Machine Learning for Signal Processing Course 2024/2025

Homework B.2 – Maritime Deep Learning Applications
(your first end-to-end CNN/RNN project on satellite data)

Release: 23 May 2025 **Due:** 15 June 2025 (23:59 CEST)

Why this homework?

This assignment is designed as the gentlest possible introduction to deep learning for MLSP students. You will start from well-commented starter code and tiny open datasets, and finish with a working model that can recognise ship types *or* predict their motion. Everything runs comfortably on a laptop CPU or a single GPU (use Kaggle if you need GPUs).

- **Hands-on & guided:** every operation you need is mapped one-to-one between the numbered steps in this document and the labelled cells in the companion template (Python notebook or MATLAB script).
- **Pick your own path:** choose the task flavour (patch classification or pixel-wise segmentation) and customise *at least* two pipeline components.
- **Learn by doing:** you will touch data wrangling, network design, training loops, metric tracking, and error analysis – the full ML lifecycle *in miniature*.

Top-tier performance is not expected, demonstrating correct ML workflow is the real goal of the homework.

Tip: work sequentially. Divide your notebook in sections corresponding to the project steps.

---

1 Learning outcomes

After completing the homework you are expected to be able to:

1. Prepare small satellite datasets (RGB or multispectral, 10–13 bands) with clean, reproducible code.
2. Build and train compact convolutional-based networks under tight compute limits.
3. Choose and calculate appropriate evaluation metrics for imbalanced remote-sensing problems.
4. Document & deliver an ML experiment so that a peer can reproduce it with one click.

---

2 Step-by-step workflow

Follow the steps in order; keep each step in a reusable function or cell so that you can refactor easily.

Step 3.1 – Environment setup

- Create the project template in Python or MATLAB and create a conda/venv environment or a MATLAB project.

Step 3.2 – Dataset selection and download

- Choose one of the following primary sources (add URLs in README): EuroSAT (RGB, 10 k tiles 64×64), scenes from Sentinel-2, 10 land-use classes; UC Merced Land-Use (RGB, 2.1 k images 256×256), 21 aerial scene classes; So2Sat LCZ42 (multispectral, 12 bands, 180 MB subset), Local Climate Zones; SEN12MS Tiny (multispectral, 10 bands, 250 MB), land-cover segmentation masks. Feel free to search for other available datasets on the web.

Step 3.3 – Data exploration & preprocessing

You can perform some of the following preprocessing examples, or similar ones depending on the project choices.

- Display sample RGB composites (or PCA for multispectral).
- Compute class histograms; plot band statistics.
- Preprocess: for instance, centre-crop/resample to 128×128; per-band normalization; optional histogram equalization.
- Stratified split train/val/test (70/15/15). Save splits for reproducibility.

Step 3.4 – Baseline model

- *Patch classification path*: Mini ResNet 18 truncated to ≤1.5 M parameters; global average pooling + softmax. Report overall accuracy, macro-F1, confusion matrix.
- *Pixel wise segmentation path*: UNet Lite (≤ 2 M parameters, depth = 4) with sigmoid/softmax output. Report mIoU, pixel accuracy; display 3 qualitative overlay examples.
- *Losses*: cross entropy (classification) + optional Dice/Focal (segmentation).
- *Initial target*: ≥ 60 % balanced accuracy or ≥ 0.40 mIoU on validation.

Step 3.5 – Training loop & logging

- Implement reproducible seed fixing.
- Log epoch-wise loss and metric curves (Matplotlib/TensorBoard or MATLAB trainingProgressMonitor).
- Stop when val-loss stops improving for 15 consecutive epochs or when budget is reached and explain.

Step 3.6 – Evaluation & error analysis

- Report overall accuracy, macro-F1, confusion matrix for patch classification, or report mIoU, pixel accuracy for segmentation.
- Discuss at least two typical failure cases if possible.

Step 3.7 – Personalization (optional before 15 June, mandatory from 16 June on)

Modify *at least two* of the following and explain your choice:

- Architecture framework (depth/width, EfficientNet B0, depthwise separable conv, attention UNet).
- Data augmentation (random crops, flips, Cutout, RandAugment, spectral jitter).
- Optimizer & scheduler (AdamW, cosine decay, cyclic LR).
- Regularization (dropout, label smoothing, Mixup).

- Loss re weighting, focal loss, Dice loss.

Discuss why the personalized model behaves differently from the baseline.

Step 3.8 – Optional extra credit

Implement at least one of the following choices for an extra score (max +2 points):

- Transfer learning from ImageNet or BigEarthNet. Lightweight super resolution pre net. Post training quantization or pruning.

Step 3.9 – Report

Write a report (Introduction, Methods, Results, Discussion) in pdf as accompany material to the code, or write a well-described notebook. Include at least: data description, model diagrams, hyper-parameter table, quantitative results, qualitative analysis.

Step 3.10 – Packaging & submission

Submit the project as a zip file or Github link via the Classroom platform by **23:59 CEST, 15 June 2025**.

---

4 Example of Project template (but feel free to create your own)

```
Satellite_DL_<surname>/
 ├─ notebooks/              # or main.m at root
 |  └─ satellite_homework.ipynb
 ├─ src/
 |   ├─ datasets.py / utils.m
 |   ├─ model.py / model.m
 |   ├─ train.py / train.m
 |   └─ inference.py / runInference.m
 ├─ data/                   # filled by download script
 ├─ results/                # metrics, figures, saved models
 ├─ report.pdf
 ├─ README.md
 └─ requirements.txt | environment.yml | project.prj
```

Good luck and enjoy exploring Earth from above!