Assignment 2 — Multimodal Sensor Fusion with Attention

ECE 532: Multimodal Machine Learning for Sensing Systems
Fall 2025

Release: October 25, 2025 Due: November 10, 2025 23:59 ET
Weight: 6%

1 The Challenge: When Sensors Fail

The Scenario: Your startup, *SenseFlow AI*, just landed a contract with a major hospital system. They want to deploy wearable sensors to monitor elderly patients at home—detecting falls, tracking activity levels, and predicting health events before they become emergencies.

The Problem: During beta testing, your CEO Maya gets an urgent call: "The system keeps crashing when Grandma forgets to charge her wristband, and it reports 100% confidence right before catastrophic failures. We can't ship this."

Your team identifies three critical issues:

1. **Missing sensors:** Patients forget devices, batteries die, sensors malfunction
2. **Misaligned timing:** Heart rate updates every 2 seconds, accelerometer at 100Hz
3. **Overconfident predictions:** Model says 98% certainty even with garbage inputs

Your Mission: Build a robust multimodal fusion system that *degrades gracefully* when sensors fail, handles asynchronous data streams, and provides *calibrated confidence estimates*.

What You're Actually Submitting

Primary deliverable: A 4-page technical report (report.pdf) analyzing whether this fusion system is ready for production deployment.

Supporting artifacts:

- Code (src/*) that proves you ran the experiments
- JSONs (experiments/*) with machine-readable results
- Plots (analysis/*) that visualize findings
- README with reproduction instructions

Think of this as a technical memo to Maya. She'll read your report to decide whether to ship, then verify your claims by running your code.

This assignment teaches production-grade sensor fusion: the kind NASA uses for Mars rovers, autonomous vehicles use for perception, and wearable companies use for health monitoring.

2 Repository Template

```
a2/
|-- src/
|   |-- fusion.py           # Fusion architectures: early/late/hybrid
|   |-- attention.py        # Cross-modal & temporal attention mechanisms
|   |-- encoders.py         # Modality-specific encoders
|   +-- uncertainty.py      # Uncertainty quantification & calibration
|-- config/
|   |-- base.yaml           # Base configuration
|   |-- fusion_strategies.yaml  # Early/late/hybrid configs
|   +-- datasets.yaml       # Dataset-specific configs
|-- experiments/            # Results JSONs (fusion, missing, uncertainty)
|-- analysis/               # Generated plots (attention, calibration)
|-- tests/                  # Optional unit tests for your implementations
|-- environment.yml
+-- README.md
```

Listing 1: Repository Structure

Expected workflow:

```
conda env create -f environment.yml -n a2 && conda activate a2
pytest -q                                    # Optional: test your
    implementations
# Follow README instructions to train models and generate outputs
```

All grading is manual. The instructor will follow your README to reproduce results.

3 Core Requirements

3.1 Fusion Architectures (35 points)

Implement three strategies for multimodal integration:

- **Early Fusion:** Concatenate encoder features and process jointly.
- **Late Fusion:** Independent per-modality classifiers with learned or average ensembling.
- **Hybrid (Attention-based):** Cross-modal attention, temporal attention, and learnable fusion weights.

Required skeletons:

```
class HybridFusion(nn.Module):
    def __init__(self, modality_dims, hidden_dim, num_heads=4):
        ...
    def forward(self, modality_features, modality_mask=None):
        ...  # Return fused representation + attention weights
```

3.2 Missing Modality Handling (25 points)

Real sensors fail. Your system must run with any subset of modalities (≥ 1) and degrade gracefully.

```
class AdaptiveFusion(nn.Module):
    def forward(self, modality_features, modality_mask):
        ...  # Learn to reweight fusion given availability patterns
```

3.3 Uncertainty Quantification (20 points)

Provide calibrated confidence estimates and incorporate uncertainty into fusion weights.

- Estimate per-modality reliability (e.g., dropout, ensembles, learned calibration).
- **Calibration target:** Aim for ECE ≈ 0.1 under full modalities; discuss deviations.
- Compare equal weighting vs. inverse-uncertainty weighting.

3.4 Temporal Alignment (Included in Fusion)

Handle different sampling rates and asynchronous arrivals. Choose one: interpolation + fixed window; continuous-time modeling; or event-based fusion. Demonstrate handling of at least two rates and compare against naive concatenation.

4 Choose Your Mission

Pick one dataset based on your application domain. **You must use a real dataset.**

4.1 Mission 1: Health Monitoring (PAMAP2) RECOMMENDED

The Story: Maya's hospital deployment needs activity recognition from wearables. You have 3 IMU sensors (hand, chest, ankle) plus heart rate. Can you predict when someone is about to fall *before* they hit the ground?

Download: `https://archive.ics.uci.edu/ml/datasets/pamap2+physical+activity+monitoring`
Challenge: Heart rate sensor is unreliable (0.5-2Hz updates), IMUs are noisy
Win condition: 75-85% accuracy with graceful degradation when sensors fail
Grading advantage: Instructor has reference baseline—you'll know if you're on track

Why This Matters: Real-World Impact

Healthcare: Apple Watch's fall detection uses multimodal fusion (accelerometer + gyroscope + altimeter). Needs 95% precision to avoid false alarms.
Autonomous Vehicles: Waymo fuses camera + lidar + radar + GPS. One sensor failing can't cause a crash.
Space Exploration: Mars rovers fuse visual + LIDAR + IMU. If camera lens gets dusty, navigation must still work.
Your hybrid fusion architecture is the same approach used in all these systems.

4.2 Mission 2: Smart Home Robotics (MHAD)

The Story: Your robot assistant needs to understand human gestures (waving, pointing, sitting) from video + IMU data. Problem: cameras get occluded, IMUs drift.

Download: `http://tele-immersion.citris-uc.org/berkeley_mhad`
Challenge: 3 different sampling rates (30Hz video, 50Hz IMU, 120Hz mocap)
Win condition: Attention mechanism correctly focuses on relevant modalities per action

4.3 Mission 3: Kitchen Safety AI (MPI Cooking)

The Story: Smart kitchen detects dangerous situations (knife handling, hot oil) from audio-visual cues. Audio can be masked by background noise.

Download: `https://www.mpi-inf.mpg.de/departments/computer-vision-and-machine-learning/research/human-activity-recognition/mpii-cooking-2-dataset`
Challenge: Correlate sounds (sizzling, chopping) with visual actions
Win condition: System works even when microphone is noisy or camera blocked

5 Building the System: Implementation Notes

5.1 The Encoder Challenge: Fast Enough to Ship

Maya's constraint: "We need this running on hospital tablets, not GPU clusters."

Your encoders must be **CPU-friendly**:

- **For time-series** (IMU, heart rate): Small 1D CNN/LSTM/GRU (2-3 layers, 64-128 hidden units)
- **For pre-extracted features** (video frames): MLP with attention pooling (2-3 layers, 256-512 hidden)
- **Target:** <5M total parameters, <30 min training time on a laptop

Think: Apple Watch runs fall detection on a tiny ARM chip. Your architecture should be similarly efficient.

5.2 Attention: What Is the Model Looking At?

The debugging story: Your model achieves 90% accuracy, but Maya asks: "How do I know it's not just memorizing noise?"

You need **interpretable attention**:

- **Cross-modal attention:** Which modality dominates for each action? (e.g., video for gestures, IMU for walking)
- **Temporal attention:** Which time steps matter? (e.g., acceleration spike before a fall)
- **Visualization:** Heatmaps showing attention weights over time and modalities

*Production systems need transparency. If the model fails, you need to debug **why** it failed.*

5.3 Training: Making It Reproducible

Maya's nightmare: "Our demo worked yesterday. Today it crashes. What changed?"

Your training must be **deterministic**:

- **Optimizer:** AdamW with cosine schedule ($10^{-3} \rightarrow 10^{-4}$)
- **Batch size:** 32–64 (fits in 8GB RAM)
- **Epochs:** 50–100 with early stopping (patience=10)
- **Augmentation:** Temporal jitter (±5%), light Gaussian noise ($\sigma = 0.01$), random modality dropout (10% prob)
- **Seeds:** `torch.manual_seed(42)` + `numpy.random.seed(42)` — **fixed everywhere**

Reproducibility = trust. If the grader can't reproduce your results, your system isn't production-ready.

6 Proving It Works: Experimental Protocol

Maya's checklist before launch: "Show me it works better than what we have, survives real-world failures, and doesn't hallucinate confidence."

6.1 Experiment 1: Fusion Strategy Shootout

Question: Which fusion strategy wins?

Run all three architectures (early, late, hybrid) on the full dataset with **all modalities present**:

- **Metrics:** Accuracy, macro-F1 (handles class imbalance), inference time per sample
- **Output:** `experiments/fusion_comparison.json` with side-by-side results
- **Expected finding:** Hybrid fusion should outperform early/late by 3–5% (if not, debug your attention!)

Real-world parallel: Tesla's Autopilot uses hybrid fusion to combine camera + radar + ultrasonic. Pure early fusion failed because sensor timings differ.

6.2 Experiment 2: Sensor Failure Stress Test

Question: What happens when sensors fail mid-deployment?

Evaluate **all modality subsets**:

- Full (both/all modalities)
- Single modalities (video-only, IMU-only, etc.)
- Random subsets if >2 modalities
- **Output:** `experiments/missing_modality.json` with degradation curve
- **Target:** <20% accuracy drop when one sensor is missing (graceful degradation)

Real-world parallel: Apple Watch's ECG feature works even when motion sensors are saturated during exercise.

6.3 Experiment 3: Confidence Calibration Audit

Question: When the model says 90% confidence, is it right 90% of the time?

Measure calibration:

- **Expected Calibration Error (ECE):** Average gap between confidence and accuracy (target: <0.1)
- **Reliability diagram:** Plot predicted confidence vs actual accuracy in 10 bins
- **Negative Log-Likelihood (NLL):** Penalizes overconfident wrong predictions
- **Output:** `experiments/uncertainty.json` + `analysis/calibration.png`

Real-world parallel: Medical AI systems must be well-calibrated. A cancer detector that's 60% confident but wrong kills patients.

6.4 Ablation Studies (Pick 2+)

Question: What design choices actually matter?

Test variations:

- Attention vs simple concatenation (does attention help?)
- Number of attention heads (1 vs 4 vs 8)
- Temporal window size (1s vs 3s vs 5s)
- Pre-trained encoders vs random init (transfer learning value?)

*This demonstrates you understand **why** your system works, not just that it does.*

6.5 Baseline Comparisons

Don't claim victory without showing you beat simple alternatives:

- **Single-modality oracle:** Best individual sensor (upper bound for "just use one")

- **Naive concatenation:** Just stack features and feed to classifier (no attention, no learned weighting)

If you can't beat concatenation, your attention mechanism is broken.

7 How You'll Be Judged: Evaluation Metrics

7.1 Primary Metrics (Make or Break)

- **Accuracy:** Overall correctness (but can hide class imbalance issues)
- **Macro-F1:** Averages F1 across classes (catches "model ignores rare classes" bugs)
- **Expected Calibration Error (ECE):** Confidence calibration quality (medical AI standard)

7.2 Secondary Metrics (Nice to Have)

- **Inference time:** Milliseconds per sample on CPU (production constraint)
- **Parameter count:** Total model size (deployment constraint)
- **Training time:** Total minutes to convergence (reproducibility check)

7.3 Robustness Metrics (The Real Test)

- **Graceful degradation:** Accuracy drop when sensors fail (all → video-only → IMU-only)
- **Calibration under stress:** Does ECE stay low when modalities are missing?

*Production systems are judged by **worst-case** performance, not average-case. Your system must survive real-world chaos.*

8 The Logging Contract: What Gets Measured Gets Managed

Why this matters: Maya's engineering team needs standardized outputs to compare your system against competitors. You can't just say "it works"—you need **machine-readable evidence**.

8.1 The Three Required JSONs

Think of these as your system's "test reports" that demonstrate it meets specifications:

8.1.1 1. Fusion Comparison Report

File: `experiments/fusion_comparison.json`
Purpose: Show which fusion strategy wins
Example schema:

```
{
  "dataset": "pamap2",
  "modalities": ["imu_hand", "imu_chest", "heart_rate"],
  "results": {
    "early_fusion":  {"accuracy": 0.867, "f1_macro": 0.854, "ece": 0.082},
    "late_fusion":   {"accuracy": 0.841, "f1_macro": 0.829, "ece": 0.094},
    "hybrid_fusion": {"accuracy": 0.892, "f1_macro": 0.881, "ece": 0.067}
  }
}
```

8.1.2 2. Robustness Stress Test Report

File: `experiments/missing_modality.json`
Purpose: Demonstrate graceful degradation when sensors fail
Example schema:

```
{
  "dataset": "pamap2",
  "modality_subsets": {
    "all":         {"accuracy": 0.892, "f1_macro": 0.881},
    "imu_only":    {"accuracy": 0.801, "f1_macro": 0.787},
    "heart_only": {"accuracy": 0.623, "f1_macro": 0.601}
  }
}
```

8.1.3 3. Calibration Audit Report

File: `experiments/uncertainty.json`
Purpose: Show confidence estimates are trustworthy
Example schema:

```
{
  "dataset": "pamap2",
  "calibration_metrics": {
    "ece": 0.067,
    "nll": 0.312,
    "bins": [0.1, 0.2, ..., 0.9, 1.0],
    "accuracy_per_bin": [0.11, 0.19, ..., 0.91, 0.98]
  }
}
```

The Contract: These JSON files are your system's "API contract" with the grading infrastructure. Fixed paths + fixed schema = automated reproducibility checking.
Think: When SpaceX launches a rocket, they don't submit a PDF—they submit telemetry logs in a standardized format. Same principle here.

9 The Primary Deliverable: Your Technical Report

⚠ Critical: The Report IS the Assignment

This is not a coding assignment with a report appendix.
This is a technical report assignment with code as verification.
Think of it this way:

- **Report (report.pdf):** Your engineering analysis and findings — what you learned, what worked, what didn't
- **Code (src/*):** Evidence that you actually ran the experiments you claim in the report
- **JSONs/plots:** Reproducible artifacts that back up your report claims

Maya doesn't want spaghetti code. She wants a **technical memo** explaining whether this fusion system is ready for production, backed by experimental evidence.

9.1 Report Structure ($\leq$ 4 pages)

9.1.1 1. Introduction (0.5 pages)

- **Motivation:** Why does multimodal fusion matter for your chosen application?
- **Dataset choice:** Which dataset (PAMAP2/MHAD/MPI Cooking) and why?
- **Key challenges:** What makes this problem hard? (missing sensors, timing misalignment, overconfidence)

9.1.2 2. Approach (1 page)

- **Architecture diagram:** Show your early/late/hybrid fusion pipelines visually
- **Attention mechanism:** How does cross-modal attention work? Include equations
- **Temporal alignment:** How do you handle different sampling rates?
- **Uncertainty quantification:** How do you compute calibrated confidence?

Focus on design decisions and tradeoffs, not implementation details.

9.1.3 3. Experiments & Results (1.5 pages)

This section demonstrates your claims. Include:

- **Fusion comparison table:** Accuracy/F1/ECE for early/late/hybrid (from `fusion_comparison.json`)
- **Missing modality plot:** Degradation curve showing performance vs available sensors (from `missing_modality.json`)
- **Attention visualization:** Heatmap showing which modalities/timesteps the model attends to (from `attention_viz.png`)
- **Calibration diagram:** Reliability plot showing confidence vs accuracy (from `calibration.png`)
- **Ablation study:** What happens when you remove attention? Change window size?
- **Baseline comparison:** How much better than single-modality or naive concatenation?

9.1.4 4. Analysis & Discussion (0.75 pages)

This is where you earn the "Analysis insights" 15 points. Answer:

- **Which fusion strategy wins and why?** (Don't just report numbers—explain the mechanism)
- **How does performance degrade?** (Graceful or catastrophic? Why?)
- **Is the model well-calibrated?** (ECE interpretation; when does it overpredict confidence?)
- **What do attention patterns reveal?** (Does the model learn sensible cross-modal dependencies?)
- **Limitations:** What didn't work? What would you do differently with more time/data?

Maya's question: "Should we deploy this?" Your analysis should help her decide.

9.1.5 5. Conclusion (0.25 pages)

- **Key findings:** 2-3 sentences summarizing your main results
- **Practical guidance:** When should practitioners use early vs late vs hybrid fusion?
- **Future work:** What would improve this system?

10 How You're Graded: Report-First Rubric (100 points)

Grading Philosophy: Report First, Code Second

The instructor reads your report first, then checks code to verify claims.
Strong report + working code = A
Strong report + buggy code = B (you understand concepts, implementation needs work)
Weak report + perfect code = C (you can implement but don't understand why)
Weak report + buggy code = D/F

Component	Report	Code	What is checked	Pts
Report Content (60 pts total)				
Analysis & insights	✓✓	—	Thoughtful discussion; limitations; design justification	20
Experimental results	✓✓	—	Tables/plots present; findings discussed; ablations	15
Approach explanation	✓✓	—	Architecture clear; attention/uncertainty explained	15
Report structure	✓	—	≤4 pages; sections present; well-written	10
Code & Artifacts (40 pts total)				
Fusion implementation	✓	✓✓	JSONs match report; code review shows correctness	15
Attention mechanism	✓	✓✓	Visualization matches report; code implements correctly	10
Missing modality handling	✓	✓	JSON has subsets; code handles gracefully	8
Uncertainty quantification	✓	✓	ECE computed correctly; calibration plot accurate	7
Total	✓✓ = primary grade source; ✓ = verification/presence check			**100**

10.1 What This Means in Practice

Scenario 1: Strong report, code has a bug

- Report clearly explains fusion strategies, discusses attention patterns, analyzes results thoughtfully
- Code has a tensor shape mismatch that breaks training
- **Grade impact:** Lose 5-10 points for "code doesn't run," keep most report points
- **Why:** You understand the concepts; the bug is fixable in production

Scenario 2: Weak report, perfect code

- Code runs flawlessly, all tests pass, metrics look good
- Report just lists numbers with no analysis: "Hybrid fusion got 89%. Late fusion got 84%."
- **Grade impact:** Lose 20+ points for missing "Analysis & insights" and weak "Experimental results"

- **Why:** You're a code monkey, not an engineer who understands trade-offs

Scenario 3: Both strong

- Report explains design decisions, analyzes results, discusses limitations
- Code runs, generates correct outputs, is well-documented
- **Grade:** 95-100 (full credit, maybe lose a few for minor polish issues)

See `GRADING_CHECKLIST.md` for the detailed evaluation rubric used during manual grading.

11 Hardware Notes

CPU-only completion is required; GPU use is optional. Amarel cluster usage is optional if you need longer jobs. No SLURM script is provided for this assignment.

12 Your 2-Week Sprint

Sprint Planning: 14 days

Sprint 1 (Days 1-5): Foundation
- Download PAMAP2, explore data
- Single-modality baseline (verify you can train *something*)
- Document setup in README (grader needs this!)

Sprint 2 (Days 6-10): Core Features
- Implement hybrid fusion with attention
- Run experiments, generate 3 JSONs + 3 plots
- Test missing sensor scenarios

Sprint 3 (Days 11-14): Polish & Ship
- Uncertainty calibration (ECE target: 0.1)
- Write report (4 pages max)
- Final README review: can grader reproduce in 30 min?

Pro tip: Maya's team ships on Friday. Don't wait until Thursday night.

13 The Deliverables: What Ships to Production

Critical: This README is Your Product Documentation

In the real world, poor documentation = unshippable product. Your README is the **primary grading artifact**. If the instructor can't run your code in <30 minutes, your "product" doesn't ship (= major point deductions).

13.1 What Maya Needs to See (maps directly to grading rubric)

1. Setup Instructions ✓ (5 pts - Code & documentation)
- Exact conda/pip commands to create environment
- Dataset download OR generation command
- Verification command (e.g., `pytest -q`)
- Expected output: "Setup complete, environment activated"

2. Training Command (maps to 20+15+15 pts - Fusion/Attention/Missing modality)
- Single command to train all fusion strategies
- Expected runtime on your hardware (e.g., "30 min on M1 Mac")
- Config file used (e.g., `python src/train.py --config config/base.yaml`)
- Expected output: Checkpoint saved, loss/accuracy logged

3. Evaluation Command (maps to 10+10 pts - Experimental completeness/Schema)
- Command to generate **all** required outputs (3 JSONs + 3 plots)
- Example: `python src/eval.py --checkpoint runs/best.ckpt`
- Expected output: Files appear in `experiments/` and `analysis/`

4. Uncertainty Analysis Command (maps to 10 pts - Uncertainty quantification)
- Command to run calibration analysis
- Expected output: `experiments/uncertainty.json` with ECE value

5. Results Summary (maps to 15 pts - Analysis quality)
- Dataset used and why
- Best fusion strategy and accuracy achieved
- Training time and hardware specs
- Key finding (1-2 sentences linking to report)

6. File Manifest Checklist
- List showing all required outputs are present
- Helps grader verify completeness quickly

Grading impact: Missing/unclear documentation costs points in corresponding components. See `template/README_TEMPLATE.md` for required structure.

14 Manual Grading Workflow

Since there is no autograder, the instructor will grade by running your code following your README:

14.1 Grading Steps (your README must support each step)

Step 1: Setup (5 min) → Code quality points
- Clone repo, run your setup commands
- If fails: -5 pts + cannot grade technical components

Step 2: Training (15 min) → Implementation points (50 pts total)
- Run your training command
- Verify: (1) fusion strategies trained, (2) attention mechanism works, (3) missing modality handling present
- If crashes: -10 pts per broken component

Step 3: Evaluation (5 min) → Experimental completeness (10 pts)
- Run your evaluation command
- Check 3 JSONs + 3 plots appear in correct locations
- Missing file: -5 pts per artifact

Step 4: Artifact Validation (10 min) → Schema compliance (5 pts)
- Open JSONs, verify required fields and reasonable values
- Open plots, check readability and labels
- Schema error: -2 pts per issue

Step 5: Report Review (15 min) → Analysis quality (15 pts)
- Read `report.pdf`, check insights and discussion

- Cross-reference with JSON results

Step 6: Code Review (10 min) → Code quality (5 pts)

- Check docstrings, config usage, no hardcoded paths

Required outputs (fixed paths - DO NOT CHANGE):

- `experiments/fusion_comparison.json` (20 pts component)
- `experiments/missing_modality.json` (15 pts component)
- `experiments/uncertainty.json` (10 pts component)
- `analysis/fusion_comparison.png` (5 pts component)
- `analysis/attention_viz.png` (15 pts component)
- `analysis/calibration.png` (10 pts component)
- `report.pdf` (root, ≤4 pages, 20 pts total)

Your documentation quality directly affects your grade. Clear README = fast grading = more time spent evaluating your work, not debugging setup.

15 Shipping to Production

Launch Checklist

Before you tag `submission` and "deploy to production":

Does it work?

- ☐ Run through your README from scratch (fresh conda env)
- ☐ All 7 files exist: 3 JSONs + 3 PNGs + 1 PDF
- ☐ pytest passes (or failures documented)

Is it production-ready?

- ☐ No hardcoded paths (use config/base.yaml)
- ☐ Handles missing sensors gracefully (doesn't crash)
- ☐ Confidence estimates are calibrated (ECE discussed)
- ☐ All three fusion strategies implemented
- ☐ Attention visualization shows what model "sees"

Can someone else deploy it?

- ☐ README has exact commands
- ☐ Expected runtime documented
- ☐ Hardware specs listed
- ☐ Reproduction takes <30 min following README

Final step:

```
git tag submission
git push origin main --tags
# Upload report.pdf to Canvas
# Send Maya a Slack: "Multimodal fusion system shipped!"
```

You're building production-grade sensor fusion—the kind that saves lives in hospitals, prevents accidents in autonomous vehicles, and enables robots to navigate Mars. Make it robust. Make it reliable. Ship it.