Exercise 5.2 Report (10 ECTS)

Exercise 5.2 Implementation Summary

Task 5.2.1: Proposal Generation and Caching

Implemented in `code/generate_proposals.py`.

- For each split (`train`, `valid`, `test`), load images and run `selective_search(...)`.
- Save proposals as compressed `.npz` files with one array `rects` (`N x 4`, (x, y, w, h)).
- Output structure: data/balloon_dataset/proposals/<split>/<image>.npz.
- This avoids recomputing selective search during training/evaluation.

Task 5.2.2: Positive/Negative Samples from IoU

Implemented in `code/detector_pipeline.py` (`build_samples()`).

- Load COCO annotations and proposals, then compute max IoU of each proposal with all GT boxes.
- Label positive if $\max IoU \geq t_p$; label negative if $\max IoU \leq t_n$.
- Per-image caps are applied (`max_pos_per_img`, `max_neg_per_img`) to control imbalance and runtime.
- Ambiguous proposals in (t_n, t_p) are ignored.

Task 5.2.2: Feature Extraction

Implemented in `code/detector_pipeline.py`, `code/run_inference.py`, and `code/evaluate_metrics.py`.

HOG option

- Resize crop to `out_size x out_size` and compute HOG: `orientations=9`, `pixels_per_cell=(8,8)`, `cells_per_block=(2,2)`, `channel_axis=-1`.

CNN option (used in final result)

- Pretrained ResNet18 (`torchvision`), `fc` replaced with `Identity` to get 512-D features.
- Preprocess: `Resize((224,224))`, `ToTensor()`, ImageNet normalization ($[0.485, 0.456, 0.406]$, $[0.229, 0.224, 0.225]$).
- Apply L2 normalization on each 512-D feature vector.
- Training and inference use the same preprocessing and feature extraction.

Task 5.2.3: SVM Training

Implemented in `code/detector_pipeline.py`.

- Classifier: `LinearSVC(class_weight='balanced', max_iter=5000)`.
- Validate on `valid` split and print `classification_report`.
- Save model with `joblib` to `results/balloon_svm.joblib`.
- Optional hard-negative mining: after first training round, collect top-scoring false positives (max $IoU \leq t_n$), append them as extra negatives, and retrain.

Task 5.2.4: Inference Script

Implemented in `code/run_inference.py`.

- For one input image: generate proposals, extract features, run SVM `decision_function`.
- Filter by `score_thresh`, apply NMS (`nms_thresh`), keep top-k boxes.
- Save visualization with red boxes to output image.

Task 5.2.5: Evaluation (COCO mAP + MABO)

Implemented in `code/evaluate_metrics.py`.

- Evaluate on `test` split with official `pycocotools` COCOeval: AP@[0.50:0.95], AP@0.50.
- Before submission to COCOeval, apply score filter + NMS + top-k to reduce duplicate false positives.
- Supports cached test proposals (`-proposals_root`); otherwise falls back to on-the-fly selective search.
- MABO is computed as mean best overlap between each GT and all proposals of the same image.

Final Configuration and Results

Configuration used for reported result

- Proposal generation: `scale=300`, `min_size=50`, `max_merges=3500`.
- Training: `feature=cnn`, `tp=0.5`, `tn=0.2`, `max_pos_per_img=100`, `max_neg_per_img=30`, `augment=True`, `aug_pos=5`, `hard_neg=True`.
- Evaluation: `feature=cnn`, `score_thresh=-1.0`, `nms_thresh=0.3`, `top_k=100`, cached test proposals.

Metrics

Metric	Value
mAP (IoU 0.50:0.95)	0.1149
AP@0.50	0.4556
MABO	0.6020

Short Analysis

- AP@0.50 is much higher than mAP@[0.50:0.95], indicating detection is often correct but localization is not always tight at high IoU.
- MABO around 0.60 suggests proposals have reasonable recall/coverage.
- Main remaining errors are duplicated/partial boxes and occasional confusing negatives (e.g., face/head-like round regions).

Q5.2 Answers

Q5.2.1

Compared to Uijlings et al., this implementation is simplified:

- Uses one selective search configuration instead of full diversification across color spaces and multiple initializations.
- Uses CNN (or HOG) features + linear SVM, rather than the original SIFT-based pipeline and full paper setup.
- Uses a lightweight hard-negative step (single optional retraining loop), not a fully iterative mining/training regime.
- No bounding-box regression/post-refinement stage.

Q5.2.2

Changing thresholds affects sample quality/quantity:

- Higher t_p: cleaner positives, fewer samples.
- Lower t_p: more positives, but noisier labels.
- Lower t_n: cleaner negatives (farther from object), often easier.
- Higher t_n: harder negatives, but risk of mislabeled near-object samples.

Two thresholds are needed to create a gray zone (t_n, t_p) that excludes ambiguous proposals. With one threshold, many borderline samples would be forced into wrong labels and hurt SVM training.

Q5.2.3

Ways to increase effective training data:

- Stronger data augmentation for positive crops (flip, brightness/contrast/color jitter, random affine/crop).
- Increase proposals and lower t_p slightly to collect more positives, then clean with hard-negative mining.
- Add external balloon-like data (same class) and convert to COCO-format boxes.
- Use pseudo-labeling from high-confidence detections on unlabeled images.
- Perform k-fold cross-validation and model averaging on a small dataset.

Reproducibility

Dependencies

```
pip install -r requirements.txt
pip install pycocotools torch torchvision
```

Commands

1) Generate proposals

```
python3 code/generate_proposals.py \
  --data_root data/balloon_dataset \
  --out_root data/balloon_dataset/proposals \
  --splits train valid test \
  --scale 300 --min_size 50 --max_merges 3500
```

2) Train detector

```
python3 code/detector_pipeline.py \
  --data_root data/balloon_dataset \
  --proposals_root data/balloon_dataset/proposals \
  --feature cnn \
  --tp 0.5 --tn 0.2 \
  --max_pos_per_img 100 --max_neg_per_img 30 \
  --augment --aug_pos 5 \
  --hard_neg \
  --model_out results/balloon_svm.joblib
```

3) Evaluate

```
python3 code/evaluate_metrics.py \
  --data_root data/balloon_dataset \
  --proposals_root data/balloon_dataset/proposals \
  --model results/balloon_svm.joblib \
  --feature cnn \
  --score_thresh -1.0 \
  --nms_thresh 0.3 \
  --top_k 100
```

4) Single-image inference visualization

```
python3 code/run_inference.py \
  --image data/balloon_dataset/valid/<image_name>.jpg \
  --model results/balloon_svm.joblib \
  --feature cnn \
  --score_thresh 0.1 \
  --nms_thresh 0.4 \
  --top_k 50 \
  --out results/inference.png
```