet | 19 |  | |
| SSD Inception  Resnet | 20.3 | 80 | |
| Faster R-CNN  MobileNet | 19 | 118 | |
|  |  |  |  |
| Faster R-CNN Inception Resnet | 34.2 |  | 860 |

**5. CONCLUSION**

We have put forward a comparative study of state of the art object detectors which use convolutional neural networks. We have addressed their issues and performance on a common hardware and also tested different combinations of them, all on the in-house dataset. We discovered that SSD performs much better with light weight feature extractors on bigger images, competing with the most accurate of models. We also found that fewer proposals increase speed without compromising much on the mAP scores. We wish to try different combinations and find better results and sweet spots which can be applied to specific use cases.

**REFERENCES**

[1]Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y. and Berg, A.C., 2016,October. Ssd: Single shot multibox detector. At European conference on computer vision (pp.

21-37). Springer, Cham.

[2]Dai, J., Li, Y., He, K. and Sun, J., 2016. R-fcn:Object detection via region-based fully convolutional networks. At Advances in neural information processing systems (pp. 379-387).

[3]Ren, S., He, K., Girshick, R. and Sun, J., 2015. Faster R-CNN: Towards real-time object detection with region proposal networks. In Advances in neural information processing systems (pp. 91-99)

[4]Girshick, R., 2015. Fast r-cnn. At Proceedings of the IEEE international conference on computer vision (pp. 1440-1448).

[5]Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M. and Ghemawat, S., 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:1603.04467.

[6]N.Silberman and S.Guadarrama .Tf-slim: A high level library tzo define complex models in tensorflow. https://research.googleblog.com/2016/08/tf- slim-high-level-library-to-define.html,2016.

[7]T. Y. Lin and P. Dollar. Ms coco api. https://github.com/pdollar/coco, 2016.

[8]Ren, S., He, K., Girshick, R., Zhang, X. and Sun, J., 2017. Object detection networks on convolutional feature maps. IEEE transactions on pattern analysis and machine intelligence,39(7), pp.1476-1481.

[9]J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection.arXiv preprint arXiv:1506.02640, 2015.

[10]A. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang,T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861,2017

[11] Nikhil Yadav , Utkarsh Binay. Comparative Study of Object Detection Algorithms.

International Research Journal of Engineering and Technology (IRJET) Volume: 04 Issue: 11 | Nov -2017