EEG Motor Imagery with Foundation-Model Transfer and Compact CNN Baselines: Comprehensive Consolidation of LaBraM Probing/Fine-Tuning/Curriculum and EEGNet Training/Tuning

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

This manuscript consolidates the full EEG motor-imagery (MI) research trail implemented in `ml/`, including script code and notebook experiments (`ml/labram-for-eegmmidb.ipynb`). We systematize two modeling tracks: (i) LaBraM transfer experiments with probing, partial/full fine-tuning, and two-stage curriculum training; and (ii) EEGNet/EEGNetResidual compact baselines with subsequent user-level Muse calibration tuning. The cross-subject PhysioNet split used subjects 1–40 for training and 41–50 for validation, with runs [4, 8, 12]. Consolidated evidence shows persistent class-collapse dynamics for several LaBraM settings, especially in 3-class mode (left/right/rest), where completed runs plateaued at 0.5000 validation accuracy near $\log 3 \approx 1.0986$**. Binary LaBraM improved modestly, with the best completed run reaching 0.5556. A full LaBraM fine-tune run reached train accuracy 0.9966 but only 0.5222 validation accuracy, indicating severe overfitting. The curriculum run improved checkpoint selection robustness (best MCC 0.0753; best val_acc 0.5378; best balanced accuracy 0.5339) without surpassing the best binary accuracy. The compact EEGNetResidual baseline achieved 0.5919 on the cross-subject split. We then ran an expanded Muse tuning sweep over six calibration files (v1–v3, left/right), comparing freezing policy, learning rate, overlap windowing, class-weighted loss, and split protocol; the strongest controlled holdout result was 0.6531 with stratified 80/20 splitting, no overlap, standard cross-entropy, and full-model fine-tuning. We provide formal propositions/proofs for diagnostic metrics (chance-level cross-entropy, collapse-accuracy bounds, MCC degeneracy for constant predictors, token-budget bounds in LaBraM preprocessing), exploratory code listings, and a reproducible run ledger suitable for critical academic audit.**

Keywords: EEG, motor imagery, LaBraM, EEGNet, transfer learning, curriculum learning, BCI, reproducibility, negative results

1. Introduction

EEG MI decoding remains difficult because discriminative information is weak relative to noise, highly nonstationary across sessions and subjects, and sensitive to preprocessing and data partitioning (Lawhern et al., 2018; Schirrmeister et al., 2017). Foundation-model (FM) transfer is attractive in principle: pre-train large representations once, then adapt to downstream tasks. In practice, reliable transfer requires that the downstream optimization interface preserve class-separable geometry under realistic channel and data constraints.

This work answers a practical question: *what does the current project actually demonstrate when all `ml/` artifacts are jointly considered*? The previous report did not fully integrate notebook and script evidence. Here we consolidate these artifacts into one reproducible, technically defensible record.

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

2. Consolidated Artifact Map

Table 1 maps code artifacts to scientific roles.

Table 1 | Consolidated artifact map for `ml/` and notebook work.

Artifact	Role in this study
`ml/eeg/dataset.py`	PhysioNet acquisition, event filtering, epoch creation, band-pass preprocessing.
`ml/models/labram_probe.py`	Script-level LaBraM probe with optional block unfreezing and token pooling.
`ml/labram_probe.py`	Reproducible LaBraM training entrypoint over subject-level split.
`ml/labram-for-eegmmidb.ipynb`	Main exploratory run trail: 3-class, binary, full fine-tune, curriculum, diagnostics, and embedding geometry analysis.
`ml/training_notebook_cells.py`	Scripted extraction of the notebook training logic: stage switch, MCC selection, per-class diagnostics, scheduler behavior.
`ml/eeg/layers/labram_encoder.py`	Tokenization interface to pre-trained LaBraM backbone, including patch budget truncation rule.
`ml/eeg/layers/labram/neural_transformers.py`	Backbone internals (patch size, depth, embedding path, class token).
`ml/__main__.py`	Cross-subject EEGNet/EEGNetResidual baseline training flow and summary logging.
`ml/tune_eegnet_muse.py`	User-level Muse fine-tuning script; includes all-data mode and optional holdout mode.
`docs/eeg/appendix_labram_training_logs.tex`	Verbatim log archive used for auditability and traceback preservation.

3. Materials and Methods

3.1. Datasets and Splits

Cross-subject PhysioNet MI setup. Runs [4, 8, 12] were used with events `T0`, `T1`, `T2` (rest, left, right), following the code configuration. Observed consolidated split statistics:

- 3-class train: 3510 epochs, shape $(3510, 64, 601)$, counts $[884, 871, 1755]$.
- 3-class val: 900 epochs, shape $(900, 64, 601)$, counts $[229, 221, 450]$.
- Binary derivation (T1/T2 only): train $[884, 871]$, val $[229, 221]$, totals 1755/450.

Muse calibration setup. Current discovered calibration files: `left_muse_v1,v2,v3.csv` and `right_muse_v1,v2,v3.csv` (6 files total). With non-overlapping 3-second windows (`stride-seconds=3.0`), epoch counts sum to 247 samples. Holdout mode with `val_split=0.2` gives 198 training and 49 validation samples. With overlapping windows (`stride-seconds=1.5`), the same recordings produce 490 epochs (392/98 under 80/20 split).

3.2. Signal Processing

Both tracks apply 8–30 Hz band-pass filtering (5th-order Butterworth in code). Let $x_{i,c}(t)$ be epoch i, channel c. Filtered signal is

$$\tilde{x}_{i,c}(t) = (h * x_{i,c})(t), \tag{1}$$

with h induced by $(\text{lowcut}, \text{highcut}) = (8, 30)$ Hz.

For LaBraM experiments, optional long-context interpolation resamples epochs to length $L' = 1600$:

$$\hat{x}_{i,c} = \mathcal{R}_{L'}(\tilde{x}_{i,c}), \tag{2}$$

followed by per-epoch/channel z-score normalization.

For EEGNet tuning, Muse epochs are resampled to the checkpoint input length ($n\_samples = 481$).

3.3. Modeling Tracks

LaBraM transfer track. The notebook explored:

- frozen/probe-like training,
- partial unfreezing of final transformer blocks,
- full fine-tuning (`full_finetune=True`),
- curriculum: stage-1 head-only then stage-2 unfreeze-last-2 blocks, with MCC-based checkpoint selection.

The backbone in code uses T =200 patch size, d =200 embedding dimension, depth 12.

EEGNet track.

- Cross-subject baseline: `ml/__main__.py` with EEGNetResidual, 4 channels, best val accuracy 0.5919 (log excerpt in appendix).
- Later tuning: `ml/tune_eegnet_muse.py` on user calibration CSVs, including a controlled sweep over freezing policy, overlap windows, learning rate, class-weighted loss, and split strategy.

3.4. Optimization Objective

Main classification objective:

$$\mathcal{L}_{\text{CE}} = -\frac{1}{B}\sum_{i=1}^{B} \log p_\theta(y_i|x_i). \tag{3}$$

Some notebook runs used label smoothing ($\epsilon = 0.05$); curriculum selection used MCC (binary) or balanced accuracy fallback.

4. Formal Diagnostic Propositions and Proofs

4.1. Chance-level Cross-Entropy Baseline

Proposition 1. *For K-class classification with uniform prediction $p(y = k|x) = 1/K$, expected cross-entropy equals* $\log K$.

Proof. For any ground-truth class y, loss is $-\log p(y|x) = -\log(1/K) = \log K$. Expectation over data leaves $\log K$ unchanged. □

Corollary 1. *Random-guess baselines here are* $\log 3 \approx 1.0986$ *for 3-class and* $\log 2 \approx 0.6931$ *for binary.*

4.2. Collapse Accuracy Bound

Proposition 2. *Let class priors be* $\pi_1, \ldots, \pi_K$, $\sum_j \pi_j = 1$. *If a classifier collapses to constant prediction* $\hat{y} = j^\star$, *then accuracy is* $\pi_{j^\star}$. *Best possible constant-prediction accuracy is* $\max_j \pi_j$.

Proof. Accuracy is $\Pr(\hat{y} = y) = \Pr(y = j^\star) = \pi_{j^\star}$. Maximizing over $j^\star$ yields $\max_j \pi_j$. □

Corollary 2. *For 3-class validation counts* $[229, 221, 450]$, $\max_j \pi_j = 450/900 = 0.5$. *Thus persistent 0.5000 accuracy is compatible with majority-class collapse.*

4.3. MCC Degeneracy Under Constant Binary Prediction

Proposition 3. *Assume both binary classes appear in ground truth. If predictions are constant (*$\hat{y} \equiv 0$ *or* $\hat{y} \equiv 1$*), then MCC is 0 under the standard safe-convention used in code (return 0 when denominator is 0).*

Proof. MCC:

$$\mathrm{MCC} = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}.$$

For constant prediction, at least one confusion-matrix marginal in the denominator is zero, yielding denominator 0. The implemented function returns 0 in this case. Therefore MCC $= 0$. □

Corollary 3. *In the binary validation split* $[229, 221]$, *predicting all class 0 gives accuracy* $229/450 = 0.5089$, *balanced accuracy 0.5, MCC 0.*

4.4. LaBraM Token Budget Bound

Proposition 4. *Given input sequence length* SL *and patch size* $T = 200$, *the implemented LaBraM encoder uses*

$$P = \min\left(\left\lfloor \frac{SL}{T} \right\rfloor, 16\right)$$

patches per channel. Patch-token count is $N \cdot P$; *with class token it is* $N \cdot P + 1$.

Proof. Directly from `ml/eeg/layers/labram_encoder.py`: P is computed as above, the input is sliced to the last $P \cdot T$ samples, then reshaped to (B, N, P, T). The transformer appends one class token before attention. □

Corollary 4. *Without long-context resampling,* $SL = 601 \Rightarrow P = 3$. *With resampling to 1600,* $P = 8$. *Hence long-context mode increases per-channel patch count from 3 to 8.*

5. Experimental Ledger

5.1. LaBraM Notebook Run Ledger

Table 2 separates completed and interrupted runs.

Table 2 | Consolidated LaBraM run ledger from `ml/labram-for-eegmmidb.ipynb`.

Cell	Task	Core setup	Selection/diagnostics	Status	Best reported result
10	3-class	Baseline-style training	Standard val acc	Completed	val_acc 0.5000, early stop epoch 44
12	3-class	Unfreeze-last-2, detailed per-class logs	Pred-count and per-class acc each epoch	Completed	val_acc 0.5000, early stop epoch 41
11	3-class	Partial unfreeze run	Standard diagnostics	Interrupted	best seen val_acc 0.2543 (epoch 3)
13	3-class	CLS pooling, freeze config variant	Includes sanity probe	Interrupted	best seen val_acc 0.4944 (sanity 0.5022)
7	Binary	Full fine-tune; long-context 1600; label smoothing 0.05; CLS pooling	Frozen-embedding sanity probe	Completed	val_acc 0.5222 (sanity 0.5444), early stop epoch 46
8	Binary	Curriculum: stage-1 head-only (12 epochs) then unfreeze last 2 blocks	MCC-based selection, per-class diagnostics	Completed	val_acc 0.5378, bal_acc 0.5339, MCC 0.0753, early stop epoch 53 (sanity 0.5600)
14	Binary	Unfreeze-last-2, no long-context variant	Standard diagnostics	Interrupted	best seen val_acc 0.5400 (epoch 74)
15	Binary	Unfreeze-last-2 + long-context + label smoothing 0.05	Standard diagnostics	Completed	val_acc 0.5556, early stop epoch 83

5.2. EEGNet Baseline and Tuning Ledger

Table 3 | EEGNet/EEGNetResidual results consolidated from script logs and direct reruns.

Run	Data regime	Key setting	Outcome
Cross-subject EEGNetResidual baseline (`ml/__main__.py`)	Train 2925 / Val 750, shape $(4, 481)$, class counts train $[1470, 1455]$, val $[379, 371]$	Early-stop patience 200, CPU log excerpt archived	Best val_acc 0.5919 at epoch 73; stop at epoch 273
Muse tuning (legacy random split)	6 CSVs, total 247 epochs, split 198/49	`-no-use-all-data`, freeze early, lr $= 10^{-4}$	Best val_acc 0.5714

Run	Data regime	Key setting	Outcome
Muse tuning (stratified, no overlap; recommended holdout protocol)	6 CSVs, total 247 epochs, stratified 198/49	`-no-use-all-data -stratified-split`, no freeze, lr = 10^{-4}, stride 3.0	Best val_acc 0.6531 at epoch 63; checkpoint `eegnet_tuned_best_stratified`
Muse tuning (stratified, no overlap, freeze ablation)	6 CSVs, total 247 epochs, stratified 198/49	Same as above + `-freeze-early`	Best val_acc 0.6327
Muse tuning (stratified, overlap ablation)	6 CSVs, total 490 epochs, stratified 392/98	stride 1.5, no freeze, lr = 10^{-4}	Best val_acc 0.6224
Muse tuning (stratified, class-weighted loss ablation)	6 CSVs, total 490 epochs, stratified 392/98	stride 1.5 + class-weighted CE	Best val_acc 0.6020
Muse deployment tuning (all-data mode)	6 CSVs, all 247 epochs in train	`-use-all-data`, no validation split, lr = 10^{-4}	Best train loss 0.5480 at epoch 64; checkpoint `eegnet_tuned_all_data_best`

6. Results

6.1. LaBraM: Main Observations

3-class regime. Completed 3-class runs repeatedly converged to 0.5000 validation accuracy with strong collapse/flip behavior in prediction histograms (e.g., all predictions in one class, then another), consistent with Proposition 2 and corollary bounds.

Binary regime. Best completed binary result was 0.5556 (cell 15), approximately +4.67 percentage points above majority-class collapse (0.5089), but still far below robust decoding thresholds typically expected for stable BCI control.

Overfitting in full fine-tune (cell 7). Train accuracy increased to 0.9966 (epoch 42), while validation accuracy peaked at 0.5222 and validation loss rose above 1.5, indicating memorization without generalization.

Curriculum behavior (cell 8). Stage switch at epoch 13 improved checkpoint selection quality using MCC (best 0.0753). This improved metric robustness but did not exceed the best binary accuracy obtained in cell 15.

6.2. Embedding Geometry Probe

From notebook exploratory analysis:

- PCA explained variance ratio: [0.19062445, 0.10972243].
- Silhouette (PCA 2D): 0.0099735.
- Silhouette (t-SNE 2D): 0.0099501.

Near-zero silhouette values indicate weakly separated clusters in low-dimensional projections, consistent with observed collapse dynamics.

6.3. EEGNet Track Interpretation

The compact cross-subject EEGNetResidual baseline (0.5919) still exceeds all completed LaBraM runs in this consolidated record (best LaBraM 0.5556). In the expanded Muse sweep, holdout performance is strongly protocol-dependent: random split settings were optimistic/noisy, while stratified holdout yielded a stable best of 0.6531 (no overlap, no freeze, lr = 10^{-4}). Overlap augmentation to 490 epochs did not improve the stratified metric (best 0.6224), and class-weighted loss further reduced performance (0.6020). This indicates the recent degradation with added files is mitigated more by split discipline and conservative tuning than by aggressive augmentation. As before, `-use-all-data` remains useful for final deployment checkpointing but must not be treated as validation evidence.

7. Exploratory Code Listings

Listing 1: LaBraM token-budget rule (`ml/eeg/layers/labram_encoder.py`)

```
T = self.model.patch_size
P = min(SL // T, 16)
x = x[:, :, -P*T:]
x = rearrange(x, "B N (P T) -> B N P T", T=T)
```

Listing 2: Curriculum + MCC logic (`ml/training_notebook_cell.py`)

```
def binary_mcc_from_tensors(y_pred, y_true):
    tp = int(((y_pred == 1) & (y_true == 1)).sum().item())
    tn = int(((y_pred == 0) & (y_true == 0)).sum().item())
    fp = int(((y_pred == 1) & (y_true == 0)).sum().item())
    fn = int(((y_pred == 0) & (y_true == 1)).sum().item())
    denom = np.sqrt(float((tp + fp) * (tp + fn) * (tn + fp) * (tn + fn)))
    if denom == 0.0:
        return 0.0
    return float((tp * tn - fp * fn) / denom)

if stage2_enabled and epoch == stage2_start_epoch:
    model.configure_trainable_encoder(
        freeze_encoder=True,
        unfreeze_last_n_blocks=stage2_unfreeze_last_n_blocks,
    )
```

Listing 3: Muse split-mode guard (`ml/tune_eegnet_muse.py`)

```
if use_all_data:
    X_train, y_train = X, y
    X_val = np.empty((0, X.shape[1], X.shape[2]), dtype=X.dtype)
    y_val = np.empty((0,), dtype=y.dtype)
    print("Training mode: using all data (no validation split).")
else:
    if args.stratified_split:
        X_train, y_train, X_val, y_val = _split_train_val_stratified(
            X, y, val_split=args.val_split, seed=args.seed
        )
    else:
        X_train, y_train, X_val, y_val = _split_train_val(
            X, y, val_split=args.val_split, seed=args.seed
        )
```

8. Discussion

The consolidated evidence supports a careful, publication-grade negative-to-mixed conclusion:

- In this project state, LaBraM transfer did not reliably surpass compact EEGNet baselines on cross-subject MI decoding.
- Multiple LaBraM runs show collapse dynamics predicted by simple metric theory (Sections 4–6).
- Curriculum learning improved checkpoint *selection quality* (MCC), but not final best accuracy relative to the best binary run.
- Muse personalization remains promising, but must be reported with explicit split protocol; all-data training cannot be treated as validation evidence.
- On the current 6-file Muse set, stratified holdout with non-overlapping windows outperformed overlap augmentation and class-weighted loss, suggesting that better evaluation protocol and moderate optimization changes dominate more complex tuning heuristics.

These findings are still scientifically useful: they isolate where transfer currently fails (representation-task interface and optimization stability), rather than only reporting top-line numbers.

9. Limitations and Threats to Validity

- Several notebook runs were interrupted; they provide trajectory evidence, not completed endpoint claims.
- The new Muse sweep is primarily single-seed ($seed = 42$); no multi-seed confidence intervals are reported yet.
- Cross-subject and within-user regimes are different tasks; direct score comparison should be interpreted cautiously.
- Muse holdout is still within-user and currently random/stratified at epoch level; strict session-wise holdout (e.g., train on v1/v2, test on v3) remains to be added.

10. Conclusion

After consolidating all available `ml/` artifacts, the current evidence shows: (i) frequent LaBraM class-collapse behavior in 3-class MI; (ii) modest binary improvements with best completed LaBraM val_acc 0.5556; (iii) strong overfitting under full fine-tuning; (iv) improved selection stability but limited headline gain under curriculum learning; and (v) competitive compact CNN performance (0.5919 cross-subject baseline, 0.6531 best stratified Muse holdout tuning on 6 calibration files). The latest tuning sweep further indicates that non-overlapping windows and standard CE with full-model updates are currently stronger than overlap augmentation or class-weighted loss for this user-specific regime. The immediate research direction is not broader claim inflation, but rigorous stabilization: multi-seed evaluation, session-wise holdout protocols, tighter calibration evaluation, and controlled comparisons against compact baselines under identical splits.

11. Reproducibility and Availability

Code paths. Core scripts are in `ml/`; this manuscript references exact files listed in Table 1.

Representative commands.

```
python ml/labram_probe.py
python ml/tune_eegnet_muse.py --data-dir ml/calibration_data --no-use-all-data \
  --stratified-split --val-split 0.2 --epochs 80 --learning-rate 1e-4
python ml/tune_eegnet_muse.py --data-dir ml/calibration_data --use-all-data \
  --output-checkpoint ml/models/baseline_2/eegnet_tuned_all_data_bestcfg.pth
```

Raw logs. Verbatim run excerpts and traceback archive are retained in Appendix A.

12. Ethics and Safety

This work is non-clinical and does not make medical claims. No personally identifying data are reported in this manuscript.

References

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.

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]
```