International Institute of Information Technology Hyderabad

Report on Machine Learning

By: (Code Crew)

Title: *Object Detection Using Faster R-CNN on COCO Dataset*

Abstract:

Object detection is a core task in the computer vision domain with wide applications. Localization and classification of objects are involved in it. Moreover, it is one state-of-the art object detection framework that has performed profoundly in localizing and classifying objects in images. In this paper, we outline how Faster R-CNN is implemented and evaluated using the challenging dataset COCO that presents scenes with complexities and diversity ranges in objects. This paper describes the intricacies of the Faster R-CNN architecture, including its region proposal network and subsequent classification and bounding box regression steps. The result of experiments throughout these papers highlights the capability of Faster R-CNN with superior accuracy on the COCO dataset rather than previous approaches. Moreover, the performance of the model is analyzed based on differences in hyperparameters and various training regimes. Besides that, we also discuss possible ways toward more substantial improvements like adding attention mechanisms and other backbone networks. This work marks a significant step in the evolution of object detection methods and provides insightful views toward future progress in the field.

Machine Learning:

In order to identify patterns and generate predictions, machine learning (ML) trains models on data. The label is the intended prediction, while the features are the input variables. Unsupervised learning looks for patterns in unlabelled data, whereas supervised learning uses labelled data. When a model does well on training data but badly on fresh data, it is said to be overfitting; when it fails to recognize patterns, it is said to be underfitting. Regression predicts continuous values, while classification predicts categories. Learning from complex data is made possible by neural networks and deep learning. Accuracy, recall, and precision are important criteria.

Natural Language Processing:

Text is processed by Natural Language Processing (NLP) using methods such as TF-IDF (word significance score), embeddings (word representations), and tokenization (dividing text). Whereas POS tagging looks for grammatical roles, NER detects entities (such as names). Fine-tuning adapts these models for tasks like sentiment analysis. Transformers such as BERT and GPT use attention processes for context. Translation is facilitated by Seq2Seq, and models may function with less data thanks to zero-shot learning. NLP can accurately comprehend and produce human language thanks to the BLEU score, which assesses translation quality.

Introduction:

Object detection is an important task in computer vision: given an image, identify and locate objects within it. The areas of application range from self-driving cars to monitoring security cameras and robotics. Faster R-

CNN is the latest state-of-the-art model from the variants that improve on the previous R-CNN approach by incorporating the region proposal process into the network to increase detection speed and accuracy. The RPN and a Fast R-CNN classifier contribute to a two-stage framework of Faster R-CNN. It allows efficient object detection across scales and contexts. Analysis in this report relies on the demanding benchmark of the COCO dataset, comprising over 200,000 images and 80 object categories that mirror real-world complexities such as occlusion and cluttered backgrounds. Testing in this study aims to identify computational challenges for Faster R-CNN and introduces possible optimizations.

Objective:

The primary objective of this project is to:

1. Adapt the COCO dataset for object detection tasks by combining multiple images.
2. Implement and train Faster R-CNN to detect individual digits within each combined image.
3. Evaluate the model using metrics such as accuracy, precision and recall analysis to understand its detection capability on this dataset.

Literature Review:

Faster R-CNN Overview

- **R-CNN Evolution into Faster R-CNN:**
  Models such as R-CNN (2014) and Fast R-CNN (2015), which distinguished between region proposal and object classification tasks, were initially used for object detection. Due to external region proposal generation (e.g., selective search), these models were accurate but expensive. An integrated Region Proposal Network (RPN) was proposed by Faster R-CNN (2015) to improve efficiency and simplify this process.

- **Network of Region Proposals (RPN)**:
  By calculating each region's "objectness" score, Faster R-CNN's RPN predicts which regions in the network are likely to contain objects. By sharing convolutional layers between the RPN and object classification stages, our end-to-end method decreased the processing effort.

- **A Two-Step Detection Framework to Increase Precision**:
  Two steps create Faster R-CNN's object detection process:

  Stage 1: Using information from a shared convolutional backbone, the RPN suggests possible object regions.
  Stage 2: Bounding boxes are modified to better fit the identified objects once these suggestions have been verified and categorised.

- **Faster R-CNN's architecture relies on:**
  1. **Feature Extraction Backbone (ResNet-50)**: This extracts features from input images.
  2. **Region Proposal Network**: Predicts bounding boxes for possible objects.
  3. **Classification and Regression Heads**: Classifies objects and refines bounding boxes.

Methodology:

Dataset Preparation

Several digit images are arranged in rows into a single, bigger image to adapt the COCO dataset for object detection:

1. **Data loading and transformation:** Pictures are scaled to 256x256 pixels and transformed to three-channel grayscale.
2. **Image Stacking and Bounding Boxes:** Four randomly selected COCO digits are used in each row of photographs to create a composite image, with bounding boxes established around each digit.

Custom Dataset Class

The combined photos and their matching bounding boxes are handled by an unique PyTorch Dataset class called COCO data. During training, this class allows Faster R-CNN to process images and return target labels with bounding boxes as well as input tensors.

Model Selection and Training

1. **Faster R-CNN with ResNet-50 Backbone**: We employ a ResNet-50 feature extractor when combined with the pretrained Faster R-CNN model. The model learns generic features faster thanks to the previously trained weights.
2. **Optimization:** Stochastic Gradient Descent (SGD) is used to train the model, using a learning rate of 0.0001, momentum of 0.9, and weight decay of 0.0005.
3. **Training Loop**: The training loop includes a forward pass, loss calculation, and backward pass to optimize the model. Losses are tracked for each epoch, and precision, recall, and accuracy are calculated to evaluate performance.

Metrics for Evaluation

To understand the model's effectiveness on this dataset, several metrics are tracked:

1. **Precision**: Measures the accuracy of the model's positive predictions.
2. **Recall**: Measures the coverage of actual positives.
3. **Accuracy**: Tracks overall prediction correctness.

Model Testing and Visualization

For testing, predictions are visualized with both ground truth and detected bounding boxes. This qualitative analysis showcases Faster R-CNN's performance in terms of accurately predicting digit locations within the combined images.

Implementation Details and Code Explanation:

Code Breakdown

1. **Libraries and Device Setup**: Necessary libraries like PyTorch, torchvision, and scikit-learn are imported. The device is set up to use GPU if available.
2. **Data Preprocessing**: The COCO dataset is loaded, transformed to a 3-channel grayscale format, and resized.
3. **Image Stacking and Bounding Boxes**: Random subsets of MNIST digits are stacked horizontally, and bounding boxes are defined for each.

4. **Custom Dataset Class (coco_Data)**: This class handles loading images and bounding boxes to be fed into the Faster R-CNN model.
5. **Model Initialization and Training Loop**: The pretrained Faster R-CNN model is loaded, optimized with SGD, and trained over a defined number of epochs. During each epoch, model loss, precision, recall, and accuracy are computed.
6. **Prediction Visualization**: Predictions for sample images are displayed with bounding boxes, differentiating true labels from model predictions.

Visualization

A helper function, plot_prediction, displays predicted bounding boxes against ground truth on sample images. Ground truth boxes are shown in green, while predictions are shown in red, helping visually assess model accuracy.

Results:

The model's performance is evaluated using metrics tracked across all epochs:

1. **Loss Curve**: The loss curve provides insight into model convergence. Lower loss across epochs indicates the model's improving accuracy.
2. **Precision, Recall, and Accuracy**: High precision and recall values suggest effective digit localization, while accuracy confirms correct predictions.

Sample outputs show the model accurately detecting and classifying most digits within combined images. However, performance could be further improved by refining bounding box accuracy and model parameters.

Conclusion:

The Faster R-CNN model demonstrates high efficacy applied to object detection on COCO with felicitous finding a very good balance between accuracy and efficiency along with its integrated methodology concerning the region proposals and classification. Prizes notwithstanding achieved, presumably even for medium-sized objects as well as large objects, but ideally improved detection of small objects and further inference speed would make it more adapted for real-time applications. Future research may investigate the integration of Faster R-CNN with feature enhancement techniques or alternative architectures in order to optimize its efficacy regarding small objects and to enhance processing efficiency.

Future Work

To build upon these results, future research could involve:

1. Experimenting with more complex and varied datasets.
2. Implementing techniques to handle overlapping objects or different spatial configurations.
3. Testing alternative object detection architectures like YOLO or SSD on this dataset for comparison.

Colab Link:

https://colab.research.google.com/drive/185fKPXrh3epTRxYL7ILlczPjnrHnQzdl