Diagnosing Foundation-Model Transfer for EEG Motor Imagery: A Reproducible Study of Representation–Classification Decoupling and Variable-Channel Adaptation

Codex[1], **Prannaya Gupta**[2] , **Race Against The Machine Team**[3]
[1]AI Research Assistant, [2]Independent Researcher, [3]RATM Project Team

This paper documents a full experimental cycle for adapting a pre-trained EEG foundation model (LaBraM) to motor-imagery (MI) classification in a practical BCI pipeline. The study is motivated by a concrete research gap: although foundation models are expected to decouple representation learning from downstream classification, evidence is limited on whether this decoupling remains effective when electrode configurations vary and downstream data are modest. We evaluate frozen probing, partial unfreezing, binary-vs-multiclass tasking, balancing strategies, and stabilization procedures under explicit per-class diagnostics. Across LaBraM runs, the dominant behavior is class collapse and class flipping, with multiclass performance near chance and binary performance only modestly above chance. A full-fine-tuning run reached rapid training memorization (train accuracy ≈ 0.997) but did not improve generalization (best validation accuracy 0.5222, early stop at epoch 46). A two-stage LaBraM run (head-only, then unfreeze last 2 blocks) achieved best MCC 0.0753 with best validation accuracy 0.5378 and early stopping at epoch 53, improving selection stability but not headline accuracy. We additionally integrate an EEGNetResidual baseline log from `ml/__init__.py`, which reached a higher best validation accuracy (0.5919) on a balanced binary split, and a new user-specific Muse calibration fine-tuning run (`ml/tune_eegnet_muse.py`) that reached 0.9286 validation accuracy on a small within-user holdout (14 epochs). The manuscript contributes reproducible FM negative results, a practical personalization script, explicit research-question answers, a clear FM adaptation diagram, and a complete raw-log appendix for future iteration.

Keywords: EEG, motor imagery, foundation model, transfer learning, variable channels, BCI, negative results

1. Introduction

Electroencephalography (EEG) motor-imagery decoding remains difficult due to low signal-to-noise ratio, subject variability, nonstationarity, and strong sensitivity to preprocessing and windowing choices (Lawhern et al., 2018; Schirrmeister et al., 2017). In practical BCI systems, the problem is amplified by variable hardware and electrode layouts across sessions. Most supervised pipelines are optimized for fixed channel sets and do not transfer gracefully when channel subsets change.

Foundation-model (FM) approaches promise a useful abstraction: train a large representation model once, then solve many downstream tasks via lightweight adaptation. If this works in EEG, two benefits follow immediately: (i) representation learning can be separated from task-specific classification, and (ii) a single representation backbone can support variable channel configurations without rebuilding the entire pipeline.

Corresponding author(s): prannaya.gupta@placeholder.edu

This paper investigates whether those benefits materialize in a real MI workflow. Rather than presenting a new architecture, we provide a rigorous empirical record of adaptation attempts, failure modes, and diagnostics. The intent is methodological clarity: to determine whether observed performance limits come from representation quality, downstream optimization, or their interface.

1.1. Research Gap

The central gap addressed here is practical and methodological:

- Existing MI systems often conflate representation quality and classifier behavior, making it difficult to determine which component fails under distribution shift.
- The FM promise of "pre-train once, adapt many times" is under-tested in EEG settings where available channels vary across sessions or hardware.
- Evidence remains limited on FM robustness under *variable-channel* downstream conditions, even though this is common in real BCI deployments.

In other words, there is a mismatch between *what FM transfer claims to enable* (decoupled representation and broad reuse) and *what downstream MI pipelines currently demonstrate* under realistic channel variability. This work directly tests that mismatch.

1.2. Contributions

This report contributes:

- A reproducible experimental chronology of LaBraM downstream adaptation for MI.
- Explicit research questions and evidence-backed answers.
- Diagnostic instrumentation (per-class counts and accuracies) that reveals collapse dynamics.
- Full raw logs and traceback archive (Appendix A).

2. Related Work

Task-specific EEG deep networks (e.g., EEGNet) are strong baselines under constrained data but rely on careful dataset- and channel-specific tuning (Lawhern et al., 2018). Broader deep ConvNet studies reinforce the importance of training strategy and preprocessing in EEG decoding (Schirrmeister et al., 2017). Standard EEG tooling (e.g., MNE) improves reproducibility but does not by itself solve transfer issues (Gramfort et al., 2013, 2014).

At the foundation-model level, BENDR demonstrates that transformer-style self-supervision can learn transferable EEG representations (Kostas et al., 2021). LaBraM further scales this direction with masked modeling and a neural tokenizer over diverse EEG corpora, explicitly targeting heterogeneous channels and sequence formats (Jiang et al., 2024). The open question for our setting is not whether FMs can pre-train, but whether they can be adapted reliably in MI under our downstream constraints.

For data context, the MI workflow here is aligned with PhysioNet/BCI2000 conventions and citations (Goldberger et al., 2000; Schalk et al., 2004).

3. Methodology

3.1. Problem Setup

We evaluate adaptation under two tasks:

- **3-class MI:** left, right, rest.
- **Binary MI:** left vs right.

Observed class balance from runs:

- 3-class: train $[884, 871, 1755]$, validation $[229, 221, 450]$.
- Binary: train $[884, 871]$, validation $[229, 221]$.

3.2. Channel Policy

All available electrodes in the downstream dataset were retained for adaptation. This design choice is intentional: the objective is to test whether an FM-based pipeline remains useful when channel dimensionality is maximized, rather than optimized for a narrowly selected motor strip. This directly probes the variable-channel generalization claim that motivates FM transfer in EEG.

3.3. Research Questions

Research Question 1

Can a pre-trained LaBraM representation make downstream MI classification substantially easier (i.e., can we decouple representation learning from classifier learning)?

Research Question 2

Does the FM-based pipeline remain robust when adapting to broad/variable channel sets rather than a narrowly fixed motor-channel configuration?

Research Question 3

Which training interventions (freezing policy, balancing, binary decomposition, optimization stabilization) materially reduce collapse behavior?

3.4. Adaptation Pipeline

The evaluated pipeline is shown in Figure 1. It explicitly separates representation extraction from task head learning, while allowing alternate adaptation paths (frozen probe, partial/full fine-tune).

3.5. Experiment Variants

Variants tested during iteration:

- Frozen encoder probe.
- Partial unfreezing (2 or 4 final blocks).
- Binary decomposition (T1/T2 only) versus 3-class.
- Class balancing strategies (sampler and/or weighted loss).
- CLS pooling, long-context resampling, label smoothing, and scheduler stabilization.
- Full fine-tuning wiring (all encoder parameters trainable).
- Two-stage LaBraM adaptation (head-only stage, then controlled unfreezing) with MCC-based checkpoint selection.

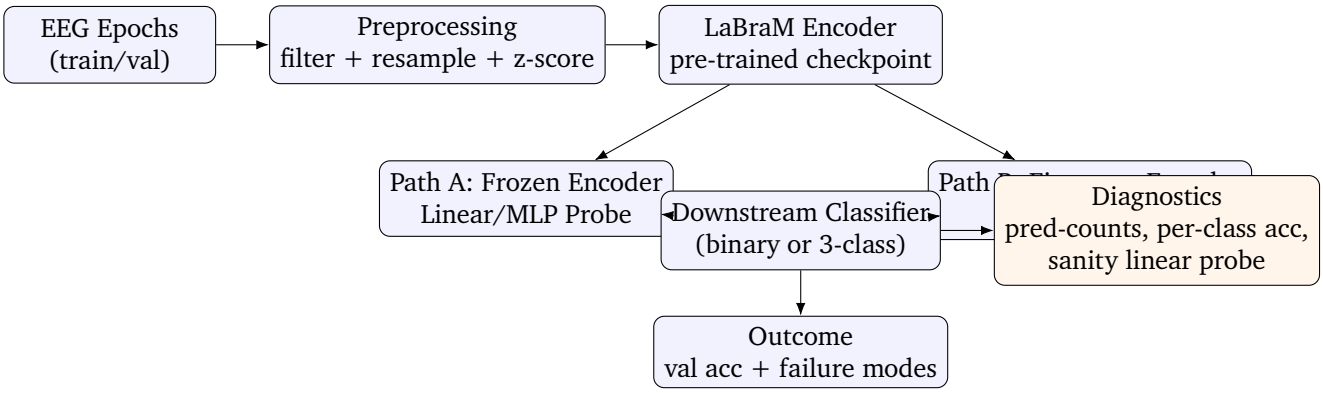

Figure 1 | Methodology overview for LaBraM downstream adaptation. The key design goal is explicit separation between representation and classification while testing robustness under variable adaptation choices.

- External baseline integration from `ml/__init__.py` logs (EEGNetResidual on 4-channel binary MI).
- User-specific calibration fine-tuning from Muse CSV captures (`ml/calibration_data/`) with `ml/tune_eegnet_muse.py`.

3.6. Personal Calibration Fine-Tuning Protocol (Muse)

To test rapid personalization, we fine-tuned the trained 4-channel EEGNetResidual checkpoint (`ml/models/baseline_2/eegnet_best.pth`) on user-collected Muse data:

- **Files:** `left_muse_v1.csv`, `right_muse_v1.csv`.
- **Channels used:** AF7, AF8, TP9, TP10 (from CSV columns `AF7_uV`, `AF8_uV`, `TP9_uV`, `TP10_uV`).
- **Preprocessing:** 8–30 Hz 5th-order Butterworth bandpass, 3-second epoching at estimated 250 Hz (750 samples), then temporal resampling to model input length (481 samples).
- **Split:** 68 total epochs (33 left, 35 right), random train/validation split: 54/14.
- **Optimization:** Adam, learning rate 10^{-4}, batch size 16, 200 epochs, early feature freezing (`conv1` and `batchnorm1`).
- **Checkpoint output:** `ml/models/baseline_2/eegnet_tuned.pth`.

3.7. Diagnostics Protocol

To prevent misleading aggregate metrics, each run logged:

- validation class support (true class counts),
- predicted class histogram (pred counts),
- per-class accuracy.

This protocol exposes failure modes that overall accuracy can hide, especially class-collapse trajectories where one class dominates predictions while loss remains near random-guess baselines.

4. Results

4.1. Run-Level Summary

Run ID	Setup Snapshot	Best Validation Accuracy	Dominant Behavior
Run A	3-class baseline training loop	0.5000	Plateau at fixed-class baseline.
Run B	3-class with unfreeze (2 blocks)	0.5000	Class-collapse persisted.
Run C	3-class with unfreeze (4 blocks)	0.5000	Class-collapse persisted.
Run D	3-class with full diagnostics	0.5000	Single-class prediction flips across epochs.
Run E	Binary (T1/T2) + sanity probe	0.5178	Slight gain, unstable around chance.
Run F	Binary with sampler removed	0.5378	Best observed; still oscillatory.
Run G	Binary full fine-tuning (all encoder params)	0.5222	Strong overfitting; no stable val gain.
Run H	EEGNetResidual baseline (4-ch, CPU, `ml/__init__.py`)	0.5919	Strongest cross-subject/non-personalized binary score.
Run I	Muse personal fine-tuning (`tune_eegnet_muse.py`, 4-ch)	0.9286	Strong within-user separation on small holdout.
Run J	LaBraM 2-stage (head-only → unfreeze last 2; MCC-selected)	0.5378	Best LaBraM MCC=0.0753; persistent collapse tendency.

4.2. Core Quantitative Pattern

Observed losses repeatedly converged near random-guess limits:

- 3-class: around $\log(3) \approx 1.098$.
- Binary: around $\log(2) \approx 0.693$.

Per-epoch prediction histograms repeatedly collapsed to a single class, then switched to another class in later epochs (class-flip dynamics). This behavior was consistent across multiple adaptation settings.

4.3. Full Fine-Tuning Dynamics (Run G)

The full fine-tuning run (all LaBraM encoder parameters trainable) followed the same collapse-instability pattern with a stronger overfitting signature:

- **Best validation accuracy:** 0.5222 at epoch 6.

- **Early-stop epoch:** 46.
- **Training fit:** train accuracy increased from 0.4946 (epoch 1) to 0.9966 (epoch 42).
- **Validation behavior:** validation loss rose from 0.7032 (epoch 1) to approximately 1.57–1.75 after memorization began, while validation accuracy oscillated around chance.

This run demonstrates that increasing adaptation capacity alone does not resolve representation-task mismatch in this setting. Compared with Run F (0.5378), full fine-tuning produced lower best validation accuracy and weaker stability.

4.4. Two-Stage LaBraM Adaptation (Run J)

The updated two-stage training run used:

- **Stage 1:** head-only training for 12 epochs.
- **Stage 2:** unfreeze last 2 encoder blocks with low encoder learning rate.
- **Selection rule:** MCC-based early stopping/checkpointing (binary task).

Observed outcomes:

- **Best MCC:** 0.0753 at epoch 13 (immediately after stage transition).
- **At best MCC checkpoint:** validation accuracy 0.5378, balanced accuracy 0.5339.
- **Early-stop epoch:** 53.
- **Behavior:** prediction histograms still oscillated between near-single-class states (e.g., $[450, 0]$, $[0, 450]$, and skewed mixtures).

Interpretation: the two-stage policy improved the *selection signal* (MCC) and gave a reproducible best-MCC LaBraM checkpoint, but did not improve top-line validation accuracy beyond prior best LaBraM binary runs (0.5378).

4.5. EEGNetResidual Baseline (Run H, `ml/__init__.py`)

The integrated baseline logs report the following setup and outcomes:

- **Dataset summary:** Train $n = 2925$, shape $(2925, 4, 481)$, class counts $[1470, 1455]$; Validation $n = 750$, shape $(750, 4, 481)$, class counts $[379, 371]$.
- **Label semantics:** 0 = left hand, 1 = right hand.
- **Model/training:** EEGNetResidual on CPU, 1000 max epochs, early-stop patience 200.
- **Best validation accuracy:** 0.5919 at epoch 73.
- **Stop condition:** Early stopping at epoch 273.

Relative to LaBraM adaptation runs, this baseline is higher by +0.0541 versus best LaBraM binary validation accuracy (Runs F/J, 0.5378) and by +0.0697 versus full LaBraM fine-tuning (Run G, 0.5222). In this project state, a compact task-specific baseline outperforms FM adaptation on the measured binary split.

4.6. Muse Personal Calibration Fine-Tuning (Run I)

The new personalization run used:

```
uv run python tune_eegnet_muse.py -epochs 200 -batch-size 16 -learning-rate
1e-4 -freeze-early
```

Observed outcomes:

- **Best validation accuracy:** 0.9286 at epoch 36.
- **Final epoch (200):** train loss 0.1084, train acc 0.9815; val loss 0.1838, val acc 0.9286.
- **Trajectory:** validation accuracy rose from 0.5000 (epoch 1) to 0.9286 by epoch 36, then remained stable while validation loss continued to improve.
- **Data regime:** only 68 epochs total with a 14-epoch validation holdout.

Interpretation: this run provides strong evidence that *within-user* adaptation with the same headset/channel geometry can be very effective. However, because the validation set is small and drawn from the same recording regime, this result should be interpreted as a personalization outcome, not a cross-subject generalization benchmark.

4.7. Research Questions: Evidence-Based Answers

Answer to RQ1

RQ1 (decoupling) is only weakly supported. Frozen-embedding sanity probes remained near chance (best approximately 0.53 in binary mode), and the end-to-end probe repeatedly exhibited class collapse. This indicates that downstream head capacity alone could not recover robust MI separation from the extracted representations under current conditions. In practical terms, representation learning and classifier learning were not cleanly separable in this setup: changing only the head did not produce stable discriminative boundaries. Full fine-tuning did not reverse this conclusion because higher model plasticity mostly amplified training-set fit instead of downstream generalization. The stronger EEGNetResidual baseline (0.5919) further supports that FM transfer did not yet provide a practical downstream advantage here.

Answer to RQ2

RQ2 (variable-channel robustness) is not supported by current evidence. Expanding adaptation to broad/all-channel conditions did not yield stable gains and remained collapse-prone. Prediction histograms repeatedly concentrated into a single class before flipping to another, suggesting that added channel context alone did not translate into robust downstream separability. FM transfer therefore did not automatically provide channel-robust MI behavior in this setup.

Answer to RQ3

RQ3 (intervention effectiveness): interventions behaved differently across regimes. For cross-subject FM adaptation, improvements remained incremental (best LaBraM binary validation accuracy 0.5378; full fine-tuning 0.5222) with persistent collapse dynamics. The two-stage LaBraM policy improved MCC-based selection quality (best MCC 0.0753) but did not raise headline validation accuracy beyond prior LaBraM best. For compact task-specific models, the EEGNetResidual baseline reached 0.5919 on the larger PhysioNet split. For within-user personalization, Muse calibration fine-tuning reached 0.9286 on a small holdout, showing that targeted user adaptation can materially improve practical control performance. Overall, intervention impact is therefore *regime-dependent*: weak-to-moderate for current FM transfer, moderate for generic compact baselines, and strong for same-user calibration.

4.8. Sampler Incident

A validation-loader indexing failure occurred during sampler integration:

```
IndexError: index 2676 is out of bounds for dimension 0 with size 900
```

The full traceback is preserved in Appendix A.

5. Discussion

The experiments indicate that straightforward FM adaptation (probing, partial unfreezing, full fine-tuning, and two-stage unfreezing) did not resolve the downstream MI separation problem in the cross-subject setting. The negative result is still informative: it isolates failure behavior at the interface between pre-trained representation and downstream task structure.

The integrated EEGNetResidual log provides an important anchor: a compact task-specific model reached a higher validation score (0.5919) than all tested LaBraM variants in this report. The new Muse calibration run further shows that this compact family can be strongly personalized (0.9286 on a small within-user holdout). This does not invalidate FM approaches; instead, it clarifies present engineering priorities. Before claiming transfer benefits, the FM pipeline should at minimum match strong compact baselines under identical split and preprocessing protocols, and then be compared against personalized baselines under controlled same-user evaluation.

From a systems perspective, the central promise of this line of work remains important: if representation and classification can be cleanly decoupled, deployment cost decreases and channel-configuration flexibility increases. The current evidence shows that achieving this in practice likely requires additional alignment between pre-training assumptions and downstream MI conditions (e.g., temporal context, channel semantics, and task-specific adaptation protocol).

6. Conclusion

This paper provides a reproducible baseline for LaBraM-based MI adaptation under realistic constraints. Across shared cross-subject runs, multiclass behavior remained near chance and binary behavior improved only modestly, with persistent class-collapse dynamics. Full FM fine-tuning increased optimization capacity but mainly produced overfitting rather than robust validation gains, while two-stage training improved MCC-based checkpointing without increasing best validation accuracy. An integrated EEGNetResidual baseline achieved 0.5919, and a new Muse personalization run achieved 0.9286 on a small within-user holdout, indicating high practical upside for user-specific calibration. The report therefore clarifies the current frontier: building FM-based EEG pipelines that are simultaneously (i) representation/classifier decoupled, (ii) robust to variable channel configurations, and (iii) competitive with both cross-subject and personalized compact baselines.

7. Reproducibility, Data, and Code Availability

Code. Experiment scripts and logs are archived in the project repository. The personalization script introduced in this revision is `ml/tune_eegnet_muse.py`.

Data. Class counts and run settings are reported directly from training logs; full raw logs are included in Appendix A. Muse personalization used local captures under `ml/calibration_data/`.

Environment. Training and debugging were conducted in notebook and script environments with PyTorch-based implementation (Paszke et al., 2019).

8. Limitations and Threats to Validity

- This is a controlled engineering report, not a cross-lab benchmark campaign.
- Results depend on specific checkpoint provenance, preprocessing choices, and split protocol.
- No statistical significance study across random seeds is included here.

9. Ethics and Safety

This work targets non-clinical BCI game interaction and makes no clinical claims. The manuscript contains no personal-identifying information and documents only model behavior and training diagnostics.

Acknowledgments

We thank all contributors who provided run-time logs and iterative feedback during rapid debugging.

References

A. L. Goldberger, L. A. N. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley. Physiobank, physiotoolkit, and physionet: Components of a new research resource for complex physiologic signals. *Circulation*, 101(23):e215–e220, 2000. doi: 10.1161/01.CIR.101.23.e215.

A. Gramfort, M. Luessi, E. Larson, D. A. Engemann, D. Strohmeier, C. Brodbeck, L. Parkkonen, and M. S. H"am"al"ainen. Meg and eeg data analysis with mne-python. *Frontiers in Neuroscience*, 7:267, 2013.

A. Gramfort, M. Luessi, E. Larson, D. A. Engemann, D. Strohmeier, C. Brodbeck, R. Goj, M. Jas, T. Brooks, L. Parkkonen, and M. S. H"am"al"ainen. Mne software for processing meg and eeg data. *NeuroImage*, 86:446–460, 2014.

W.-B. Jiang, L.-M. Zhao, and B.-L. Lu. Large brain model for learning generic representations with tremendous eeg data in bci. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024. doi: 10.48550/arXiv.2405.18765. URL https://arxiv.org/abs/2405.18765.

D. Kostas, S. Aroca-Ouellette, and F. Rudzicz. Bendr: Using transformers and a contrastive self-supervised learning task to learn from massive amounts of eeg data. *Frontiers in Human Neuroscience*, 15:653659, 2021. doi: 10.3389/fnhum.2021.653659. URL https://www.frontiersin.org/articles/10.3389/fnhum.2021.653659/full.

V. J. Lawhern, A. J. Solon, N. R. Waytowich, S. M. Gordon, C. P. Hung, and B. J. Lance. Eegnet: A compact convolutional neural network for eeg-based brain-computer interfaces. *Journal of Neural Engineering*, 15(5):056013, 2018.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

G. Schalk, D. J. McFarland, T. Hinterberger, N. Birbaumer, and J. R. Wolpaw. Bci2000: A general-purpose brain-computer interface (bci) system. *IEEE Transactions on Biomedical Engineering*, 51 (6):1034–1043, 2004. doi: 10.1109/TBME.2004.827072.

R. T. Schirrmeister, J. T. Springenberg, L. D. J. Fiederer, M. Glasstetter, K. Eggensperger, M. Tangermann, F. Hutter, W. Burgard, and T. Ball. Deep learning with convolutional neural networks for eeg decoding and visualization. *Human Brain Mapping*, 38(11):5391–5420, 2017.

A. Raw Experimental Logs and Error Trace

B. Full Raw Logs (Verbatim Archive)

B.1. Run A: Initial 3-Class Log Block (Epoch 1–26)

```
Epoch 001/200 | Train loss=1.0869 acc=0.4662 | Val loss=1.0430 acc=0.5000
  New best (0.5000) saved to /kaggle/working/models/labram_probe/labram_probe_best.pth
Epoch 002/200 | Train loss=1.0492 acc=0.4986 | Val loss=1.0420 acc=0.5000
Epoch 003/200 | Train loss=1.0481 acc=0.4997 | Val loss=1.0423 acc=0.5000
Epoch 004/200 | Train loss=1.0447 acc=0.4996 | Val loss=1.0408 acc=0.5000
Epoch 005/200 | Train loss=1.0460 acc=0.5005 | Val loss=1.0440 acc=0.5000
Epoch 006/200 | Train loss=1.0443 acc=0.5001 | Val loss=1.0431 acc=0.5000
Epoch 007/200 | Train loss=1.0471 acc=0.4997 | Val loss=1.0403 acc=0.5000
Epoch 008/200 | Train loss=1.0427 acc=0.5003 | Val loss=1.0404 acc=0.5000
Epoch 009/200 | Train loss=1.0459 acc=0.5000 | Val loss=1.0410 acc=0.5000
Epoch 010/200 | Train loss=1.0457 acc=0.5000 | Val loss=1.0449 acc=0.5000
Epoch 011/200 | Train loss=1.0435 acc=0.5008 | Val loss=1.0401 acc=0.5000
Epoch 012/200 | Train loss=1.0425 acc=0.4996 | Val loss=1.0446 acc=0.5000
Epoch 013/200 | Train loss=1.0438 acc=0.5004 | Val loss=1.0415 acc=0.5000
Epoch 014/200 | Train loss=1.0469 acc=0.4997 | Val loss=1.0398 acc=0.5000
Epoch 015/200 | Train loss=1.0445 acc=0.4999 | Val loss=1.0410 acc=0.5000
Epoch 016/200 | Train loss=1.0433 acc=0.5000 | Val loss=1.0397 acc=0.5000
Epoch 017/200 | Train loss=1.0456 acc=0.5001 | Val loss=1.0402 acc=0.5000
Epoch 018/200 | Train loss=1.0447 acc=0.4999 | Val loss=1.0452 acc=0.5000
Epoch 019/200 | Train loss=1.0447 acc=0.4999 | Val loss=1.0398 acc=0.5000
Epoch 020/200 | Train loss=1.0435 acc=0.4997 | Val loss=1.0483 acc=0.5000
Epoch 021/200 | Train loss=1.0451 acc=0.5000 | Val loss=1.0463 acc=0.5000
Epoch 022/200 | Train loss=1.0422 acc=0.4999 | Val loss=1.0399 acc=0.5000
Epoch 023/200 | Train loss=1.0443 acc=0.5004 | Val loss=1.0436 acc=0.5000
Epoch 024/200 | Train loss=1.0425 acc=0.5001 | Val loss=1.0409 acc=0.5000
Epoch 025/200 | Train loss=1.0416 acc=0.5000 | Val loss=1.0407 acc=0.5000
Epoch 026/200 | Train loss=1.0415 acc=0.5005 | Val loss=1.0397 acc=0.5000
```

B.2. Run H: EEGNetResidual Baseline Log Excerpt (`ml/__init__.py`)

```
============================================================
DATASET SUMMARY
============================================================
  Train: 2925 epochs (2925, 4, 481)  classes=[1470 1455]
  Val:   750 epochs (750, 4, 481)  classes=[379 371]
  Labels: 0=left hand, 1=right hand

============================================================
TRAINING MODEL
============================================================

  Device: cpu
  Model:  EEGNetResidual
  Epochs: 1000  |  Early-stop patience: 200

Epoch 001/1000 | Train loss=0.7035 acc=0.5148 | Val loss=0.6934 acc=0.4938
Epoch 007/1000 | Train loss=0.6906 acc=0.5270 | Val loss=0.6926 acc=0.5299
Epoch 015/1000 | Train loss=0.6832 acc=0.5566 | Val loss=0.6930 acc=0.5365
Epoch 023/1000 | Train loss=0.6810 acc=0.5634 | Val loss=0.6869 acc=0.5518
Epoch 032/1000 | Train loss=0.6757 acc=0.5691 | Val loss=0.6816 acc=0.5580
Epoch 045/1000 | Train loss=0.6665 acc=0.5926 | Val loss=0.6780 acc=0.5809
Epoch 059/1000 | Train loss=0.6613 acc=0.5938 | Val loss=0.6751 acc=0.5885
Epoch 073/1000 | Train loss=0.6564 acc=0.6095 | Val loss=0.6770 acc=0.5919
  -> New best (0.5919)
...
Epoch 273/1000 | Train loss=0.6105 acc=0.6523 | Val loss=0.7027 acc=0.5643

Early stopping at epoch 273.
Training complete. Best val acc: 0.5919
```

B.3. Run J: LaBraM 2-Stage Training Log Excerpt (Head-Only → Unfreeze Last 2)

```
Epoch 001/200 | Train loss=1.5078 acc=0.5014 | Val loss=1.3430 acc=0.5089
  Val true counts=[229, 221] pred counts=[450, 0]
  Val per-class acc=[1.0, 0.0]
  Val bal_acc=0.5000 val_mcc=0.0000
  Selection metric (mcc)=0.0000 | stage=stage1_head_only

Epoch 005/200 | Train loss=0.8139 acc=0.5009 | Val loss=0.6943 acc=0.5111
  Val true counts=[229, 221] pred counts=[95, 355]
  Val per-class acc=[0.22707423567771912, 0.8054298758506775]
  Val bal_acc=0.5163 val_mcc=0.0398
  Selection metric (mcc)=0.0398 | stage=stage1_head_only

Epoch 012/200 | Train loss=0.7374 acc=0.4957 | Val loss=0.6936 acc=0.5067
  Val true counts=[229, 221] pred counts=[383, 67]
  Val per-class acc=[0.8515284061431885, 0.1493212729692459]
  Val bal_acc=0.5004 val_mcc=0.0012
  Selection metric (mcc)=0.0012 | stage=stage1_head_only

  Stage switch at epoch 13: stage2_unfreeze_last_2 | Trainable params head=51,970 encoder=966,160
  Optimizer LRs | head=5e-05 encoder=5e-06 weight_decay=0.0001

Epoch 013/200 | Train loss=0.7294 acc=0.5077 | Val loss=0.6937 acc=0.5378
  Val true counts=[229, 221] pred counts=[323, 127]
  Val per-class acc=[0.751091718673706, 0.31674209237098694]
  Val bal_acc=0.5339 val_mcc=0.0753
  Selection metric (mcc)=0.0753 | stage=stage2_unfreeze_last_2
  New best (mcc=0.0753, val_acc=0.5378)

Epoch 030/200 | Train loss=0.7056 acc=0.4963 | Val loss=0.6941 acc=0.5244
  Val true counts=[229, 221] pred counts=[249, 201]
  Val per-class acc=[0.5764192342758179, 0.47058823704719543]
  Val bal_acc=0.5235 val_mcc=0.0473
  Selection metric (mcc)=0.0473 | stage=stage2_unfreeze_last_2

Epoch 053/200 | Train loss=0.6956 acc=0.5117 | Val loss=0.6938 acc=0.5111
  Val true counts=[229, 221] pred counts=[303, 147]
  Val per-class acc=[0.6812227368354797, 0.33484163880348206]
  Val bal_acc=0.5080 val_mcc=0.0171
  Selection metric (mcc)=0.0171 | stage=stage2_unfreeze_last_2

Early stopping at epoch 53.

Training complete. Best mcc: 0.0753 | Best val_acc: 0.5378 |
Best val_bal_acc: 0.5339 | Best val_mcc: 0.0753
```

B.4. Run B/C: Partial Unfreeze (2 blocks and 4 blocks)

```
Epoch 001/200 | Train loss=1.1468 acc=0.3321 | Val loss=1.1015 acc=0.2457
  New best (0.2457) saved to /kaggle/working/models/labram_probe/labram_probe_best.pth
Epoch 002/200 | Train loss=1.1103 acc=0.3623 | Val loss=1.0987 acc=0.2457
Epoch 003/200 | Train loss=1.1001 acc=0.2557 | Val loss=1.0987 acc=0.2543
  New best (0.2543) saved to /kaggle/working/models/labram_probe/labram_probe_best.pth
Epoch 004/200 | Train loss=1.0998 acc=0.3086 | Val loss=1.0987 acc=0.2457
Epoch 005/200 | Train loss=1.0989 acc=0.2485 | Val loss=1.0987 acc=0.2457
Epoch 006/200 | Train loss=1.0992 acc=0.2826 | Val loss=1.0986 acc=0.2543
Epoch 007/200 | Train loss=1.0988 acc=0.2801 | Val loss=1.0986 acc=0.5000
  New best (0.5000) saved to /kaggle/working/models/labram_probe/labram_probe_best.pth
Epoch 008/200 | Train loss=1.0987 acc=0.3364 | Val loss=1.0986 acc=0.5000
Epoch 009/200 | Train loss=1.0993 acc=0.4317 | Val loss=1.0987 acc=0.2457
Epoch 010/200 | Train loss=1.0986 acc=0.4933 | Val loss=1.0987 acc=0.5000
Epoch 012/200 | Train loss=1.0989 acc=0.4741 | Val loss=1.0987 acc=0.5000
Epoch 013/200 | Train loss=1.0986 acc=0.5001 | Val loss=1.0987 acc=0.5000
Epoch 014/200 | Train loss=1.0986 acc=0.4986 | Val loss=1.0987 acc=0.5000
Epoch 015/200 | Train loss=1.0988 acc=0.4825 | Val loss=1.0987 acc=0.5000
Epoch 016/200 | Train loss=1.0987 acc=0.5005 | Val loss=1.0986 acc=0.5000
Epoch 017/200 | Train loss=1.0987 acc=0.4997 | Val loss=1.0987 acc=0.5000
Epoch 018/200 | Train loss=1.0986 acc=0.5001 | Val loss=1.0987 acc=0.5000
```

```
with 4 blocks:
Epoch 001/200 | Train loss=1.1505 acc=0.3389 | Val loss=1.1059 acc=0.2457
  New best (0.2457) saved to /kaggle/working/models/labram_probe/labram_probe_best.pth
Epoch 002/200 | Train loss=1.1035 acc=0.3532 | Val loss=1.1044 acc=0.5000
  New best (0.5000) saved to /kaggle/working/models/labram_probe/labram_probe_best.pth
Epoch 003/200 | Train loss=1.1018 acc=0.3513 | Val loss=1.0996 acc=0.2543
Epoch 004/200 | Train loss=1.1002 acc=0.2946 | Val loss=1.0989 acc=0.5000
Epoch 005/200 | Train loss=1.0993 acc=0.3279 | Val loss=1.0989 acc=0.2457
Epoch 006/200 | Train loss=1.0993 acc=0.3907 | Val loss=1.0987 acc=0.5000
Epoch 007/200 | Train loss=1.0988 acc=0.3175 | Val loss=1.0987 acc=0.5000
Epoch 008/200 | Train loss=1.0991 acc=0.2967 | Val loss=1.0988 acc=0.2457
Epoch 009/200 | Train loss=1.0988 acc=0.4218 | Val loss=1.0987 acc=0.2457
Epoch 010/200 | Train loss=1.0988 acc=0.3952 | Val loss=1.0987 acc=0.5000
Epoch 011/200 | Train loss=1.0987 acc=0.4996 | Val loss=1.0987 acc=0.5000
```

B.5. WeightedRandomSampler Error Traceback

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
/tmp/ipykernel_55/1629047982.py in <cell line: 0>()
    208     val_losses, val_accs = [], []
    209     with torch.no_grad():
--> 210         for X_batch, y_batch in val_loader:
    211             X_batch = X_batch.to(device)
    212             y_batch = y_batch.to(device)

/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.py in __next__(self)
    730                 # TODO(https://github.com/pytorch/pytorch/issues/76750)
    731                 self._reset()  # type: ignore[call-arg]
--> 732             data = self._next_data()
    733             self._num_yielded += 1
    734             if (

/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataloader.py in _next_data(self)
    786     def _next_data(self):
    787         index = self._next_index()  # may raise StopIteration
--> 788         data = self._dataset_fetcher.fetch(index)  # may raise StopIteration
    789         if self._pin_memory:
    790             data = _utils.pin_memory.pin_memory(data, self._pin_memory_device)

/usr/local/lib/python3.12/dist-packages/torch/utils/data/_utils/fetch.py in fetch(self, possibly_batched_index)
     50                 data = self.dataset.__getitems__(possibly_batched_index)
     51             else:
---> 52                 data = [self.dataset[idx] for idx in possibly_batched_index]
     53         else:
     54             data = self.dataset[possibly_batched_index]

/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataset.py in __getitem__(self, index)
    205
    206     def __getitem__(self, index):
--> 207         return tuple(tensor[index] for tensor in self.tensors)
    208
    209     def __len__(self):

/usr/local/lib/python3.12/dist-packages/torch/utils/data/dataset.py in <genexpr>(.0)
    205
    206     def __getitem__(self, index):
--> 207         return tuple(tensor[index] for tensor in self.tensors)
    208
    209     def __len__(self):

IndexError: index 2676 is out of bounds for dimension 0 with size 900
```

B.6. Run D: 3-Class Diagnostic Run (Counts + Pred Distributions)

```
============================================================
TRAINING LaBraM PROBE
```

```
============================================================
  Train class counts: [884.0, 871.0, 1755.0]
  Val class counts:   [229.0, 221.0, 450.0]

  Device: cuda
  Trainable params | head=52,227 encoder=966,160 (unfreeze_last_n_blocks=2)
  Optimizer LRs | head=0.001 encoder=1e-05 weight_decay=0.01
  Epochs: 200  |  Early-stop patience: 40

Epoch 001/200 | Train loss=1.1433 acc=0.3379 | Val loss=1.0696 acc=0.5000
  Val true counts=[229, 221, 450] pred counts=[0, 0, 900]
  Val per-class acc=[0.0, 0.0, 1.0]
  New best (0.5000) saved to /kaggle/working/models/labram_probe/labram_probe_best.pth
Epoch 002/200 | Train loss=1.1067 acc=0.3330 | Val loss=1.0779 acc=0.5000
  Val true counts=[229, 221, 450] pred counts=[0, 0, 900]
  Val per-class acc=[0.0, 0.0, 1.0]
Epoch 003/200 | Train loss=1.1010 acc=0.3410 | Val loss=1.0949 acc=0.2544
  Val true counts=[229, 221, 450] pred counts=[900, 0, 0]
  Val per-class acc=[1.0, 0.0, 0.0]
Epoch 004/200 | Train loss=1.0996 acc=0.3376 | Val loss=1.1028 acc=0.2456
  Val true counts=[229, 221, 450] pred counts=[0, 900, 0]
  Val per-class acc=[0.0, 1.0, 0.0]
Epoch 005/200 | Train loss=1.0994 acc=0.3387 | Val loss=1.0998 acc=0.2456
  Val true counts=[229, 221, 450] pred counts=[0, 900, 0]
  Val per-class acc=[0.0, 1.0, 0.0]
Epoch 006/200 | Train loss=1.0990 acc=0.3274 | Val loss=1.0971 acc=0.2456
  Val true counts=[229, 221, 450] pred counts=[0, 900, 0]
  Val per-class acc=[0.0, 1.0, 0.0]
Epoch 007/200 | Train loss=1.0986 acc=0.3422 | Val loss=1.0985 acc=0.2456
  Val true counts=[229, 221, 450] pred counts=[0, 900, 0]
  Val per-class acc=[0.0, 1.0, 0.0]
Epoch 008/200 | Train loss=1.0985 acc=0.3396 | Val loss=1.0982 acc=0.2456
  Val true counts=[229, 221, 450] pred counts=[0, 900, 0]
  Val per-class acc=[0.0, 1.0, 0.0]
Epoch 009/200 | Train loss=1.0988 acc=0.3353 | Val loss=1.0986 acc=0.2456
  Val true counts=[229, 221, 450] pred counts=[0, 900, 0]
  Val per-class acc=[0.0, 1.0, 0.0]
Epoch 010/200 | Train loss=1.0990 acc=0.3299 | Val loss=1.0833 acc=0.5000
  Val true counts=[229, 221, 450] pred counts=[0, 0, 900]
  Val per-class acc=[0.0, 0.0, 1.0]
```

B.7. Run E: Binary Mode (T1/T2) with Diagnostic Output

```
============================================================
TRAINING LaBraM PROBE
============================================================
  Binary mode enabled (T1/T2 only). Kept labels=[0, 1], remap={0: 0, 1: 1}
  Train class counts: [884.0, 871.0]
  Val class counts:   [229.0, 221.0]
  Overrides | pooling=cls unfreeze_last_n_blocks=0 head_lr=0.0001 encoder_lr=1e-05

  Device: cuda

[Sanity] Running frozen-embedding linear probe...
[Sanity] Feature shapes train=(1755, 200) val=(450, 200)
[Sanity] Epoch 01/25 train_loss=1.2102 train_acc=0.5048 val_acc=0.5089
[Sanity] Val true counts=[229, 221] pred counts=[450, 0]
[Sanity] Epoch 05/25 train_loss=0.9566 train_acc=0.5003 val_acc=0.5089
[Sanity] Val true counts=[229, 221] pred counts=[450, 0]
[Sanity] Epoch 10/25 train_loss=0.8089 train_acc=0.5100 val_acc=0.5089
[Sanity] Val true counts=[229, 221] pred counts=[450, 0]
[Sanity] Epoch 15/25 train_loss=0.7810 train_acc=0.5060 val_acc=0.4911
[Sanity] Val true counts=[229, 221] pred counts=[14, 436]
[Sanity] Epoch 20/25 train_loss=0.7993 train_acc=0.5140 val_acc=0.5089
[Sanity] Val true counts=[229, 221] pred counts=[450, 0]
[Sanity] Epoch 25/25 train_loss=0.7463 train_acc=0.5322 val_acc=0.5089
[Sanity] Val true counts=[229, 221] pred counts=[450, 0]
[Sanity] Best val acc: 0.5111
```

```
  Trainable params | head=51,970 encoder=0 (unfreeze_last_n_blocks=0)
  Optimizer LR  | head=0.0001 weight_decay=0.01
  Epochs: 200  |  Early-stop patience: 40

Epoch 001/200 | Train loss=0.8636 acc=0.5088 | Val loss=0.7030 acc=0.5089
  Val true counts=[229, 221] pred counts=[450, 0]
  Val per-class acc=[1.0, 0.0]
  New best (0.5089) saved to /kaggle/working/models/labram_probe/labram_probe_best.pth
Epoch 002/200 | Train loss=0.8154 acc=0.5088 | Val loss=0.6962 acc=0.4933
  Val true counts=[229, 221] pred counts=[311, 139]
  Val per-class acc=[0.6812227368354797, 0.2986425459384918]
Epoch 003/200 | Train loss=0.7915 acc=0.5202 | Val loss=0.6965 acc=0.4933
  Val true counts=[229, 221] pred counts=[385, 65]
  Val per-class acc=[0.8427947759628296, 0.1312217265367508]
Epoch 004/200 | Train loss=0.7586 acc=0.5157 | Val loss=0.6965 acc=0.4756
  Val true counts=[229, 221] pred counts=[91, 359]
  Val per-class acc=[0.18340611457824707, 0.7782805562019348]
Epoch 005/200 | Train loss=0.7461 acc=0.5060 | Val loss=0.7042 acc=0.4911
  Val true counts=[229, 221] pred counts=[0, 450]
  Val per-class acc=[0.0, 1.0]
Epoch 006/200 | Train loss=0.7417 acc=0.5014 | Val loss=0.6994 acc=0.4933
  Val true counts=[229, 221] pred counts=[5, 445]
  Val per-class acc=[0.013100436888635159, 0.9909502267837524]
Epoch 007/200 | Train loss=0.7348 acc=0.5054 | Val loss=0.6965 acc=0.4978
  Val true counts=[229, 221] pred counts=[315, 135]
  Val per-class acc=[0.6943231225013733, 0.29411765933036804]
Epoch 008/200 | Train loss=0.7140 acc=0.5265 | Val loss=0.6982 acc=0.5000
  Val true counts=[229, 221] pred counts=[24, 426]
  Val per-class acc=[0.061135370284318924, 0.9547511339187622]
Epoch 009/200 | Train loss=0.7264 acc=0.4963 | Val loss=0.6954 acc=0.5067
  Val true counts=[229, 221] pred counts=[361, 89]
  Val per-class acc=[0.8034934401512146, 0.19909502565860748]
Epoch 010/200 | Train loss=0.7089 acc=0.4997 | Val loss=0.6957 acc=0.4689
  Val true counts=[229, 221] pred counts=[94, 356]
  Val per-class acc=[0.18340611457824707, 0.7647058963775635]
Epoch 011/200 | Train loss=0.7093 acc=0.5083 | Val loss=0.6990 acc=0.5089
  Val true counts=[229, 221] pred counts=[450, 0]
  Val per-class acc=[1.0, 0.0]
Epoch 012/200 | Train loss=0.7064 acc=0.5083 | Val loss=0.6987 acc=0.4911
  Val true counts=[229, 221] pred counts=[0, 450]
  Val per-class acc=[0.0, 1.0]
Epoch 013/200 | Train loss=0.7060 acc=0.4986 | Val loss=0.7060 acc=0.4911
  Val true counts=[229, 221] pred counts=[0, 450]
  Val per-class acc=[0.0, 1.0]
Epoch 014/200 | Train loss=0.7057 acc=0.5009 | Val loss=0.6998 acc=0.4911
  Val true counts=[229, 221] pred counts=[0, 450]
  Val per-class acc=[0.0, 1.0]
Epoch 015/200 | Train loss=0.7013 acc=0.5071 | Val loss=0.6951 acc=0.5089
  Val true counts=[229, 221] pred counts=[188, 262]
  Val per-class acc=[0.42794761061668396, 0.5927602052688599]
Epoch 016/200 | Train loss=0.7031 acc=0.5123 | Val loss=0.6990 acc=0.4911
  Val true counts=[229, 221] pred counts=[0, 450]
  Val per-class acc=[0.0, 1.0]
Epoch 017/200 | Train loss=0.6975 acc=0.5202 | Val loss=0.6963 acc=0.5067
  Val true counts=[229, 221] pred counts=[449, 1]
  Val per-class acc=[0.9956331849098206, 0.0]
Epoch 018/200 | Train loss=0.6975 acc=0.4883 | Val loss=0.6982 acc=0.4911
  Val true counts=[229, 221] pred counts=[0, 450]
  Val per-class acc=[0.0, 1.0]
Epoch 019/200 | Train loss=0.6976 acc=0.4969 | Val loss=0.6950 acc=0.5067
  Val true counts=[229, 221] pred counts=[449, 1]
  Val per-class acc=[0.9956331849098206, 0.0]
Epoch 020/200 | Train loss=0.7004 acc=0.5094 | Val loss=0.7073 acc=0.5089
  Val true counts=[229, 221] pred counts=[450, 0]
  Val per-class acc=[1.0, 0.0]
Epoch 021/200 | Train loss=0.6989 acc=0.5037 | Val loss=0.6961 acc=0.5089
  Val true counts=[229, 221] pred counts=[450, 0]
  Val per-class acc=[1.0, 0.0]
Epoch 022/200 | Train loss=0.6948 acc=0.5191 | Val loss=0.6967 acc=0.4911
  Val true counts=[229, 221] pred counts=[2, 448]
```

```
  Val per-class acc=[0.0043668122962117195, 0.9954751133918762]
Epoch 023/200 | Train loss=0.6953 acc=0.5151 | Val loss=0.6982 acc=0.5089
  Val true counts=[229, 221] pred counts=[450, 0]
  Val per-class acc=[1.0, 0.0]
Epoch 024/200 | Train loss=0.6982 acc=0.4963 | Val loss=0.6941 acc=0.5089
  Val true counts=[229, 221] pred counts=[450, 0]
  Val per-class acc=[1.0, 0.0]
Epoch 025/200 | Train loss=0.6997 acc=0.4997 | Val loss=0.6954 acc=0.5089
  Val true counts=[229, 221] pred counts=[450, 0]
  Val per-class acc=[1.0, 0.0]
Epoch 026/200 | Train loss=0.6961 acc=0.5014 | Val loss=0.7014 acc=0.4911
  Val true counts=[229, 221] pred counts=[0, 450]
  Val per-class acc=[0.0, 1.0]
Epoch 027/200 | Train loss=0.6928 acc=0.5214 | Val loss=0.6972 acc=0.5089
  Val true counts=[229, 221] pred counts=[450, 0]
  Val per-class acc=[1.0, 0.0]
Epoch 028/200 | Train loss=0.6946 acc=0.5105 | Val loss=0.6985 acc=0.4911
  Val true counts=[229, 221] pred counts=[0, 450]
  Val per-class acc=[0.0, 1.0]
Epoch 029/200 | Train loss=0.6946 acc=0.4912 | Val loss=0.6952 acc=0.4844
  Val true counts=[229, 221] pred counts=[15, 435]
  Val per-class acc=[0.026200873777270317, 0.959276020526886]
Epoch 030/200 | Train loss=0.6938 acc=0.5174 | Val loss=0.6987 acc=0.4911
  Val true counts=[229, 221] pred counts=[0, 450]
  Val per-class acc=[0.0, 1.0]
Epoch 031/200 | Train loss=0.6989 acc=0.4849 | Val loss=0.6966 acc=0.4911
  Val true counts=[229, 221] pred counts=[0, 450]
  Val per-class acc=[0.0, 1.0]
Epoch 032/200 | Train loss=0.6949 acc=0.5123 | Val loss=0.6941 acc=0.4844
  Val true counts=[229, 221] pred counts=[25, 425]
  Val per-class acc=[0.04803493618965149, 0.9366515874862671]
Epoch 033/200 | Train loss=0.6922 acc=0.5271 | Val loss=0.6952 acc=0.5089
  Val true counts=[229, 221] pred counts=[450, 0]
  Val per-class acc=[1.0, 0.0]
Epoch 034/200 | Train loss=0.6969 acc=0.4838 | Val loss=0.6940 acc=0.5111
  Val true counts=[229, 221] pred counts=[97, 353]
  Val per-class acc=[0.23144105076789856, 0.8009049892425537]
  New best (0.5111) saved to /kaggle/working/models/labram_probe/labram_probe_best.pth
Epoch 035/200 | Train loss=0.6932 acc=0.5031 | Val loss=0.6940 acc=0.5111
  Val true counts=[229, 221] pred counts=[449, 1]
  Val per-class acc=[1.0, 0.004524887073785067]
Epoch 036/200 | Train loss=0.6929 acc=0.5259 | Val loss=0.6951 acc=0.4933
  Val true counts=[229, 221] pred counts=[1, 449]
  Val per-class acc=[0.0043668122962117195, 1.0]
Epoch 037/200 | Train loss=0.6919 acc=0.5379 | Val loss=0.6936 acc=0.5044
  Val true counts=[229, 221] pred counts=[444, 6]
  Val per-class acc=[0.9825327396392822, 0.009049774147570133]
Epoch 038/200 | Train loss=0.6939 acc=0.5100 | Val loss=0.6944 acc=0.4867
  Val true counts=[229, 221] pred counts=[40, 410]
  Val per-class acc=[0.08296943455934525, 0.9049773812294006]
Epoch 039/200 | Train loss=0.6950 acc=0.5123 | Val loss=0.6943 acc=0.5178
  Val true counts=[229, 221] pred counts=[126, 324]
  Val per-class acc=[0.3013100326061249, 0.7420814633369446]
  New best (0.5178) saved to /kaggle/working/models/labram_probe/labram_probe_best.pth
Epoch 040/200 | Train loss=0.6947 acc=0.5020 | Val loss=0.6954 acc=0.4889
  Val true counts=[229, 221] pred counts=[1, 449]
  Val per-class acc=[0.0, 0.9954751133918762]
Epoch 041/200 | Train loss=0.6935 acc=0.5151 | Val loss=0.6934 acc=0.5089
  Val true counts=[229, 221] pred counts=[444, 6]
  Val per-class acc=[0.9868995547294617, 0.013574660755693913]
Epoch 042/200 | Train loss=0.6931 acc=0.5117 | Val loss=0.6943 acc=0.4911
  Val true counts=[229, 221] pred counts=[2, 448]
  Val per-class acc=[0.0043668122962117195, 0.9954751133918762]
```

B.8. Run F: Binary Mode After Removing Weighted Sampler

```
============================================================
TRAINING LaBraM PROBE
============================================================
```

```
  Binary mode enabled (T1/T2 only). Kept labels=[0, 1], remap={0: 0, 1: 1}
  Train class counts: [884.0, 871.0]
  Val class counts:   [229.0, 221.0]
  Overrides | pooling=cls unfreeze_last_n_blocks=0 head_lr=0.0001 encoder_lr=1e-05

  Device: cuda

[Sanity] Running frozen-embedding linear probe...
[Sanity] Feature shapes train=(1755, 200) val=(450, 200)
[Sanity] Epoch 01/25 train_loss=0.9123 train_acc=0.4860 val_acc=0.5089
[Sanity] Val true counts=[229, 221] pred counts=[74, 376]
[Sanity] Epoch 05/25 train_loss=0.7581 train_acc=0.5185 val_acc=0.4889
[Sanity] Val true counts=[229, 221] pred counts=[1, 449]
[Sanity] Epoch 10/25 train_loss=0.7403 train_acc=0.5037 val_acc=0.5133
[Sanity] Val true counts=[229, 221] pred counts=[172, 278]
[Sanity] Epoch 15/25 train_loss=0.7156 train_acc=0.5282 val_acc=0.5089
[Sanity] Val true counts=[229, 221] pred counts=[450, 0]
[Sanity] Epoch 20/25 train_loss=0.7585 train_acc=0.5276 val_acc=0.4978
[Sanity] Val true counts=[229, 221] pred counts=[419, 31]
[Sanity] Epoch 25/25 train_loss=0.8477 train_acc=0.5481 val_acc=0.4911
[Sanity] Val true counts=[229, 221] pred counts=[0, 450]
[Sanity] Best val acc: 0.5267

  Trainable params | head=51,970 encoder=0 (unfreeze_last_n_blocks=0)
  Optimizer LR  | head=0.0001 weight_decay=0.01
  Epochs: 200  |  Early-stop patience: 40

Epoch 001/200 | Train loss=0.8512 acc=0.4929 | Val loss=0.6916 acc=0.5289
  Val true counts=[229, 221] pred counts=[131, 319]
  Val per-class acc=[0.3231441080570221, 0.7420814633369446]
  New best (0.5289) saved to /kaggle/working/models/labram_probe/labram_probe_best.pth
Epoch 002/200 | Train loss=0.7797 acc=0.5060 | Val loss=0.7054 acc=0.4911
  Val true counts=[229, 221] pred counts=[0, 450]
  Val per-class acc=[0.0, 1.0]
Epoch 003/200 | Train loss=0.7725 acc=0.4997 | Val loss=0.7047 acc=0.4911
  Val true counts=[229, 221] pred counts=[0, 450]
  Val per-class acc=[0.0, 1.0]
Epoch 004/200 | Train loss=0.7447 acc=0.5145 | Val loss=0.6918 acc=0.5378
  Val true counts=[229, 221] pred counts=[183, 267]
  Val per-class acc=[0.44541484117507935, 0.6334841847419739]
  New best (0.5378) saved to /kaggle/working/models/labram_probe/labram_probe_best.pth
Epoch 005/200 | Train loss=0.7568 acc=0.5026 | Val loss=0.7316 acc=0.5089
  Val true counts=[229, 221] pred counts=[450, 0]
  Val per-class acc=[1.0, 0.0]
Epoch 006/200 | Train loss=0.7458 acc=0.4963 | Val loss=0.6944 acc=0.5089
  Val true counts=[229, 221] pred counts=[450, 0]
  Val per-class acc=[1.0, 0.0]
Epoch 007/200 | Train loss=0.7251 acc=0.4969 | Val loss=0.7080 acc=0.5089
  Val true counts=[229, 221] pred counts=[450, 0]
  Val per-class acc=[1.0, 0.0]
Epoch 008/200 | Train loss=0.7123 acc=0.5123 | Val loss=0.6939 acc=0.4889
  Val true counts=[229, 221] pred counts=[9, 441]
  Val per-class acc=[0.017467249184846878, 0.9773755669593811]
Epoch 009/200 | Train loss=0.7184 acc=0.4980 | Val loss=0.6941 acc=0.4956
  Val true counts=[229, 221] pred counts=[14, 436]
  Val per-class acc=[0.034934498369693756, 0.9728506803512573]
Epoch 010/200 | Train loss=0.7059 acc=0.5100 | Val loss=0.6982 acc=0.5089
  Val true counts=[229, 221] pred counts=[450, 0]
  Val per-class acc=[1.0, 0.0]
Epoch 011/200 | Train loss=0.7098 acc=0.5088 | Val loss=0.6935 acc=0.5111
  Val true counts=[229, 221] pred counts=[115, 335]
  Val per-class acc=[0.27074235677719116, 0.7601810097694397]
Epoch 012/200 | Train loss=0.6998 acc=0.4974 | Val loss=0.6929 acc=0.4822
  Val true counts=[229, 221] pred counts=[356, 94]
  Val per-class acc=[0.7685589790344238, 0.18552036583423615]
Epoch 013/200 | Train loss=0.7011 acc=0.5128 | Val loss=0.6940 acc=0.5089
  Val true counts=[229, 221] pred counts=[450, 0]
  Val per-class acc=[1.0, 0.0]
Epoch 014/200 | Train loss=0.7003 acc=0.5048 | Val loss=0.6929 acc=0.5156
  Val true counts=[229, 221] pred counts=[409, 41]
```

```
  Val per-class acc=[0.9170305728912354, 0.09954751282930374]
Epoch 015/200 | Train loss=0.7064 acc=0.4769 | Val loss=0.6941 acc=0.5089
  Val true counts=[229, 221] pred counts=[450, 0]
  Val per-class acc=[1.0, 0.0]
Epoch 016/200 | Train loss=0.7053 acc=0.4940 | Val loss=0.6997 acc=0.5089
  Val true counts=[229, 221] pred counts=[450, 0]
  Val per-class acc=[1.0, 0.0]
Epoch 017/200 | Train loss=0.6953 acc=0.5111 | Val loss=0.6934 acc=0.4733
  Val true counts=[229, 221] pred counts=[82, 368]
  Val per-class acc=[0.16157205402851105, 0.7963801026344299]
Epoch 018/200 | Train loss=0.7021 acc=0.4815 | Val loss=0.6934 acc=0.5133
  Val true counts=[229, 221] pred counts=[430, 20]
  Val per-class acc=[0.960698664188385, 0.04977375641465187]
```