University Echahid Hama Lakhdar, El-oued
Institute of Exact Sciences
Department of Computer Science
$2^{ed} Master$:
Artificial Intelligence and Data Science
Semester : 3.2025

Lab 6 : Object Detection using R-CNN
From Region Proposals to Deep Feature Classification

Master Level – Computer Vision

Introduction

This lab session aims to teach step-by-step implementation of the R-CNN (Region-based Convolutional Neural Network) architecture using Python. You will explore the process of generating region proposals, extracting CNN features, classifying them, and evaluating detections. A subset of the COCO dataset will be used as a case study, with an optional extension for vignette (drug sticker) data.

—

1 Objective

— Understand R-CNN architecture components.
— Implement Selective Search for region proposals.
— Extract CNN features using pre-trained networks.
— Train and evaluate SVM classifiers.
— Apply Non-Maximum Suppression (NMS).
— Evaluate performance on COCO and vignette datasets.

—

2 Task 1 : Dataset Preparation

COCO Subset

Download a subset of COCO (e.g., categories : `person, dog, car`). Resize images to 416×416 for uniformity.

Listing 1 – Load and prepare COCO subset

```
from pycocotools.coco import COCO
import requests, os, cv2

dataDir = 'coco'
dataType = 'val2017'
annFile = f'{dataDir}/annotations/instances_{dataType}.json'

coco = COCO(annFile)
catIds = coco.getCatIds(catNms=['person','dog','car'])
imgIds = coco.getImgIds(catIds=catIds)

img = coco.loadImgs(imgIds[0])[0]
img_data = requests.get(img['coco_url']).content
with open('sample.jpg', 'wb') as f:
    f.write(img_data)
```

Questions :

a) Why is resizing necessary for CNN feature extraction ?

b) What challenges arise from using only a subset of COCO ?

Vignette Dataset (Extension Questions)

a) How does lighting variation in vignette images affect region proposals ?

b) Which preprocessing methods could improve OCR detection ?

—

3 Task 2 : Region Proposals using Selective Search

Listing 2 – Generate region proposals with Selective Search

```
import selectivesearch
import matplotlib.pyplot as plt

img = cv2.imread('sample.jpg')
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
_, regions = selectivesearch.selective_search(img_rgb, scale=500, sigma=0.9,
    min_size=50)

candidates = []
for r in regions:
    if r['rect'] in candidates:
        continue
    x, y, w, h = r['rect']
    if w * h < 2000:
        continue
    candidates.append(r['rect'])

for (x, y, w, h) in candidates[:50]:
    cv2.rectangle(img_rgb, (x, y), (x+w, y+h), (255, 0, 0), 2)

plt.imshow(img_rgb)
plt.title('Region Proposals')
plt.show()
```

Questions :

a) What are the trade-offs in adjusting `scale`, `sigma`, and `min_size` ?

b) Why does Selective Search tend to over-generate proposals ?

Vignette Dataset (Extension Questions)

a) How does Selective Search handle text-dense vignette images ?

b) Could MSER yield better proposals for text ? Why ?

—

4 Task 3 : CNN Feature Extraction

Use a pre-trained CNN (e.g., VGG16) to extract deep features from proposed regions.

Listing 3 – Extract CNN features using VGG16

```
from tensorflow.keras.applications.vgg16 import VGG16, preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
import numpy as np

model = VGG16(weights='imagenet', include_top=False, pooling='avg')
```

```
6
7  features = []
8  for (x, y, w, h) in candidates[:200]:
9      roi = img_rgb[y:y+h, x:x+w]
10     roi = cv2.resize(roi, (224, 224))
11     roi = img_to_array(roi)
12     roi = np.expand_dims(roi, axis=0)
13     roi = preprocess_input(roi)
14     f = model.predict(roi)
15     features.append(f.flatten())
```

Questions :

a) Which CNN layers provide the most robust object features ?

b) How does feature dimensionality affect SVM training time ?

Vignette Dataset (Extension Questions)

a) Which CNN features best capture character shapes ?

b) Would fine-tuning on text patches improve OCR accuracy ?

—

5 Task 4 : SVM Classification

Train an SVM classifier on the extracted features.

Listing 4 – Train an SVM classifier

```
1  from sklearn.svm import LinearSVC
2  from sklearn.model_selection import train_test_split
3
4  X = np.array(features)
5  y = np.random.randint(0, 2, len(X))  # Placeholder labels
6
7  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
8  svm = LinearSVC(max_iter=10000)
9  svm.fit(X_train, y_train)
10
11 print("Accuracy:", svm.score(X_test, y_test))
```

Questions :

a) Why is an SVM used instead of softmax here ?

b) How could we handle class imbalance in object detection ?

Vignette Dataset (Extension Questions)

a) How would you label “entête” regions versus background ?

b) Can SVMs separate characters from background efficiently ?

—

6 Task 5 : Detection and Non-Maximum Suppression

Apply SVM predictions to region proposals and refine detections with NMS.

Listing 5 – Apply NMS to reduce overlapping detections

```
1  from imutils.object_detection import non_max_suppression
2
3  boxes = np.array([[x, y, x+w, y+h] for (x,y,w,h) in candidates[:200]])
4  scores = np.random.rand(len(boxes))  # Simulated scores
```

```
5
6 final_boxes = non_max_suppression(boxes, probs=scores, overlapThresh=0.3)
7 print("Detections after NMS:", len(final_boxes))
```

Questions :

a) What effect does the IoU threshold have on detections ?

b) How could overlapping boxes cause false positives ?

Vignette Dataset (Extension Questions)

a) How can NMS be tuned for dense character regions ?

b) What IoU threshold suits OCR-style text boxes ?

—

7 Task 6 : Evaluation and Visualization

Evaluate the system using mean Average Precision (mAP).

Listing 6 – Evaluate detection performance

```
1 from sklearn.metrics import average_precision_score
2
3 y_true = np.random.randint(0, 2, len(scores))
4 mAP = average_precision_score(y_true, scores)
5 print("Mean Average Precision (mAP):", mAP)
```

Questions :

a) Why is mAP a better metric than accuracy for detection ?

b) How do localization errors impact mAP ?

Vignette Dataset (Extension Questions)

a) Which metric suits character-level accuracy best ?

b) How can we evaluate both detection and OCR jointly ?

—

8 Task 7 : Backbone Variation (Optional)

Try different CNN backbones for feature extraction (ResNet, MobileNet).

Listing 7 – Use ResNet backbone for feature extraction

```
1 from tensorflow.keras.applications import ResNet50
2
3 resnet = ResNet50(weights='imagenet', include_top=False, pooling='avg')
4 features_resnet = [resnet.predict(preprocess_input(np.expand_dims(cv2.resize(
    img_rgb[y:y+h, x:x+w], (224,224)), axis=0))).flatten() for (x,y,w,h) in
    candidates[:100]]
```

Questions :

a) How does backbone depth affect computation time ?

b) What trade-offs exist between MobileNet and ResNet ?

Vignette Dataset (Extension Questions)

a) Which backbone best captures small-text details ?

b) How could OCR-specific backbones improve detection ?

—

9 Conclusion

Students have implemented a complete R-CNN pipeline step by step :

1. Region Proposals (Selective Search)
2. CNN Feature Extraction
3. SVM Classification
4. Non-Maximum Suppression
5. Evaluation

By extending the same workflow to vignette data, students can adapt R-CNN principles for OCR and printed-text detection.