Lab 2: Deep Learning Training Clinic — Robust MLPs + RNNs for Real Sensor Sequences

Projects in ML and AI — Spring 2026 | CSCI-4170/6170
Mini-project (2-3 weeks)

Release Week: Week 6
Lab Time: 90–110 minutes per week
Recommended Duration: 3 weeks
Team Policy: Individual or groups of 2–4

Learning objectives

- Create a reproducible deep learning training pipeline (data loaders, training loop, checkpointing, fixed seeds).
- Diagnose and fix common training failures in real projects: overfitting, unstable loss/NaNs, exploding gradients, and poor generalization.
- Implement and compare a strong MLP baseline (non-sequence) and a sequence model (GRU/LSTM) for windowed sensor data.
- Make model decisions using validation discipline (no peeking at test) and document experiments in a structured log (CSV).
- Produce deployment-readiness artifacts: a saved model checkpoint, a simple inference function, and a basic latency measurement.
- Communicate results clearly via a short report section and a one-page model card (intended use, risks, limitations).

Constraints and allowed methods

Required

- Python + Jupyter Notebook / Google Colab (Pro is fine).
- PyTorch (preferred) or TensorFlow/Keras.
- NumPy, pandas, matplotlib; scikit-learn allowed for metrics only.
- Git repository with clear commit history.
- Experiment Log (CSV): one row per experiment run (model, params, seed, metrics, timestamp).

Allowed (encouraged)

- Early stopping, learning-rate scheduling, dropout, weight decay (L2), gradient clipping.
- Simple imbalance handling (class weights or light re-sampling).
- Mixed precision is allowed only if you can explain what changed and why.

Problem setup: on-device activity recognition from IMU sequences

You are building a prototype for on-device activity recognition using smartphone accelerometer and gyroscope data. This resembles real deployments: windowing and temporal structure, subject generalization, and inference latency constraints.

Dataset (default)

- Use the UCI Human Activity Recognition (HAR) dataset using the "Inertial Signals" (windowed time series).
- You must use sequential windows so an RNN model is meaningful (do not use only engineered summary features).

Leakage and generalization requirement (required)

- Use a subject-disjoint split: no subject IDs shared between train/validation and test.
- Prove it in the notebook by printing the set intersection size (must be 0).
- Your tuning decisions must use only the training subjects (with a validation split). Do not tune on test.

Deliverables (what to submit)

- A Git repo called Lab 2, containing notebooks + any source code.
- Notebook 1: 01_data_and_baselines.ipynb (dataset card + leakage checks + MLP baseline).
- Notebook 2: 02_training_clinic.ipynb (training curves, overfitting control, instability debugging, experiment log).
- Notebook 3: 03_rnn_and_inference.ipynb (GRU/LSTM model, final evaluation, inference function, latency).
- Model Card (1 page, markdown): intended use, evaluation protocol, metrics, limitations, risks.
- Experiment Log (CSV): one row per experiment run (model, params, seed, metrics, timestamp).

What you must build

1. Week 1 Checkpoints

Part A — Dataset card + leakage-safe split + MLP baseline

1. Mini datasheet (short dataset card): motivation, target definition, data source/license, signal description (channels, window length), and limitations/risks.
2. Data sanity checks: confirm shapes (num_windows, T, C), label distribution, and absence of NaN/inf after loading.
3. Leakage audit: implement a subject-disjoint split and print overlap = 0 (required).
4. Baseline 0 (trivial): majority-class predictor or random predictor with fixed seed.
5. Baseline 1 (required): MLP on flattened windows (T × C) with a train/validation split drawn only from training subjects.

6. Report one primary metric (accuracy or macro-F1) plus one supporting artifact (confusion matrix and per-class F1 table).

2. Week 2 Checkpoints

Part B — Training clinic: overfitting + instability (Week 2)

7. Training curves + reproducibility: plot train/val loss and metric across epochs; fix seeds and record them in your log.
8. Overfitting control (required): intentionally induce overfitting (oversized MLP or too many epochs), then fix it using at least two of: dropout, weight decay, early stopping, light noise augmentation, or LR scheduling. Report before/after validation results.
9. Instability debug (required): intentionally trigger one failure mode (NaNs from high LR, exploding gradients, or severe underfitting). Then fix it using two appropriate interventions (e.g., lower LR, gradient clipping, improved normalization, adjusted regularization).
10. Experiment log discipline: record ≥5 runs (teams) or ≥3 runs (individual) with key hyperparameters, seed, best validation metric, and a short note per run.

3. Week 3 Checkpoints

Part C — Sequence model + inference readiness (Week 3)

11. Core sequence model (required): GRU or LSTM that consumes (batch, T, C) windows and outputs class logits. Include at least one regularization choice (dropout and/or weight decay).
12. Training robustness: use gradient clipping (required for RNNs unless you justify why not). Use early stopping or LR scheduling.
13. Final evaluation: evaluate your best checkpoint on held-out test subjects. Provide confusion matrix, per-class performance, and a 3–5 sentence error analysis (which classes confuse, why plausible).
14. Inference function + latency (required): implement predict_activity(window) -> (label, probs). Time average inference per window over 100–500 windows on CPU in Colab and interpret feasibility for near-real-time use.
15. Model card + limitations: include intended use, non-intended use, evaluation protocol, known failure modes, and a short privacy/ethics note.

Required for Graduate (6170) (choose any ONE)

- Uncertainty: bootstrap confidence intervals for test macro-F1 (≥500 resamples).
- Interpretability: saliency/gradient-based attribution over timesteps/channels with a short discussion of a failure mode.
- Generalization: report performance variability across subjects and discuss deployment risk.

Weekly milestones and Thursday check-ins

Week	Milestone (due by Thursday lab session)	Evidence to show in the 3–5 minute explanation
Week 1	Datasheet + subject-safe split + MLP baseline	Notebook outputs; overlap=0 proof; baseline metrics; confusion matrix; commits
Week 2	Training clinic: overfitting fix + instability fix + experiment log	Before/after curves; one debug story; experiment log rows; commits
Week 3	GRU/LSTM + test eval + inference + latency + model card	Final metrics; confusion matrix; inference demo; timing results; model card; commits

End-of-lab explanation (required for full credit)

At the end of each Thursday lab session, your team must give a 3–5-minute explanation using this structure:

- What changed since last session? (show git commits or notebook diffs)
- What did you measure and what did you learn? (show one table/plot)
- One technical decision: what you tried, expected outcome, actual outcome, next step
- Instructor verification questions (any team member may answer)

Note: If the team cannot demonstrate evidence of progress and shared understanding, the oral-check component may be reduced for that week.

Submission checklist

- All notebooks run top-to-bottom without manual edits; all random seeds are fixed and reported.
- Repo contains an Experiment Log CSV with ≥5 runs (teams) or ≥3 runs (individual).
- Repo contains: baseline results, training-clinic evidence (curves + fixes), RNN results, and final test evaluation.
- Model Card included as markdown in the repo.
- Inference function works and latency measurement is reported.