Proposal for Objective 3

February 16, 2026

1 Motivation and Research Problems

Based on prior sleep modeling literature, the optimal amount of temporal context is *task-dependent* (e.g., local patterns for per-epoch staging versus longer-range structure or sparse events for disease severity and phenotype risk). Meanwhile, real-world PSG is affected by *distribution change*: montage differences, missing channels, sensor artifacts, sampling mismatches, and cross-site dataset shifts. These two issues make fixed design choices brittle and limit deployment reliability.

This proposal focuses on two research problems:

- **Problem A: Context usage uncertainty.** How much context (and potentially which night region) is truly needed for different sleep tasks, and how can we learn it automatically rather than selecting it manually?
- **Problem B: Distribution change robustness.** How can we maintain (or safely recover) performance under montage/missingness/artifact/domain shift without labels, and avoid adaptation drift?

For each problem, I propose two method directions. Before developing new models, we will run targeted validation experiments (Phase 0) to confirm the problem on selected tasks/datasets and to produce reference "oracle" behaviors that we can later use to verify whether the proposed methods are selecting reasonable context lengths and behaving robustly under shift.

2 Phase 0: Validation Experiments (Proving the Problems Exist)

2.1 Phase 0A: Validate Context-Length and Coverage Dependence (Problem A Evidence)

Datasets and Tasks (To start with). Choose 2–3 tasks spanning temporal needs:

- **Per-epoch task:** sleep staging (e.g., 5-class).
- **Night-level tasks:** AHI severity, apnea presence, dementia, insomnia (depending on label availability per dataset).

Train on one source dataset (e.g., STAGES or SHHS) and evaluate in-domain; optionally add cross-dataset evaluation (e.g., train STAGES $\rightarrow$ test SHHS) as a stress test.

Validation Protocol: Manual Length and Coverage Sweeps. For each task, train the same downstream architecture multiple times with fixed context lengths:

$$L \in \{30\text{s}, 2\text{m}, 5\text{m}, 10\text{m}, 20\text{m}, 40\text{m}, 80\text{m}\} \quad \text{(log-spaced)}$$

- For **staging**, define context around an anchor epoch t (e.g., past-only window ending at t).
- For **night-level tasks**, evaluate both *length* and *coverage* (night region), since "best length" is confounded with "where to look." Use a small menu of windows: {start, middle, end, full-night}.

Outcomes (Problem A).

- Different tasks have different performance-vs-length curves; optima differ.
- Some tasks plateau quickly; some benefit from intermediate/longer context; sometimes full context can be worse than intermediate.
- Night-level tasks may prefer different night regions (coverage), motivating optional coverage selection.

How Phase 0A ties to main experiments. The best-performing length(s) and region(s) from Phase 0A become:

- a reference "oracle" for comparison (what a human would choose with hindsight),
- a target behavior to validate that learned policies (Ideas 1–2) recover similar length/coverage preferences automatically.

2.2 Phase 0B: Validate Robustness Failure Modes (Problem B Evidence)

PSG-specific stress-test suite. We quantify what breaks robustness using controlled perturbations:

- **Missing modalities/channels:** drop ECG/Resp/EOG/EMG; simulate EEG montage changes via channel-subset sampling.
- **Intermittent missingness:** drop contiguous segments (e.g., 10–30 min gaps).
- **Artifact injection:** noise bursts / amplitude scaling / baseline drift proxies.
- **Sampling mismatches:** resampling/filtering differences.
- **Cross-dataset transfer:** train on dataset A and test on dataset B (site/capture shift).

Outcomes (Problem B).

- Identify which modalities, montage changes, and artifact types degrade each task the most.
- Establish robust baselines and a consistent evaluation protocol for Ideas 3–4.

3 Problem A: Context Usage Uncertainty

We assume a backbone that produces per-window embeddings (e.g., SleepFM) and keep long-context modules lightweight and length-flexible.

4 Idea 1: AMTA + CSL (Multi-Timescale Adapter with Context Sufficiency Learning)

4.1 Goal

Learn a lightweight long-context module that (i) adaptively mixes short/medium/long temporal evidence and (ii) is trained to behave stably across context lengths. This turns context-length choice from a manual hyperparameter into a learned/measurable property. It also yields interpretability via gates and post-hoc sufficiency/importance analysis.

4.2 Inputs (Backbone + Multimodal, Montage-Agnostic Interface)

- Channel-agnostic backbone produces embeddings per time step/window: $\mathbf{h}_m[t]$ for each modality family m (EEG/EOG/EMG/ECG/Resp), plus a mask $\mathbf{z}_m[t] \in \{0, 1\}$ indicating availability.

4.3 Architecture: Adaptive Multi-Timescale Adapter (AMTA)

We build three temporal pathways with different receptive fields and mix them with a learned gate:

- **Short path:** local minutes-scale context (e.g., small TCN/GRU over recent steps).
- **Medium path:** tens-of-minutes context (e.g., dilated TCN or small SSM / Mamba-style block).
- **Long path:** hours-scale memory (e.g., compact SSM state or hierarchical pooling state).

Each path outputs features at time t: $\mathbf{f}_{\text{short}}[t]$, $\mathbf{f}_{\text{med}}[t]$, $\mathbf{f}_{\text{long}}[t]$. A gate outputs mixing weights $\mathbf{g}[t] \in \mathbb{R}^3$ (softmaxed), optionally conditioned on missingness:

$$\mathbf{f}[t] \;=\; g_{\text{short}}[t]\mathbf{f}_{\text{short}}[t] \;+\; g_{\text{med}}[t]\mathbf{f}_{\text{med}}[t] \;+\; g_{\text{long}}[t]\mathbf{f}_{\text{long}}[t].$$

A task-specific head consumes $\mathbf{f}[t]$ (staging) or a pooled summary of $\mathbf{f}[\cdot]$ (night-level).

Options for modality fusion.

- **Fusion-first:** fuse modalities at each t into one vector, then AMTA processes the fused sequence.
- **Per-modality states:** run small temporal adapters per modality family, then fuse their states with a mask-aware gate.

4.4 Training: Context Sufficiency Learning (CSL) via Truncated Multi-View Batches

For each training sample, we create two views with different history lengths that share the same prediction target.

- **Sleep Staging:** anchor an epoch t with label y_t. Create:
  - short view: last L_s minutes ending at t,
  - long view: last L_ℓ minutes ending at t (with $L_\ell > L_s$).
- **Night-level:** choose a window (see optional coverage) and create short/long views inside the chosen window.

4.4.1 Losses

Let p_s and p_ℓ be predictions from short and long views, and y be the label.

1. **Supervised losses (always on):**

   $$\mathcal{L}_{\text{sup}} \;=\; \mathcal{L}(p_s, y) \;+\; \mathcal{L}(p_\ell, y)$$

   where $\mathcal{L}$ is CE for classification or MAE/MSE for regression tasks.

2. **Consistency loss (selectively on):** encourage the short prediction to match the long prediction *only when long actually helps.* Two practical gating options:

   - **Confidence-gated:** apply consistency if the long prediction is confident (e.g., low entropy or high margin).
   - **Better-with-label gated:** apply consistency only if $\mathcal{L}(p_\ell, y) < \mathcal{L}(p_s, y)$ for this sample.

   Using a distillation-style match (e.g., KL divergence) gives:

   $$\mathcal{L}_{\text{cons}} \;=\; \mathbf{1}[\text{gate passes}] \cdot \mathrm{KL}\big(p_\ell \,\|\, p_s\big).$$

Total:

$$\mathcal{L}_{\text{total}} \;=\; \mathcal{L}_{\text{sup}} \;+\; \lambda_{\text{cons}}\mathcal{L}_{\text{cons}}.$$

4.5 Testing and Outcomes

Testing.

- Evaluate performance across a sweep of lengths L (as in Phase 0A) and compare to the validation curves.
- Under missingness/montage change, evaluate robustness (drop modality families; channel-subset perturbations).

Outcomes.

- AMTA gates reveal when the model relies on short vs long evidence.
- CSL yields stable behavior across lengths and supports a principled "sufficiency length" estimate (smallest L within ϵ of best).
- Validate alignment with Phase 0A: do learned sufficiency trends match the manually observed task-specific optima?

4.6 Interpretability for Idea 1

- **Timescale importance over time:** visualize $g_{\text{short}}[t], g_{\text{med}}[t], g_{\text{long}}[t]$ and summarize by task/subject.
- **Modality reliance:** if modality-aware gating is used, report average modality weights per task; show shifts under missingness.
- **Window importance:** rank time segments by gate mass or attribution; validate by masking top-ranked segments vs random segments and measuring performance drop.

4.7 Possible Add-on: Coverage Selection (Where to Look) for Night-Level Tasks

Length alone does not address the "last hour matters" case. Add coverage selection only for night-level outcomes:

- **Option C1:** candidate windows {start, middle, end, full}; learn a window selector and run AMTA+CSL inside the selected window.
- **Option C2:** cheap coarse scan (downsampled embeddings) predicts a window location/duration; run AMTA inside that window.

5 Idea 2: Adaptive Early-Exit Length Selection (Learn Which Length to Use)

5.1 Goal

Train a model that produces predictions at multiple predefined lengths and learns a selector that chooses which length to use per sample/task. The selector is trained to choose the *earliest near-best* exit, enabling intermediate optima and avoiding the assumption that longer is always better.

5.2 Architecture

- **Incremental long-context module:** any temporal module that can produce a state after consuming a prefix of the sequence (SSM/GRU/TCN/AMTA).
- **Exit heads:** K small heads at checkpoints $L_1 < L_2 < \cdots < L_K$ (e.g., 2/5/10/20/40 minutes). Each head produces a prediction p_k from state $\mathbf{s}_k$.
- **Selector:** a small MLP that outputs a categorical distribution over exits ("predict k" formulation), conditioned on features (e.g., $\mathbf{s}_K$ plus missingness mask).

Selector input. Use the longest-context state $\mathbf{s}_K$ plus missingness mask:

$$q \;=\; \operatorname{softmax}\big(\mathrm{MLP}([\mathbf{s}_K;\, \mathbf{z}])\big) \in \mathbb{R}^K.$$

5.3 Training: One Forward Pass, Multiple Exits

For each training example:

1. Run the long-context module once up to L_K and record checkpoint states $\mathbf{s}_1, \dots, \mathbf{s}_K$.
2. Compute predictions at all exits: $p_k = \mathrm{Head}_k(\mathbf{s}_k)$.
3. Compute supervised loss per exit:
$$\ell_k \;=\; \mathcal{L}(p_k, y).$$

5.3.1 Defining the target exit $k^\star$ (Oracle-exit supervision; no RL)

Compute:

$$k_{\text{best}} = \arg\min_k \ell_k, \qquad k^\star = \min\{k :\; \ell_k \le \ell_{k_{\text{best}}} + \delta\},$$

i.e., the earliest exit within a small margin δ of the best exit.

5.3.2 Losses

1. **Multi-exit supervised loss (train all exits):**
$$\mathcal{L}_{\text{exit}} \;=\; \sum_{k=1}^{K} w_k\, \ell_k.$$
2. **Selector loss (cross-entropy over exits):**
$$\mathcal{L}_{\text{sel}} \;=\; \mathrm{CE}(q, k^\star).$$

Total:

$$\mathcal{L}_{\text{total}} \;=\; \mathcal{L}_{\text{exit}} \;+\; \lambda_{\text{sel}}\mathcal{L}_{\text{sel}}.$$

5.4 Testing and Outcomes

Testing. At test time (labels unavailable), we cannot compute ℓ_k. The selector provides an actionable decision:

- Compute $q = \operatorname{softmax}(\cdot)$ and choose $\hat{k} = \arg\max_k q[k]$.
- Output prediction $p_{\hat{k}}$ as the model's final decision.

Outcomes.

- The model automatically selects different exits for different tasks/samples.
- Validate against Phase 0A: does the distribution of chosen $\hat{k}$ align with the lengths that were optimal in the manual sweep?
- Potentially improve performance by avoiding overly long context when it hurts and by adapting length under missingness.

5.5 Interpretability for Idea 2

- **Chosen-length histograms:** report $\hat{k}$ (or minutes) distributions by task, cohort, and missingness condition.
- **Why it chose longer:** correlate larger $\hat{k}$ with missing modalities or low-confidence indicators.
- **Masking validation:** if combined with coverage/segment importance, mask selected segments and quantify performance drop vs random masks.

5.6 Possible Add-on: Risk-Controlled Early Exit (Safety Under Shift)

After training, calibrate confidence thresholds on a validation set to only exit early when risk is controlled. This can make the method more conservative under dataset shift:

- Calibrate confidence per exit on validation.
- At test time, only accept early exit if confidence passes calibrated threshold; otherwise fall back to later exits or abstain.

5.7 Possible Add-on: Coverage Selection (Where to Look) for Night-Level Tasks

For night-level outcomes, augment length selection with coverage selection:

- **Option C1:** choose among {start, middle, end, full} and then choose exit $\hat{k}$ within the selected window.
- **Option C2:** cheap coarse scan predicts window start/duration; run exits within that window and select $\hat{k}$.

6 Problem B: Distribution Change Robustness

We focus on robustness under montage variation, missing channels, artifacts, sampling mismatches, and cross-dataset transfer. We propose two complementary directions: safe label-free test-time adaptation (Idea 3) and modality-aware mixture-of-experts routing (Idea 4).

7 Idea 3: PSG Stress Suite + Safe Test-Time Adaptation (TTA) with Safety Gates

7.1 Goal

Measure robustness failures using a PSG-specific stress-test suite (Phase 0B), then design a safe, label-free test-time adaptation procedure that can recover performance under shift *without* drifting and collapsing.

7.2 Train vs Test-Time Separation

- **Training time (source):** train the base predictor on labeled source data (optionally with missingness augmentation).
- **Test time (target, unlabeled):** adapt only a small parameter subset using unlabeled objectives, but *only* when safety gates pass.

7.3 What is Adapted (Tiny Parameter Sets)

We update only a small subset to reduce drift risk:

- **Option P1:** normalization parameters/statistics (e.g., affine scales/shifts).
- **Option P2:** small adapters/LoRA modules in the temporal adapter or head.
- **Option P3:** classifier head only (bias/scale or a small MLP head).
- **Option P4 (optional):** only a modality fusion gate (if present).

7.4 Unlabeled TTA Objectives (Label-free)

Given an unlabeled test batch, we minimize an unsupervised objective:

- **Option U1: entropy minimization:** encourage confident predictions on target data.
- **Option U2: augmentation consistency:** predictions should match under mild perturbations (noise/cropping/masking).
- **Option U3 (optional): prototype consistency:** keep embeddings close to source prototypes of predicted class.

7.5 Safety Gates (Update Only if All Pass)

We apply updates only if all three gates pass:

1. **Gate 1 — Confidence gate (entropy).** Compute prediction entropy $H(p)$. Update only if:
$$H(p) < \tau_H.$$
2. **Gate 2 — Prototype margin gate.** Precompute fixed class prototypes from source training (mean embedding per class). For a sample embedding, compute cosine similarities to prototypes; let s_1 and s_2 be top-1 and top-2 similarities. Update only if:
    - predicted class matches the top prototype class, and
    - $(s_1 - s_2) > \tau_M$.
3. **Gate 3 — Drift control (rollback/stop).** Maintain an EMA copy of adapted parameters. Track a rolling prediction distribution over the last W batches. If the distribution shifts too abruptly (e.g., KL divergence exceeds τ_D), rollback to EMA and freeze updates for N steps.

7.6 Test-Time Adaptation Loop (What Happens at Inference)

For each unlabeled target batch:

- Compute predictions and gate statistics.
- If all gates pass: take a small gradient step on the unlabeled objective (U1/U2/U3) updating only the chosen parameter subset (P1–P4).
- Else: do not update (and possibly rollback/freeze under Gate 3).

7.7 Evaluation and Outcomes

- Evaluate on Phase 0B stress tests and cross-dataset transfer.
- Show recovery of performance under shift while avoiding collapse (compare gated vs ungated TTA).

7.8 Interpretability for Idea 3

- Gate pass rates: fraction of samples/batches passing each gate.
- Rollback analysis: when drift is detected and how often updates are frozen.
- Confidence/margin trajectories over time (per night or per subject).

7.9 Core Ablations for Idea 3

- No TTA vs TTA (ungated) vs TTA (gated).
- Remove each gate (Gate 1 / Gate 2 / Gate 3) to show necessity.
- Parameter subset: head-only vs adapter-only vs norm-only.
- Unlabeled objective: entropy vs consistency vs hybrid.

8 Idea 4: Modality-Aware Mixture-of-Experts (MoE) Routing for PSG Robustness

8.1 Goal

Improve robustness to missing/unreliable modalities and montage variability by using multiple small specialist sub-models (experts) and a router that selects (and mixes) experts based on available signals. This is designed to go beyond simple modality weighting, by enabling different *mappings* under different modality regimes.

8.2 Architecture

- **Router:** a small network that takes missingness mask $\mathbf{z}$ (and optional reliability proxies) and outputs expert weights.
- **Experts:** multiple lightweight specialists (recommended: small temporal adapters, not only MLP heads) that can encode different modality regimes and temporal cues.
- **Combiner:** weighted mixture of expert outputs (or expert states) to form the final prediction.

Expert design options.

- **Option E1 (modality-regime experts):** experts specialize in EEG-rich vs ECG/Resp-dominant vs missing-EEG regimes.
- **Option E2 (timescale experts):** experts specialize in short vs long temporal evidence (can connect to sufficiency).
- **Option E3 (artifact-robust experts):** experts specialize in noisy conditions (learned via artifact augmentation).

8.3 Training

- **Supervised task loss:** train end-to-end on labeled source data.
- **Missingness/montage augmentation (recommended):** simulate realistic channel drop and montage changes during training so the router learns meaningful routing.
- **MoE regularization (recommended):** use a load-balancing term to prevent routing collapse to one expert.

8.4 Testing and Outcomes

- Evaluate robustness on Phase 0B stress tests and cross-dataset transfer.
- Compare against a **simple gating baseline** (weighted modality fusion) to demonstrate that MoE provides gains beyond reweighting.

8.5 Interpretability for Idea 4

- Router weight trajectories over time (e.g., per chunk): when EEG is missing, does routing shift toward ECG/Resp experts?
- Expert usage summaries: average expert weights by task, dataset, and stress condition.

8.6 Core Ablations for Idea 4

- Simple modality gating (baseline) vs MoE routing.
- MLP-only experts vs temporal experts (stronger, more distinct).
- Time-varying routing (per chunk) vs one routing per night.
- With vs without load balancing regularization (collapse ablation).
- With vs without missingness/montage augmentation.

8.7 Optional Integration with Problem A (Context Robustness)

MoE can be combined with CSL (Idea 1) to improve robustness to varying context length:

- Train MoE under truncated multi-view batches (short/long views).
- Apply selective consistency (confidence-gated or better-with-label gated) to stabilize behavior across lengths.
- Produce sufficiency curves under different missingness regimes.