DGSIN: DISCREPANCY-GATED SOURCE–INSTANCE NORMALIZATION FOR CROSS-DOMAIN TIME-SERIES SEMANTIC SEGMENTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Cross-domain time-series semantic segmentation aims to label temporal segments in unseen domains. Representative approaches use feature alignment or statistical normalization to reduce feature discrepancies across domains. However, these discrepancies can reflect both domain factors and semantic variation, suppressing them may also weaken the features that help distinguish semantic states. Normalization methods in visual recognition learn a trade-off between batch and instance normalization, but their input-independent mixing coefficients do not explicitly respond to changes between temporal segments. We propose discrepancy-gated source–instance normalization (DGSIN), which learns segment-dependent trade-offs through the segmentation objective. DGSIN uses source–instance statistical discrepancies to generate layer–channel coefficients that control the correction from source-normalized to instance-normalized representations. At inference, a shared network averages prediction probabilities under different frozen source statistics. Across five leave-one-dataset-out sleep-staging tasks, DGSIN achieves 72.60% mean macro-F1, 74.45% balanced accuracy, and 0.690 Cohen's κ. Its mean macro-F1 exceeds DSON's 72.14% and PSDNorm's 71.47%, with the highest macro-F1 on four targets. DGSIN provides a concrete approach to learning segment-dependent source–instance normalization trade-offs for cross-domain temporal segmentation. Source code is available at `https://anonymous.4open.science/r/Dgsin-E9DB/`.

1 INTRODUCTION

Time-series semantic segmentation assigns a label to each segment using information from the segment itself and its temporal context. In cross-domain settings, models must make accurate predictions on unseen domains whose data distributions differ from those encountered during training. Sleep staging exemplifies this challenge: each physiological epoch must be assigned a stage label by jointly modeling its waveform characteristics and temporal context (Perslev et al., 2019; Phan et al., 2019). Beyond learning these representations, the model must also determine how to handle statistical variations while preserving information relevant to semantic discrimination.

Representative approaches seek cross-dataset generalization through feature alignment, statistical normalization, or related consistency objectives. SleepDG aligns epoch- and sequence-level features (Wang et al., 2024), PSDNorm adjusts temporal correlations through spectral statistics (Gnassounou et al., 2026), and FF-TRUST studies time–frequency consistency under noisy labels (Wang et al., 2026). A less-explored question is how strongly each segment should rely on instance rather than source statistics. Statistical differences can reflect domain factors, semantic states, or both. Suppressing them indiscriminately may therefore also attenuate task-relevant variation, while a smaller cross-domain discrepancy alone does not establish that the resulting representation better supports labeling.

We examine this premise empirically using a separate source-only diagnostic protocol. Table 1 shows that the same set of 22 handcrafted EEG/EOG statistical descriptors supports both source-domain prediction within a fixed sleep stage and sleep-stage prediction within a fixed source domain. We then compare SleepDG with a matched NoAlign variant that retains the classification and reconstruction

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Table 1: Statistical descriptors are informative about both source identity and sleep stage. Both probes use the same set of 22 handcrafted EEG/EOG statistical descriptors.

Probe task	Mean AUC	BAcc
Fixed stage → source domain	0.8366	0.6197
Fixed source → sleep stage	0.8808	0.6381

Table 2: Effect of SleepDG alignment on domain discrepancy and representation separability. NoAlign retains the classification and reconstruction objectives but removes epoch- and sequence-level alignment. $|\Delta|$ denotes the absolute difference.

Diagnostic	NoAlign	SleepDG	$\lvert\Delta\rvert$
Epoch-level discrepancy	0.0659	0.0009	0.0650
Sequence-level discrepancy	0.9230	0.0001	0.9229
Source-domain probe AUC	0.9186	0.8931	0.0255
Source-domain probe BAcc	0.7620	0.7156	0.0464
Sleep-stage probe AUC	0.9231	0.9233	0.0002
Sleep-stage probe BAcc	0.7299	0.7277	0.0022
Held-out target MF1 (%)	67.62	66.99	0.64
Held-out target BAcc (%)	70.02	69.32	0.71

objectives but removes epoch- and sequence-level alignment. As summarized in Table 2, alignment substantially reduces the measured cross-domain discrepancies and source-domain separability, whereas sleep-stage separability remains essentially unchanged and held-out target performance does not improve consistently across datasets. Together, these diagnostics indicate that statistical variation is not purely domain-specific and that reducing domain discrepancy alone does not establish that the resulting representation better supports semantic labeling.

These observations suggest that statistical adjustment should be treated as a learnable choice rather than an objective to be maximized uniformly across inputs. Normalization methods in visual recognition provide a concrete way to realize such a choice by learning a trade-off between population-level and instance statistics. Specifically, batch–instance normalization (BIN) combines batch-normalized and instance-normalized outputs, whereas domain-specific optimized normalization (DSON) uses learned weights to combine domain-specific batch and instance means and variances (Nam & Kim, 2018; Seo et al., 2020). We build on this perspective for cross-domain temporal segmentation, where the appropriate trade-off may vary across temporal segments.

Even under approximately unchanged acquisition conditions, consecutive segments can correspond to different semantic states and exhibit different activation patterns across learned feature channels. A normalization coefficient that remains fixed with respect to the current input cannot capture segment-wise changes in the balance between source and instance statistics. BIN learns channel-specific coefficients that are independent of the current input, whereas DSON learns a single coefficient for each source domain and shares it across inputs, layers, and channels. Their outputs still vary with the input features and the corresponding instance statistics, but the coefficients controlling the normalization trade-off remain fixed with respect to the current input. Consequently, these methods cannot adapt the strength of the normalization adjustment when the statistical characteristics of the current segment change. This limitation motivates an input-dependent source–instance trade-off with finer granularity across layers and channels.

Figure 1 summarizes the motivation for a finer-grained source–instance trade-off that can respond to the current segment. To realize this trade-off, we introduce discrepancy-gated source–instance normalization (DGSIN), which compares the current segment–modality feature statistics with source reference statistics and uses the resulting discrepancy to determine the normalization correction. The backbone and gate response parameters are jointly optimized under the supervised segmentation objective. Unlike DSON, which uses one learned coefficient per source domain, DGSIN learns gate response parameters separately for each source, layer, and channel. For each segment, the current source–instance statistical discrepancy is mapped through these response parameters to produce the corresponding correction coefficients, providing a finer-grained, input-dependent normalization trade-off. The remaining trainable network parameters are shared across sources. At inference, the

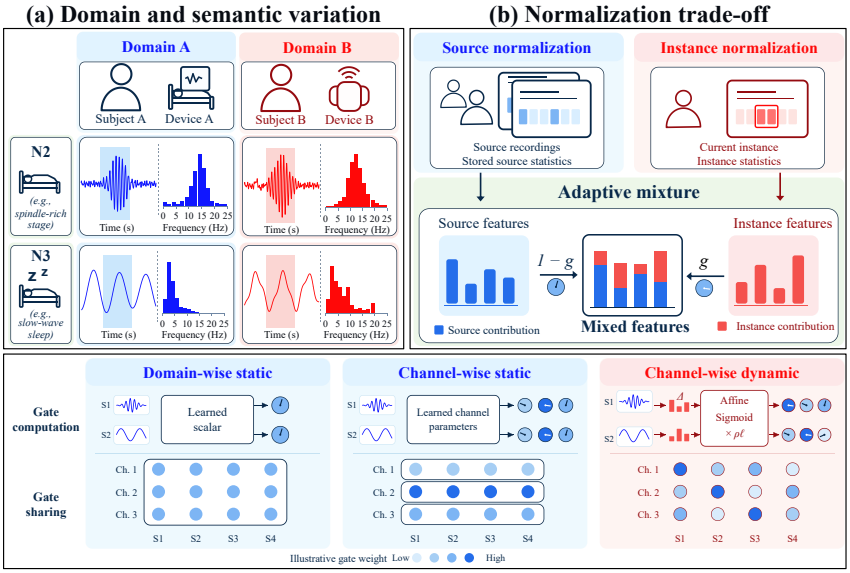

Figure 1: Motivation for DGSIN. (a) Domain and sleep-stage variation can both alter signal characteristics. (b) Source–instance normalization trades off source-normalized and instance-normalized representations. (c) Domain-wise static, channel-wise static, and discrepancy-conditioned dynamic gating differ in coefficient granularity and input dependence. Colors indicate illustrative gate coefficients, not measured values.

shared network is run once for each source using the corresponding frozen source statistics and gate response parameters, and the resulting prediction probabilities are averaged. No target labels or test-time parameter updates are required.

Our contributions are twofold. First, we formulate source–instance normalization as a segment-dependent trade-off for cross-domain temporal segmentation and provide diagnostic evidence that statistical variation is associated with both source identity and sleep stage, while reducing domain discrepancy alone does not establish improved semantic labeling. Second, we introduce DGSIN, a discrepancy-gated normalization module that parameterizes the source–instance trade-off at the source–layer–channel level and uses each segment's statistical discrepancy from a source reference to produce segment-specific correction coefficients. This design provides a structured, source-referenced form of input-adaptive normalization while retaining a shared backbone across source domains.

We evaluate DGSIN across five leave-one-dataset-out sleep-staging tasks and analyze source-conditioned predictions from the same checkpoints. DGSIN achieves 72.60% mean macro-F1 in the main benchmark. We retrain ablation variants to compare coarse and fine static parameterizations and evaluate discrepancy-dependent adaptation against a matched fine-static control. Branch comparisons within each checkpoint separately assess the effect of multi-source probability fusion. Together with the motivating diagnostics, these experiments examine overall system performance, normalization design, and multi-source aggregation.

2 Related Work

Domain generalization for sleep staging. Sleep staging can be viewed as a fixed-epoch time-series segmentation task, where each physiological epoch is labeled using local signal characteristics and temporal context. Generalizable sleep staging has been studied under both unseen-subject and unseen-dataset settings. For cross-subject generalization, MSTGCN combines adversarial domain generalization with spatial–temporal graph modeling (Jia et al., 2021), SIDA introduces stage-specific domain adversarial learning (Ma et al., 2024), and DREAM learns subject-invariant representations while modeling sleep dynamics (Lee et al., 2025). For cross-dataset generalization, which is the setting considered in this work, SleepDG aligns epoch- and sequence-level features across source

datasets (Wang et al., 2024), PSDNorm normalizes temporal correlation structure through spectral statistics (Gnassounou et al., 2026), and FF-TRUST studies time–frequency consistency under noisy multi-source supervision (Wang et al., 2026). These methods improve generalization through domain-invariant representation learning, feature alignment, statistical normalization, or consistency regularization. DGSIN complements these directions by explicitly learning how strongly each temporal segment should rely on source versus instance statistics.

Statistical Trade-offs in Normalization. A central distinction among normalization methods is how they use population-level and instance-specific statistics. BN and IN compute normalization statistics over different populations (Ioffe & Szegedy, 2015; Ulyanov et al., 2016). IBN-Net incorporates both operations within visual networks, BIN learns channel-wise mixtures of batch-normalized and instance-normalized outputs, and DSON uses learned domain-specific weights to combine batch and instance means and variances (Pan et al., 2018; Nam & Kim, 2018; Seo et al., 2020). BNE takes a different approach, using normalization statistics to characterize relationships among source domains and aggregate predictions (Segu et al., 2023). DGSIN builds on this normalization trade-off perspective but makes the correction input dependent: at each layer, it uses the current segment–modality feature's statistical discrepancy from a source reference to adapt channel-wise correction coefficients.

Adaptive Normalization in Vision and Time Series. Adaptive normalization has been studied in both vision and time-series modeling. In vision, instance-level meta-normalization predicts normalization parameters from the current instance, while dynamically transformed instance normalization uses input-dependent information to transform instance-normalized features (Jia et al., 2019; Jiao et al., 2022). In time series, DAIN adaptively shifts, scales, and gates the input sequence, RevIN combines instance normalization with an inverse transformation for forecasting, and Non-stationary Transformers examine information loss introduced by stationarization (Passalis et al., 2020; Kim et al., 2021; Liu et al., 2022). DGSIN shares the general idea of adapting normalization to the current input, but differs in both its conditioning signal and the quantity being adapted. It uses the statistical discrepancy between the current segment and a source reference to control the trade-off between source-normalized and instance-normalized representations at the layer and channel levels.

3 Discrepancy-Gated Source–Instance Normalization

3.1 Problem Setup and Source–Instance Normalization Trade-off

We are given K labeled source datasets $\{\mathcal{D}_k\}_{k=1}^{K}$ and evaluate the model on an unseen target dataset. Each input sequence $x \in \mathbb{R}^{L\times M\times T_0}$ contains L valid segments, M modalities, and T_0 samples per segment, with corresponding labels $y \in \{0,\dots,Q-1\}^{L}$. Both model training and checkpoint selection use source data only. In our experiments, $L = 20$, $M = 2$, $T_0 = 3000$, $Q = 5$, and $K = 4$. Segments with invalid labels are removed before sequence construction, which may introduce gaps in the original recording timeline. The task is to assign a semantic label to each fixed-duration segment using its surrounding sequence context.

Neither source nor instance normalization is universally appropriate. In our setting, the relevant trade-off concerns learned intermediate features after input z-scoring; Appendix C provides a feature-space analysis of this trade-off.

3.2 Normalization Endpoints for Intermediate Features

Figure 2 provides an overview of the network architecture and the DGSIN normalization sites. The waveform encoder consists of the stem and two Inception blocks and processes EEG and EOG using shared weights. At DGSIN site ℓ, let $H_n^{(\ell)} \in \mathbb{R}^{C_\ell\times T_\ell}$ denote the learned intermediate feature immediately before normalization for the n-th segment–modality pair. For a batch of B input sequences, flattening the L segment and M modality dimensions yields $N = B \times L \times M$ such instances. Each $H_n^{(\ell)}$ is treated as an instance. Thus, an instance is neither an entire recording nor the full L-segment sequence. Figure 2 suppresses the instance and layer indices and denotes the features of a single instance simply as $H \in \mathbb{R}^{C\times T}$.

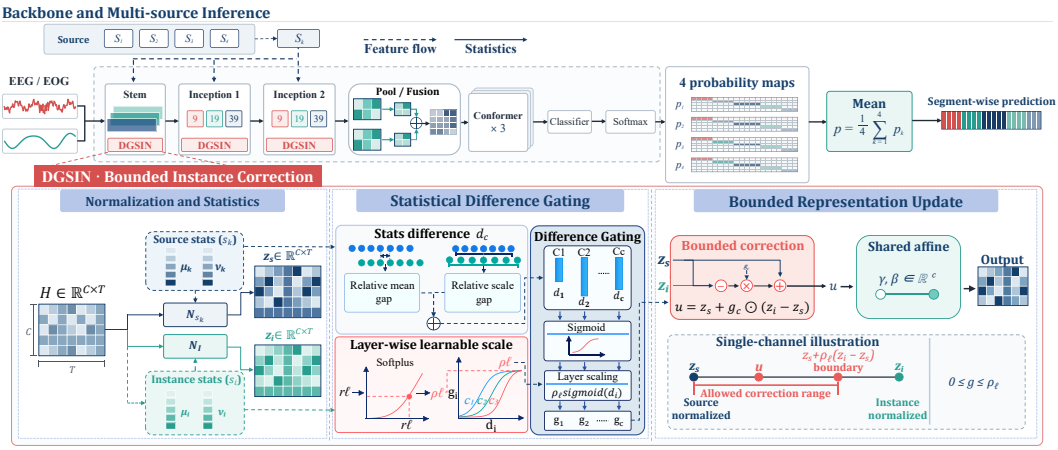

Figure 2: Overview of DGSIN and multi-source inference. DGSIN replaces the normalization layers in the stem and two Inception blocks, using source–instance statistical discrepancies to produce channel-wise correction coefficients with a learnable layer-wise scale. At inference, the shared network is evaluated with each frozen source-statistics bank, and the resulting prediction probabilities are averaged.

Suppressing the layer index ℓ, let $(\mu_{k,c}, \nu_{k,c})$ denote the source mean and variance for source k and channel c, and let $(\mu_{I,nc}, \nu_{I,nc})$ denote the mean and variance computed over the temporal positions of instance n. We define

$$z_{s,nkct} = \frac{H_{nct} - \mu_{k,c}}{\sqrt{\nu_{k,c} + \varepsilon}}, \qquad z_{i,nkct} = \frac{H_{nct} - \mu_{I,nc}}{\sqrt{\nu_{I,nc} + \varepsilon}}, \tag{1}$$

where $\varepsilon = 10^{-5}$. During training, source statistics are computed separately from the instances belonging to each source-specific subbatch and update the corresponding running-statistics bank; at inference, source normalization uses the frozen running statistics. Instance statistics are recomputed from the current instance's features in both training and inference. Appendix A.1 gives the corresponding estimators and update rules.

The source index k on z_i denotes the source-conditioned forward route rather than a direct dependence of instance normalization on source statistics. Because preceding DGSIN layers use source-specific statistics and gate response parameters, different source-conditioned passes can produce different intermediate features for the same input; each pass therefore recomputes its own instance statistics.

3.3 Discrepancy-Gated Correction

For each instance n, source k, and channel c, we define the dimensionless source–instance statistical discrepancy as

$$d_{nkc} = \frac{|\mu_{I,nc} - \mu_{k,c}|}{\sqrt{\nu_{k,c} + \varepsilon}} + \frac{1}{2}\left|\log \frac{\nu_{I,nc} + \varepsilon}{\nu_{k,c} + \varepsilon}\right|. \tag{2}$$

The first term measures the mean difference relative to the source standard deviation, while the second measures the absolute log-ratio between the regularized instance and source standard deviations. Together, they quantify the statistical mismatch between the current features and the source reference.

The gate maps this discrepancy to a correction coefficient:

$$g^{(\ell)}_{nkc} = \rho_\ell \ \mathrm{sigmoid}(b_{k\ell c} + \mathrm{softplus}(a_{k\ell c}) \ \mathrm{sg}(d_{nkc})), \tag{3}$$

where $b_{k\ell c}$ and the positive sensitivity $\mathrm{softplus}(a_{k\ell c})$ are learned separately for each source, layer, and channel, and $\mathrm{sg}(\cdot)$ denotes stop-gradient. The layer-wise scale is

$$\rho_\ell = \mathrm{softplus}(\alpha r_\ell + \eta), \qquad 0 \le g^{(\ell)}_{nkc} \le \rho_\ell, \tag{4}$$

where r_ℓ is trainable and α, η are fixed constants chosen such that ρ_ℓ is initialized to 0.25. The response parameters are source-specific, whereas ρ_ℓ and the channel-wise affine parameters are

shared across sources. For fixed response parameters, the gate is monotone in d_{nkc}, introducing a bias toward greater instance reliance as the mismatch from the source reference increases. Only the discrepancy supplied to the gate is detached; both normalization endpoints remain differentiable. Initialization and gradient details are given in Appendix A.2.

DGSIN applies the resulting correction directly between the source-normalized and instance-normalized representations:

$$u_{nkct} = z_{s,nkct} + g^{(\ell)}_{nkc}\left(z_{i,nkct} - z_{s,nkct}\right), \qquad Y_{nkct} = \gamma_{\ell c} u_{nkct} + \beta_{\ell c}. \tag{5}$$

The coefficient $g^{(\ell)}_{nkc}$ is shared over temporal positions within channel c. When $g^{(\ell)}_{nkc} = 0$, the representation remains at the source-normalized endpoint; when $g^{(\ell)}_{nkc} = 1$, it reaches the instance-normalized endpoint. Values between zero and one interpolate between the endpoints, while values above one extrapolate beyond the instance-normalized endpoint. The checkpoints analyzed in our experiments operate in the interpolation regime. Appendix C derives the corresponding effects on feature means, standard deviations, and representation displacement.

Unlike DSON, which combines batch and instance statistics before normalization, DGSIN combines normalized representations in output space. Its source–layer–channel response parameters determine how each channel responds to discrepancy, while the current discrepancy determines the segment-specific correction coefficient.

3.4 Training and Multi-source Inference

The architecture and sleep-specific prediction objective are detailed in Appendix B. These components are kept unchanged across matched normalization comparisons and are not contributions of DGSIN. During training, each sequence follows the route corresponding to its known source identity, and the gate response parameters are optimized jointly with the shared network through the supervised segmentation objective. No additional loss minimizes the source–instance statistical discrepancy or explicitly encourages the gate toward instance normalization.

At test time, the source identity of the target input is unknown. For each segment e, we compute

$$p_e(y \mid x) = \frac{1}{K}\sum_{k=1}^{K} \mathrm{softmax}(f_\theta(x; S_k)_e)_y\,, \qquad \hat{y}_e = \arg\max_y p_e(y \mid x), \tag{6}$$

where S_k denotes the frozen statistics bank and source-specific gate response parameters associated with source k. Softmax is applied within each source-conditioned pass before probability averaging. The entire shared network is evaluated once per source, while all trainable parameters and source-statistics banks remain fixed. Instance statistics and correction coefficients are recomputed from the current features in each pass. The predictions are combined with uniform weights and require neither target labels, target-population statistics, nor test-time parameter optimization.

4 Experiments

We evaluate held-out-dataset performance, normalization design, discrepancy-conditioned behavior, and probability fusion, then measure the cost of source-conditioned inference.

4.1 Evaluation Setting

Data and task. We evaluate on SleepEDF, HMC, ISRUC, SHHS1, and P2018 (Kemp et al., 2000; Alvarez-Estevez & Rijsman, 2021; Khalighi et al., 2016; Quan et al., 1997; Ghassemi et al., 2018). Each leave-one-dataset-out task uses four datasets for training and validation and the fifth as the target. EEG and EOG are sampled at 100 Hz and organized into sequences of twenty 30-second epochs, labeled W, N1, N2, N3, or REM.

Model selection and evaluation. Source data were split into 80/20 sequence blocks, where the training split was used for parameter optimization and the validation split was used for model selection and checkpoint saving. This split was performed at the sequence-block level without additionally enforcing recording- or subject-level disjointness. Both optimization objectives and accuracy-based

checkpoint selection relied exclusively on source-domain validation performance. The selected model was subsequently evaluated on multiple target-domain benchmarks to assess its cross-domain generalization ability.

We report five-class macro-F1 (MF1) as the primary metric, balanced accuracy (BAcc; mean class recall), and unweighted Cohen's κ. All use the same valid predictions within each target and are averaged equally across targets.

Comparison scope. Comparison scope. DSON, BNE, PSDNorm, DGSIN, and the standard-normalization controls share the same waveform–temporal backbone and a 20-epoch training budget. The BatchNorm, LayerNorm, and InstanceNorm controls replace the corresponding normalization modules while retaining the same preprocessing pipeline, optimization objective, and checkpoint selection strategy. An adapted one-dimensional implementation of BIN (Nam & Kim, 2018) is evaluated using the same data split and training budget. SleepDG and FF-TRUST retain their original architectures and are trained for 50 epochs; FF-TRUST is evaluated without additional label noise. Implementation details and comparison protocols are provided in Appendix D.

4.2 Overall Held-out-Dataset Performance

Table 3: Main leave-one-dataset-out benchmark. I–V denote SleepEDF, HMC, ISRUC, SHHS1, and P2018. SD lists the four training source datasets; TD is the held-out target. ACC, MF1, and BAcc are percentages; κ is unweighted Cohen's kappa. Avg is the equal-target mean. Mean branch averages scores from four source branches; fusion scores their averaged probabilities. Bold marks column maxima; underlining marks the second-highest distinct value in each column.

SD	II, III, IV, V		I, III, IV, V		I, II, IV, V		I, II, III, V		I, II, III, IV		–			
TD	I		II		III		IV		V		Avg			
Method	ACC	MF1	ACC	MF1	ACC	MF1	ACC	MF1	ACC	MF1	ACC	MF1	BAcc	κ
FF-TRUST	77.21	71.49	74.38	72.38	77.02	73.76	74.48	66.10	72.97	68.49	75.21	70.44	72.83	0.670
BatchNorm	73.87	69.30	74.13	72.17	75.95	74.02	74.85	68.07	73.42	71.33	74.44	70.98	73.53	0.663
LayerNorm	75.02	69.76	74.20	71.78	75.06	73.13	74.12	67.29	74.19	71.54	74.52	70.70	72.93	0.662
InstanceNorm	75.66	70.67	75.07	**72.85**	76.34	73.74	77.52	69.96	73.60	70.96	75.64	71.64	73.94	0.677
BIN	74.38	69.79	73.64	71.52	76.12	74.03	75.38	68.46	73.75	71.87	74.66	71.13	73.52	0.665
DSON	**77.56**	71.54	**75.13**	72.13	77.05	74.53	77.05	69.85	75.98	72.64	76.55	72.14	73.78	0.686
BNE	75.41	70.34	71.98	69.84	74.83	72.72	74.54	67.53	72.98	70.23	73.95	70.13	72.50	0.655
SleepDG	76.01	70.93	73.48	71.46	**77.47**	74.18	73.08	65.51	73.16	68.89	74.64	70.19	72.73	0.664
PSDNorm	76.11	71.50	74.60	72.44	75.34	72.80	76.50	68.88	74.56	71.72	75.42	71.47	73.57	0.673
DGSIN (mean branch)	76.51	71.42	73.28	70.69	75.91	73.87	77.29	69.52	74.75	72.20	75.55	71.54	73.45	0.675
DGSIN (fusion)	77.31	**71.95**	74.79	72.06	77.12	**74.98**	**78.25**	**70.50**	**76.41**	**73.53**	**76.78**	**72.60**	**74.45**	**0.690**

As shown in Table 3, DGSIN achieves 72.60% mean MF1, compared with 72.14% for DSON, the strongest baseline on this metric. Its corresponding equal-target means are 74.45% for BAcc and 0.690 for Cohen's κ. DGSIN obtains the highest MF1 on SleepEDF, ISRUC, SHHS1, and P2018, while InstanceNorm performs best on HMC.

Among the standard-normalization controls, InstanceNorm achieves the highest mean MF1 at 71.64%. Complete target-level metrics are provided in Appendix G.

4.3 Analysis of Discrepancy-Conditioned Correction

Normalization controls. All three configurations are independently retrained with matched preprocessing, backbone, objective, and checkpoint selection. Without discrepancy conditioning, DGSIN uses learned source–layer–channel coefficients fixed across inputs, with layer-wise scales shared across sources. This is the fine-static control; coarse static uses one coefficient and scale per source, changing both granularity and scale sharing. DGSIN dynamically generates correction coefficients based on the discrepancy between source and instance statistics for each input, whereas both static controls use fixed coefficients while still computing normalization statistics from the current instance.

Table 4 shows that fine static increases mean MF1 from 70.94% to 71.97%, while DGSIN reaches 72.60%. Relative to fine static, DGSIN improves MF1 on four targets but lowers it by 0.27 percentage

Table 4: Normalization ablations (MF1, %). All variants are independently retrained. Mean averages the five targets equally; bold marks column maxima.

Configuration	SleepEDF	HMC	ISRUC	SHHS1	P2018	Mean
Coarse static	70.83	71.06	73.12	69.17	70.53	70.94
w/o discrepancy conditioning	71.20	**72.33**	73.97	70.24	72.09	71.97
DGSIN (full)	**71.95**	72.06	**74.98**	**70.50**	**73.53**	**72.60**

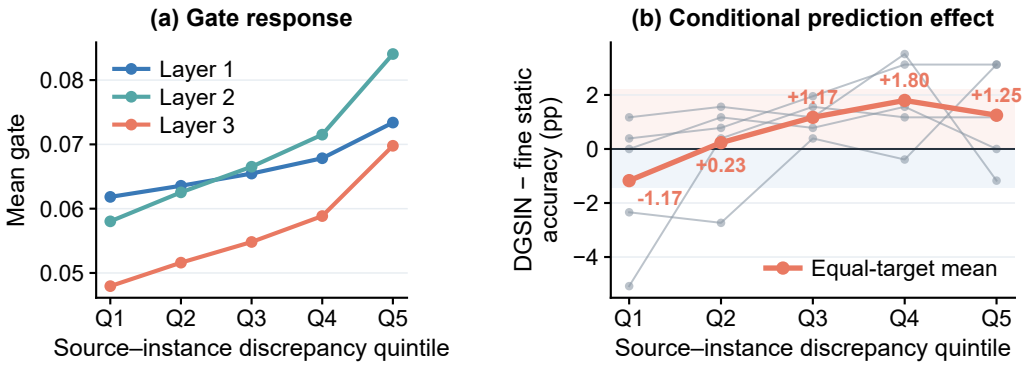

Figure 3: Discrepancy-conditioned correction. (a) Mean correction coefficients at three layers. (b) Accuracy differences between DGSIN and fine static, per target (gray) and averaged equally (orange). Each panel groups discrepancies from Q1 (lowest) to Q5 (highest).

points on HMC. The benefit therefore varies across targets. Appendix E presents paired evaluation-cluster analyses for fixed checkpoints.

Gate response and conditional performance. Figure 3(a) forms discrepancy quintiles within each target, separately by source-conditioned branch, layer, modality, and channel. Mean correction coefficients increase across quintiles at all three layers, consistent with the monotone gate in Equation 3. For these checkpoints, larger coefficients increase reliance on the instance-normalized representation.

Figure 3(b) instead groups segments within each target using discrepancies averaged over branches, layers, modalities, and channels. On average across targets, DGSIN improves accuracy in Q3–Q5, with the largest gain of 1.80 percentage points in Q4. Fine static performs better in Q1 by 1.17 percentage points, while the average difference in Q2 is small. Effects vary across targets and are not monotonic in discrepancy. The panels use different groupings; their juxtaposition does not establish a causal link between correction coefficients and accuracy.

4.4 Source-Conditioned Inference

Following Equation 6, we evaluate four source-conditioned routes through the same shared network. Each route recomputes instance statistics and correction coefficients with all learned parameters and source-statistics banks fixed. We compare mean branch MF1 with MF1 after probability fusion.

Figure 4 shows that probability fusion exceeds the mean branch MF1 on all five targets. The equal-target mean increases from 71.54% to 72.60%, with target-wise gains ranging from 0.53 to 1.37 percentage points. Some individual branches nevertheless score higher than fusion, as detailed in Appendix F.

4.5 Computational Cost

We measure full-model inference on an RTX A6000 using the HMC checkpoints and batches of 20-epoch sequences.

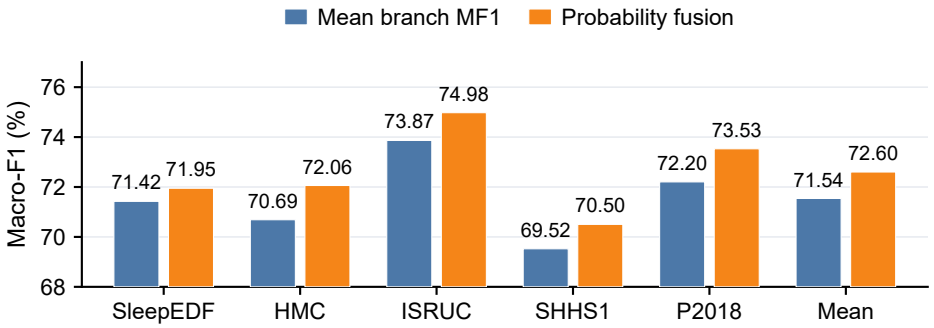

Figure 4: Within-checkpoint probability fusion. Mean branch MF1 averages four individual branch scores; probability fusion averages class probabilities before prediction and scoring.

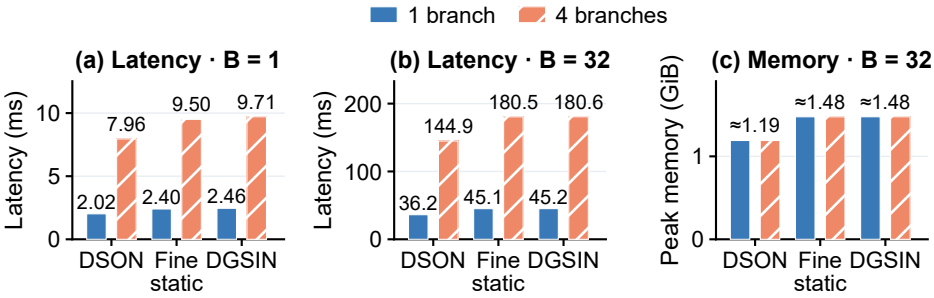

Figure 5: Inference cost on an RTX A6000. One- and four-branch latency and peak allocated memory are measured on batches of 20-epoch sequences.

As shown in Figure 5, DGSIN latency at $B = 32$ increases from 45.2 to 180.6 ms when evaluating four routes, approximately fourfold. Four-branch fine static takes 180.5 ms, indicating comparable measured latency for dynamic and static conditioning. Sequential execution keeps peak allocated memory close to the single-branch level. Full measurements and timing procedures are provided in Appendix H.

5 Discussion and Conclusion

Our central perspective is to learn statistical adjustment alongside segmentation features. Table 2 shows that reducing measured discrepancy alone does not guarantee better target labeling. DGSIN extends visual normalization trade-offs to discrepancy-conditioned responses across segments, layers, and channels and is competitive across five datasets. Its mean–variance descriptor cannot distinguish acquisition from task-state changes, while its monotone, time-constant gate limits the available transformations.

Discrepancy conditioning increases mean MF1 from 71.97% to 72.60% in the matched ablation, with improvements on four targets and a decrease on HMC. The perturbation analyses in Appendix E show that this benefit also depends on noise structure and strength. Richer descriptors and state-transition analyses could clarify when conditioning helps.

The evaluation concerns sleep staging; source training and validation can share recordings, whole-method budgets differ, and benchmark results informed architecture development. Broader tasks, independent development data, and repeated matched training would better establish generality.

DGSIN achieves 72.60% mean macro-F1 across five sleep datasets, and probability fusion improves over individual source branches. Diagnostic and matched-control analyses support learning adaptive source–instance normalization trade-offs.

AI USE STATEMENT

Generative AI was used to assist with literature review and organization, structuring the manuscript argument, and preparing manuscript materials. All scientific decisions, experimental designs, implementations, analyses, and reported results were conducted and verified by the authors.

ETHICS STATEMENT

This study analyzes physiological recordings for sleep-stage prediction and does not establish clinical readiness. Performance differences across datasets and the comparatively weak N1 results warrant caution about reliability for particular populations or acquisition settings. No claim is made about patient-level benefit, diagnostic safety, or prospective deployment.

REPRODUCIBILITY STATEMENT

Section 3 defines DGSIN and its inference rule. Appendices A and B specify the estimation of source and instance means and variances, initialization, gradient computation, architecture, and training objective. Appendix C gives the mathematical derivations. Appendix D describes the data splits, baseline configurations, evaluation metrics, and resampling units. Perturbation analyses and source-branch results appear in Appendices E and F, with complete benchmark metrics in Appendix G. Appendix H specifies the hardware, timing procedure, memory measurement, and parameter counts used for the inference-cost comparison.

REFERENCES

Diego Alvarez-Estevez and Roselyne M Rijsman. Inter-database validation of a deep learning approach for automatic sleep scoring. *PloS one*, 16(8):e0256111, 2021. doi: 10.1371/journal.pone.0256111. URL https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0256111.

Mohammad M Ghassemi, Benjamin E Moody, Li-Wei H Lehman, Christopher Song, Qiao Li, Haoqi Sun, Roger G Mark, M Brandon Westover, and Gari D Clifford. You snooze, you win: the physionet/computing in cardiology challenge 2018. In *2018 Computing in Cardiology Conference (CinC)*, volume 45, pp. 1–4. IEEE, 2018. doi: 10.22489/CinC.2018.049. URL https://ieeexplore.ieee.org/abstract/document/8743916.

Theo Gnassounou, Antoine Collas, Rémi Flamary, and Alexandre Gramfort. Psdnorm: Temporal normalization for deep learning in sleep staging. In *International Conference on Learning Representations*, volume 2026, pp. 41514–41539, 2026. URL https://openreview.net/forum?id=BZMQotjBwW.

Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, and Ruoming Pang. Conformer: Convolution-augmented transformer for speech recognition. In *Interspeech*, pp. 5036–5040, 2020. doi: 10.21437/Interspeech.2020-3015. URL https://www.isca-archive.org/interspeech_2020/gulati20_interspeech.html.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. pmlr, 2015. URL https://proceedings.mlr.press/v37/ioffe15.html.

Songhao Jia, Ding-Jie Chen, and Hwann-Tzong Chen. Instance-level meta normalization. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4860–4868. IEEE, 2019. doi: 10.1109/CVPR.2019.00500. URL https://ieeexplore.ieee.org/abstract/document/8953796.

Ziyu Jia, Youfang Lin, Jing Wang, Xiaojun Ning, Yuanlai He, Ronghao Zhou, Yuhan Zhou, and Li-wei H Lehman. Multi-view spatial-temporal graph convolutional networks with domain

generalization for sleep stage classification. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 29:1977–1986, 2021. doi: 10.1109/TNSRE.2021.3110665. URL https://pubmed.ncbi.nlm.nih.gov/34487495/.

Bingliang Jiao, Lingqiao Liu, Liying Gao, Guosheng Lin, Lu Yang, Shizhou Zhang, Peng Wang, and Yanning Zhang. Dynamically transformed instance normalization network for generalizable person re-identification. In *European conference on computer vision*, pp. 285–301. Springer, 2022. doi: 10.1007/978-3-031-19781-9_17. URL https://link.springer.com/chapter/10.1007/978-3-031-19781-9_17.

Bob Kemp, Aeilko H Zwinderman, Bert Tuk, Hilbert AC Kamphuisen, and Josefien JL Oberye. Analysis of a sleep-dependent neuronal feedback loop: the slow-wave microcontinuity of the eeg. *IEEE Transactions on Biomedical Engineering*, 47(9):1185–1194, 2000. doi: 10.1109/10.867928. URL https://ieeexplore.ieee.org/abstract/document/867928.

Sirvan Khalighi, Teresa Sousa, José Moutinho Santos, and Urbano Nunes. Isruc-sleep: A comprehensive public dataset for sleep researchers. *Computer methods and programs in biomedicine*, 124: 180–192, 2016. doi: 10.1016/j.cmpb.2015.10.013. URL https://www.sciencedirect.com/science/article/abs/pii/S0169260715002734.

Taesung Kim, Jinhee Kim, Yunwon Tae, Cheonbok Park, Jang-Ho Choi, and Jaegul Choo. Reversible instance normalization for accurate time-series forecasting against distribution shift. In *International conference on learning representations*, 2021. URL https://openreview.net/forum?id=cGDAkQo1C0p.

Seungyeon Lee, Thai-Hoang Pham, Zhao Cheng, and Ping Zhang. Domain-invariant representation learning and sleep dynamics modeling for automatic sleep staging. *ACM transactions on computing for healthcare*, 6(4):1–20, 2025. doi: 10.1145/3757066. URL https://doi.org/10.1145/3757066.

Yong Liu, Haixu Wu, Jianmin Wang, and Mingsheng Long. Non-stationary transformers: Exploring the stationarity in time series forecasting. In *Advances in neural information processing systems*, volume 35, pp. 9881–9893, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/4054556fcaa934b0bf76da52cf4f92cb-Abstract-Conference.html.

Shuo Ma, Yingwei Zhang, Yiqiang Chen, Tao Xie, Shuchao Song, and Ziyu Jia. Exploring structure incentive domain adversarial learning for generalizable sleep stage classification. *ACM Transactions on Intelligent Systems and Technology*, 15(1):1–30, 2024. doi: 10.1145/3625238. URL https://doi.org/10.1145/3625238.

Hyeonseob Nam and Hyo-Eun Kim. Batch-instance normalization for adaptively style-invariant neural networks. In *Advances in Neural Information Processing Systems*, volume 31, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/018b59ce1fd616d874afad0f44ba338d-Abstract.html.

Xingang Pan, Ping Luo, Jianping Shi, and Xiaoou Tang. Two at once: Enhancing learning and generalization capacities via ibn-net. In *European conference on computer vision*, pp. 484–500. Springer, 2018. URL https://link.springer.com/chapter/10.1007/978-3-030-01225-0_29.

Nikolaos Passalis, Anastasios Tefas, Juho Kanniainen, Moncef Gabbouj, and Alexandros Iosifidis. Deep adaptive input normalization for time series forecasting. *IEEE transactions on neural networks and learning systems*, 31(9):3760–3765, 2020. doi: 10.1109/TNNLS.2019.2944933. URL https://arxiv.org/abs/1902.07892.

Mathias Perslev, Michael Jensen, Sune Darkner, Poul Jørgen Jennum, and Christian Igel. U-time: A fully convolutional network for time series segmentation applied to sleep staging. In *Advances in neural information processing systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/hash/57bafb2c2dfeefba931bb03a835b1fa9-Abstract.html.

Huy Phan, Fernando Andreotti, Navin Cooray, Oliver Y Chén, and Maarten De Vos. Seqsleepnet: end-to-end hierarchical recurrent neural network for sequence-to-sequence automatic sleep staging. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 27(3):400–410, 2019. doi: 10.1109/TNSRE.2019.2896659. URL http://ieeexplore.ieee.org/abstract/document/8631195.

Stuart F Quan, Barbara V Howard, Conrad Iber, James P Kiley, F Javier Nieto, George T O'Connor, David M Rapoport, Susan Redline, John Robbins, Jonathan M Samet, et al. The sleep heart health study: design, rationale, and methods. *Sleep*, 20(12):1077–1085, 1997. doi: 10.1093/sleep/20.12.1077. URL https://doi.org/10.1093/sleep/20.12.1077.

Mattia Segu, Alessio Tonioni, and Federico Tombari. Batch normalization embeddings for deep domain generalization. *Pattern Recognition*, 135:109115, 2023. doi: 10.1016/j.patcog.2022.109115. URL https://doi.org/10.1016/j.patcog.2022.109115.

Seonguk Seo, Yumin Suh, Dongwan Kim, Geeho Kim, Jongwoo Han, and Bohyung Han. Learning to optimize domain specific normalization for domain generalization. In *European conference on computer vision*, pp. 68–83. Springer, 2020. URL https://www.ecva.net/papers/eccv_2020/papers_ECCV/html/3935_ECCV_2020_paper.php.

Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization, 2016. URL https://arxiv.org/abs/1607.08022.

Jiquan Wang, Sha Zhao, Haiteng Jiang, Shijian Li, Tao Li, and Gang Pan. Generalizable sleep staging via multi-level domain alignment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 265–273, 2024. doi: 10.1609/aaai.v38i1.27779. URL https://ojs.aaai.org/index.php/AAAI/article/view/27779.

Kening Wang, Di Wen, Yufan Chen, Ruiping Liu, Junwei Zheng, Jiale Wei, Kailun Yang, Rainer Stiefelhagen, and Kunyu Peng. Towards multi-source domain generalization for sleep staging with noisy labels, 2026. URL https://arxiv.org/abs/2604.10009.

A DGSIN IMPLEMENTATION DETAILS

This appendix describes the estimation of source and instance means and variances, parameter initialization, and the DGSIN training and inference procedures introduced in Section 3. As in the main text, an instance denotes the intermediate features of one segment–modality pair at a given layer.

A.1 MEAN AND VARIANCE ESTIMATION AND SOURCE-STATISTICS UPDATES

At one layer, the instance mean and variance of the intermediate features $H_n \in \mathbb{R}^{C\times T}$ are

$$\mu_{I,nc} = \frac{1}{T}\sum_t H_{nct}, \qquad \nu_{I,nc} = \frac{1}{T}\sum_t (H_{nct} - \mu_{I,nc})^2. \tag{7}$$

For the set $\mathcal{B}_k$ of flattened segment–modality instances belonging to source k, the training source mean and variance are

$$\mu_{kc} = \frac{1}{m_k}\sum_{n\in\mathcal{B}_k,t} H_{nct}, \qquad \nu_{kc} = \frac{1}{m_k}\sum_{n\in\mathcal{B}_k,t} (H_{nct} - \mu_{kc})^2, \quad m_k = |\mathcal{B}_k|T. \tag{8}$$

Both forward variances divide by the number of observations, rather than by one less. The source subbatch spans all available epochs and both modalities for that source. It does not mix statistics across sources. Instance normalization uses neither other epochs nor the other modality.

Only present source banks are updated. With BN momentum $\lambda = 0.1$ and $m_k > 1$,

$$\bar{\mu}_{kc} \leftarrow (1-\lambda)\bar{\mu}_{kc} + \lambda\mu_{kc}, \qquad \bar{\nu}_{kc} \leftarrow (1-\lambda)\bar{\nu}_{kc} + \lambda\frac{m_k}{m_k-1}\nu_{kc}. \tag{9}$$

The running variance uses the unbiased subbatch estimate, whereas the forward pass uses the population estimate. At inference, $(\mu_{kc}, \nu_{kc}) = (\bar{\mu}_{kc}, \bar{\nu}_{kc})$ in Equations 1 and 2; these source statistics remain fixed. We use a numerical stabilizer of 10^{-5} and initialize the running means and variances to zero and one, respectively. The post-mixture affine parameters γ, β are shared across source branches.

Table 5: State and computation scopes. A instance contains the features of one segment–modality pair at the current layer.

Quantity	Scope	Inference behavior
Running means/variances	Source, layer, channel	Frozen
Instance statistics	Instance, layer, channel within a forward	Recomputed
Gate bias and sensitivity b, a	Source, layer, channel	Frozen learned parameters
Scale parameter r	Layer	Frozen learned parameter
Shared affine γ, β	Layer, channel	Frozen learned parameters
Discrepancy and gate d, g	Instance, source, layer, channel	Recomputed
Waveform/temporal/classifier weights	Shared across sources	Frozen

A.2 INITIALIZATION AND RANGE

Let

$$\alpha = \frac{0.125}{1-e^{-0.25}}, \quad \eta = \log(e^{0.25}-1), \quad r_\ell = 0, \quad b_{k\ell c} = \log\frac{0.2}{0.8}, \quad a_{k\ell c} = \log(e^{0.5}-1). \tag{10}$$

Then $\rho_\ell = 0.25$, $\partial\rho_\ell/\partial r_\ell = 0.125$, $\mathrm{softplus}(a_{k\ell c}) = 0.5$, and the initial gate is

$$g(d) = 0.25\ \mathrm{sigmoid}(\log(0.2/0.8) + 0.5d)\,. \tag{11}$$

The initial coefficient is 0.05 at zero discrepancy and increases with d. When source and instance statistics coincide, the endpoints are equal, so the representation correction vanishes despite the nonzero coefficient. We initialize the shared affine parameters to $\gamma = 1$ and $\beta = 0$.

A finite checkpoint gives a finite positive layer scale and hence a bound on its correction coefficients. The softplus parameterization imposes no fixed ceiling of 0.5 or 1. Appendix F reports the learned scales of the evaluated checkpoints.

A.3 Gradient paths and task learning

Write $q = \mathrm{sigmoid}(b + \mathrm{softplus}(a)\,\mathrm{sg}(d))$ and $\Delta = z_i - z_s$. With the endpoint values fixed, the gradients of the pre-affine output with respect to gate parameters are

$$\frac{\partial u}{\partial b} = \rho q(1-q)\Delta, \tag{12}$$

$$\frac{\partial u}{\partial a} = \rho q(1-q)\,\mathrm{sigmoid}(a)\,\mathrm{sg}(d)\Delta, \tag{13}$$

$$\frac{\partial u}{\partial r} = \alpha\,\mathrm{sigmoid}(\alpha r + \eta) q\Delta. \tag{14}$$

The task gradient therefore adjusts how strongly the representation moves along the current source–instance difference. These expressions do not imply that the learned movement is always useful. After the affine map, the corresponding partial derivatives are multiplied by the channel's γ.

With d detached, the backpropagated feature Jacobian combines the ordinary endpoint Jacobians with weights $1-g$ and g. This describes the implemented computational graph, not the full analytical derivative of the input-conditioned value function without detachment. Source normalization still differentiates through its training subbatch statistics, and IN differentiates through its instance statistics. The running statistics are buffers and receive no gradient.

A.4 Training and inference algorithms

Table 6: Training and inference procedure. Each source-conditioned inference branch evaluates the entire shared network with its frozen source statistics and gate response parameters.

Training step: labeled source sequences (x_b, y_b, k_b)	
1	Apply the fixed recording preprocessing and in-network epoch/modality standardization.
2	At each DGSIN site, flatten epochs/modalities and group instances by their known source IDs.
3	Compute source-subbatch statistics and update only the present source banks using Equation 9. Compute the statistics of each instance.
4	Form z_s, z_i, compute the discrepancy, detach only its gate input, and apply Equations 3–5.
5	Fuse modality features, model the 20-epoch context, and compute the common refined logits. Each sequence has followed its known-source route once.
6	Compute Equation 23, backpropagate, clip the gradient norm, and update the source-specific gate response parameters together with the shared network parameters.
Inference: preprocessed target sequence x, frozen $\theta, S_1, \ldots, S_K$	
1	For each k, evaluate $f_\theta(x; S_k)$ in evaluation mode, where S_k contains the source-statistics bank and gate response parameters. Recompute instance statistics and correction coefficients while keeping parameters and buffers fixed.
2	Apply softmax to each branch's refined logits and average the K probability vectors for every epoch.
3	Return the argmax of the averaged distribution. Do not use target labels to select a branch.

B Architecture, Preprocessing, and Training Objective

B.1 Input processing and tensor shapes

The network input has shape $B \times 20 \times 2 \times 3000$. Fixed record-wise preprocessing precedes sequence construction. Inside every forward, each epoch and modality is standardized independently:

$$x'_t = \frac{x_t - \bar{x}}{\max(\sigma_x, 10^{-4})}, \quad \bar{x} = \frac{1}{T_0}\sum_t x_t, \quad \sigma_x = \sqrt{\frac{1}{T_0}\sum_t (x_t - \bar{x})^2}. \tag{15}$$

The 10^{-4} floor applies to the standard deviation; DGSIN instead adds $\varepsilon = 10^{-5}$ to the variance in its normalization endpoints. Recording-level preprocessing also depends on statistics unavailable from an isolated 20-epoch block. Appendix D summarizes the documented evaluation protocol.

If $\tilde{x} = ax + b\mathbf{1}$, $a > 0$, and both standard deviations exceed the floor, Equation 15 yields the same standardized input for x and $\tilde{x}$. In deterministic evaluation, downstream predictions are consequently

Table 7: Waveform encoder and temporal representation dimensions. A DGSIN shape is measured immediately before normalization.

Stage	Operation	Shape / setting
Flatten	Epoch–modality pairs	$40B \times 1 \times 3000$
Stem	Conv1D; DGSIN; GELU	Kernel 31, stride 4, pad 15; $40B \times 80 \times 750$
Stem pool	MaxPool1D	Kernel 5, stride 2, pad 2; length 375
Inception 1	Four branches; concat; DGSIN; GELU; dropout; residual	$40B \times 80 \times 375$
Inception 2	Same structure	$40B \times 80 \times 375$
Per-modality pool	Temporal mean and max concatenation	160 features per epoch/modality
Modality projection	Linear–LN–GELU–dropout	$160 \to 192$; add modality embedding
Modality fusion	Mean and max over modalities; projection	$384 \to 192$; $B \times 20 \times 192$
Temporal encoder	Learned positions; three residual temporal blocks	192 features, 4 attention heads
Prediction	Five-class linear head and N1 logit refinement	$B \times 20 \times 5$ refined logits

identical for these separately applied epoch/modality transformations. This is a preprocessing identity, not evidence that raw amplitude is irrelevant to sleep staging or that DGSIN preserves it. DGSIN only acts on information available after this preprocessing.

Flattening the epoch and modality axes produces $40B$ epoch–modality pairs. The detailed waveform encoder is listed in Table 7. All convolutions in the stem and Inception branches are bias-free. EEG and EOG share the same waveform weights.

Each Inception block has three convolutional branches with kernels 9, 19, and 39, paddings 4, 9, and 19, and 20 output channels each. A fourth branch uses max pooling with kernel 5, stride 1, padding 2, followed by a kernel-1 convolution with 20 channels. Concatenation yields 80 channels. The normalized, activated, dropout-regularized output is added to the residual input. Both blocks preserve temporal length 375.

The modality projection and fusion projection each use Linear–LayerNorm–GELU–Dropout. After learned epoch-position embeddings are added, each of the three temporal blocks applies pre-normalized four-head self-attention, a depthwise temporal convolution with kernel 5 and padding 2 followed by GELU and a pointwise convolution, and a feed-forward sublayer of widths 192–768–192. These sublayers have residual connections and dropout 0.1. The attention has no causal mask. "Conformer-style" here denotes this documented attention–convolution–feed-forward structure, not an assertion that every detail matches the original Conformer architecture (Gulati et al., 2020).

B.2 N1 LOGIT REFINEMENT

The sleep-specific prediction components are held fixed across matched normalization comparisons. Let $h_i \in \mathbb{R}^{192}$ be an epoch feature and $\ell_i \in \mathbb{R}^5$ its base logits. An auxiliary head

$$q_i = \mathrm{Linear}_{96\to 1} \circ \mathrm{Dropout}_{0.1} \circ \mathrm{GELU} \circ \mathrm{Linear}_{192\to 96} \circ \mathrm{LayerNorm}(h_i) \tag{16}$$

predicts N1 versus the remaining classes. Index 1 denotes N1. With $p_i = \mathrm{softmax}(\ell_i)$ and $\ell_{i,-1} = \max_{j\neq 1} \ell_{ij}$, define

$$E_i = -\frac{\sum_j p_{ij} \log p_{ij}}{\log 5}, \qquad v_i = \mathrm{sg}\left[E_i \,\mathrm{sigmoid}\left(\frac{1.25 - |\ell_{i1} - \ell_{i,-1}|}{0.35}\right)\right]. \tag{17}$$

The refined logits are

$$\tilde{\ell}_{i1} = \ell_{i1} + 0.35 v_i \tanh(q_i), \qquad \tilde{\ell}_{ij} = \ell_{ij} \quad (j \neq 1). \tag{18}$$

This changes only the N1 component and is applied separately inside every source-conditioned inference pass. Its uncertainty gate is distinct from the DGSIN discrepancy gate.

B.3 Losses and optimization

For all N_{ep} valid epoch labels in a training batch, normally $20B$, the label-smoothed classification loss is

$$\mathcal{L}_{\text{cls}} = -\frac{1}{N_{\text{ep}}} \sum_i \sum_{j=0}^{4} \left(0.9\mathbf{1}[y_i = j] + \frac{0.1}{5} \right) \log \text{softmax}(\tilde{\ell}_i)_j. \tag{19}$$

Let $b_i = \mathbf{1}[y_i = 1]$, $s_i = \text{sg}(\text{sigmoid}(q_i))$, and $p_i^{\text{true}} = b_i s_i + (1 - b_i)(1 - s_i)$. The auxiliary loss is

$$\mathcal{L}_{\text{N1-aux}} = \frac{1}{N_{\text{ep}}} \sum_i (1 + 2b_i)(1 - p_i^{\text{true}})^{1.5} \, \text{BCEWithLogits}(q_i, b_i). \tag{20}$$

Only the focal modulator is detached; the binary cross-entropy retains its gradient. With $[z]_+ = \max(z, 0)$ and $\tilde{\ell}_{i,-1} = \max_{j \neq 1} \tilde{\ell}_{ij}$,

$$\mathcal{L}_{\text{margin}} = \text{mean}_{i:y_i=1}[0.40 - (\tilde{\ell}_{i1} - \tilde{\ell}_{i,-1})]_+ \tag{21}$$

$$+ 0.25 \, \text{mean}_{i:y_i \neq 1}[0.20 - (\tilde{\ell}_{i,-1} - \tilde{\ell}_{i1})]_+. \tag{22}$$

An empty group contributes zero. Classification and margin losses use the refined logits. With the weights in Table 8, the complete supervised objective is

$$\mathcal{L} = \mathcal{L}_{\text{cls}} + 0.08\mathcal{L}_{\text{N1-aux}} + 0.03\mathcal{L}_{\text{margin}}. \tag{23}$$

These prediction components are shared by the matched normalization controls. No additional loss minimizes the source–instance discrepancy or encourages the correction coefficient toward the instance-normalized endpoint.

Table 8: Common settings for DGSIN and its matched normalization controls. Whole-method baseline reproductions have the different training budgets stated in the main text.

Setting	Value
Optimizer	AdamW
Training epochs / sequence batch size	20 / 64
Initial learning rate / weight decay	5×10^{-4} / 5×10^{-5}
Learning-rate schedule	Cosine, stepped after optimizer updates
Gradient norm clipping	0.5
Dropout / label smoothing	0.1 / 0.1
Checkpoint metric	Source-validation accuracy
Initial layer scale / initial derivative	0.25 / 0.125
Initial gate at zero discrepancy	0.05
N1 auxiliary / margin weight	0.08 / 0.03
N1 residual logit scale	0.35

C Local Geometry of the Normalization Update

A simple feature-space example explains why normalization strength is a task question. For a nonconstant feature vector h and $h' = ah + b\mathbf{1}$, $a > 0$, instance normalization gives $\text{IN}(h') = \text{IN}(h)$ when the numerical stabilizer is ignored. This invariance is useful when a, b represent nuisance, but removes a distinction when they are label-relevant. A fixed source reference retains both kinds of distinction. Neither endpoint therefore determines a universally appropriate adjustment. In our pipeline, each input epoch and modality is already z-scored. The relevant trade-off therefore concerns *learned intermediate features*, whose channel responses can differ across segments after this preprocessing.

C.1 Effects on feature statistics and representation displacement

Proposition 1 (One-channel representation correction)**.** *For a single-channel feature vector $h \in \mathbb{R}^T$, let z_s and z_i be the source- and instance-normalized endpoints in Equation 1. Let g be constant over*

temporal positions, and define $u = (1-g)z_s + gz_i$*. Using the population definitions of the mean and standard deviation, write* $\delta = \mathrm{mean}(z_s)$*,* $\sigma_s = \mathrm{std}(z_s)$*, and* $\sigma_i = \mathrm{std}(z_i)$*. Then*

$$\mathrm{mean}(u) = (1-g)\delta, \tag{24}$$

$$\mathrm{std}(u) = |(1-g)\sigma_s + g\sigma_i|, \tag{25}$$

$$\mathrm{RMS}(u - z_s) = |g|\sqrt{\delta^2 + (\sigma_i - \sigma_s)^2}. \tag{26}$$

For DGSIN, $g \geq 0$*, so* $|g| = g$*.*

Proof. If the instance variance $\nu_I > 0$, define $w = (h - \mu_I \mathbf{1})/\sqrt{\nu_I}$. Then $\mathrm{mean}(w) = 0$ and $\mathrm{mean}(w^2) = 1$. Direct substitution gives

$$z_s = \delta\mathbf{1} + \sigma_s w, \qquad z_i = \sigma_i w, \tag{27}$$

where

$$\delta = \frac{\mu_I - \mu_k}{\sqrt{\nu_k + \varepsilon}}, \quad \sigma_s = \frac{\sqrt{\nu_I}}{\sqrt{\nu_k + \varepsilon}}, \quad \sigma_i = \frac{\sqrt{\nu_I}}{\sqrt{\nu_I + \varepsilon}}. \tag{28}$$

Hence $u = (1-g)\delta\mathbf{1} + [(1-g)\sigma_s + g\sigma_i]w$. Taking its mean and standard deviation yields the first two identities. Also, $u - z_s = g[-\delta\mathbf{1} + (\sigma_i - \sigma_s)w]$. Its mean squared value is $g^2[\delta^2 + (\sigma_i - \sigma_s)^2]$, since the cross term has zero mean. Taking the square root proves the final identity. If $\nu_I = 0$, the feature vector is constant, $z_i = 0$, and $z_s = \delta\mathbf{1}$; substitution gives the same identities with $\sigma_s = \sigma_i = 0$. □

For $0 \leq g \leq 1$, the pre-affine standard deviation is a nonnegative interpolation of the endpoint standard deviations. In particular, $\mathrm{std}(u) - \sigma_i = (1-g)(\sigma_s - \sigma_i)$ and $\mathrm{mean}(u) = (1-g)\delta$. The gate thus controls the remaining fraction of these endpoint contrasts. It does not necessarily reduce every difference between classes or every difference between source and target distributions.

The gate acts on intermediate waveform features before modality fusion and contextual modeling. Within a channel, its coefficient is shared across temporal positions, so the update applies a common scaling and offset to the features. Different channels can receive different corrections, changing their relative contributions to the pooled epoch representation. The temporal encoder then models dependencies within the resulting sequence of epoch tokens. Segment dependence enters through the instance statistics, without requiring a predicted stage label to select the correction coefficient.

C.2 Extrapolation, shared affine maps, and relative bounds

If $g > 1$, the operator goes beyond the instance-normalized endpoint. The coefficient multiplying the centered waveform can then be negative, which requires the absolute value in the standard deviation formula. DGSIN is therefore not restricted to convex mixtures by its parameterization. For the evaluated checkpoints, however, all learned layer scales are below one, as reported in Appendix F. Their correction coefficients consequently remain in the interpolation regime.

For $Y = \gamma u + \beta$, the channel mean and standard deviation become

$$\mathrm{mean}(Y) = \gamma(1-g)\delta + \beta, \qquad \mathrm{std}(Y) = |\gamma|\,|(1-g)\sigma_s + g\sigma_i|. \tag{29}$$

The shared affine map can change scale and sign; cross-channel and later-layer operations further change the representation. These facts prevent a local contraction statement from becoming a claim about class separability.

With $\Delta = z_i - z_s$ and entrywise multiplication, $u - z_s = g \odot \Delta$. For any entrywise p-norm, $1 \leq p \leq \infty$,

$$\|u - z_s\|_p \leq \rho_\ell \|\Delta\|_p. \tag{30}$$

This follows from $0 \leq g \leq \rho_\ell$ componentwise. It is a *relative* bound for a fixed layer scale. Without a separate bound on the endpoint difference, it gives no input-independent absolute displacement guarantee. No classification margin, semantic label, or unseen-domain generalization bound follows from this inequality alone.

C.3 Why output mixing differs from mixing means and variances

Consider centered endpoints with the same mean but different variances. Output mixing scales the centered feature vector by

$$\frac{1-g}{\sqrt{\nu_k+\varepsilon}}+\frac{g}{\sqrt{\nu_I+\varepsilon}}, \tag{31}$$

whereas normalization with a linearly mixed variance scales it by

$$\frac{1}{\sqrt{(1-g)\nu_k+g\nu_I+\varepsilon}}. \tag{32}$$

The expressions are generally unequal because inverse square root is nonlinear. DSON and DGSIN therefore differ in the normalized quantity as well as coefficient parameterization. The effect of discrepancy conditioning is assessed against the matched fine-static output-mixing control in Table 4.

D Evaluation Protocol and Metrics

Each leave-one-dataset-out task uses four source datasets and one held-out target, as described in Section 4. Source data are split 80/20 into disjoint blocks of 20 retained epochs, without enforcing recording- or subject-disjoint partitions. Removing invalid labels can leave gaps in the original recording timeline. Optimization and within-run checkpoint selection use source data only, with source-validation accuracy selecting the checkpoint.

The BatchNorm, LayerNorm, and InstanceNorm controls replace the three waveform-normalization sites used by DGSIN and otherwise retain the same preprocessing, backbone, sleep-specific prediction package, objective, data splits, 20-epoch schedule, and source-validation checkpoint rule. BatchNorm uses affine parameters and running statistics. LayerNorm normalizes the full channel–time feature shape at each site and uses elementwise affine parameters; its shape-dependent affine maps increase the trainable count to 2,381,310. InstanceNorm normalizes each channel over time without affine parameters or running statistics. BatchNorm and InstanceNorm contain 2,141,790 and 2,141,310 trainable parameters, respectively. All five held-out targets use the same initialization seed, while each target remains an independently trained leave-one-dataset-out model.

The BIN baseline adapts batch–instance normalization to one-dimensional convolutional sites with learned static layer–channel mixture coefficients clipped to $[0, 1]$. It uses one shared BatchNorm statistics bank and a single forward pass. The five target runs use the same data and 20-epoch budget, with source-validation accuracy selecting each checkpoint. This implementation differs from DGSIN in its statistics bank and prediction configuration, as noted in the main text.

DSON, BNE, PSDNorm, DGSIN, and the standard-normalization controls share the backbone and a 20-epoch training budget. SleepDG and FF-TRUST retain their own architectures and use 50 training epochs; FF-TRUST is evaluated without added label noise. These whole-method comparisons complement the matched normalization ablations but do not isolate the normalization operator alone.

Table 9: Target evaluation sets and resampling units. Clusters group dependent observations for paired bootstrap resampling.

Target	Blocks	Clusters	Identity scope
SleepEDF	11806	100	Subjects, 197 nights
HMC	6791	151	Records
ISRUC	5603	118	Subjects within subgroup, 126 nights
SHHS1	7367	150	Participants, visit 1
P2018	6636	150	Records

Table 9 lists the target cohorts and the clusters used for paired resampling. SleepEDF includes 153 cassette nights and 44 telemetry nights. HMC and P2018 use recording-level clusters because subject linkage across recordings is unavailable. ISRUC uses subject-level clusters within each subgroup; subject identities cannot be linked across subgroups. All 38,203 blocks are included in the paired evaluation.

MF1 averages the five class F1 scores, BAcc averages the five class recalls, and accuracy is the fraction of correctly classified epochs. Unweighted Cohen's $\kappa = (p_o - p_e)/(1 - p_e)$ uses observed agreement and expected agreement from the prediction and label marginals. Each metric is computed within a target before averaging the five target scores equally. Each configuration is trained once per target. Paired confidence intervals quantify evaluation-cluster sampling for fixed checkpoints, not variability across training runs.

E Paired Perturbation Evaluation

E.1 Protocol and paired resampling

We compare DGSIN with the independently retrained fine-static control labeled *w/o discrepancy conditioning* in Table 4. Both models receive identical target blocks and perturbations. The seven conditions comprise clean input, stationary AR(1) noise with coefficient 0.9, and white noise. Each noise type is applied at RMS ratios 0.05, 0.10, and 0.20 after input normalization. The five targets, two models, and seven conditions give 70 evaluation cells.

Noise amplitude is constant within an input sequence. The experiment therefore tests sensitivity to noise structure and strength, rather than transitions between local corruption regimes. The static control is retrained independently; it is not obtained by freezing the gates of a DGSIN checkpoint.

We draw 5,000 bootstrap replicates by resampling the clusters in Table 9 with replacement within each target. All blocks from a cluster receive the same multiplicity. Identical resampling weights are applied to both models and all seven conditions. Confusion matrices are aggregated under those weights before recalculating MF1, and target scores are then averaged equally. The 95% percentile intervals are pointwise and do not provide simultaneous family-wise coverage. They quantify evaluation-sample uncertainty for fixed checkpoints, not training variability.

E.2 Equal-target performance

Figure 6 summarizes MF1 across the seven conditions. On clean input, DGSIN reaches 72.60% mean MF1, compared with 71.97% for fine static. The mean paired difference is 0.64 percentage points, with a 95% interval of $[0.431, 0.890]$, supporting a positive difference for these evaluated checkpoints.

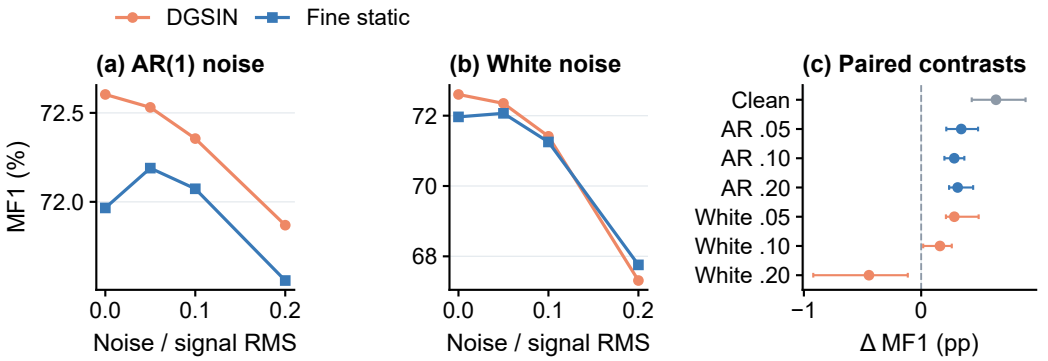

Figure 6: Paired perturbation results. Left and middle: equal-target MF1 under AR(1) and white noise. Right: DGSIN minus fine-static MF1, with pointwise 95% paired cluster-bootstrap intervals. Fine static denotes the retrained control without discrepancy conditioning.

Under AR(1) noise, DGSIN exceeds fine static by 0.28–0.34 percentage points across the three RMS ratios, with all three paired intervals above zero. Under white noise, DGSIN leads by 0.28 points at ratio 0.05 and 0.16 points at ratio 0.10; both paired intervals also lie above zero. At ratio 0.20, DGSIN falls below fine static by 0.45 points, with a DGSIN-minus-static interval of $[-0.920, -0.114]$ percentage points. The benefit therefore depends on the perturbation type and strength.

E.3 TARGET-LEVEL EFFECTS

Figures 7 and 8 show substantial variation across targets. SleepEDF favors DGSIN in all seven conditions. P2018 also favors DGSIN in its point estimates across all seven conditions, although its intervals include zero at the strongest AR(1) and white-noise levels. At white-noise ratio 0.20, DGSIN trails fine static by 2.43 MF1 points on HMC but leads by 0.82 points on SleepEDF. These results do not support a universal robustness benefit or identify why the independently trained models differ.

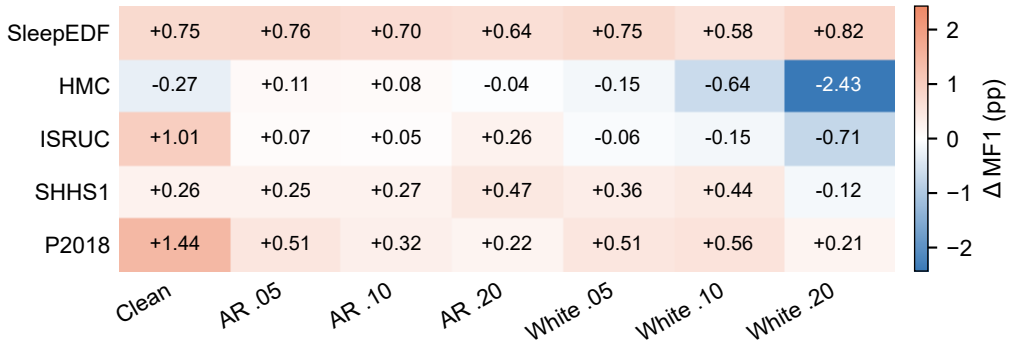

Figure 7: Target-level DGSIN-minus-static MF1 differences across all seven conditions. Positive values favor DGSIN; negative values favor the fine-static control.

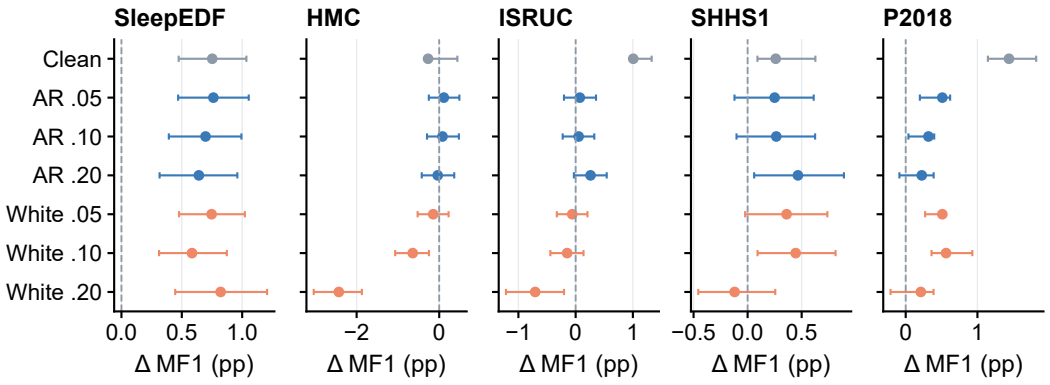

Figure 8: Target-level paired cluster-bootstrap intervals. Point estimates favor DGSIN on SleepEDF and P2018 in all seven conditions, but the P2018 intervals include zero at the strongest noise level of each type. Other targets show mixed effects.

F SOURCE-CONDITIONED PREDICTIONS AND LEARNED SCALES

Table 10 compares mean branch MF1 with probability fusion for the same DGSIN checkpoints used in Section 4.4. Mean branch MF1 is the arithmetic average of four independently scored branches. Probability fusion first averages their class probabilities and then computes MF1 from the resulting predictions. Probability fusion increases the equal-target mean MF1 from 71.54% to 72.60%.

Table 11 lists all 20 individual branch scores. Fusion exceeds every individual branch on SHHS1 and P2018, whereas at least one branch scores higher than fusion on SleepEDF, HMC, and ISRUC. These are retrospective comparisons: the inference rule averages probabilities uniformly without using target labels to select a branch.

Table 10: Source-conditioned predictions from the same DGSIN checkpoints. MF1 is reported in percent and fusion gain in percentage points. Mean branch averages four separate MF1 scores; fusion scores predictions from averaged class probabilities. Fusion gain is the difference between fused MF1 and mean branch MF1.

Target	Mean branch MF1	Fused MF1	Fusion gain (pp)
SleepEDF	71.42	71.95	+0.53
HMC	70.69	72.06	+1.37
ISRUC	73.87	74.98	+1.11
SHHS1	69.52	70.50	+0.98
P2018	72.20	73.53	+1.33
Mean	71.54	72.60	+1.06

Table 11: Individual source-conditioned branch MF1 (%). Source columns identify the frozen source-statistics bank and gate response parameters used for a full forward pass. Each row contains four branches from the same checkpoint; the held-out target has no source branch. Mean branch averages their MF1 scores, while fusion scores predictions from averaged probabilities.

	Source-conditioned branch						
Target	SleepEDF	HMC	ISRUC	SHHS1	P2018	Mean branch	Fusion
SleepEDF	–	69.20	70.73	70.93	74.85	71.42	71.95
HMC	70.54	–	72.45	70.76	69.00	70.69	72.06
ISRUC	74.10	76.07	–	73.71	71.58	73.87	74.98
SHHS1	69.20	69.90	69.23	–	69.77	69.52	70.50
P2018	73.34	71.50	71.07	72.91	–	72.20	73.53

Across the five checkpoints and three layers, learned scales range from 0.2120 to 0.3095. Since each correction coefficient is bounded by its layer scale, these checkpoints remain in the interpolation regime described in Section 3. Table 12 reports the per-class F1 scores of their probability-fused predictions.

Table 12: Per-class F1 of DGSIN probability fusion (%).

Target	W	N1	N2	N3	REM
SleepEDF	91.63	45.96	76.02	61.56	84.57
HMC	81.87	45.15	77.27	74.14	81.87
ISRUC	86.92	46.00	77.66	79.51	84.79
SHHS1	89.16	31.58	78.00	73.84	79.93
P2018	80.12	43.59	80.07	77.48	86.39

G Complete Main-Benchmark Metrics

Table 13 expands the main benchmark with accuracy, MF1, BAcc, and unweighted Cohen's κ for each target. Each baseline row and DGSIN fusion row evaluates all four metrics on the same valid predictions within a target. DGSIN mean-branch rows instead average the four separately computed branch metrics; they are not scores from an averaged confusion matrix. The target cohorts and checkpoint identities are the same as in the main benchmark.

Table 13: Complete target-level main-benchmark metrics. ACC, MF1, and BAcc are percentages; κ is unweighted Cohen's kappa. The DGSIN mean-branch row averages four separately scored source branches, while fusion scores averaged probabilities.

Method	Target	ACC	MF1	BAcc	κ
FF-TRUST	SleepEDF	77.21	71.49	75.28	0.695
FF-TRUST	HMC	74.38	72.38	72.40	0.664
FF-TRUST	ISRUC	77.02	73.76	73.06	0.699
FF-TRUST	SHHS1	74.48	66.10	71.02	0.653
FF-TRUST	P2018	72.97	68.49	72.39	0.641
BatchNorm	SleepEDF	73.87	69.30	74.05	0.657
BatchNorm	HMC	74.13	72.17	72.34	0.661
BatchNorm	ISRUC	75.95	74.02	72.69	0.685
BatchNorm	SHHS1	74.85	68.07	73.75	0.659
BatchNorm	P2018	73.42	71.33	74.83	0.651
LayerNorm	SleepEDF	75.02	69.76	73.94	0.669
LayerNorm	HMC	74.20	71.78	71.26	0.659
LayerNorm	ISRUC	75.06	73.13	71.50	0.673
LayerNorm	SHHS1	74.12	67.29	73.36	0.650
LayerNorm	P2018	74.19	71.54	74.61	0.659
InstanceNorm	SleepEDF	75.66	70.67	74.82	0.678
InstanceNorm	HMC	75.07	72.85	72.64	0.672
InstanceNorm	ISRUC	76.34	73.74	72.55	0.690
InstanceNorm	SHHS1	77.52	69.96	75.45	0.695
InstanceNorm	P2018	73.60	70.96	74.23	0.651
BIN	SleepEDF	74.38	69.79	74.43	0.664
BIN	HMC	73.64	71.52	71.48	0.654
BIN	ISRUC	76.12	74.03	72.55	0.687
BIN	SHHS1	75.38	68.46	74.04	0.666
BIN	P2018	73.75	71.87	75.07	0.655
DSON	SleepEDF	77.56	71.54	74.79	0.699
DSON	HMC	75.13	72.13	71.08	0.668
DSON	ISRUC	77.05	74.53	73.11	0.698
DSON	SHHS1	77.05	69.85	74.86	0.686
DSON	P2018	75.98	72.64	75.03	0.679
BNE	SleepEDF	75.41	70.34	74.39	0.674
BNE	HMC	71.98	69.84	70.10	0.633
BNE	ISRUC	74.83	72.72	71.31	0.670
BNE	SHHS1	74.54	67.53	73.09	0.654
BNE	P2018	72.98	70.23	73.63	0.644
SleepDG	SleepEDF	76.01	70.93	75.16	0.682
SleepDG	HMC	73.48	71.46	71.47	0.652
SleepDG	ISRUC	77.47	74.18	73.45	0.704
SleepDG	SHHS1	73.08	65.51	70.72	0.635
SleepDG	P2018	73.16	68.89	72.87	0.645
PSDNorm	SleepEDF	76.11	71.50	75.26	0.683
PSDNorm	HMC	74.60	72.44	71.69	0.664
PSDNorm	ISRUC	75.34	72.80	71.18	0.676
PSDNorm	SHHS1	76.50	68.88	75.15	0.682
PSDNorm	P2018	74.56	71.72	74.55	0.662
DGSIN (mean branch)	SleepEDF	76.51	71.42	74.59	0.687
DGSIN (mean branch)	HMC	73.28	70.69	70.57	0.647
DGSIN (mean branch)	ISRUC	75.91	73.87	72.57	0.685
DGSIN (mean branch)	SHHS1	77.29	69.52	74.93	0.691
DGSIN (mean branch)	P2018	74.75	72.20	74.61	0.665
DGSIN (fusion)	SleepEDF	77.31	71.95	75.34	0.697
DGSIN (fusion)	HMC	74.79	72.06	71.89	0.673
DGSIN (fusion)	ISRUC	77.12	74.98	73.56	0.700
DGSIN (fusion)	SHHS1	78.25	70.50	75.72	0.703
DGSIN (fusion)	P2018	76.41	73.53	75.74	0.679

Table 14: RTX A6000 inference cost. Latency: ms per 20-epoch block batch; memory: peak allocated MiB at $B = 32$. Parameters are trainable counts; fusion runs four sequential branches.

Method	Branches	Params (M)	$B = 1$	$B = 32$	Memory
DSON	1	2.1432	2.02	36.22	1219.6
DSON	4 (fusion)	2.1432	7.96	144.92	1219.7
Fine static	1	2.1428	2.40	45.14	1512.2
Fine static	4 (fusion)	2.1428	9.50	180.52	1512.3
DGSIN	1	2.1437	2.46	45.21	1512.6
DGSIN	4 (fusion)	2.1437	9.71	180.60	1512.7

H Inference Cost and Measurement Protocol

Models and inputs. Table 14 summarizes the HMC-checkpoint measurements for DGSIN, its independently retrained fine-static control, and the DSON implementation used in the main benchmark. Fine static is the *w/o discrepancy conditioning* configuration in Table 4. The three models have 2,143,713, 2,142,753, and 2,143,234 trainable parameters, respectively. These are full-model counts; the four branches reuse the backbone rather than store four complete networks. Every method receives the same preprocessed HMC inputs, with shape $B \times 20 \times 2 \times 3000$ at $B \in \{1, 32\}$. Each block contains 20 labeled 30-second epochs, so latency is measured per batch of blocks.

Execution and timing. All methods run sequentially in separate processes on one NVIDIA RTX A6000, using PyTorch 2.1.0, CUDA 12.1, cuDNN 8.9.2, and two CPU threads. Evaluation uses inference mode and FP32 tensors without autocast; cuDNN TF32 is enabled and matrix-multiplication TF32 is disabled. cuDNN deterministic mode is enabled and algorithm benchmarking is disabled. Inputs remain on the GPU. Timing includes complete forward passes, softmax, and the applicable aggregation, but excludes file reading, preprocessing, and host-to-device transfer.

Each configuration has three measurement rounds, with eight warm-up calls before 24 timed calls in each round. CUDA synchronization brackets each wall-clock measurement. The single-branch and four-branch measurement order reverses between rounds, and single-branch calls rotate the four source banks equally. Table 15 reports the median and 95th percentile of the resulting 72 timings. These repetitions measure runtime variability, not uncertainty in predictive accuracy. Throughput is $1000 \times 20B/t$ epochs per second for median batch latency t in milliseconds.

Table 15: Complete inference measurements. Latency and P95 are per-batch milliseconds; throughput counts 30-second epochs per second. Memory is peak allocated MiB.

Method	Branches	Batch	Median	P95	Epochs/s	Memory
DSON	1	1	2.02	2.06	9910.4	54.0
DSON	4	1	7.96	8.42	2513.8	54.0
Fine static	1	1	2.40	2.65	8326.5	63.2
Fine static	4	1	9.50	9.96	2104.8	63.2
DGSIN	1	1	2.46	2.76	8140.3	63.2
DGSIN	4	1	9.71	10.14	2060.1	63.2
DSON	1	32	36.22	36.48	17668.2	1219.6
DSON	4	32	144.92	145.27	4416.4	1219.7
Fine static	1	32	45.14	45.48	14178.8	1512.2
Fine static	4	32	180.52	180.89	3545.3	1512.3
DGSIN	1	32	45.21	45.48	14156.4	1512.6
DGSIN	4	32	180.60	181.22	3543.8	1512.7

Memory measurement and fusion. We reset PyTorch peak-memory statistics after warm-up and cache cleanup, then report the largest allocated-memory peak across the three rounds. This includes the resident model and input, but excludes reserved allocator memory and allocations outside PyTorch. Sequential fusion releases intermediate activations between source passes, keeping peak memory close to that of a single branch. All methods use identical input tensors, and model parameters remain

fixed during timing. DGSIN and fine static average class probabilities, whereas DSON averages logits before applying softmax. The measurements cover the complete forward passes and these fusion operations.

Cost comparison. At $B = 32$, DGSIN fusion processes 3,543.8 epochs per second and uses 1,512.7 MiB. DSON processes 4,416.4 epochs per second and uses 1,219.7 MiB. DGSIN and fine-static fusion differ by less than 0.1 ms at this batch size, consistent with Section 4.5. Four DGSIN passes take 3.95–3.99 times the single-pass latency. Table 15 reports all 12 combinations of method, branch count, and batch size.

These measurements characterize sequential inference on one hardware platform with fixed input shapes. They exclude data loading and preprocessing and are not averaged across target datasets. GPU utilization and temperature were not monitored, which limits the characterization of runtime conditions. We do not measure FLOPs, energy consumption, or parallel-branch execution.

Parameter counts. Table 14 reports trainable parameters. DGSIN has $4\times3\times80\times2+3\times80\times2+3 = 2403$ active normalization parameters, compared with 1,443 for fine static and 1,924 for DSON. Discrepancy conditioning therefore adds 960 trainable parameters to the matched static control. The dynamic and static models also contain 483 frozen parameters excluded from the trainable counts. Including these parameters gives model totals of 2,144,196 and 2,143,236, respectively. DSON contains 2,143,234 parameters, all trainable. The source-conditioned branches share the backbone and affine maps, so four-branch inference does not require four separate networks.

I Evaluation Scope

The experiments evaluate transfer across five sleep-staging datasets using EEG and EOG. They do not cover CAP or tasks beyond sleep staging. Each matched ablation uses one independently trained model per target. The paired intervals in Appendix E describe evaluation-sample uncertainty for these checkpoints; they do not quantify variability across repeated training runs.

The normalization controls in Table 4 answer two distinct questions. Coarse static versus fine static changes both coefficient granularity and scale sharing. Fine static versus DGSIN tests discrepancy conditioning within the same source-specific parameterization.

J Diagnostic Experiments

Tables 1 and 2 report separate diagnostic studies; their SleepDG results are not substituted for the main benchmark. Table 1 uses the same 22 EEG/EOG statistical descriptors for two conditional probe tasks. A fixed-stage probe predicts the source dataset; a fixed-source probe predicts the sleep stage. Probe training and testing are disjoint by recording, and the reported AUC is macro one-versus-rest AUC. Balanced accuracy (BAcc) is mean recall across classes. These probes show that the chosen descriptors contain information about both labels; they do not establish that the very same coordinates independently encode domain and stage or identify causal factors.

For Table 2, each of five leave-one-dataset-out folds trains a matched SleepDG pair from the same initialization and recording-level data split. SleepDG retains classification, reconstruction, and epoch- and sequence-level alignment objectives. NoAlign retains classification and reconstruction but removes the two alignment objectives. Frozen learned representations are evaluated with source-domain and sleep-stage probes on source recordings disjoint from backbone training and validation. The discrepancy columns measure the corresponding epoch- and sequence-level representation differences. Target MF1 and BAcc use the held-out dataset; all entries are equal-weight means across the five target folds. The final column reports the absolute difference between the equal-target means for SleepDG and NoAlign.

SleepDG reduces both discrepancy measures and source-domain probe AUC and BAcc in every fold. Sleep-stage probe AUC increases by 0.0002 on average, while BAcc decreases by 0.0022, indicating little change in stage separability under these probes. Held-out MF1 improves for HMC, ISRUC, and P2018 but decreases for SleepEDF and SHHS1; the equal-target mean is 0.64 percentage points lower. Each paired comparison uses one initialization per target and does not establish that alignment

generally removes semantic information. Consistent with the introduction, the results show that reducing measured domain discrepancy alone does not guarantee better target labeling.

K Normalization-Operator Comparison

Table 16 summarizes the distinctions discussed in Section 2. BIN and DGSIN combine normalized representations, whereas DSON combines means and variances before normalization. BIN and DSON use coefficients fixed with respect to the current input. DGSIN instead computes correction coefficients from current source–instance discrepancies using learned source–layer–channel response parameters. Instance statistics remain input dependent in all three methods.

Table 16: Normalization trade-offs. Input conditioning refers to the mixing or correction coefficient. Instance statistics depend on the input in all three methods. For DGSIN, parameter scope describes the gate response parameters; its learnable scale is layer-wise.

Method	Mixed quantity	Parameter scope	Input conditioning
BIN	Normalized outputs	Channel within layer	No
DSON	Mean and variance	Source domain	No
DGSIN	Normalized outputs	Source–layer–channel	Source–instance statistical discrepancy