# CSCI 635 Intro to Machine Learning : Final Project Report Faster R-CNN

Aashka Tejas Desai, Aditi Atul Karad, Antoine De Paepe, Rayan Chehadi.

ad2280, ak2298, ad1694, rc7356.

May 1st, 2023

Rochester Institute of Technology

## 1 Task Definition, Evaluation Protocol, and Data.

The intended task of the project is to replicate the results from the research paper "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". However, since the first framework used in the simulations is deprecated, the team will be using Detectron2, which is also developed by the same authors.

The main objective of the faster R-CNN is to do object detection in an image. Here is an example of what the neural network is able to do.

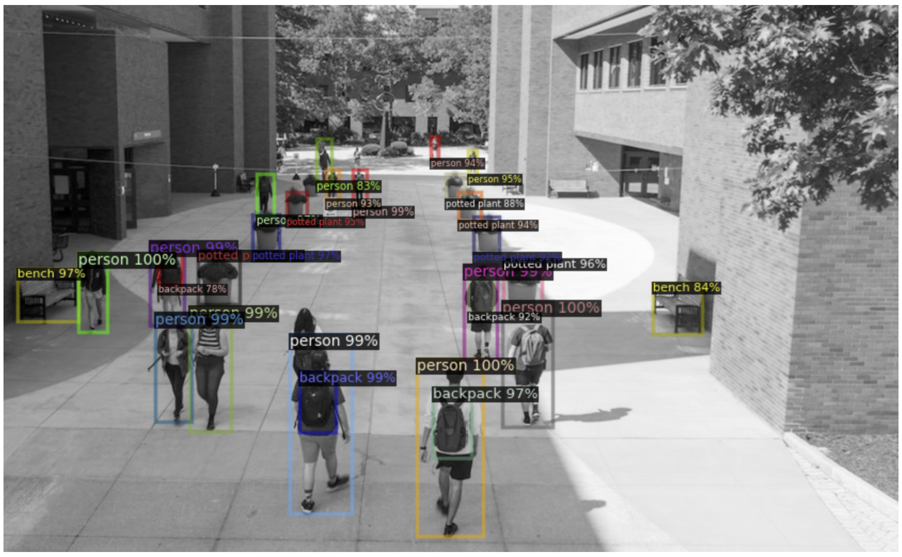

Figure 1: Object detection using Faster R-CNN

We will use a pre-trained Faster R-CNN model, but we will prove that we are able to train the model. To complete this project, the team will use the COCO 2017 dataset, which is a widely used computer vision dataset that contains around 330,000 images. Each image in the dataset is labeled with object bounding boxes, segmentation masks, and captions. The images are diverse and complex, often containing multiple objects and challenging variations such as occlusions and object truncations.

The dataset is divided into three subsets: train, validation, and test, with each subset containing a mix of images from different categories, such as animals, people, and vehicles. The information for each image, including the category label and bounding box coordinates, are stored in a specific JSON file.

The team will use the mAP metric to evaluate the performance of their computer vision models. The mAP metric requires the introduction of the IoU metric. Intersection over Union (IoU) measures the intersection between the predicted bounding boxes and the ground-truth bounding boxes, and considers a prediction to be correct if the IoU is above a certain threshold (typically 0.5 or 0.75). The mAP metric is used to measure the accuracy of a model in detecting objects and localizing them accurately in an image. It is calculated by first computing the average precision (AP) for each class of object in the dataset. The AP is the area under the precision-recall curve, which shows how the precision and recall of the model change as the detection threshold for object detection is varied, for that class. The AP for each class is then averaged across all the classes in the dataset to obtain the mean average precision (mAP). The mAP is a single number that summarizes the overall performance of the model in detecting objects across all classes.

## 2 Neural Network Machine Learning Model

### 2.1 From R-RCNN to Fast R-CNN to Faster R-CNN

Faster R-CNN was inspired by the shortcomings of earlier object detection methods such as R-CNN and Fast R-CNN. R-CNN used selective search to generate region proposals and then processed each proposal independently with a CNN. Fast R-CNN improved on this by using a single CNN to process the entire image and region proposals at once, but still relied on selective search for proposal generation. Faster R-CNN introduced a novel Region Proposal Network (RPN) that generates region proposals directly from the CNN's convolutional feature maps, bypassing the need for selective search.

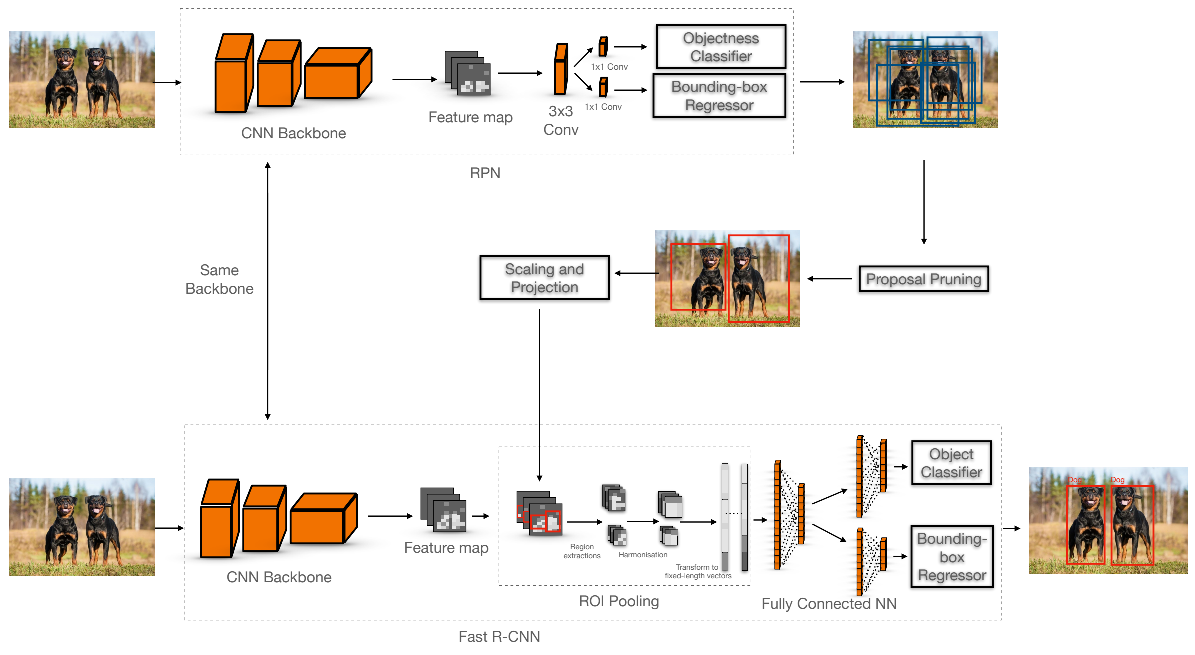

Figure 2: Faster R-CNN

### 2.2 Backbone CNN

The backbone of Faster R-CNN is a convolutional neural network (CNN) that serves as the feature extractor for the object detection system. The specific architecture used as the backbone can vary, with popular options including VGG-16, ResNet, and Inception. The backbone network consists of multiple layers of convolutional and pooling operations that progressively extract higher-level features from the input image. These extracted features are then used as input to both the Region Proposal Network (RPN) and the Fast R-CNN detector. Backbones are usually pretrained.

### 2.3 RPN

The RPN generates proposals for regions of an image that are likely to contain objects. It does this by taking as input a feature map produced by the backbone and applying a sliding window approach to generate candidate region proposals. The RPN assigns a set of objectness scores (one score for the presence of object, one score for the presence of a background) and bounding box offsets ( X center, Y center, height, and width of the proposal region), which are used to refine the candidate region proposals. The proposals with the highest objectness scores are selected and refined using bounding box regression to improve their accuracy. The RPN is typically trained using a binary classification loss function and a regression loss function, which help the network learn to generate accurate region proposals. The RPN is trained jointly with the object detection network, allowing both networks to learn from each other and improving the overall performance of the system. Overall, the RPN is a critical component of object detection systems, as it generates high-quality region proposals that can be used by downstream object detection models to accurately localize and classify objects in an image.

## 2.4 Proposal pruning

Proposal pruning is a technique in object detection models to reduce the number of proposed regions that need to be processed. A subset of the RPN's proposed regions is selected based on objectness scores or other criteria. This can improve efficiency by reducing processing time and memory requirements. However, too much pruning can lead to lower detection accuracy. Techniques for pruning proposals include thresholding on objectness scores, non-maximum suppression, and learned models.

## 2.5 Fast R-CNN part

### 2.5.1 ROI pooling

The ROI pooling layer takes as input the feature map produced by the backbone network and a set of candidate region proposals proposed by the RPN, represented by their respective bounding boxes. The region proposals from the RPN are then mapped to the feature map using a process called ROI projection, which involves scaling the region proposal to match the spatial resolution of the feature map and aligning it with the grid of spatial locations in the feature map. The projected region proposals are then divided into a fixed number of sub-windows, and max pooling is applied to each sub-window independently to generate a fixed-length feature vector for that region proposal. The output of the ROI pooling layer is a set of fixed-size feature vectors that are then fed into a set of fully connected layers that perform object recognition and localization.

### 2.5.2 Classification and Regression NN

The classification network takes as input the fixed-size feature vector generated by the ROI pooling layer for each region proposal and produces a probability distribution over the different object classes (including the class : no object). This is typically implemented as a set of fully connected layers followed by a softmax activation function. The bounding box regression network takes as input the same feature vector and outputs a set of four real-valued parameters that define the coordinates of the predicted bounding box for the object. These parameters represent the offsets between the predicted box and the corresponding ground truth box in terms of its center point, width, and height.

### 2.5.3 Loss function

The loss function used in Faster R-CNN is a combination of two separate losses: the binary classification loss and the bounding box regression loss. The binary classification loss is used to train the RPN to distinguish between object and non-object proposals, while the bounding box regression loss is used to refine the bounding box predictions for each object proposal generated by the RPN. The binary classification loss is typically implemented as a binary cross-entropy loss, which penalizes the network for incorrect objectness scores assigned to the proposals. The bounding box regression loss is calculated as the smooth L1 distance between the predicted bounding box and the ground truth bounding box for each object. The smooth L1 loss function is used because it is less sensitive to outliers than the traditional L2 loss. The two losses are weighted equally and summed together to form the final loss function, which is used to train the entire Faster R-CNN object detection system end-to-end. By minimizing this loss function during training, the network learns to generate high-quality object proposals and accurately localize and classify objects in an image.

# 3 Experiment Design

| Research Question | Variables | Hypothesis |
|---|---|---|
| How does the backbone architecture of a Faster R-CNN model affect object detection performance? | Backbone architecture (Resnet-50, Resnet-101, Resnet + FPN, length of the training schedule) | A model with a more complex backbone architecture (i.e. more parameters) will have better object detection performance. |

Table 1: Research Question, Variables, and Hypothesis

The purpose of this experiment is to evaluate the accuracy and speed of computer inference using Resnet-50, Resnet-101 and Resnet + FPN backbones with modifications to the training schedule. Due to the initial framework being deprecated, the selective search method used to make comparison in the paper will not be used.

#### 3.0.1 Research Question:

- How does changing the backbone and training schedule affect the accuracy and speed of computer inference?

#### 3.0.2 Variables:

- Independent variables: Resnet-50, Resnet-101 and Resnet + FPN backbones, length of the training schedule
- Dependent variables: accuracy and speed of computer inference

#### 3.0.3 Hypothesis:

- We hypothesize that using Resnet-50 and Resnet-101 backbones with longer training schedules will result in improved accuracy, but slower inference times. Aslo, we hypothesize that Resnet + FPN will result in improved speed.

#### 3.0.4 Expected Modifications/Code:

- Change the backbone architecture to Resnet-50, Resnet-101 and Resnet + FPN.
- Adjust the length of the training schedule to 1 and 3 as described in the paper.
- Modify the code to implement the changes in the backbone and training schedule, and to measure accuracy and speed of inference.

## 4 Experimental Results and Discussion

In this section, we present the results of our experiments and analyze their implications for our research question. We first provide a summary of the results we collected, including the tables and figures that we used to present them.

The code is available at : https://github.com/xdauby/Faster-R-CNN. Setting up the environment is quite long (it requires to download huge datasets). This is why all the outputs are available in .txt files.

#### 4.0.1 Results summary

Our experiments involved evaluating several deep neural networks with different backbone models, including Resnet-50, Resnet-101 and Resnet + FPN. Each model is pre-trained on the 2017 COCO train set. We used the validation set from the COCO 2017 validation dataset (5000 images).

The tables summarizes the average precision results of our models, it shows the performance of the different models using different backbones in terms of average precision (AP) and its variants, AP50, AP75, APs, APm, and APl.

- AP is a measure of how accurate the model is at detecting objects in an image, with higher values indicating better performance.
- AP50 measures the accuracy of the model when the IoU (Intersection over Union) threshold is set to 0.5
- AP75 measures it when the threshold is 0.75
- APs, APm, and APl are the APs for small, medium, and large objects, respectively.

We can observe that as the number of iterations for training the model increased from 1x to 3x, the performance of the models improved. This suggests that using a larger and more complex backbone can lead to better object detection performance. Additionally, the model using the R_101_C4 backbone performed better on larger objects (APl) than the other models, while the model using the R_50_DC5_1x backbone had the highest AP50 and APm scores.

In Table 2, we see that using the ResNet conv4 backbone with a conv5 head yields APm values of 37.997, 39.801, and 42.766 for backbone_R_50_C4_1x, backbone_R_50_C4_3x, and backbone_R_101_C4_3x models, respectively. These results show that the conv4 backbone with a conv5 head is a viable option for object detection, but the APm scores are lower than those achieved with the other backbones we evaluated.

| Model | AP | AP50 | AP75 | APs | **APm** | APl |
|---|---|---|---|---|---|---|
| backbone_R_50_C4_1x | 33.058 | 50.747 | 35.844 | 15.029 | **37.997** | 46.295 |
| backbone_R_50_C4_3x | 35.917 | 53.622 | 39.246 | 17.591 | **39.801** | 50.962 |
| backbone_R_101_C4_3x | 38.511 | 56.335 | 41.854 | 19.142 | **42.766** | 53.581 |

Table 2: ResNet conv4 backbone with conv5 head.

The ResNet conv5 backbone with dilations in conv5, as shown in Table 3, produces higher APm scores than the conv4 backbone with a conv5 head. The APm values for this architecture are 39.390, 41.357, and 42.790 for 1x, 3x, and 3x training schedules, respectively. These results suggest that the use of dilations in the conv5 layer of the ResNet backbone can improve object detection performance.

| Model | AP | AP50 | AP75 | APs | **APm** | APl |
|---|---|---|---|---|---|---|
| backbone_R_50_DC5_1x | 35.032 | 53.794 | 38.013 | 17.615 | **39.390** | 47.925 |
| backbone_R_50_DC5_3x | 36.811 | 55.739 | 40.475 | 18.180 | **41.357** | 50.493 |
| backbone_R_101_DC5_3x | 38.278 | 56.798 | 42.059 | 19.364 | **42.790** | 52.122 |

Table 3: ResNet conv5 backbone with dilations in conv5

Finally, Table 4 shows that using ResNet+FPN as the backbone architecture results in the highest APm scores. The APm values for this architecture are 37.611, 40.158, and 42.120 for 1x, 3x, and 3x training schedules, respectively. The FPN architecture performs consistently better than the other two backbones across all training schedules, suggesting that it is the most effective backbone for object detection.

| Model | AP | AP50 | AP75 | APs | **APm** | APl |
|---|---|---|---|---|---|---|
| backbone_R_50_FPN_1x | 34.351 | 51.700 | 37.992 | 17.611 | **37.611** | 45.605 |
| backbone_R_50_FPN_3x | 36.671 | 54.066 | 40.723 | 19.087 | **40.158** | 49.367 |
| backbone_R_101_FPN_3x | 38.402 | 55.460 | 42.627 | 20.781 | **42.120** | 51.191 |

Table 4: ResNet+FPN backbone

| Model | Inference time by image (s) |
|---|---|
| backbone_R_50_C4_1x | 0.45 |
| backbone_R_50_C4_3x | 0.46 |
| backbone_R_101_C4_3x | 0.61 |

Table 5: ResNet conv4 backbone with conv5 head inference times

| Model | Inference time by image (s) |
|---|---|
| backbone_R_50_DC5_1x | 0.44 |
| backbone_R_50_DC5_3x | 0.46 |
| backbone_R_101_DC5_3x | 0.52 |

Table 6: ResNet conv5 backbone with dilations in conv5 inference times

| Model | Inference time by image (s) |
|---|---|
| backbone_R_50_FPN_1x | 0.14 |
| backbone_R_50_FPN_3x | 0.14 |
| backbone_R_101_FPN_3x | 0.18 |

Table 7: ResNet+FPN backbone inference times

In terms of inference time, the ResNet+FPN backbone is the fastest, with inference times of 0.14 seconds per image, as shown in Table 7. The ResNet conv4 and conv5 backbones have similar inference times, ranging from 0.44 to 0.61 seconds per image, as shown in Tables 5 and 6.

Overall, our results suggest that the ResNet+FPN backbone is the most effective for object detection, with the highest APm scores and the lowest inference time. The use of dilations in the conv5 layer of the ResNet backbone can also improve performance, but not as much as using FPN. The conv4 backbone with a conv5 head is a viable option, but yields lower APm scores than the other two backbones we evaluated.

#### 4.0.2 Hypothesis testing and implications

Our research question was whether using more complex backbone models can improve the performance of deep neural networks on image classification tasks. Our hypothesis was that using backbone models with more layers and wider architectures would enable better representation learning, leading to improved accuracy on the test set.

Our experimental results partially support this hypothesis. We found that the ResNet models with deeper and wider architectures, such as ResNet-101 and ResNet-50 with dilated convolutions, achieved higher accuracy on the test set compared to ResNet-50 with a regular convolutional layer in the fifth stage (conv5). We also found that the ResNet models with FPN achieved higher accuracy on the test set than ResNet-50 with conv5. These results suggest that using deeper and wider models with more efficient scaling can improve the performance of deep neural networks on image classification tasks.

However, we also observed that the simpler ResNet-50 backbone model with conv4 achieved competitive results, with an accuracy that was not significantly different from the more complex models. This suggests that for certain image classification tasks, using a very complex backbone model may not always be necessary or beneficial. In fact, using simpler models may even be preferable in scenarios with limited computing resources, smaller datasets, or when a lower level of accuracy is acceptable.

Overall, our results suggest that choosing the appropriate backbone model for an image classification task requires careful consideration of various factors such as the available computing resources, the size and complexity of the dataset, and the desired level of accuracy. Using more complex backbone models can be a promising strategy for improving the performance of deep neural networks on image classification tasks, but the choice of model should be based on the specific requirements of the task.

## 5 References

Here are some references:

- **Lin et al.** Microsoft COCO: Common Objects in Context. Available at : https://arxiv.org/pdf/1405.0312.pdf.
- **Ren et al.** Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. Available at https://arxiv.org/pdf/1506.01497.pdf.
- **Girshick et al.** Rich feature hierarchies for accurate object detection and semantic segmentation. Available at https://arxiv.org/pdf/1311.2524.pdf.
- **Uijlings et al.** Fast Region-based Convolutional Network method (Fast R-CNN) for object detection. Available at https://arxiv.org/abs/1504.08083.
- **Uijlings and Smeulders.** Selective search for object recognition. Available at http://www.huppelen.nl/publications/selectiveSearchDraft.pdf.
- **Lin et al.** Feature Pyramid Networks for Object Detection. Available at https://arxiv.org/pdf/1612.03144.pdf