Faster R-CNN from Scratch

Ismail Cebi
guscebis@student.gu.se

Shivneshwar Velayutham
shivel@chalmers.se

ABSTRACT

The objective of this project is to create an implementation of the Faster Region-based Convolutional Neural Network (Faster R-CNN) from scratch. Known for its high accuracy and efficiency in various computer vision tasks, Faster R-CNN is a widely used object detection framework. Our motivation for this project is to gain a comprehensive understanding of the model and its components. We utilize the Region Proposal Network (RPN) and Region-based Convolutional Network (RCNN) in our approach for detecting objects in images. The results of our experiments on training on the PASCAL VOC dataset show that our implementation of Faster R-CNN achieves fairly decent results with additional unnecessary bounding boxes. Additionally, we provide a brief overview of the relevant literature and techniques used in our project to demonstrate our grasp on the underlying theory.

1. INTRODUCTION

The field of computer vision has seen significant advancements in recent years, thanks to the application of deep learning techniques. One of the most challenging tasks in computer vision is object detection, where the goal is to detect and classify objects of interest in an image. This task has numerous real-world applications, such as in detecting standing dead trees[4], medical imaging eg. detection and classification of COVID-19 [6], solid waste management [5] , autonomous vehicles, surveillance, and defect detection in industrial productions[11] .

To achieve state-of-the-art results in object detection, many researchers have turned to the use of deep learning models. One such model is Faster Region-based Convolutional Neural Network (Faster R-CNN), which has been shown to achieve high accuracy and efficiency in various object detection tasks.

In this project, our objective is to implement Faster R-CNN from scratch and evaluate its performance on the PASCAL VOC dataset [1]. Our motivation for this project is to gain a deeper understanding of the model and its components. To approach this task, we utilize the Region Proposal Network (RPN) and the Region-based Convolutional Network (RCNN). The RPN generates candidate object regions, while the RCNN classifies and refines these regions. This project is heavily based on the original paper on Faster R-CNN (Ren et al., 2015) and its implementation in the Pytorch library. We also draw on relevant literature and techniques used in deep learning, specifically in the area of object detection. We draw inspiration for building Faster R-CNN from scratch from this guide [2].

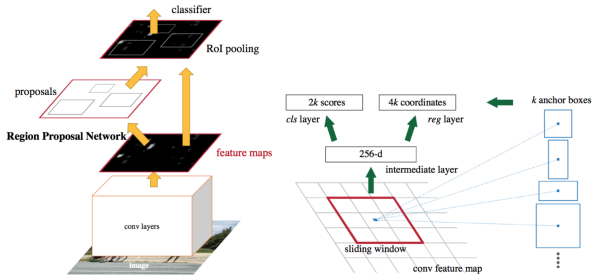

Figure 1: Faster Region-based Convolutional Neural Network

2. BACKGROUND THEORY

2.1 Evolution of Faster RCNN

Faster Region-based Convolutional Neural Network (Faster R-CNN) is a popular object detection framework introduced by Ren et al. [8]. It consists of two main components: the Region Proposal Network (RPN) and the Fast Region-based Convolutional Network (RCNN). The RPN generates potential object proposals, while the Fast RCNN applies a classifier and a regressor to refine these proposals. In earlier implementations of Faster R-CNN,, the RPN used a CPU based exhaustive Selective Search algorithm to generate proposals, which was computationally expensive. To overcome these limitations, research efforts have focused on improving the speed and accuracy of the framework.

2.1.1 Region-based Convolutional Neural Network

Figure 2 illustrates the operation of the Region-based Convolutional Neural Network (RCNN) [3], which represents the initial step that paved the way for the development of Faster RCNN. In this network, an input image is partitioned into 2K region proposals, each of which is then subjected to classification using a Convolutional Neural Network (CNN).

2.1.2 Fast RCNN

Figure 3 illustrates the operational principles of Fast R-CNN, a significant advancement over its predecessor R-CNN and a precursor to Faster R-CNN. One of the key enhancements introduced in Fast R-CNN is the transformation of

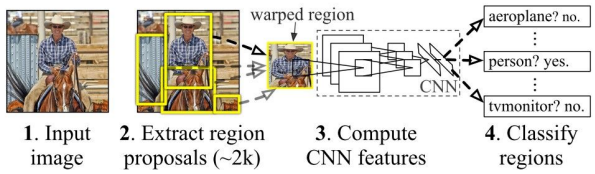

Figure 2: Region-based Convolutional Neural Network

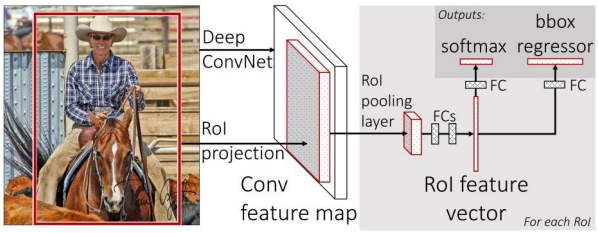

Figure 3: Fast Region-based Convolutional Neural Network

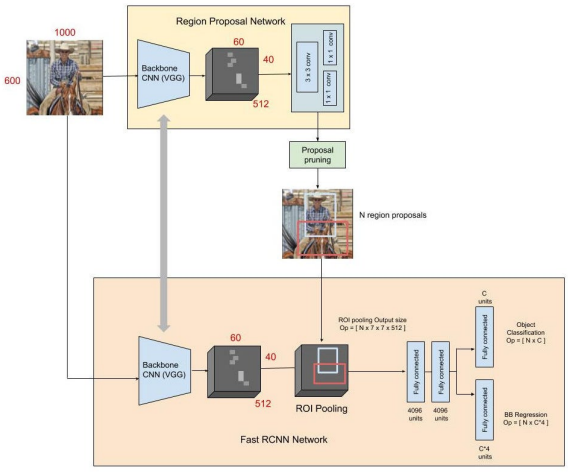

Figure 4: Faster Region-based Convolutional Neural Network

```
RegionProposalNetwork(
  (backbone): VGG16(weights='IMAGENET1K_V1')
  (conv): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (dropout): Dropout(p=0.3, inplace=False)
  (conf_head): Conv2d(512, 18, kernel_size=(1, 1), stride=(1, 1))
  (reg_head): Conv2d(512, 36, kernel_size=(1, 1), stride=(1, 1))
)
```

Figure 5: Region Proposal Network Model created

region proposals, also known as Regions of Interest (ROIs), which are generated from the feature maps extracted by the Convolutional Neural Network (CNN), as opposed to being computed from the original input image. This results in substantial computational efficiency gains, as opposed to the previous approach in R-CNN, which necessitated feature computation for every individual proposal.

Another pioneering contribution of Fast R-CNN is the concept of Region of Interest Pooling (RoI Pooling). RoI Pooling ensures that feature vectors of uniform length are extracted from all ROIs within the same image. This uniformity simplifies subsequent processing and allows for seamless integration of ROIs into the CNN pipeline.

2.2 Further work

As Faster R-CNN continues to be a widely used object detection framework, researchers continue to introduce new methods to improve its speed and accuracy capabilities. One such approach is the use of a single-stage detector network, which removes the need for a separate RPN and RCNN, simplifying the architecture and reducing computational requirements. Examples of these single-stage detector networks include YOLO [7] , and Single Shot Detectors (SSD). Another potential path for future improvements is the integration of attention mechanisms into Faster R-CNN. Attention mechanisms have shown promising results in other computer vision tasks, such as image captioning and visual question answering. By integrating attention mechanisms into Faster R-CNN, it may be possible to improve the model's ability to focus on relevant features and discard irrelevant information. Furthermore, researchers have also explored incorporating localization information in the training process of Faster R-CNN to improve the model's ability to handle occlusion and varying object sizes. For example, Zhang et al.[10]. proposed an offset-guided focal loss technique to handle occlusion and scale variance in object detection. Overall, the continuous efforts of researchers to improve the speed and accuracy of Faster R-CNN show its potential to remain a prominent object detection framework in the future.

3. METHOD

3.1 Model

The Faster R-CNN model takes the approach of passing the image through a backbone network to obtain an output feature map, onto which the ground truth bounding boxes are projected. This feature map is created by a dense convolutional network such as ResNet or VGG16 [9], and represents the learned features of the image. In our case we used VGG16 which was pretrained on the ImageNet dataset. Each point on this feature map is treated as an anchor, and 9 boxes of varying sizes and shapes are generated for each anchor to capture objects in the image. Next, a 1x1 convolutional network is used to predict the category and

```
FastRCNN(
  (flatten): Flatten(start_dim=1, end_dim=-1)
  (backbone): Sequential(
    (0): Linear(in_features=25088, out_features=4096, bias=True)
    (1): ReLU(inplace=True)
    (2): Dropout(p=0.5, inplace=False)
    (3): Linear(in_features=4096, out_features=4096, bias=True)
    (4): ReLU(inplace=True)
    (5): Dropout(p=0.5, inplace=False)
  )
  (classify): Linear(in_features=4096, out_features=7, bias=True)
  (reg): Linear(in_features=4096, out_features=28, bias=True)
)
```

Figure 6: Fast RCNN Model created

offsets of all the anchor boxes. During training, positive or activated anchor boxes are selected based on their overlap with the projected ground truth boxes, while negative boxes with little to no overlap are also sampled. These anchor boxes are then labeled as object or background, and the network is trained using binary cross-entropy loss. To address any misalignments between the positive anchor boxes and the ground truth boxes, a 1x1 convolutional network is also trained to predict offsets to bring the anchor boxes closer to the ground truth. This network uses L1 regression loss to learn the offsets, and the transformed anchor boxes are referred to as region proposals. The entire process described above is called the region proposal network, and is the first stage of the detector. For the second stage, the input is the region proposals generated from the first stage. A simple convolutional network is used to learn and predict the category of the object in each region proposal, using ROI pooling to resize the proposals before passing them through. A separate network is also used to predict offsets for further alignment with the ground truth boxes, and both the loss from the first and second stages are combined to compute the final loss. This is referred to as multi-task learning. During inference, the image is passed through the backbone network and anchor boxes are generated. However, only some boxes with high classification scores from the first stage are selected for the second stage. Non-max suppression is also used as a post-processing step to remove duplicate bounding boxes. In summary, the Faster R-CNN model uses a two-stage approach to first generate anchor boxes and then predict the final categories and offsets for object detection. This is achieved through training a region proposal network (Figure 5) and a multi-task learning network (Figure 6), and during inference, only the top-scoring boxes are selected for the final prediction. [8].

3.2 Dataset

We have used the PASCAL Visual Object Classes (VOC) dataset, which consists of over 20,000 images across 20 categories, including person, car, and chair. For the purpose of making it easier and faster to train we have limited the number of classes to train to 6. The following are the classes that have chosen: aeroplane, bicycle, boat, bus, train, motorbike.

Bounding boxes are used to specify the location of an object in an image by defining a rectangular box around it. These boxes are annotated by specifying the coordinates of the corners of the box. Additionally, each bounding box is also assigned a class label, which identifies the type of object contained within it. The annotations in the PASCAL VOC Dataset is provided as xml files.

4. RESULTS

Our research involved building and training models, as depicted in Figure 6 and Figure 5, utilizing the PASCAL VOC dataset. We managed to train these models for a limited period, specifically for six epochs, due to significant time constraints, as the training process was quite time-consuming on the cloud-based machine we used. As a result, our ability to draw meaningful comparisons with state-of-the-art models, such as Faster R-CNN or other object detection models, was somewhat constrained. This was due to the fact that most of the available time and compute resources were spent on training. Therefore, our evaluation method primarily involved monitoring the training loss and testing the final trained model with images and its corresponding output. The following are our observations.

```
Starting epoch 0
Average loss for 500 training images = tensor(0.6206, device='cuda:0')
Average loss for 500 training images = tensor(0.4388, device='cuda:0')
Starting epoch 1
Average loss for 500 training images = tensor(0.4209, device='cuda:0')
Average loss for 500 training images = tensor(0.3502, device='cuda:0')
Average loss for 500 training images = tensor(0.3249, device='cuda:0')
Starting epoch 2
Average loss for 500 training images = tensor(0.3149, device='cuda:0')
Average loss for 500 training images = tensor(0.2808, device='cuda:0')
Starting epoch 3
Average loss for 500 training images = tensor(0.2819, device='cuda:0')
Average loss for 500 training images = tensor(0.2532, device='cuda:0')
Average loss for 500 training images = tensor(0.2462, device='cuda:0')
Starting epoch 4
Average loss for 500 training images = tensor(0.2277, device='cuda:0')
Average loss for 500 training images = tensor(0.2073, device='cuda:0')
Starting epoch 5
Average loss for 500 training images = tensor(0.2216, device='cuda:0')
Average loss for 500 training images = tensor(0.1936, device='cuda:0')
Average loss for 500 training images = tensor(0.1896, device='cuda:0')
```

Figure 7: Average loss during training

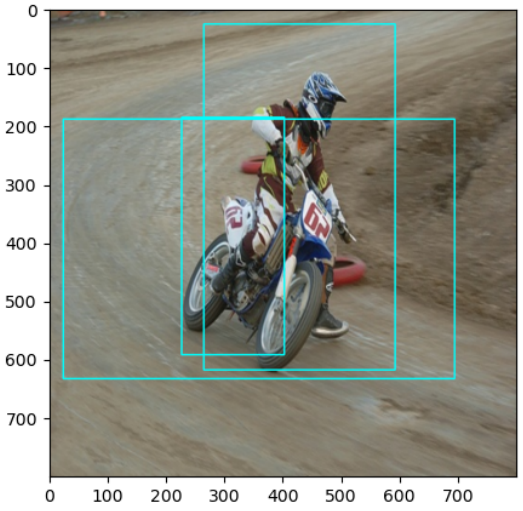

Figure 8: Motorbike test

Figure 7 shows the average loss experienced as the model was trained for 6 epochs. We can see the loss declined as the model was trained. Our model consistently avoided misclassifying objects in the tested scenarios. However, we encountered a recurring issue where the model generated a higher number of bounding boxes than expected. This often resulted in multiple bounding boxes for a single object, as shown in Figure8 and Figure 10. On the flip side, there were instances where our model performed well, producing accurate results as illustrated in Figure 9.

5. CONCLUSIONS

In conclusion, our research was constrained by limited time and training resources, which prevented us from conducting a comprehensive assessment of our model. Despite these limitations, our model demonstrated promising object recognition capabilities, showcasing its potential for further

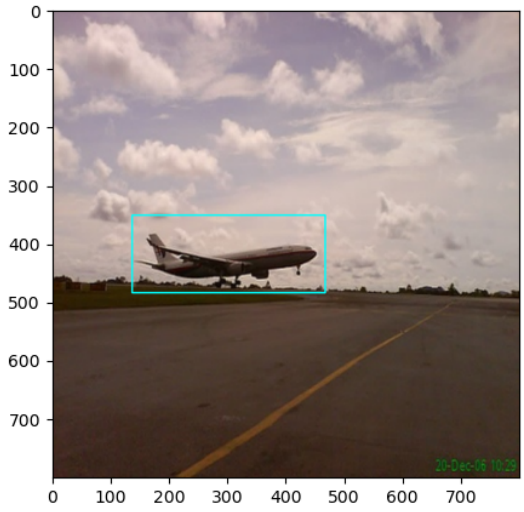

Figure 9: Aeroplane test

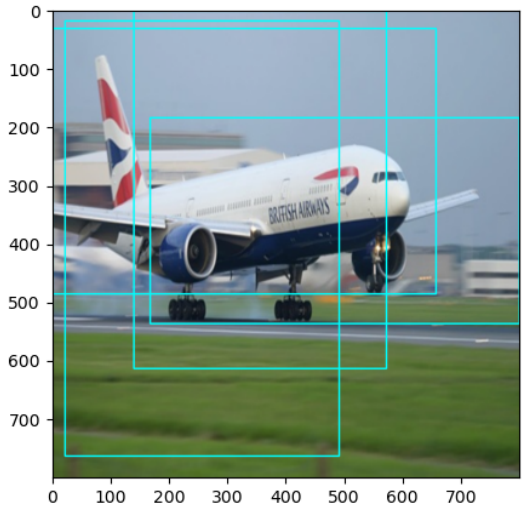

Figure 10: Aeroplane test 2

development and refinement. Notably, the issue of excess bounding boxes warrants investigation and improvement in future iterations.

The inability to incorporate batching during training, with each image being trained individually, proved to be a major bottleneck that hindered the training process. This highlights the need for addressing batch processing in future research, as it has the potential to greatly improve the model's performance and scalability.

Despite being faced with various challenges and limitations, this research project provided us with a profound comprehension of object detection using Faster R CNN. Building a model from the ground up was a demanding and intellectually stimulating experience. While the results may not be entirely satisfactory, we view this undertaking as a meaningful and worthwhile pursuit that provided us with insights and knowledge that could not have been acquired without actively developing an object detection model.

6. REFERENCES

[1] M. Everingham, L. V. Gool, C. K. I. Williams, J. M. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *Int. J. Comput. Vis.*, 88(2):303–338, 2010.

[2] M. V. Fractal AI@Scale. Guide to build faster rcnn in pytorch, May 2022.

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

[4] X. Jiang and et al. A multi-scale approach to detecting standing dead trees in uav rgb images based on improved faster r-cnn. *PLoS One*, 18(2):e0281084, 2023.

[5] J. Jose and T. Sasipraba. An optimal model for municipal solid waste management using hybrid dual faster r-cnn. *Environ Monit Assess*, 195(4):462, 2023.

[6] S. Podder, S. Bhattacharjee, and A. Roy. An efficient method of detection of covid-19 using mask r-cnn on chest x-ray images. *AIMS Biophysics*, 8(3):281–290, 2021.

[7] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(1):1561–1569, 2016.

[8] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in Neural Information Processing Systems*, 28:91–99, 2015.

[9] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[10] S. Zhang, L. Wen, X. Bian, Z. Lei, G. Huang, and T. Liu. Single-shot refinement neural network for object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(1):920–931, 2018.

[11] J. Zheng and T. Zhang. Wafer surface defect detection based on background subtraction and faster r-cnn. *Micromachines (Basel)*, 14(5), 2023.