FASTER

R-CNN

ARIHARASUDHAN

EXPLAINED

Faster R-CNN (Region-based Convolutional Neural Network) is an object detection algorithm that combines the concepts of region proposals and convolutional neural networks to accurately detect and classify objects in an image.

Faster R-CNN = RPN + CNN

It consists of two main components : the Region Proposal Network (RPN) and the Region-based Convolutional Network (RCNN).

Region Proposal Network

The RPN takes an image as input and generates a set of proposed regions (bounding boxes) likely to contain objects of interest. It operates on a sliding window over the convolutional feature map

generated by a backbone network (e.g., ResNet). At each sliding window position, the RPN predicts two outputs for each anchor box: objectiveness score (whether the anchor contains an object or not) and refined bounding box coordinates. The RPN uses anchor boxes of different scales and aspect ratios to handle objects of various sizes and shapes.

Region-based Convolutional Network

Once the proposed regions are generated by the RPN, they are used to extract fixed-sized feature maps from the backbone network. These region proposals are then fed into an RoI (Region of Interest) pooling layer, which transforms them into a fixed size, allowing them to be processed by a classifier. The RCNN applies fully connected layers and softmax

classification to predict object labels and refine the bounding box coordinates for each proposed region. The RoI pooling layer aligns the extracted features with the corresponding regions, ensuring spatial correspondence between the features and the regions.

TRAINING

The entire Faster R-CNN model is trained in a two-stage process:

1. Pre-training:

The backbone network (e.g., ResNet) is pre-trained on a large-scale image classification dataset (e.g., ImageNet) to learn general visual features. This pre-trained backbone is used to initialize the weights of the RPN and the RCNN components.

2. Fine-tuning:

The RPN and the RCNN are then jointly fine-tuned on a smaller dataset specifically annotated for object detection tasks. During fine-tuning, the RPN learns to generate accurate proposals, and the RCNN learns to classify the proposed regions and refine their bounding box coordinates.

KEY CONCEPT

The key idea behind Faster R-CNN is that it combines region proposal generation (RPN) and object classification / regression (RCNN) into a single unified network. This design allows for more efficient and accurate object detection by sharing convolutional feature computation between the RPN and the RCNN, reducing redundant calculations.

Overall, Faster R-CNN provides a robust and accurate framework for object detection by effectively integrating region proposals and convolutional neural networks.

IMPORTS

In [4]:
```
import torch
import torchvision
from torchvision.models.detection import FasterRCNN
from torchvision.models.detection.rpn import AnchorGenerator
from torchvision.transforms import ToTensor
from PIL import Image
```

THE MODEL

In [ ]:
```
# Define the Faster R-CNN model
def create_faster_rcnn_model(num_classes):
    backbone = torchvision.models.resnet50(pretrained=True)
    in_features = backbone.fc.in_features
    backbone.fc = torch.nn.Linear(in_features, num_classes)
    anchor_generator = AnchorGenerator(sizes=((32, 64, 128, 256, 512),),
                                       aspect_ratios=((0.5, 1.0, 2.0),))
    model = FasterRCNN(backbone, num_classes=num_classes,
                       rpn_anchor_generator=anchor_generator)
    return model
```

PREPROCESSING

In [ ]:

```python
num_classes = 21  # Number of object classes + background
model = create_faster_rcnn_model(num_classes)
image = Image.open('example_image.jpg')
transform = ToTensor()
input_tensor = transform(image)
input_batch = input_tensor.unsqueeze(0)
```

INFER

In [ ]:

```python
# Pass the input batch through the Faster R-CNN model
model.eval()
with torch.no_grad():
    prediction = model(input_batch)

# Process the prediction (e.g., get the bounding boxes, labels, and scores)
boxes = prediction[0]['boxes'].tolist()
labels = prediction[0]['labels'].tolist()
scores = prediction[0]['scores'].tolist()

# Print the bounding boxes, labels, and scores
for box, label, score in zip(boxes, labels, scores):
    print('Label: {}, Score: {:.2f}'.format(label, score))
    print('Bounding Box:', box)
```

MERCI