International Institute of Information Technology Hyderabad

Report on Machine Learning

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

Abstract:

Object detection is one of the core tasks in the computer vision domain with wide applications, where objects involved are localization and classification. It is also one of the state-of the art object detection frameworks that have performed profoundly to localize and classify objects in images. We outline how the challenging task of the MNIST dataset is met in the implementation and evaluation of the Faster R-CNN. In this paper, we describe the complexities involved in the architecture of the Faster R-CNN, most notably its region proposal network, and the subsequent steps of classification and bounding box regression. Results of experiments over these papers unveil Faster R-CNN's strength of better accuracy over MNIST dataset as compared with the previously proposed methods. In addition to that, we also review the performance of the model through differences in hyperparameters as well as various training regimes. Besides, we also comment on other ways toward more meaningful improvements, for example, including attention mechanism as well as other backbone networks. It certainly represents a critical step in the advancement and continuous evolution of object detection methods and these vivid and insightful views toward future progress in the area.

Machine Learning:

In terms of recognition of patterns and prediction, ML trains models on data. That is to say, the label refers to the intended prediction, and features are referred to as input variables. Unlabelled data looks for patterns, and labelled data looks for patterns as well, but it uses them for supervised learning. When a model performs well over the training data but poorly over new data, then it is said to be overfitting; if it fails to identify the pattern, then it is said to be underfitting. Regression predicts continuous values, whereas classification predicts categories. Neural networks and deep learning enable learning from complex data. Accuracy, recall, and precision are some of the significant criteria.

Introduction:

Object detection: Given an image, recognize and locate objects in it. Application areas range from self-driving cars to monitoring security cameras and robotics. Faster R-CNN is the newest state-of-the-art model from the variants that improve on the previous R-CNN approach by introducing the region proposal process inside the network to increase the speed and accuracy of detection. A two-stage framework contributed by the RPN and a Fast R-CNN classifier makes Faster R-CNN an efficient tool for object detection across the scales and contexts. Most of the analysis contained in this report depends on a very challenging benchmark: the MNIST dataset, more than 200,000 images and 80 object categories that match real-world complexities like occlusion and cluttered backgrounds. In this study, testing would identify the computational problems that are present for Faster R-CNN as well as the possible optimizations available.

Objective:

The primary objective of this project is to:

1. Adapt the MNIST 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 instead separated the region proposal and object classification tasks, were mainly used for object detection. Because external region proposal generation, such as selective search, was necessary in these models, they were accurate but costly. The integrated Region Proposal Network (RPN) was developed by Faster R-CNN in 2015 to enhance efficiency and make the process easier..

- **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 MNIST 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 MNIST digits are used in each row ofphotographs 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 MNIST 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 MNIST 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 shows high efficiency when applied to object detection on MNIST with felicitous finding a very good balance between accuracy and efficiency together with its integrated methodology regarding the region proposals and classification. The 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 work will try to combine this Faster R-CNN architecture with feature enhancement techniques or other architectures for an even better performance with small objects and as an efficient processing.

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