Global Wheat Detection

Raja Haseeb Ur Rehman
Department of Aerospace
KAIST
South Korea
rajahaseeb147@kaist.ac.kr

1 Introduction

To understand an image completely, not only we should concentrate on classifying different images but also try to precisely locate and estimate the position of various objects in an image. This task is referred to as object detection [3], and is one of the fundamental problems in computer vision. Object detection can provide valuable information for semantic understanding of images and videos and is related to many applications including image classification [9][11], human behaviour analysis, face recognition [19], autonomous driving and so on.

In this project, our task is to identify wheat heads in Global Wheat Head Dataset using object detection and image analysis. We will detect wheat heads from outdoor images of wheat plants including wheat datasets from around the globe. We will focus on a generalized solution to estimate the number and size of wheat heads using worldwide data. To better gauge the performance for unseen genotypes, environments, and observational conditions, the training dataset covers multiple regions.

However, it can be visually challenging to accurately detect wheat heads in outdoor field images. There is often overlap of wheat dense plants, blur in photographs due to wind, matching colors. Additionally, appearances vary due to maturity, color, genotype, and head orientation. We also need to obtain better generalization results between different growing environments. Improved detection farmers can better assess crops, ultimately bringing cereal, toast, and other favorite dishes to your table.

1.1 Conventional Methods

The frameworks of generic object detection can mainly be categorized into two types (see Figure 1). One follows traditional object detection pipeline, in which first region proposals are generated and then each proposal is classified into different object categories. The other regards object detection as a regression or cclassification problem, adopting a unified framework to achieve final results (category and location) directly. The region proposal based methods mainly include R-CNN [6], SPP-net [7], Fast R-CNN [5] , Faster R-CNN [10], R-FCN [1], FPN [12] and Mask R-CNN [8], some of which are correlated with each other (e.g. SPP-net modifies RCNN with a SPP layer). The regression/classification based methods mainly includes MultiBox [2], AttentionNet [20], G-CNN [14], YOLO [16], SSD [13], YOLOv2 [17], DSSD [4] and DSOD [18].

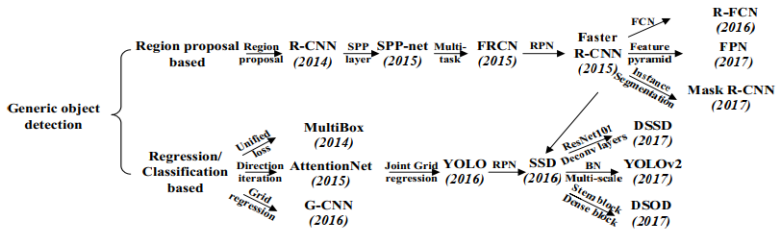

Figure 1: Two types of frameworks: region proposal based and regression/classification based [21].

1.1.1 You Only Loo Once (YOLO)

For YOLO, detection is a straightforward regression dilemma that takes an input image and learns the class possibilities with bounding box coordinates. YOLO divides every image into a grid of S x S and every grid predicts N bounding boxes and confidence. The confidence reflects the precision of the bounding box and whether the bounding box contains an object despite the defined class. YOLO even forecasts the classification score for every box for each class. You can merge both the classes to work out the chance of every class being in attendance in a predicted box.

So, a total of SxSxN boxes are forecasted. On the other hand, most of these boxes have lower confidence scores and if we set a doorstep say 30% confidence, we can get rid of most of them.

1.1.2 Single Shot Detector (SSD)

The SSD object detection composes of 2 parts i.e. extract feature maps, and apply convolution filters to detect objects. SSD uses VGG16 to extract feature maps. Then it detects objects using the convolution layer. SSD attains a better balance between swiftness and precision. SSD runs a convolutional network on input image only one time and computes a feature map. Now, we run a small 3×3 sized convolutional kernel on this feature map to foresee the bounding boxes and categorization probability.

SSD does not use a delegated region proposal network. Instead, it resolves to a very simple method. It computes both the location and class scores using small convolution filters. After extracting the feature maps, SSD applies 3 × 3 convolution filters for each cell to make predictions.

1.1.3 Faster R-CNN

The most widely used state of the art version of the R-CNN family — Faster R-CNN was first published in 2015. These networks usually consist of — a) A region proposal algorithm to generate "bounding boxes" or locations of possible objects in the image; b) A feature generation stage to obtain features of these objects, usually using a CNN; c) A classification layer to predict which class this object belongs to; and d) A regression layer to make the coordinates of the object bounding box more precise.

Both R-CNN and Fast R-CNN use CPU based region proposal algorithms, Eg- the Selective search algorithm which takes around 2 seconds per image and runs on CPU computation. The Faster R-CNN [10] paper fixes this by using another convolutional network (the RPN) to generate the region proposals. This not only brings down the region proposal time from 2s to 10ms per image but also allows the region proposal stage to share layers with the following detection stages, causing an overall improvement in feature representation.

2 Method

In my approach, I used a Faster-RCNN ResNet50 model, which is pre-trained on MS COCO dataset. Transfer learning, where the goal is to transfer knowledge from a related source task, is commonly used to compensate for the lack of sufficient training data in the target task [15]. Fine-tuning is arguably the most widely used approach for transfer learning when working with deep learning models. Compared with training from scratch, fine-tuning a pre-trained convolutional neural network

on a target dataset can significantly improve performance, while reducing the target labeled data requirements.

Faster-RCNN is one of the most well-known object detection neural networks [10]. It is also the basis for many derived networks for segmentation, 3D object detection, the fusion of LIDAR point cloud with image, etc. At the conceptual level, Faster-RCNN is composed of 3 neural networks — Feature Network, Region Proposal Network (RPN), Detection Network.

The Feature Network is usually a well known pre-trained image classification network such as VGG or ResNet minus a few 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.

The RPN is usually a simple network with 3 convolutional layers. The purpose of RPN is to generate a number of bounding boxes called Region of Interests (ROIs) that have a high probability of containing any object. The output from this network is several bounding boxes identified by the pixel co-ordinates of two diagonal corners, and a value (1, 0, or -1, indicating whether an object is in the bounding box or not or the box can be ignored respectively).

The Detection Network (sometimes also called the RCNN network) takes input from both the Feature Network and RPN, then 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. Both the RPN and Detection Network needs to be trained. This is where most of the complexities of Faster-RCNN lies.

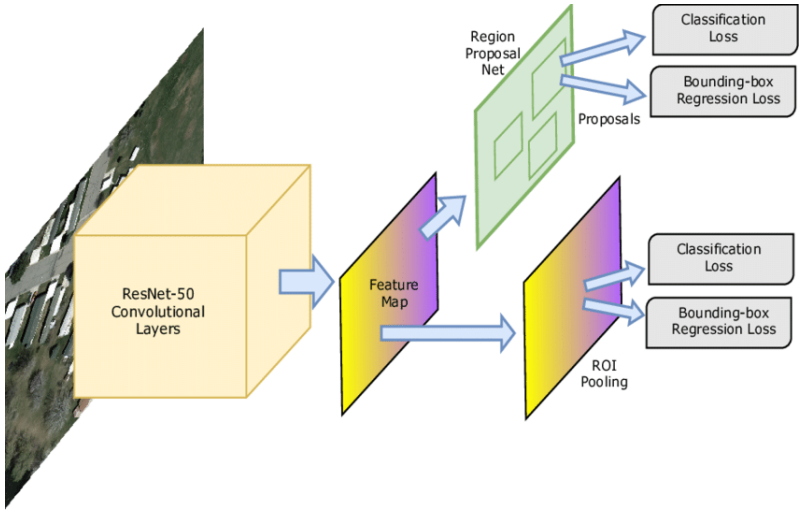

Figure 2: Model architecture

3 Experiment

3.1 Implementation Details

For the framework, PyTorch is used. Since I am going to fine-tune a pre-trained Faster R-CNN model, so the pre-trained model is also imported from torchvision models. For the fine-tuning stage, I will be modifying the Faster R-CNN predictor for which predictor class is also imported at the start. Next I use pandas to open the 'train.csv' file. Here we can see that for each image, we have an image_id and also for each image_id we have multiple bounding box definitions. The format of the bounding box is coordinates of the top left corner along with the width and height of the box.

Next, we have some function to load the training images. OpenCV is used to load images. Next, we obtain the coordinates of the bottom right corner as well to match the format expected by the Faster R-CNN predictor. I also create a simple function to draw bounding boxes on an image just to visualize them. Then defining the model that I will be training and eventually use to detect bounding boxes of wheat plants. Here I am using the Faster R-CNN model, which is pre-trained on COCO dataset. After downloading the model, I expand it to take a look at the network architecture. Here we

Figure 3: Validation example

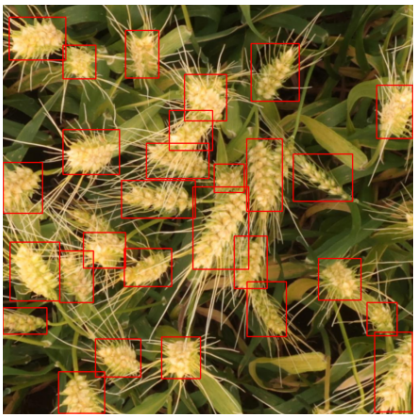

Figure 4: Test example

make some slight changes in the predictor layer. In the downloaded model, we have 91 output classes, which corresponds to the number of classes in the COCO dataset. For each class, there are four parameters for the bounding boxes, and this is why we have 364 output features for the bounding box predictions. We change this number of outputs to two classes since in our dataset we have two classes i.e. wheat and background class. Now we have eight features for the bounding box predictions.

Data is divided into two subsets, training and validation set with 80/20 split. In total, we have 2699 training samples and 674 validation samples. We also create labels with index 1 which corresponds to the wheat class. We don't have to explicitly create any labels for the background class. Then we create training and validation data loaders. Now we proceed to the training part. For optimizer, we used SGD with momentum to optimize over all parameters in the model. A learning rate of 0.005 is used. Loss is calculated for each image in the training batch and the new take average over the losses. The model is trained for 10 epochs and the loss decreases gradually.

3.2 Results and Analysis

After that, we check what our model has learned on the validation dataset. The state of the model is turned to evaluation so that we don't calculate all the unnecessary gradients. After that, we make the generator and then go through some examples and see how our model has learned. According to results, it looks like the recall is relatively good. Ground truth bounding boxes are shown in green and predicted boxes by red color. We can see that wherever there is a green box showing a ground truth wheat plant, there is some kind of red box prediction around it. In some cases predicted box is almost similar to the ground truth.

However, we have one issue with precision. We see that there are some spurious detections, where there isn't a ground truth wheat plant. So there is room for improvement. For now, we save the model, and then in a separate kernel, we can load up this fine-tuned model and run it on the test dataset. Examples of testing and validation are shown in Figures 3 and 4 respectively. From the inference results, we can see that model is performing very well and detects almost all the wheat heads in the image.

4 Conclusion

R-CNN algorithms have truly been a game-changer for object detection tasks. There has suddenly been a spike in recent years in the amount of computer vision applications being created, and R-CNN is at the heart of most of them. Faster R-CNN is one of the models that proved that it is possible to solve complex computer vision problems with the same principles that showed such amazing results at the start of this new deep learning revolution. Experiments on the Global Wheat Dataset also shows the effectiveness of Faster R-CNN models, and the model performs relatively well during the inference stage as well. However further improvements can be made via fine-tuning and other methods.

References

[1] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 379–387, Red Hook, NY, USA, 2016. Curran Associates Inc.

[2] Dumitru Erhan, Christian Szegedy, Alexander Toshev, and Dragomir Anguelov. Scalable object detection using deep neural networks, 2013.

[3] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010.

[4] Cheng-Yang Fu, Wei Liu, Ananth Ranga, Ambrish Tyagi, and Alexander C. Berg. Dssd : Deconvolutional single shot detector, 2017.

[5] Ross Girshick. Fast r-cnn, 2015.

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

[7] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1904–1916, 2015.

[8] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn, 2017.

[9] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding, 2014.

[10] H. Jiang and E. Learned-Miller. Face detection with the faster r-cnn. In *2017 12th IEEE International Conference on Automatic Face Gesture Recognition (FG 2017)*, pages 650–657, 2017.

[11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[12] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection, 2016.

[13] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. *Lecture Notes in Computer Science*, page 21–37, 2016.

[14] Mahyar Najibi, Mohammad Rastegari, and Larry S. Davis. G-cnn: an iterative grid based object detector, 2015.

[15] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[16] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection, 2015.

[17] Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger, 2016.

[18] Zhiqiang Shen, Zhuang Liu, Jianguo Li, Yu-Gang Jiang, Yurong Chen, and Xiangyang Xue. Dsod: Learning deeply supervised object detectors from scratch, 2017.

[19] Zhenheng Yang and Ramakant Nevatia. A multi-scale cascade fully convolutional network face detector. pages 633–638, 12 2016.

[20] Donggeun Yoo, Sunggyun Park, Joon-Young Lee, Anthony S. Paek, and In So Kweon. Attentionnet: Aggregating weak directions for accurate object detection, 2015.

[21] Zhong-Qiu Zhao, Peng Zheng, Shou tao Xu, and Xindong Wu. Object detection with deep learning: A review, 2018.