Object Detection with Faster R-CNN

Nemanja Ilic
nemanja.ilic@studenti.unipd.it

Lucrezia Rossi
lucrezia.rossi.2@studenti.unipd.it

Abstract

Object recognition is a fundamental task in computer vision, with applications in autonomous systems, surveillance, and medical imaging. This study explores the performance of Faster R-CNN, a widely used deep learning model for object detection, on the COCO 2017 dataset. We investigate the impact of modifying the model architecture by adjusting dropout rates, adding dense layers and changing the optimization strategy to improve feature extraction and generalization. The evaluation is conducted using standard COCO metrics, such as mean Average Precision (mAP) and Intersection over Union (IoU). Preliminary results indicate that architectural modifications influence detection performance. This research provides insights into optimizing Faster R-CNN for real-world object recognition tasks.

1. Introduction

Object recognition is a critical task in computer vision, enabling machines to detect and classify objects within images. It has widespread applications in autonomous driving, medical imaging, surveillance, and robotics. Among deep learning-based approaches, Faster R-CNN (Region-Based Convolutional Neural Network) 6 is one of the most effective models for object detection, leveraging a Region Proposal Network (RPN) to efficiently identify object locations while maintaining high accuracy.

Despite its strong performance, optimizing Faster R-CNN for better generalization and robustness remains an open challenge. The COCO 2017 dataset 6 serves as a benchmark for object detection tasks, containing a diverse set of images with annotated objects across multiple categories. However, object detectors often struggle with overfitting, viewpoint variations, and suboptimal feature representations, leading to performance degradation in real-world scenarios. Several factors, such as dropout regularization, additional dense layers, and different optimization techniques, can influence model performance.

This study investigates the impact of architectural modifications and data augmentation strategies on Faster R-CNN's performance. Specifically, we investigate:

- **Dropout modifications** – Adjusting dropout rates to reduce overfitting and enhance model generalization6.
- **Additional dense layers** – Introducing fully connected layers to refine feature representations before final predictions.
- **Optimization strategy** – Evaluating alternative training strategies to enhance convergence and accuracy.

By systematically evaluating these modifications on the COCO 2017 dataset, we aim to identify strategies that enhance object detection performance. The results are assessed using the standard COCO evaluation metric, mean Average Precision (mAP), and Intersection over Union (IoU), to determine the effectiveness of these approaches.

2. Related Work

2.1. Faster R-CNN with ResNet-50 Backbone

Faster R-CNN is a state-of-the-art object detection model that follows a two-stage architecture, combining a Region Proposal Network (RPN) with a classification and bounding box regression network. The model's backbone network plays a crucial role in extracting features, which are then used both for generating region proposals and for classifying and refining object predictions. In this study, we adopt the Faster R-CNN model with a ResNet-50 6 backbone, a variant of the Residual Network (ResNet) designed to improve the depth and accuracy of the model by employing residual connections.

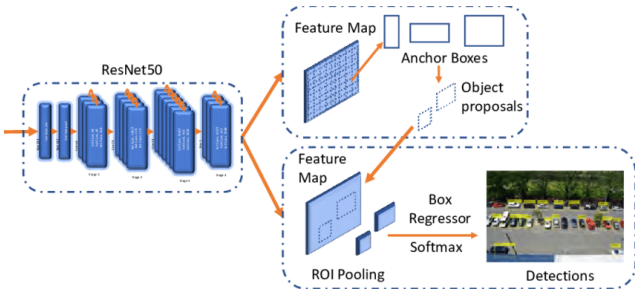

Figure 1. Architecture of Faster-RCNN containing a ResNet50 backbone[6]

The Faster R-CNN pipeline consists of the following key components:

1. **Backbone Network (ResNet-50)**

   The role of ResNet-50 is to extract high-level features from the input image that are essential for both the Region Proposal Network (RPN) and the object classification module.

   It is a deep convolutional network with 50 layers and utilizes residual connections to mitigate the vanishing gradient problem commonly encountered in deeper networks. Residual connections skip one or more layers, allowing the network to learn more complex features without the degradation problem. These features are used to generate proposals and classify objects in the image.

2. **Region Proposal Network (RPN)**

   The RPN is responsible for generating potential object proposals from the feature maps produced by the ResNet-50 backbone. It is an integral part of Faster R-CNN and differentiates the model from previous methods like R-CNN and Fast R-CNN.

   The RPN is a fully convolutional network that slides over the feature map produced by the backbone. At each location, it generates a set of proposals (potential object bounding boxes) by predicting:

   - **Objectness score**: A binary classification (foreground or background).
   - **Bounding box regressions**: Adjustments to refine the coordinates of the proposed boxes.

   These proposals are then filtered using non-maximum suppression (NMS) to remove redundant or low-confidence proposals, and the best ones are passed to the next stage of the model.

3. **Region of Interest (RoI) Pooling**

   After the RPN generates object proposals, RoI Pooling is used to convert the varying-sized proposals into fixed-size feature maps. RoI Pooling takes each proposal and extracts a fixed-size feature map from the feature maps produced by ResNet-50, regardless of the proposal's original size.

   This operation involves dividing each proposal into a grid of cells and applying max pooling to each cell to down-sample the feature map. This process allows the model to work with fixed-size feature maps for classification and regression, which is required by fully connected layers.

4. **Object Classification and Bounding Box Regression**

   After RoI pooling, the resulting fixed-size feature maps are passed to two separate fully connected layers:

   - **Object Classification**: This part of the network classifies each proposal into one of the object categories. It is typically followed by a softmax activation function to output class probabilities.
   - **Bounding Box Regression**: This module refines the coordinates of the bounding boxes by predicting offsets for each proposal. These predicted offsets are applied to adjust the initial proposal locations, making them more accurate for detecting objects in the image.

   Both the classification and bounding box regression heads are jointly trained using a multi-task loss function, which combines classification loss (cross-entropy) and regression loss (smooth L1 loss).

5. **Final Detection**

   The output of the model includes class labels and refined bounding boxes that tightly enclose each detected object. These detections are then evaluated using a NMS, to eliminate redundant detections that overlap heavily. The remaining detections are considered the final output.

3. Dataset

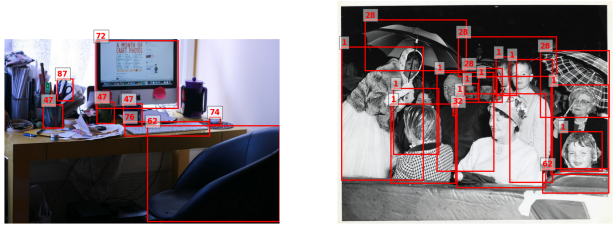

Figure 2. Images from COCO17

In this study, we utilize the COCO 2017 dataset, one of the most widely used benchmarks for object detection. The COCO dataset provides high-quality annotations across a diverse set of images, making it an ideal choice for evaluating deep learning-based object detection models such as Faster R-CNN.

The dataset consists of a total of 123,287 labeled images (118,287 training + 5,000 validation images) and 40,670 unlabeled images for the test set. However, since the official test set lacks publicly available labels and this amount of images is too heavy for us to compute, we constructed our own dataset by selecting 5% of the total labeled images

and then creating a test set from the training images with a 4:1 ratio. This approach ensures that we have labeled data for both training and evaluation while significantly reducing computational costs. The resulting sets are: 4731 train, 250 validation, and 1183 test.

The images are collected from real-world scenes, featuring objects in complex environments with varying lighting conditions, occlusions, and background clutter. Each image in the dataset contains multiple objects from 80 different categories.

For our study, we extracted from the dataset just the bounding boxes (in the $x_{\text{min}}, y_{\text{min}}, width, height$ format) and category labels to serve as ground truth for the model, ignoring the instance segmentation data.

COCO 2017 presents several challenges that make it a strong benchmark for object detection:

1. **Diverse Object Scales**: Objects appear in various sizes, requiring the model to be robust across scale variations.

2. **Occlusions and Crowded Scenes**: Many images contain overlapping objects, making detection more difficult.

3. **Class Imbalance**: Some categories appear more frequently than others, necessitating techniques to handle class imbalance.

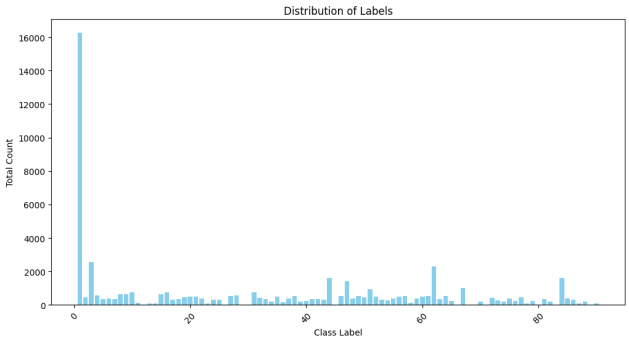

Figure 3. Label distribution on custom dataset (1 = person)

4. Method

4.1. Data Preprocessing

Before feeding the images into the model, we apply several preprocessing transformations to ensure consistency and improve performance.

Since the images have various dimensions, all of them are resized to a fixed dimension (128 x 128 and 256 x 256) while preserving aspect ratio, with padding applied when necessary. This ensures uniform input size across the dataset and faster computation.

Pixel values are normalized using the mean and standard deviation of the ImageNet dataset, as the backbone is pre-trained on it, following standard practice for pre-trained models. In addition the images are transformed in RGB format making their dimensions [3, 128, 128].

The COCO dataset provides bounding boxes in the format (x, y, w, h) so we convert them to the format $(x_{\text{min}}, y_{\text{min}}, x_{\text{max}}, y_{\text{max}})$ required by Faster R-CNN using:

$$x_{\text{max}} = x + w, \quad y_{\text{max}} = y + h$$

This format is essential for defining the regions of interest used by the model.

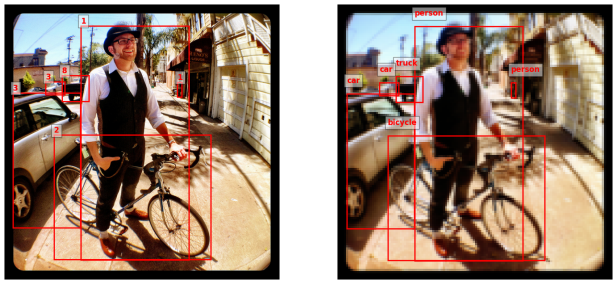

Figure 4. Preprocessed image example

4.2. Model Modifications

To analyze the impact of different architectural choices, we modify the standard Faster R-CNN model.

The classification and regression heads of Faster R-CNN are modified by adding additional dense layers to explore their effect on feature extraction and decision boundaries. Both the default classifier and the regression in Faster R-CNN have only one dense layer, so we tried adding two extra to the classifier to allow the network to learn richer feature representations and one extra layer to the regression that now transforms the features once before predicting the bounding boxes.

We introduce dropout layers at the classifier after each fully connected layer, it's not used for the regression tasks (bounding box prediction) as there we want precise outputs, and dropping neurons randomly might add too much noise to the predicted bounding boxes. Dropout is used in our model to prevent overfitting, improve generalization and reduce co-adaptation. The dropout rates experimented with are 0.1 and 0.5.

We also experiment with freezing early layers of ResNet-50 to retain low-level feature representations while fine-tuning deeper layers.

Layer	Input Features	Output Features
Classification Head		
FC1	1024	1024
ReLU	-	-
Dropout	-	-
FC2	1024	512
ReLU	-	-
Dropout	-	-
FC3	512	91 (classes)
Regression Head		
FC1	1024	1024
ReLU	-	-
FC2	1024	364 (bbox coordinates)

Table 1. Modified predictor architecture

4.3. Loss Function

Faster R-CNN has a two-part loss function, since it performs both object classification and bounding box regression. The total loss is:

$$L = L_{\text{cls}} + L_{\text{reg}}$$

where:

- L_{cls} = Classification Loss (which object is in the box)
- L_{reg} = Bounding Box Regression Loss (how well the predicted box matches the ground truth)

4.3.1 Classification Loss

This loss is used twice in the model:

- **Region Proposal Network (RPN):** Classifies if a region is an object or background.
- **Final Detection Head:** Classifies the object category.

Both use Cross-Entropy Loss to compare predicted class probabilities with ground truth. The classification loss is defined as:

$$L_{\text{cls}} = -\sum y \log(\hat{y})$$

where:

- y = ground truth label (one-hot encoded)
- $\hat{y}$ = predicted class probabilities

4.3.2 Bounding Box Regression Loss

This loss measures how close the predicted bounding box is to the ground truth. Faster R-CNN uses Smooth L1 Loss, which behaves like L1 loss (absolute error) for large errors and L2 loss (squared error) for small errors. The regression loss is defined as:

$$L_{\text{reg}} = \sum_{i \in (x,y,w,h)} \text{SmoothL1}(t_i - t_i^*)$$

where:

- t_i = Predicted bounding box coordinates (center x, y, width w, height h)
- t_i^* = Ground truth bounding box coordinates

The Smooth L1 Loss formula is:

$$\text{SmoothL1}(x) = \begin{cases} 0.5x^2, & \text{if } |x| < 1 \\ |x| - 0.5, & \text{otherwise} \end{cases}$$

This helps stabilize training and prevents exploding gradients.

4.4. Evaluation

To evaluate the performance of the model, we use two commonly used metrics in object detection: mean Average Precision (mAP) and Intersection over Union (IoU).

4.4.1 Intersection over Union

IoU is a metric that evaluates how well the predicted bounding box overlaps with the ground truth bounding box. It is calculated as:

$$IoU = \frac{\text{Area of Intersection}}{\text{Area of Union}}$$

where:

- Area of Intersection is the overlap between the predicted and ground truth bounding boxes.
- Area of Union is the total area covered by both the predicted and ground truth bounding boxes.

IoU is used to filter out low-quality predictions in the evaluation process. We use the common threshold for a prediction to be considered correct of $IoU > 0.5$, meaning that the predicted bounding box must overlap with the ground truth box by at least 50

4.4.2 Mean Average Precision

MAP is the primary evaluation metric for object detection tasks. It is calculated as the average of the Average Precision (AP) for each object class across all classes in the dataset. AP is computed by calculating the precision-recall curve, which evaluates the trade-off between precision and recall at different detection thresholds.

The mAP provides a global measure of how well the model is performing across all object classes. The calculation for AP is as follows:

$$AP = \int_0^1 P(r)\, dr$$

where $P(r)$ is the precision at recall level r. The mAP is then computed as:

$$mAP = \frac{1}{N} \sum_{c=1}^{N} AP_c$$

where N is the number of classes and AP_c is the Average Precision for class c.

4.5. Optimizers

In this study, we experiment with two widely used optimization algorithms: Stochastic Gradient Descent (SGD) and Adaptive Moment Estimation (Adam)6.

SGD updates the model parameters by computing the gradient of the loss function with respect to each parameter and applying the update in the direction that minimizes the loss. Adam maintains two moving averages: one for the gradient and another for the squared gradient, which helps adaptively adjust the learning rate for each parameter.

The parameters we used are:

- SGD: lr = 0.005, momentum = 0.9, weight decay = 0.0005.
- Adam: lr = 0.0005, β_1 = 0.9, β_2 = 0.999.

SGD is generally more stable for large-scale deep learning models and tends to generalize better, especially when combined with momentum. Adam is often faster in terms of convergence, as it adapts the learning rate dynamically for each parameter.

5. Experiments

In this section, we present and analyze the results of our experiments using Faster R-CNN with different configurations. The models were trained for 6 epochs, a decision made after observing that the validation loss curve exhibited a "elbow" point at epoch 6 when trained for 10 epochs. This indicates that the model's improvement becomes less significant after this point, and it might even lead to overfitting. We used a batch size of 16 for the training process, balancing memory constraints and model performance. To compare different model configurations, we used the mAP with threshold 0.5.

The table below summarizes the results for different configurations:

Architecture	**Training procedure**	**mAP%**
128x128 images	Baseline + SGD	25.77
128x128 images	ResNet50 frozen + Adam	7.80
128x128 images	ResNet50 frozen + SGD	24.45
128x128 images + dense layers + dropout(0.1)	ResNet50 frozen + SGD	25.98
128x128 images + dense layers + dropout(0.5)	ResNet50 frozen + SGD	24.94
128x128 images + dropout(0.5)	ResNet50 frozen + SGD	26.45

Table 2. Comparison of Different Model Configurations

5.1. Analysis of Experimental Results

The experimental results reveal key trends in model performance based on different architectural modifications and training strategies.

One important factor to keep in mind for the evaluation is the result of the baseline Faster R-CNN which is pretrained to evaluate images of the COCO 2017 dataset. It will be used as a benchmark to properly evaluate the following models since the global mAP% is not very impressive (the reason for it will be analyzed later).

The choice of optimizer also plays a role in performance. As seen SGD outperforms Adam significantly, due to it's better generalization. The only advantage of the Adam optimazer is it's faster convergence rate in the early epochs.

The following models are mainly focused on freezing the ResNet-50 backbone and training the rest of the model since without the ResNet-50 Backbone the performance drops.

Another critical characteristic is the addition of dense layers and dropout regularization. Introducing additional dense layers enhances the model's capacity to learn complex patterns, however, the dropout rate plays a crucial role in determining the effectiveness of these additional layers, evidenced by the mAP. A moderate dropout rate of 0.1 yields an improvement, indicating that controlled regularization helps mitigate overfitting while retaining important features. In contrast, a higher dropout rate of 0.5 leads to a decline in mAP, which can suggest that excessive dropout can hinder learning by removing too much information from the model.

Interestingly, results with a dropout rate of 0.5 show some inconsistencies, when tested without any additional dense layers. This variation could stem from different training conditions, weight initializations, or other hyperparameter interactions. This highlights the importance of conducting multiple trials to ensure stability and consistency in model evaluation.

5.2. Impact of image resolution

Following the initial experiments with 128x128 image resolution, we conducted additional trials using a higher resolution of 256x256 to analyze its impact on performance. The results are reported below:

Architecture	Training procedure	mAP%
256x256 images + dense layers + dropout(0.1)	ResNet50 frozen + SGD	33.11
256x256 images + dense layers + dropout(0.5)	ResNet50 frozen + SGD	31.94

Table 3. Higher resolution images models

The results showed a significant improvement in mAP, indicating that the model benefited from the increased image detail, leading to better object detection accuracy. Maintaining the previous results regarding the comparison of the dropout rates.

However, this improvement came at the cost of higher computational requirements. The increased resolution led to longer training times and higher memory consumption, making the training process more resource-intensive.

5.3. Future improvements

Despite the careful selection of model architecture and hyperparameters, the overall mAP values remain relatively low. A key limitation of our experiments is the small subset of the dataset used. With only 5% of the total COCO17 dataset utilized for training, the model had significantly fewer examples to learn from, leading to limited generalization capability. Expanding the dataset size would very likely result in substantial performance improvements.

Additionally, the number of training epochs was relatively low. A graph of loss vs. epochs shows that the loss was still somewhat decreasing, indicating that further training could lead to little better convergence and improved accuracy. Increasing the number of epochs would allow the model to refine its learned features and achieve potentially higher mAP scores.

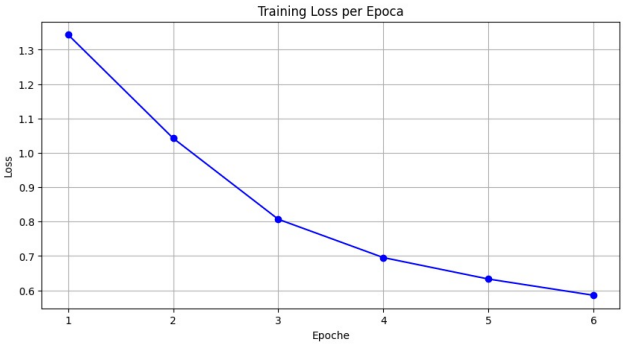

Figure 5. Loss vs. Number of Epochs of 128x128 images + dense layers + dropout(0.1)

Another important factor is image resolution. Our experiments with 128x128 images yielded suboptimal results, while just by increasing the resolution to 256x256 demonstrated noticeable improvements. Further increasing the resolution could enhance the model's ability to detect fine details and improve overall performance, albeit at a greater computational cost.

Additional ideas are the fine-tuning of dropout rates and batch sizes that could also optimize the regularization effect and prevent overfitting. Also the implementation of data augmentation techniques could be used to improve generalization.

One of the major constraints in our experiments was hardware limitations. Training deeper networks with higher resolution images and longer epochs demands significant computational power. Our available hardware resources limited the extent of these experiments.

6. Conclusion

To conclude, this study examined the impact of architectural modifications and optimization strategies on Faster R-CNN's performance for object detection. Our findings show that increasing image resolution significantly improved mean Average Precision (mAP), and moderate dropout rates (0.1) provide better enhanced generalization than excessive rates (0.5). Freezing the ResNet-50 backbone preserved essential features, and while Adam converged faster, Stochastic Gradient Descent (SGD) achieved better final mAP.

The overall mAP remained limited due to a small dataset subset and limited . Future work should explore larger datasets, extended training, advanced data augmentation, and refined hyperparameters to further improve detection accuracy. Leveraging hardware accelerations could also enhance model efficiency for real-world applications.

References

[1] Acharya, Debaditya, and Kourosh Khoshelham. ”11 Parking Occupancy Detection and Slot Delin-eation Using Deep Learning: A Tutorial.” SMART PARKING IN FAST-GROWING CITIES (2021): 143.

[2] Shaoqing Ren, Kaiming He, Ross B. Girshick, Jian Sun, ”Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, In NeurIPS 2015.

[3] Tsung-Yi Lin, Michael Maire, Serge Belongie, et al., ”Microsoft COCO: Common Objects in Context”, In ECCV 2014.

[4] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov, ”Dropout: A Simple Way to Prevent Neural Networks from Overfitting”, Journal of Machine Learning Research 2014.

[5] Karen Simonyan, Andrew Zisserman, ”Very Deep Convolutional Networks for Large-Scale Image Recognition”, In NeurIPS 2014.

[6] D.P. Kingma, J.Ba, ”Adam: A Method for Stochastic Optimization”, In ICLR 2015.