Faster RCNN Bird Object Detection Model

Yuvraj Patra, Mitsumi Nandi and Alvi Rahman
Department of Electrical Engineering
City University of Hong Kong

Abstract

We present an object detection model that detects birds in their natural habitat. We implement different machine learning and deep learning algorithms to determine the best combination of pre-trained models with methods such as non-maximum suppression. The model that resulted in the highest confidence score prediction with the lowest mean square of errors on the bounded boxes values predicted was chosen for the final submission to the Kaggle Competition EE4211-Det.

1. INTRODUCTION

Object detection garnered significant attention over the past couple of years for its vital importance in real work applications such as surveillance and autonomous vehicles. In object detection, there are three fundamental tasks : object classification, object localization, and object detection. Classification is needed to give target labels to our objects and localization is required to give the position of the given label in the image, which is usually labeled by the bounding box. Object detection is recursively doing multiple object localization and classification over an image at the same time. Traditional object detection methods include HOG and DPM detectors. The performance of these models reached their saturation point in the last decade after which deep learning methods were used in object detection which was a considerable turning point. There are primarily two sorts of deep learning based object detection methods: ”two stages detectors” and ”one stage detectors”. The “two stages detectors” usually have better accuracy.

In this paper, firstly, how RCNN works and the different variations of RCNN that are available will be illustrated. After this, MobileNet V3, which is the base network used for our RCNN model will be explained and a brief description of the COCO dataset used to train our RCNN model will be given.

In order to understand how RCNN works, we need to do a recap on CNN. A Convolutional Neural Network is a Deep Learning algorithm which takes an image, assigns weights and biases to various segments in the image and differentiate one from the other.. The architecture mimics the connectivity pattern of neurons in the Human Brain and was inspired by the organization of the Visual Cortex [1].

1.1. R-CNN

In RCNN, a selective search algorithm is used to generate 2000 sub-segmentations of the image that may correspond to a single object and these are called region proposals. These candidate region proposals are then warped into a square and inputted into a CNN that produces a 4096-dimensional feature vector as output. The CNN acts as a feature extractor and the output dense layer consists of the features extracted from the image. SVM is then used to analyze the extracted features and classify the presence of the object within that candidate region proposal. Furthermore, the algorithm also predicts four values which are offset values to increase the precision of the bounding box [2].

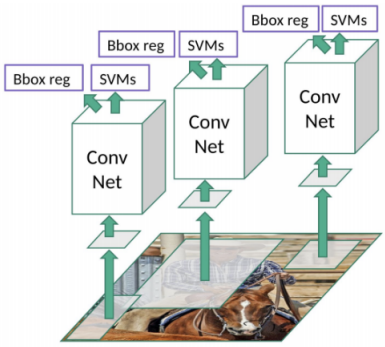

Fig 1.1 R-CNN

1.2. Faster R-CNN

With RCNN, there were a couple of problems such as long duration of time to train the network and no learning taking place in the selective search algorithm which could lead to bad candidates for region proposals.

These flaws were fixed with the development of faster R-CNN. It had a similar approach to R-CNN with the difference being that the image was fed to the CNN to generate a convolutional feature map. Moreover, in Faster R-CNN, a separate network is used to predict the region proposals. From this map. the region of proposals is identified and warped into squares. By using a RoI pooling layer, they are then reshaped into a fixed size so that it can be fed into a fully connected layer. From the RoI feature vector, a softmax layer is used to predict the class of the proposed region and also the offset values for the bounding box [2].

The output size of each convolutional and pooling layer can be calculated precisely by the following two formulas[4]:

$$\text{output\_size}_{conv} = \left\lfloor \frac{\text{input\_size} + 2 * \text{pad} - [\text{dilation} * (\text{kernel\_size} - 1) + 1]}{\text{stride}} \right\rfloor + 1$$

$$\text{output\_size}_{pool} = \left\lceil \frac{\text{input\_size} + 2 * \text{pad} - \text{kernel\_size}}{\text{stride}} \right\rceil + 1,$$

Faster R-CNN used with its own Region Proposal Network is much better and faster than previous iterations. This Region proposal network has an improvement in the region proposal generation model while training. This also helps to reduce the overall detection time as compared to previous versions.

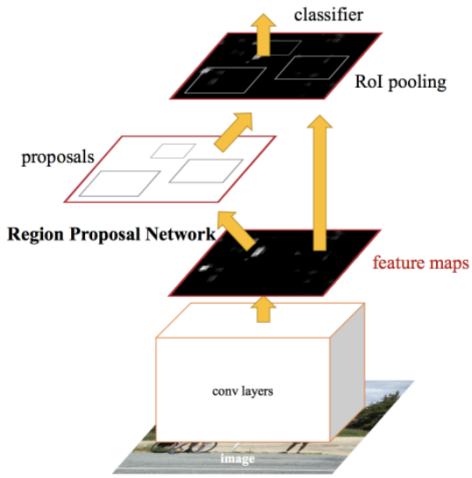

Fig 1.2 Faster R-CNN

1.3. MobileNet V3

MobileNetV3 is optimized for mobile phone CPUs through an amalgamation of hardware aware network architecture search (NAS) complemented by the NetAdapt algorithm and then subsequently improved through novel architecture advances. MobileNetV3 has two models: MobileNetV3-Large and MobileNetV3-Small. These models are targeted at high and low resource use cases respectively. The models are created through applying platform-aware NAS and NetAdapt for network search and incorporating the network improvements. MobileNets utilize depth-wise separable convolutions to build light weight deep neural networks. Two simple global hyper-parameters are used which allow the model builder to choose the right sized model for specific application based on the limits of the problem [5].

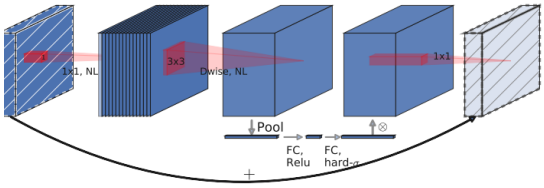

Fig 1.3 MobileNetv3

1.4. COCO Dataset

A large-scale object detection, segmentation, and captioning dataset.Images of complex everyday scenes containing common objects in their natural context. Objects are labeled using per-instance segmentations to aid in precise object localization. The dataset contains photos of 91 easily recognizable objects types with a total of 2.5 million labeled instances in 328k images. Baseline performance analysis for bounding box and segmentation detection results using a Deformable Parts Model is also provided. In the Faster RCNN MobileNet V3 model,Coco dataset is used to pretrain the CNN and the label used is 16 which corresponds to bird segmentations in the dataset. [4]

Fig 1.4 Segmented Bird Image from COCO dataset

2. METHOD

This project used a number of techniques for performing object detection. Owing to the detection models offered by Torchvision and PyTorch, different detection models were tested on the Test dataset to measure the accuracy of the model using mean squared error as the evaluation metric. The model that was finally adopted for the object detection of birds is the Faster RCNN with MobileNetv3 as the backbone CNN Architecture.

2.1. Environment and Setup

The program for implementing the Faster RCNN MobileNetv3 was written in python. Both Visual Studio Code (VS Code) and Google Colab were used for this purpose. In VS Code, a new python environment was set up and conda commands were used to install the required libraries such as PyTorch, Torchvision and Tensorflow.

2.2. PyTorch and Torchvision Pre-trained Model

PyTorch is an open source machine learning framework that accelerates the path from research prototyping to production development. TorchVision is the python library for computer vision that works with PyTorch for efficient image and video transformations, as well as some pre-trained models. For this project the pre-trained models of torchvision were used.

The torchvision.models subpackage contains definitions of models for solving different problems, such as image classification, segmentation, object detection etc. For the purpose of this project we used the object detection models that were pre-trained on the COCO dataset (See 1.4). The Faster RCNN MobileNetv3 Large FPN model was finally used for this project as it generated the best results. It constructs a high resolution Faster R-CNN model with a MobileNetV3-Large FPN backbone.

2.3. NMS: Non Maximum Suppression

Non-maximum suppression can be defined as a computer vision method to filter out the proposed bounded boxes generated by the model and select one detection box out of the overlapping boxes. This can be determined based on certain criteria. The arguments for the NMS function are the generated bounded boxes in the form of 4 coordinates (Xmin, Ymin, Xmax, Ymax), confidence score of each bounded box and the overlapping threshold which has been set to 0.5.
Algorithm: The bounded box with the highest confidence score is inputted to the output list and the remaining proposed boxes are compared with this bounded box. The IoU(Union of Intersection) score is applied to compare the boxes, if the score is greater than the threshold, 0.5 then the proposed box is discarded. This process is repeated until there are no more proposed bounded boxes left in the input list to compare with the bounded box in the output list. This way we are able to generate one bounded box to detect the bird in the test image.

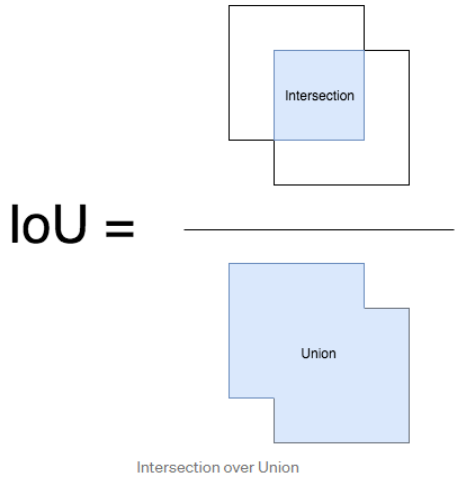

Fig 2.1 IoU is a measure of overlap between 2 bounded boxes.

Fig 2.2 Illustration of NMS

2.4. Program Design Flow

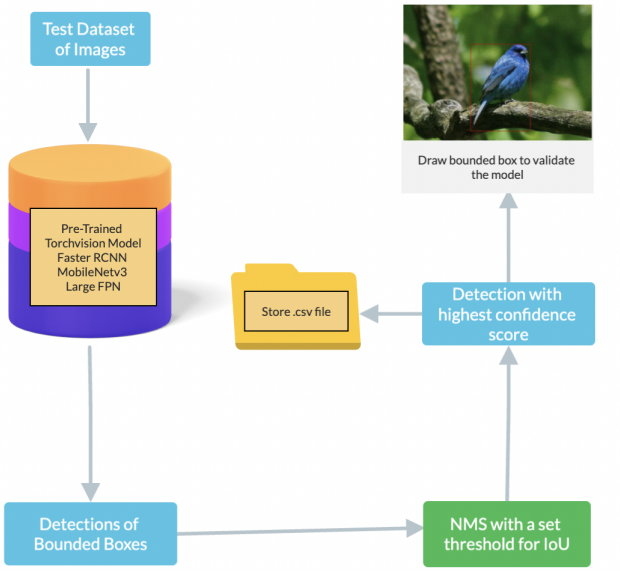

Fig 2.3

The program iteratively reads through the folder containing the test data images. The torchvision.models.detection.fasterrcnn_mobilenet_v3_large_fpn model is imported and the image is converted to a tensor, after which it is passed into the model to generate a result. The result from the model consists of multiple detections in the form of lists that have the coordinates of the bounded boxes in [xmin, ymin, xmax, ymax] format. The bird detections are screened out from this result with the label id that corresponds to birds in the COCO dataset (16). However, there are still a number of detections that have been generated. At this stage, the detections are passed into a function that performs non-maximum suppression. A 0.5 threshold for IoU is set and the function returns a single desired detection with the highest confidence score. This detection is used to draw the bounded box on the image for which it has been computed and the resultant image is stored in a folder. The bounded box values of xmin, ymin, xmax, and ymax are systematically stored in lists and finally inputted into a DataFrame. The DataFrame is then stored into a .csv file using the pandas library in python. The .csv file is saved at a definite location on the system. The file is then submitted to Kaggle to check the mean squared error which was the metric for evaluation in the competition.

3. RESULTS

We obtained the best result for Mean Squared Error with Faster RCNN MobileNetv3 out of all the machine learning and deep learning models that were implemented. The MSE score obtained was 392.93416. The confidence scores of all the birds detected were mostly above 90%. Another problem that was resolved by using MobileNetv3 as our backbone architecture was that camouflaged birds were detected more accurately. Other backbone architectures like ResNet50, could not accurately detect these camouflaged birds and this issue was overcome by changing to MobileNetv3.

Fig 3.1 Test image with bounded box

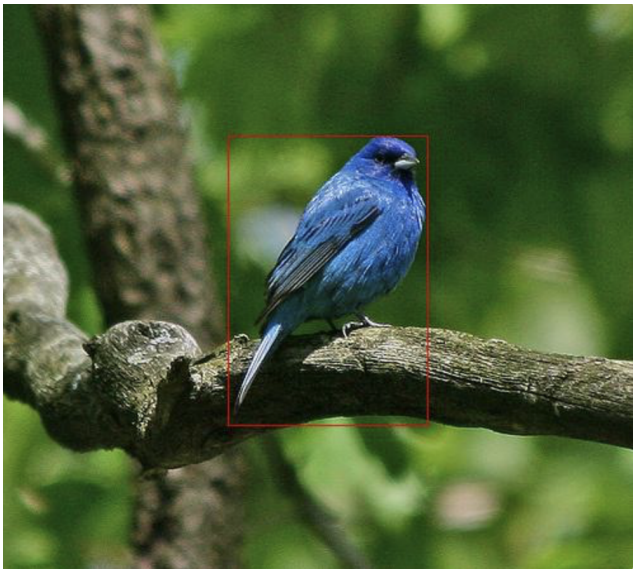

Fig 3.2 Test image with bounded box

Fig 3.3 ResNet50 v/s MobileNetv3

4. Conclusion

In this project we experimented with miscellaneous models like:

- Yolov3 and Yolov4
- Machine Learning (HOG+ML+NMS)
- Deep Learning models:
  - ❖ Simple RCNN
  - ❖ Mask RCNN with ResNet50
  - ❖ Faster RCNN with ResNet50
  - ❖ Faster RCNN with MobileNetv3

After implementing these models, it was observed that the best result was given by Faster RCNN with MobileNetv3.

5. References

[1] S. Saha, "A comprehensive guide to Convolutional Neural Networks-the eli5 way," *Medium*, 17-Dec-2018. [Online]. Available:https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53. [Accessed: 17-Apr-2022].

[2] R. Gandhi, "R-CNN, fast R-CNN, Faster R-CNN, YOLO - object detection algorithms," *Medium*, 09-Jul-2018. [Online].Available:https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e. [Accessed: 17-Apr-2022].

[3] Y. Ren, C. Zhu, and S. Xiao, "Object detection based on fast/faster RCNN employing fully convolutional architectures," *Mathematical Problems in Engineering*, 09-Jan-2018. [Online]. Available: https://www.hindawi.com/journals/mpe/2018/3598316/. [Accessed: 17-Apr-2022].

[4] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft Coco: Common Objects in Context," *arXiv.org*, 21-Feb-2015. [Online]. Available: https://arxiv.org/abs/1405.0312v3.[Accessed:17-Apr-2022].

[5] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam, "Searching for MobileNetV3," *arXiv.org*, 20-Nov-2019.[Online].Available: https://arxiv.org/abs/1905.02244. [Accessed: 17-Apr-2022].

[6] S. K, "Non-maximum suppression (NMS)," *Medium*, 30-Apr-2021. [Online]. Available: https://towardsdatascience.com/non-maximum-suppression-nms-93ce178e177c. [Accessed: 16-Apr-2022].