FINAL-REPORT: Faster RCNN

SongSeungWu

1 WHAT IS Faster RCNN?

Faster R-CNN is a highly effective deep learning model for object detection, designed to quickly and accurately detect and locate objects within an image. Its name reflects its improvement over previous R-CNN models in terms of both efficiency and speed.

Key Component:

- Backbone Network: This is a convolutional neural network used to extract high-level features from the input image. Common networks like VGG16 or ResNet serve as the backbone, converting the input image into a lower-resolution feature map.
- Region Proposal Network: The RPN is a core innovation in Faster R-CNN, responsible for proposing regions in the feature map where objects are likely to be present.
- ROI Pooling: Proposed regions vary in size, so ROI Pooling is used to standardize these regions to a fixed size, making them compatible with subsequent layers.
- Classification and Bounding box regression This involves both classification, to identify the type of object, and bounding box regression, to precisely adjust its position.

Advantages of Faster R-CNN:

- Faster R-CNN enables efficient, end-to-end training through the use of RPN, significantly improving detection speed over previous R-CNN models.

2 Util Functions

- loc2bbox: Decodes new bounding box coordinates from a given src_bbox and loc. It calculates the center and dimensions of src_bbox and applies the loc offset and scale adjustments to determine the new box's position and size.
- bbox2loc: This function calculates the offsets and scales required to move src_bbox to dst_bbox, providing the changes in position and size. This is useful for tracking object location and scale adjustments.

3 Hyper parameters

- epochs: Sets the number of epochs for training, running the entire dataset 14 times.
- learning_rate: Sets the initial learning rate to 0.001.
- lr_decay: Specifies the rate to decay the learning rate by a factor of 0.1 after a certain epoch
- weight_decay: Adds weight decay for regularization of model parameters.
- use_drop: Decides whether to use dropout in the RoIHead.
- rpn_sigma and roi_sigma: Controlling the impact on the loss function.
- data_dir: Specifies the dataset directory path.
- train_load_path: Sets the path for loading checkpoints during training.
- inf_load_path: Specifies the path for the checkpoint to use during inference.

4 RPN(Region Proposal Network)

- generate_anchor_base: Generates 9 anchor boxes based on base_size, ratios, and anchor_scales.
  Output: A (9, 4) shaped anchor_base array, where each row contains anchor box coordinates (y_min, x_min, y_max, x_max).

Each anchor box is centered and scaled according to width and height, with offsets applied to calculate final positions.

5 ProposalCreator

Generates Regions of Interest from rpn_loc and anchors obtained from the RPN. To reduce the number of RoIs, it filters proposals based on a predefined minimum size and applies Non-Maximum Suppression to return the final RoIs.

6 Region Proposal Network

The Region Proposal Network generates region proposals based on the feature map obtained from passing an image through VGG16.

- Intermediate Feature Map Creation: A 3x3 convolution is applied to the feature map from the backbone, resulting in an intermediate feature map.
- Bounding Box Offset Prediction: A 1x1 convolution is applied to middle, producing an output feature map with channels representing bounding box offsets for each anchor box.
- Object Score Prediction: Another 1x1 convolution is applied to middle to produce an output feature map with channels representing objectness scores for each anchor box.
- Region Proposal Generation: Using the Proposal Creator function, the final regions of interest are generated based on the predicted bounding box offsets and scores.

7 Faster R-CNN head

The VGG16RoIHead class is designed to serve as the head for a Faster R-CNN model using VGG16 as a backbone. This head processes the Region of Interest feature maps obtained after RoI pooling, classifying and regressing bounding boxes.

Forward Pass:

- Concatenates RoI indices and coordinates for pooling.
- Applies RoI pooling on the input feature map x for each RoI.
- Flattens pooled features and passes them through a fully connected classifier.
- Computes bounding box offsets and class scores for each RoI.

8 Faster R-CNN

The Faster R-CNN class is the main model for object detection, combining a feature extractor, Region Proposal Network, and a RoI head.

- Initialization: Takes in an extractor, RPN, and head components, with settings for location normalization.

- Forward Pass: In forward, the image goes through the extractor, then the RPN, and finally the RoI head for classification and bounding box regression.
- Prediction: The predict function uses the trained model to perform object detection on input images. It includes scaling, bounding box conversion, and post-processing to return final bounding boxes, labels, and scores.
- Suppression: _suppress removes overlapping bounding boxes using NMS, filtering out the background class and adjusting thresholds.
- Optimizer Setup: The get_optimizer and scale_lr methods handle SGD optimizer creation and learning rate scaling.

9 Trainer

includes key components for training and managing the Faster R-CNN model, including loss calculations and target generation.

- bbox_iou Function: Calculates the Intersection over Union between two bounding boxes, a fundamental metric for identifying object overlap in object detection.
- AnchorTargetCreator: Generates targets for anchors by assigning labels (positive, negative) based on IoU values. It also selects a balanced set of positive and negative samples from the anchors.
- ProposalTargetCreator: Selects RoIs from RPN output for further processing. RoIs are assigned positive or negative labels based on IoU thresholds, and only a fixed number of RoIs are selected.
- FasterRCNNTrainer: Main class for training the Faster R-CNN model.
  - forward: Computes losses for RPN and RoI heads based on ground truth targets.
  - train_step: A single training step updating the model weights.
  - Loss Calculation: Includes both localization loss (bounding box regression) and classification loss.
  - Checkpoints: Save and load checkpoints to resume training efficiently.
- Smooth L1 Loss Functions: _smooth_l1_loss and _fast_rcnn_loc_loss calculate the loss for bounding box predictions, considering only positive examples to focus on object localization.

10 Train

- Dataset Loading: Loads the training dataset using the TrainCustom class and prepares the DataLoader.
- Model and Trainer Initialization: Initializes the Faster R-CNN model and the trainer on GPU.
- Checkpoint Loading: Optionally loads a pretrained checkpoint if train_load_path is specified.
- Training Loop:
  - Runs through the dataset for a specified number of epochs.
  - For each batch, performs a training step on the GPU and updates the model's parameters based on the computed loss.
  - Tracks and displays the total loss for each epoch, saving the model checkpoint if it achieves a lower loss than previously recorded.
- Learning Rate Adjustment: Reduces the learning rate by a decay factor after 9 epochs.

11 Train Result

The following details capture the progress of training the Faster R-CNN model on the given dataset. The model was trained for 14 epochs, with each epoch demonstrating improvements in loss metrics, indicating enhanced model performance as training progressed.

Training Loss Details by Epoch:

- Epoch1:
  - RPN Localization Loss: 0.1629
  - RPN Classification Loss: 0.2931
  - ROI Localization Loss: 0.3066
  - ROI Classification Loss: 0.5294
  - Total Loss: 1.2921
- Epoch2:
  - RPN Localization Loss: 0.1520
  - RPN Classification Loss: 0.2357
  - ROI Localization Loss: 0.2650
  - ROI Classification Loss: 0.4775
  - Total Loss: 1.1303
- Epoch14:
  - RPN Localization Loss: 0.1242
  - RPN Classification Loss: 0.1400
  - ROI Localization Loss: 0.1713
  - ROI Classification Loss: 0.2611
  - Total Loss: 0.6966

Conclusion:
Over 14 epochs, the Faster R-CNN model showed continuous improvement in reducing the total loss, with the final epoch achieving a total loss of 0.6966. This suggests that the model effectively learned to reduce localization and classification errors in both the RPN and ROI phases.

12 Inference

- eval function: Receives test data from dataloader and uses the Faster R-CNN model for inference. It stores the predicted bounding boxes, class labels, and scores in a list of dictionaries.
- inference function: Loads the test dataset and prepares it with DataLoader. It loads the pre-trained Faster R-CNN model to perform predictions, filters bounding boxes based on a score threshold, and formats the results for submission. The final output is saved as faster_rcnn_scr

13 Conclusion

In this study, a Faster R-CNN object detection model was developed using VGG16 as the backbone, combining RPN and RoI head components into a complete system. Training results indicate a stable reduction in loss values, reflecting improved model performance, and the final inference yielded effective predictions on the test dataset. This model demonstrates potential applicability across various object detection tasks.

14 Modification

- loc2bbox function
  - Updated to handle PyTorch Tensor format. When src_bbox and loc are Tensors, they are converted to numpy for processing, and the final result is returned as a Tensor on the appropriate device.
  - Modified to return torch.zeros for an empty src_bbox, improving GPU compatibility.
- bbox2loc function
  - Adjusted to work seamlessly with Tensor format for better GPU support.