

Online Resource 1

Neuroinformatics

HP-GSBS: Accelerating greedy state boundary search
with reusable interval statistics

Yanzhe Zhao^{1,*}, Yongzhang Ji²

¹School of Future Technology, Tianjin University, Tianjin, China

²Faculty of Psychology, Southwest University, Chongqing, China

*Correspondence: justinzhaoyz@outlook.com

S1 Interval decomposition and candidate reuse

The target is the matching objective used by GSBS (Geerligs et al., 2022; Geerligs, van Gerven, & Güçlü, 2021). Let $H = I - \mathbf{1}\mathbf{1}^\top/P$ be the spatial centering operator, $c_t = Hx_t$, and $z_t = \sqrt{P-1}c_t/\|c_t\|_2$. For an interval $S = [a, b)$ of length L , its mean pattern is $m_S = L^{-1}\sum_{t \in S} x_t$. The contribution to the mean spatial correlation is

$$F(a, b) = \frac{1}{T(P-1)} \sum_{t=a}^{b-1} z_t^\top z(m_S). \quad (\text{S1})$$

Define $A_b = \sum_{t < b} c_t$ and $B_b = \sum_{t < b} z_t$. Because

$$Hm_S = \frac{A_b - A_a}{L}, \quad z(m_S) = \sqrt{P-1} \frac{A_b - A_a}{\|A_b - A_a\|_2}, \quad (\text{S2})$$

the length factor cancels, giving

$$F(a, b) = \frac{(B_b - B_a)^\top (A_b - A_a)}{T\sqrt{P-1}\|A_b - A_a\|_2}. \quad (\text{S3})$$

The identity applies to nonconstant spatial state means. It preserves the distinction between row-wise spatial standardization in GSBS and any preprocessing applied before the matrix reaches the software.

The full score is additive over intervals. A single boundary replaces one interval by two; a double boundary replaces it by three. HP-GSBS reuses all unaffected contributions. In a local search block of length L_b , scores are multiplied by T/L_b to reproduce the original local normalization. The CPU implementation computes prefix differences in bounded tiles. PyTorch constructs interval scores through batched Gram products and recomputes short-interval differences directly; Triton fuses the prefix differences, dot product and norm.

Streaming reduction preserves candidate order and the original blockwise domain. The unscored sentinel, first-maximum rule and nonfinite-score handling are explicitly tested against the reference implementation. Candidate proposals maximize the matching score; the subsequent single-versus-double decision uses t-distance. Refinement deletes and reinserts boundaries sequentially, retaining their insertion labels. In a double insertion, adjacent state-count slots can share a partition; the last accepted partition may contain $K_{\max} + 1$ states. The full arrays retain this indexing.

S2 T-distance from sufficient statistics

Let U denote row-standardized data with its residual row means removed, matching the centering in `numpy.cov`. Correlations are $r_{ij} = U_i^\top U_j / (P-1)$. Let $s(i)$ be the state label and $d = \text{dmin}$.

Table S1: Principal computational costs. C is the number of legal candidates, L a searched block length, q the prefix chunk size and K the current state count. Costs describe the named operation rather than a universal complete-fit bound.

Operation	Reference-style calculation	HP-GSBS calculation
One candidate score	Reconstruct and normalize $O(TP)$ template	$O(1)$ Full-Q lookup; $O(P)$ prefix mode
Interval precomputation	Repeated within candidates	$O(T^2P)$ time, $O(T^2)$ table
Double-score storage	$O(L^2)$ score array	Bounded running summary; legal candidates still visited
Fixed-partition TD	Scan $O(T^2)$ pair structure	$O(K)$ rectangle queries after endpoints are obtained
Feature-moment TD	Pairwise time matrix	$O(TP^2 + KP^2 + TdP)$ arithmetic, with $d = \mathbf{dmin}$
Full output history	$O(K_{\max}T)$	Same required boundary history

The two distributions are

$$W = \{r_{ij} : j - i \geq d, s(i) = s(j)\}, \quad (\text{S4})$$

$$A = \{r_{ij} : j - i \geq d, s(j) = s(i) + 1\}. \quad (\text{S5})$$

Only these eligible upper-triangular pairs enter the score. For either distribution, the sufficient statistics are count n , sum S and squared sum R . The interval-template table retains the symbol Q used in the main article. These statistics give

$$\mu = S/n, \quad v = \frac{R - S^2/n}{n - 1}, \quad \text{TD} = \frac{\mu_W - \mu_A}{\sqrt{v_W/n_W + v_A/n_A}}. \quad (\text{S6})$$

This is the Welch statistic (Welch, 1947) used as the GSBS partition contrast. The statistic serves as a partition-selection criterion.

Two-dimensional prefixes

The stable backend builds zero-padded two-dimensional prefixes of the valid-pair indicator, masked correlation matrix and squared matrix. Four array references give the sum in any rectangle. Within-state rectangles and adjacent-state rectangles assemble the six statistics. Constructing interval endpoints from a boundary vector costs $O(T)$; the rectangle queries then cost $O(K)$. Small-sample handling follows the original implementation. Near-zero variances and ambiguous single/double comparisons invoke the reference calculation.

Feature moments for long inputs

The optional low-memory route works in feature space. With $h = P - 1$, an interval has summaries

$$L_S = |S|, \quad u_S = \sum_{t \in S} U_t, \quad M_S = \sum_{t \in S} U_t U_t^\top, \quad D_{1S} = \sum_{t \in S} \|U_t\|^2, \quad D_{2S} = \sum_{t \in S} \|U_t\|^4. \quad (\text{S7})$$

Before short-lag exclusion, its within-state statistics are

$$n_S = L_S(L_S - 1)/2, \quad (\text{S8})$$

$$S_S = (\|u_S\|^2 - D_{1S})/(2h), \quad (\text{S9})$$

$$R_S = (\|M_S\|_F^2 - D_{2S})/(2h^2). \quad (\text{S10})$$

For distinct adjacent intervals S, R , the cross statistics are

$$n_{SR} = L_S L_R, \quad S_{SR} = u_S^\top u_R / h, \quad R_{SR} = \langle M_S, M_R \rangle_F / h^2. \quad (\text{S11})$$

Pairs with $0 < j - i < d$ are subtracted from the relevant distribution. The implementation retains state summaries and bounded short-band caches, using bounded feature-space working storage. Candidate scoring uses raw-input and standardized-input time prefixes, with at most $q \times P$ working arrays. Boundary-history output still contributes $O(K_{\max} T)$ storage.

Moment differences can lose precision when correlations are nearly constant. The implementation uses bounded tiled two-pass pair statistics for risky cases and conservatively refuses unresolved nearly degenerate results. This is an experimental execution option; Full-Q/Prefix2D remains the fast default for the ordinary high- K inputs evaluated here.

S3 Execution interfaces and numerical scope

The stable configuration combines FP64, Full-Q, streaming candidates and Prefix2D t-distance. CPU, PyTorch CUDA and Triton share this controller. An optional independent input Y supplies t-distance data while X continues to determine candidates and state patterns.

Near-tied candidate scores are reevaluated with the original expression. Such guards preserve the measured decision paths, while practical tolerances remain distinct from an error bound over all floating-point inputs. Spatially constant rows and nonfinite inputs are rejected; the documented handling of ill-conditioned but otherwise valid matrices follows the reference pathway. Resource budgets limit candidate enumeration, original-expression reevaluation, workspace and tiled pair evaluation. Budget refusal does not change the candidate domain or scientific parameters.

Table S2: Released functionality and optional extensions.

Mode	Status	Role
FP64 Full-Q + Prefix2D, three backends	Stable	Default numerical path; sequential search and refinement
Raw-prefix + feature moments	Experimental	CPU low-memory option with guarded numerical refusal
Selective Triton resident candidates	Experimental	Device reduction for selected large candidate domains
FP32	Experimental	GPU t-distance differs from strict FP64 comparisons
Incremental TD; no-Q GPU prototypes	Research	Transactional partition updates and component-level GPU studies; not public complete-fit defaults

Saved results include the full boundary history, t-distance curve, selected states, state patterns and strengths. Batch execution processes recordings independently. GPU groups use compatible shapes and resource settings; CPU workers parallelize whole recordings. Resume requires both successful output and matching versioned execution identity. Historical results remain readable after a version change, while rerunning the new implementation uses a new output directory.

Example analysis workflow

The accompanying executable `scripts/research_workflow.py` generates a small prepared input, fits it on CPU, retrieves boundaries and t-distance, saves and reloads a versioned result, and processes two recordings independently. The CPU example in the main article uses this result interface; selecting `backend='torch-cuda'` or `backend='triton-cuda'` retains the interface on a compatible CUDA installation. CPU multiprocessing is configured through `GSBSConfig(n_jobs=...)`; on platforms using spawn, calls belong inside a `__main__` guard.

For a recording collection, the released command-line interface accepts a task manifest:

```
python -m hpgsbs fit-batch tasks.json --backend cpu-fast \
    --workers 4 --output outputs
```

The manifest records input paths and scientific configuration. Results are written atomically and completed entries are reused when their execution identity matches. The same command can therefore resume an interrupted collection; a change of software version, input or configuration requires the corresponding new execution identity. Exact manifest fields and examples are provided in the versioned software documentation (Zhao, 2026a).

S4 Experimental protocols and data provenance

Controlled paired benchmark

Seven unique configurations were evaluated with seeds 20261017 and 20261018. The centre $T = 512$, $P = 32$, $K_{\max} = 20$ is reused in each one-factor series. Gaussian means were generated as an 8×64 matrix, followed by a $T \times 64$ Gaussian noise matrix scaled by 0.8. Eight contiguous states have boundaries from rounded equally spaced endpoints. The first P features are retained. Thus feature subsets are nested, and the $K_{\max}$ comparison reuses identical inputs. The 14 configuration–seed cases contain ten distinct input matrices; seed reuse also couples comparisons across conditions.

Each official observation times construction and `GSBS.fit(showProgressBar=False)`. Import and numerical initialization used a separate short input. Each HP observation has one untimed warm-up and three timed calls to the released public API. The route ordering rotates by configuration and seed. GPU calls are synchronized before and after timing. Validation and serialization occur outside the timed interval. In total, 14 official fits and 126 timed HP fits were collected, plus 42 HP warm-ups. All 168 HP outputs passed the prescribed comparisons; no configuration was removed.

For input i , backend b and repetitions r , define

$$\tilde{t}_{i,b} = \text{median}_r t_{i,b,r}, \quad G_{i,b} = t_{i,\text{official}} / \tilde{t}_{i,b}. \quad (\text{S12})$$

Runtime summaries are medians of the two per-input observations, and speedups are medians of the two paired ratios. Whiskers span the two observations. They represent observed ranges, not confidence intervals. The official implementation has one timed observation per input; the optimized routes have three. Tables S5–S7 retain this distinction. The data bundle includes all 140 timed observations.

Mechanism illustration and data provenance

Figure 1 uses the public Sherlock fMRI dataset (Baldassano, 2020; Chen et al., 2017). The input is the mean angular-gyrus response over 16 participants, restricted to the first 120 TRs (TR = 1.5 s). Fitting uses all 4,230 voxels; 64 voxels are displayed in the activity heatmap. The supplied interval and correlation matrices are derived from this input, with smaller subsets used to explain rectangle queries. The complete symmetric correlation matrix is displayed for interpretation; the temporal mask and eligible upper-triangular pairs define the sufficient statistics actually counted. Thus the interval-template table Q and the correlation prefixes have different meanings despite their related matrix displays.

The saved illustration uses HP-GSBS 0.2.0 CPU, FP64, $K_{\max} = 30$, `dmin=1`, `finetune=1`, and block size 50. Supplied numerical checks compare its output with `statesegmentation 0.0.6`. The three backend icons explain the software architecture; cross-backend evidence is provided by the

synthetic and EEG comparisons. Derived arrays, the search trace, attribution and the parity report accompany the figure source.

Illustrative signal and EEG provenance

The main illustrative signal uses seed 19, four 16-feature Gaussian means repeated for 20 samples each, and noise with standard deviation 0.05. Its 80 samples have generating boundaries 20, 40 and 60. All four routes select these boundaries, with the exact state-count and numerical differences shown in Fig. 2 of the main article. Supplementary Table S12 reports the other previously recorded seed/noise cases, preserving the distinction between implementation agreement and recovery of generating boundaries.

The EEG evidence comprises four frozen analysis inputs derived from BBBD (Madsen, Kuppa, & Parra, 2026), a dataset by Jens Madsen, Nikhil Kuppa and Lucas C. Parra distributed under CC BY 4.0. Table S3 identifies the recordings and retained analysis dimensions. Recorded processing comprises event cropping, explicit artifact masks, nonoverlapping 25-sample block means from 128 Hz to 5.12 Hz, and within-recording channel z-scoring using valid-bin sample SD. The stored float32 values are converted to FP64 for fitting. All four use `dmin=5`, statewise detection, `finetune=1`, ordered sequential refinement, and block size 50, with recording-specific $K_{\max}$. The frozen-input provenance establishes the available recording identifiers and processing history; the exact upstream download version remains unspecified.

Table S3: BBBD frozen analysis inputs. Recording numbers are one-based in the article; file identifiers are zero-based. Experiment information follows the archived provenance. Duration is $T/5.12$ seconds; time maps store bin centres. The selected $K_{\max} + 1$ index for Recording 3 follows the double-insertion history described in Section S1.

Rec.	Recording identifier	Exp.	$T \times P$	Channels	$K_{\max}$	Selected K
1	sub-01_ses-01_stim01	Unspecified	1336×64	All 64	131	126
2	sub-02_ses-01_stim02	3	1857×32	Frontal 32	121	120
3	sub-03_ses-02_stim03	3	1754×64	All 64	172	173
4	sub-04_ses-02_stim06	3	1374×64	All 64	135	131

Table S4: Input and comparison-reference files, relative to the accompanying source-package root and the fixed public EEG snapshot. Each input contains `x` and `times`. Recording 1 reuses its figure-source input. All backend paths, algorithm settings, channel names and hashes are enumerated in `data/eeg/provenance.json`.

Rec.	Input file	Comparison reference
1	<code>figure_sources/figure3/data/sub-01_ses-01_stim01_all64_5.12hz_d5_k131.npz</code>	<code>figure_sources/figure3/data/0_full_reference.npz</code> (complete original GSBS output)
2	<code>data/eeg/inputs/recording_1.npz</code>	<code>data/eeg/outputs/recording_1_full_cpu.npz</code> (official-validated CPU chain)
3	<code>data/eeg/inputs/recording_2.npz</code>	<code>data/eeg/outputs/recording_2_full_cpu.npz</code> (official-validated CPU chain)
4	<code>data/eeg/inputs/recording_3.npz</code>	<code>data/eeg/outputs/recording_3_full_cpu.npz</code> (official-validated CPU chain)

CPU, PyTorch and Triton outputs use suffixes `_full_cpu.npz`, `_torch-cuda_float64.npz` and `_triton-cuda_float64.npz`. Recording 1 uses the stem `figure_sources/figure3/data/0`; Recordings 2–4 use `data/eeg/outputs/recording_1` through `recording_3`. The provenance includes the historical pipeline outputs and the separate HP-GSBS 0.2.0 validation history. File SHA-256 values identify complete files; processed-array hashes identify explicitly typed contiguous

array bytes. Historical base-matrix identifiers retain their recorded byte definition. These hash domains give complementary identities for the frozen evidence.

The saved-output comparison for recording 1 uses its complete official reference. Recordings 2–4 use CPU outputs previously checked against official results, providing a reference chain for the GPU comparisons. Their CPU self-comparisons are identity checks. The supplied release evidence additionally records all three v0.2 backends against the frozen full outputs. Full boundary history, final state count, t-distance, state patterns and strengths are checked. Table 3 of the main article retains all four historical full-output comparisons.

Figure 3 uses the newly supplied frozen input and four complete saved outputs for Recording 1 (1336×64). The entire input is used for segmentation; the local window is selected by fixed sample indices 617–718, corresponding to plotted bin edges 120.5078125–140.4296875 s. Input values are already standardized per channel. Heatmap colours are clipped to ± 3 for display without changing the analysis arrays. Correlation uses all 64 channels and is displayed symmetrically, with selected state blocks clipped at the window edges. The full-record state sequence and local boundaries all come from the saved selected partition. Evaluated t-distance indices are 2–131; the uncomputed padded slot is omitted. All four outputs select 126 states and have identical complete boundary histories and state patterns. Maximum absolute t-distance differences from the official output are 2.43361×10^{-13} (CPU) and 9.59233×10^{-14} (both GPU routes). The figure is regenerated directly from these files without fitting the excerpt.

Numerical criteria

Boundary arrays and selected state counts use exact equality. T-distance, state patterns and strengths use $\text{atol} = \text{rtol} = 10^{-8}$, with shapes and nonfinite masks checked separately. The maximum absolute t-distance difference in the controlled benchmark is 1.87583×10^{-11} ; state-pattern and strength differences are zero. The illustrative signal was compared at the strict per-refinement-move level. Missing required trace fields cause the strict comparison to fail.

Environment

The controlled experiment ran on an AMD Ryzen 7 7840H and NVIDIA RTX 4050 Laptop GPU (6 GB) under Ubuntu 22.04/WSL2, using Python 3.10.12, NumPy 2.2.6, SciPy 1.15.3, Numba 0.68.0, PyTorch 2.11.0+cu128, Triton 3.6.0 and CUDA runtime 12.8. BLAS, Numba and PyTorch intra-/inter-operation thread counts were one, and TF32 was disabled. The pre-existing component, memory and throughput measurements below used the documented release-development environment on the same laptop family. They are reported separately from the new paired benchmark.

S5 Complete controlled-benchmark tables

Table S5: Complete-fit runtime in seconds: median [minimum, maximum] across two input seeds. Each HP value is first summarized over three timed runs.

T	P	$K_{\max}$	Official	CPU	PyTorch	Triton
256	32	20	11.234 [10.144, 12.324]	0.073 [0.073, 0.073]	0.068 [0.065, 0.070]	0.064 [0.063, 0.064]
512	32	20	64.667 [58.876, 70.458]	0.142 [0.138, 0.146]	0.081 [0.075, 0.087]	0.099 [0.086, 0.111]
768	32	20	189.610 [169.973, 209.247]	0.251 [0.226, 0.277]	0.110 [0.105, 0.116]	0.131 [0.124, 0.138]
512	8	20	53.323 [52.454, 54.192]	0.122 [0.120, 0.123]	0.081 [0.077, 0.085]	0.075 [0.073, 0.076]
512	64	20	79.365 [79.157, 79.573]	0.153 [0.149, 0.156]	0.077 [0.073, 0.082]	0.100 [0.094, 0.106]
512	32	10	55.603 [50.179, 61.028]	0.118 [0.117, 0.120]	0.065 [0.063, 0.068]	0.076 [0.069, 0.082]
512	32	40	75.652 [69.669, 81.635]	0.165 [0.160, 0.170]	0.111 [0.106, 0.116]	0.126 [0.126, 0.126]

Table S6: Paired speedups (times): median [minimum, maximum] of the two same-input ratios. Ratios use unrounded values before summarization.

T	P	$K_{\max}$	CPU	PyTorch	Triton
256	32	20	154.0 [139.5, 168.5]	166.9 [144.1, 189.7]	177.0 [158.6, 195.4]
512	32	20	454.1 [425.1, 483.2]	794.8 [782.0, 807.6]	675.0 [528.2, 821.9]
768	32	20	770.1 [613.5, 926.6]	1729.7 [1470.9, 1988.4]	1458.3 [1235.0, 1681.7]
512	8	20	438.5 [427.1, 450.0]	662.2 [616.9, 707.4]	715.7 [690.1, 741.2]
512	64	20	520.5 [508.8, 532.2]	1027.7 [967.7, 1087.7]	797.3 [751.1, 843.5]
512	32	10	469.9 [429.9, 509.8]	854.8 [740.3, 969.4]	735.1 [723.7, 746.6]
512	32	40	457.6 [434.4, 480.7]	683.6 [599.8, 767.4]	599.3 [551.7, 646.9]

Table S7: Per-input complete-fit observations (seconds). HP columns are the median of three timed runs; the official column is one timed run.

T	P	$K_{\max}$	Seed	Official	CPU	PyTorch	Triton
256	32	20	20261017	12.3239	0.0731	0.0650	0.0631
256	32	20	20261018	10.1440	0.0727	0.0704	0.0639
512	32	20	20261017	70.4578	0.1458	0.0872	0.0857
512	32	20	20261018	58.8762	0.1385	0.0753	0.1115
768	32	20	20261017	209.2472	0.2258	0.1052	0.1244
768	32	20	20261018	169.9726	0.2770	0.1156	0.1376
512	8	20	20261017	52.4543	0.1228	0.0850	0.0760
512	8	20	20261018	54.1916	0.1204	0.0766	0.0731
512	64	20	20261017	79.5733	0.1564	0.0822	0.1059
512	64	20	20261018	79.1572	0.1487	0.0728	0.0938
512	32	10	20261017	61.0280	0.1197	0.0630	0.0817
512	32	10	20261018	50.1787	0.1167	0.0678	0.0693
512	32	40	20261017	81.6352	0.1698	0.1064	0.1262
512	32	40	20261018	69.6693	0.1604	0.1162	0.1263

S6 Complementary performance and recovery results

Historical extended workloads

Table S8 preserves the published v0.2.0 release comparison (Zhao, 2026b). Earlier official timings and later release-wheel timings were collected in separate sessions on the same documented machine. Synthetic workloads use the median of three timed complete fits after warm-up; the

EEG entry uses one complete fit per route. Complete-fit timing includes computation and required transfers, with validation and file output outside the timed interval. These observations extend the workload range of the main paired experiment and remain a separate evidence series.

Table S8: Historical extended workload performance: complete-fit seconds, with official/HP ratios in parentheses. Ratios use unrounded observations.

Workload	$T \times P$	$K_{\max}$	Official	CPU	PyTorch	Triton
Small	450×16	20	48.0	0.103 (467x)	0.075 (636x)	0.057 (847x)
Medium	900×32	30	349.9	0.317 (1,105x)	0.098 (3,556x)	0.098 (3,577x)
Large	1500×64	40	2213.5	0.868 (2,550x)	0.172 (12,903x)	0.213 (10,377x)
XL	2100×64	50	5011.3	1.952 (2,567x)	0.363 (13,816x)	0.456 (10,988x)
EEG	1336×64	131	6115.4	1.610 (3,798x)	0.853 (7,165x)	0.838 (7,299x)

The XL PyTorch ratio is approximately 13,816 times when recomputed from the archived unrounded values. The corresponding 5,011 s to 0.363 s comparison illustrates the practical value of reuse for larger analyses. Main-text speedup conclusions retain the same-input paired design.

Ablation and memory

The ablation uses one frozen EEG input (1336×64 , $K_{\max} = 131$, $\mathbf{dmin}=5$, statewise detection, sequential refinement and block size 50). Its materialized baseline already uses the earlier Full-Q optimization. The successive changes isolate streaming reduction, window refinement and active-block reuse, within the precomputed interval-score representation.

Table S9: CPU ablation: three timed fits after one warm-up, seconds. All compared outputs agree.

Configuration	Median	Minimum	Maximum
Materialized	3.3638	3.2950	3.3870
Streaming	1.8035	1.7705	1.9462
Streaming + window	1.6014	1.5863	1.6605
Streaming + window + cache	1.4288	1.4275	1.4291

The separate memory comparison uses a 5000×8 input, single-boundary search, no refinement, block size zero and $K_{\max} = 16$. The Full-Q/materialized path takes 9.754 s and peaks at 1598.0 MiB process RSS; raw-prefix/feature-moments takes 0.110 s and peaks at 190.6 MiB. These are process totals, including runtime state, sampled every 5 ms. Their timing ranges are 9.384–10.432 s and 0.105–0.113 s.

Table S10: Long-input low-memory measurements under the same single-boundary/no-refinement configuration. Each median uses three timed runs. The final column is the maximum selected/final-partition TD difference from the tiled oracle.

$T \times P$	Median (s)	Range (s)	Peak RSS (MiB)	Max. TD difference
5000×8	0.1077	0.1074–0.1079	191.1	1.78×10^{-15}
10000×16	0.1843	0.1724–0.1850	204.5	8.88×10^{-16}
20000×32	0.3745	0.3652–0.4073	247.5	8.88×10^{-16}
50000×64	1.4439	1.4177–1.4842	469.5	1.15×10^{-14}

The long-input series changes both T and P and records a restricted search configuration. The 50000×64 run selects 15 states, and its final 16-state partition is also checked. Full statewise search retains a quadratic double-candidate domain, so the released resource budget can reject that larger computation without altering its scientific parameters.

Batch throughput

Short workloads use independent synthetic inputs of size 450×16 , $K_{\max} = 20$, $\mathbf{dmin}=1$. High- K tasks use repeated copies of one 1336×64 EEG input, $K_{\max} = 131$, $\mathbf{dmin}=5$. Each configuration has one warm-up batch and three timed batches. Timings include pool construction, serialization where applicable, fitting and synchronization; validation occurs afterwards. Fits/minute is computed per batch, then summarized. CPU batches have 32 short tasks or eight high- K tasks; the GPU batch count is explicit. Table S11 contains the selected configurations used to interpret the main figure, including larger GPU batches. Admission skips are not encoded as zero throughput.

Table S11: Batch throughput: median [minimum, maximum], completed fits/minute.

Workload	Backend	Workers	Batch	Fits/minute
Small	CPU	1	32	274.3 [269.4, 282.9]
Small	CPU	2	32	839.6 [815.0, 843.6]
Small	CPU	4	32	1362.2 [1328.0, 1419.7]
Small	CPU	8	32	1766.8 [1638.0, 1781.1]
Small	PyTorch	1	1	646.7 [641.3, 730.3]
Small	PyTorch	1	8	1418.4 [1383.6, 1425.6]
Small	PyTorch	1	32	1459.9 [1378.7, 1538.1]
Small	PyTorch	1	64	1399.3 [1398.8, 1484.2]
Small	Triton	1	1	658.6 [652.8, 660.1]
Small	Triton	1	8	1451.5 [933.0, 1654.5]
Small	Triton	1	32	1529.9 [1384.9, 1598.3]
Small	Triton	1	64	1340.3 [1313.8, 1358.1]
High-K	CPU	1	8	24.4 [24.0, 24.9]
High-K	CPU	2	8	74.3 [74.1, 74.7]
High-K	CPU	4	8	125.7 [124.5, 129.4]
High-K	CPU	8	8	154.9 [149.0, 161.9]
High-K	PyTorch	1	1	42.9 [42.0, 47.9]
High-K	PyTorch	1	4	40.5 [38.8, 40.7]
High-K	PyTorch	1	8	41.2 [40.4, 41.5]
High-K	Triton	1	1	41.9 [40.5, 45.6]
High-K	Triton	1	4	44.6 [44.2, 45.1]
High-K	Triton	1	8	43.1 [42.2, 43.3]

Generating-boundary recovery

Table S12: Previously recorded synthetic recovery examples, all with exact implementation agreement. The generating boundaries are 20, 40 and 60. Precision and recall compare selected boundaries with those exact indices.

Seed	Noise SD	Selected K	Selected boundaries	Precision	Recall
12	0.05	3	20, 40	1.00	0.67
12	2.00	2	42	0.00	0.00
19	0.05	4	20, 40, 60	1.00	1.00
19	2.00	6	3, 7, 20, 41, 58	0.20	0.33

These examples show how noise and state-count selection affect GSBS outputs while the optimized implementation follows the same decisions. The main figure uses the existing seed-19 low-noise

example to illustrate the interface and numerical agreement.

S7 Reproducibility package and data access

HP-GSBS 0.2.0 is distributed under the MIT license at <https://doi.org/10.5281/zenodo.23422148> (Zhao, 2026a). The original benchmark archive, version 1.0.1, is available at <https://doi.org/10.5281/zenodo.23457342> (Zhao & Ji, 2026b). The expanded reproducibility archive, version 1.1.0, is available at <https://doi.org/10.5281/zenodo.23573202> (Zhao & Ji, 2026a); it corresponds to Git tag `v1.1.0` (commit `126043d11050368dbbcffae8f512eda65e6c4648`). It includes the four-record EEG collection, the complete Sherlock application, six main figures, benchmark observations, provenance, licenses and verification scripts. The all-versions DOI is <https://doi.org/10.5281/zenodo.23457341>. The earlier EEG snapshot remains identifiable by commit `f90797909882a067578f7a50c6814de46d333d98`. These identifiers distinguish the software release, the original benchmark archive and the expanded reproducibility release.

From a checkout of the public reproducibility repository at tag `v1.1.0`, the following commands check the archived files and saved outputs and regenerate all six figures:

```
python scripts/verify_manifest.py
python scripts/verify_data.py
python scripts/make_figures.py
```

The public acquisition and rerun scripts in `scripts/download_sherlock.py` and `scripts/run_sherlock.py` provide the full fMRI application described in Section S8.

The accompanying source archive contains the editable main article and Online Resource 1, six figure PDFs, plotting scripts, numerical figure inputs, the four-record EEG collection, and the compact fMRI outputs in Section S8. The standalone figure archive supplies `Fig1.pdf` through `Fig6.pdf` and the corresponding EPS files. `FIGURE_SOURCES.json` maps each canonical figure to its plotting script and data, with file SHA-256 values. `SOURCES_SHA256.json` identifies every archived source file. Numerical-library versions and the single-thread benchmark configuration are specified in Section S4.

From the extracted manuscript source directory, the following commands check the saved EEG arrays and fMRI results, regenerate the figures and compile the article and supplement:

```
python data/eeg/verify_eeg_all.py
python verify_fmri.py
python make_figures.py
python build_paper.py
```

The EEG verifier checks all four matrices and time maps, 23 file hashes, saved input identity, algorithm settings, boundary histories, selected states, t-distance, state patterns and strengths. Boundary comparisons are exact; floating-point comparisons use $\text{atol} = \text{rtol} = 10^{-8}$. The fMRI verifier checks the 17 compact outputs and figure cache, independently recomputes observed boundary matching, and reconstructs the seeded circular-shift statistics. Verification and figure regeneration read the archived numerical outputs. Compilation produces `HP_GSBS_Neuroinformatics_Main.pdf`, `HP_GSBS_Neuroinformatics_ESM_1.pdf` and the upload alias `ESM_1.pdf`. The flattened sources `main.tex` and `ESM_1.tex` compile independently with pdfLaTeX, the supplied figure files and, for the main article, the supplied Springer class.

An optional new EEG analysis uses the fixed software release and a fresh output directory:

```
python scripts/run_eeg_all.py --backend cpu-fast \
    --output new_eeg_cpu
```

The `-recordings` option uses zero-based identifiers 0–3; `torch-cuda` and `triton-cuda` select the corresponding FP64 GPU routes on a compatible installation. This command compares the newly saved optimized outputs with the references in Table S4. Recorded historical timings and validation reports remain associated with their original executions.

All plotting-data paths below are relative to the source-package root. `data/paired_timed_runs.csv` contains the 140 timed controlled observations. `data/paired_per_input.csv`, `data/paired_runtime.csv` and `data/paired_speedup.csv` contain successive summaries; `data/paired_inputs.csv` records shapes, seeds and input hashes. `data/synthetic_figure_data.npz` stores the illustrative signal and extracted backend outputs. The ablation, memory, throughput and EEG-comparison CSVs retain their measured observations. `data/figure6_data.npz` contains the Sherlock display matrix, state-count curves, selected boundaries and null scores. Table rows in `generated/` are derived from these observations. `verify_sources.py` checks archive hashes and numerical table consistency.

BBBD and Sherlock retain attribution to their creators (Baldassano, 2020; Chen et al., 2017; Madsen et al., 2026); processing histories and recording identifiers accompany the derived inputs. Code is MIT licensed. Distributed data, derived figures and documentation retain CC BY 4.0 attribution. The source archive fixes the reported numerical observations, while independent reruns yield timings for the researcher’s documented environment.

S8 Naturalistic fMRI protocol and complete results

Input identity and analysis configuration

The angular-gyrus array `Sherlock_AG_movie.npy` is distributed in Baldassano (2020), version 1 (<https://doi.org/10.6084/m9.figshare.12436955.v1>), and derives from Chen et al. (2017). The source file is available at <https://ndownloader.figshare.com/files/22927253>. Its size is 256,101,248 bytes, its MD5 is `f80dde9a451e140c39b98c2e5214d93c`, and its shape is $16 \times 946 \times 4230$ in float32. The time axis covers the first movie half supplied in this derivative: 946 TRs at 1.5 s, or 1,419 s. Participant numbering below follows the array order.

Each participant matrix was converted to FP64. The group response was computed by accumulating the 16 participant arrays divided by 16 in FP64. Analysis retained the supplied preprocessing. HP-GSBS 0.2.0 used the CPU backend, Full-Q interval scores, Prefix2D t-distance, streaming candidates, latency mode, $K_{\max} = 60$, `dmin=1`, `statewise_detection=True`, `finetune=1`, `finetune_order=True`, and `blocksize=50`. BLAS and Numba thread counts were one. The host CPU was an AMD Ryzen 7 7840H under Ubuntu 22.04/WSL2 with Python 3.10.12. These are 17 complete application fits, one per group or participant matrix, with a total fit time of 213.1028 s. A separate initial group run used the same configuration and took 16.7959 s; the cohort table reports the group result from the complete cohort execution.

Boundary matching and null construction

Selected internal boundary indices define each participant’s boundary train. For each participant pair, the matching algorithm maximizes the number of one-to-one matches whose absolute index difference is at most three. The sorted matching implementation was independently checked against maximum bipartite matching. If M boundaries match between trains of sizes n_i and n_j , $F_1 = 2M/(n_i + n_j)$. The observed statistic averages these scores across all $\binom{16}{2} = 120$ pairs.

For each null draw, every participant receives an independent integer offset sampled uniformly from 1 through 945. Adding that offset modulo 946 preserves the boundary count and circular interboundary spacing. Matching uses absolute distance on the linear time axis after shifting,

including any boundary shifted to index zero. The 120 pair scores are averaged to produce one null cohort statistic. One thousand draws with seed 20261020 yield a null mean of 0.3248678 and central 95% interval [0.3148120, 0.3353518]. The observed mean is 0.4140970, giving an excess of 0.0892292 and a one-sided Monte Carlo probability of $1/1001 = 0.000999$. The interval describes the distribution of shifted cohort statistics; the inferential unit is the complete set of shifted participant trains.

Saved outputs and figure construction

Each compact fMRI NPZ contains the five fields in Table S13. The JSON files `application_fmri/manifest.json` and `application_fmri/results.json` supply the configuration, source identity, software version, measured times and circular-shift scores. Table S14 gives selected state counts and measured times. Seven participants select index 60 or 61, and the group selects 60, locating these selections at the upper end of the evaluated search range. The complete curves support inspection and further state-count sensitivity analysis.

Table S13: Fields in each compact Sherlock segmentation file. Participant filenames use zero-based indices 00–15, corresponding to Participants 1–16 in the article.

Field	Content
<code>selected_K</code>	Scalar selected state-count index
<code>boundary_indices</code>	Internal boundary positions for the selected partition
<code>evaluated_K_indices</code>	Evaluated state-count slots, including an accepted $K_{\max} + 1$ slot
<code>all_bounds</code>	Complete indexed boundary history
<code>tdists</code>	Full padded t-distance array; evaluated slots are identified separately

For the activity panel of main Fig. 6, 72 evenly spaced voxel indices are selected from the group response. Each displayed time point is spatially z-scored across these voxels with sample SD, and colors are clipped at ± 3 . This display transformation is confined to the heatmap; fitting uses all 4,230 supplied voxels. The raster shows selected group and participant boundaries, the curve panel shows all evaluated state-count indices, and the histogram shows the 1,000 null cohort means with the observed statistic. The compact figure cache and scripts reproduce these panels from the saved outputs.

Table S14: Complete Sherlock cohort fits. The group row is the arithmetic mean response. Boundary counts equal selected states minus one.

Matrix	Selected K	Boundaries	Fit time (s)
Group mean	60	59	13.5556
Participant 1	60	59	13.0830
Participant 2	41	40	13.1543
Participant 3	21	20	12.4916
Participant 4	60	59	12.3681
Participant 5	45	44	12.4939
Participant 6	38	37	12.2782
Participant 7	61	60	12.3665
Participant 8	60	59	12.3386
Participant 9	60	59	12.2270
Participant 10	55	54	12.3911
Participant 11	19	18	12.9135
Participant 12	61	60	12.6413

Matrix	Selected K	Boundaries	Fit time (s)
Participant 13	58	57	12.4916
Participant 14	34	33	12.3703
Participant 15	39	38	11.8765
Participant 16	61	60	12.0616

Reproducing the application

The archived 17 segmentations, their 120-pair observed statistic and all 1,000 null scores can be checked from the manuscript source directory with `python verify_fmri.py`. The figure can be regenerated with `python figure_sources/draw_figure6.py`. These operations use the compact saved outputs and display cache supplied in the source archive.

For a new fit, the download helper retrieves the versioned Figshare input and checks its byte count, MD5, shape and dtype:

```
python scripts/download_sherlock.py \
    --output external/Sherlock_AG_movie.npy
```

Install the version 0.2.0 wheel linked from the software release (Zhao, 2026a). Its filename is `hpgsbs-0.2.0-py3-none-any.whl` and its complete-file SHA-256 is

43f1819b7d8c31adea736345210550b0a5aae4a247e77e363c6650b1b5db749b

The source README provides a hash-checked installation command. With numerical-library thread counts set to one, the archived analysis script produces a new group fit, 16 participant fits, their cohort statistic and circular-shift scores:

```
OMP_NUM_THREADS=1 OPENBLAS_NUM_THREADS=1 \
MKL_NUM_THREADS=1 NUMBA_NUM_THREADS=1 \
python application_fmri/sherlock_full_application.py \
    --input external/Sherlock_AG_movie.npy \
    --output new_sherlock_results --kmax 60 \
    --subjects all --perm 1000 --tolerance 3 \
    --null-seed 20261020
```

The new output directory stores its own measured times and configuration. The application times summarize HP-GSBS execution on these fMRI matrices; the paired implementation speedups remain the controlled measurements in Section S5.

References

- Baldassano, C. (2020). *Sherlock data for OHBM*. figshare. Retrieved from <https://doi.org/10.6084/m9.figshare.12436955.v1> (Dataset. CC BY 4.0)
- Chen, J., Leong, Y.C., Honey, C.J., Yong, C.H., Norman, K.A., Hasson, U. (2017). Shared memories reveal shared structure in neural activity across individuals. *Nature Neuroscience*, 20, 115–125, doi: <https://doi.org/10.1038/nn.4450>
- Geerligs, L., Gözükar, D., Oetinger, D., Campbell, K.L., van Gerven, M., Güçlü, U. (2022). A partially nested cortical hierarchy of neural states underlies event segmentation in the human brain. *eLife*, 11, e77430, doi: <https://doi.org/10.7554/eLife.77430>

- Geerligs, L., van Gerven, M., Güçlü, U. (2021). Detecting neural state transitions underlying event segmentation. *NeuroImage*, 236, 118085, doi: <https://doi.org/10.1016/j.neuroimage.2021.118085>
- Madsen, J., Kuppa, N., Parra, L.C. (2026). The Brain, Body, and Behavior Dataset (BBBD): Multimodal recordings during educational videos. *Scientific Data*, 13, 920, doi: <https://doi.org/10.1038/s41597-026-07215-1>
- Welch, B.L. (1947). The generalization of ‘Student’s’ problem when several different population variances are involved. *Biometrika*, 34(1–2), 28–35, doi: <https://doi.org/10.1093/biomet/34.1-2.28>
- Zhao, Y. (2026a). *HP-GSBS: High-performance greedy state boundary search*. Zenodo, version 0.2.0. Retrieved from <https://doi.org/10.5281/zenodo.23422148>
- Zhao, Y. (2026b). *HP-GSBS v0.2.0: historical official comparison and release-wheel benchmarks*. Versioned software benchmark record. Retrieved from https://github.com/JustinZHAO-05/HP-GSBS/blob/6f3dac95f38d2d1b132677ec5e702f7b162b8a63/docs/v020_official_comparison.md (Archived numerical observations are included in the accompanying reproducibility package)
- Zhao, Y., & Ji, Y. (2026a). *Hp-gsbs: Reproducibility materials for neural-state segmentation and performance evaluation*. Zenodo. Retrieved from <https://doi.org/10.5281/zenodo.23573202> (Version 1.1.0)
- Zhao, Y., & Ji, Y. (2026b). *Hp-gsbs: Software reproducibility materials*. Zenodo. Retrieved from <https://doi.org/10.5281/zenodo.23457342> (Version 1.0.1)