Lucidify EEG Viewer: A Browser-Based EDF Visualization and Sleep Staging Tool with Multi-Model Inference and Reference Hypnogram Overlay

Author(s): [Ryan Cameron, Divya Chander]
Affiliation(s): [Lucidify Inc.]
Corresponding author: [ryan@lucidifyai.com]

Working example here:

https://lucidifyai.github.io/LucidifyWeb/

Abstract

We describe a client-side, browser-based tool ("Lucidify EEG Viewer") for interactive visualization and automated sleep staging from European Data Format (EDF/EDF+) recordings. The tool loads local EDF files, renders multi-channel waveforms and time–frequency representations, and generates hypnograms via three selectable staging pipelines: (1) a Lucidify logistic-regression model trained on PhysioNet Sleep-EDF Expanded recordings, (2) a Lucidify logistic-regression model trained on the OpenNeuro Bitbrain Open Access Sleep (BOAS) dataset, and (3) a pure-JavaScript implementation of the YASA sleep-staging algorithm using an exported LightGBM classifier. To support transparent validation, the tool can load an external reference hypnogram (EDF+ annotations or BIDS `events.tsv`) and overlay it on model output, computing epoch-wise agreement metrics (accuracy and Cohen's κ) over overlapping, non-missing epochs. The system is designed to run fully locally (no server-side inference), improving accessibility for exploratory research workflows, device prototyping, and reproducible inspection of staging behavior across datasets.

1. Introduction

Sleep staging is commonly performed on polysomnography (PSG) signals and summarized as a hypnogram over standardized epochs (typically 30 s). Open datasets such as PhysioNet Sleep-EDF Expanded and OpenNeuro BOAS provide widely used benchmarks for development and evaluation of automated sleep staging methods. PhysioNet Sleep-EDF Expanded includes whole-night PSG with expert-scored hypnograms provided as EDF+ annotations, while BOAS provides PSG and wearable “headband” EEG recordings in BIDS format with events files suitable for alignment and comparison. PhysioNet+2OpenNeuro+2

Many staging pipelines depend on Python/MATLAB environments and assume batch processing. In contrast, browser-based inference enables frictionless inspection of individual recordings, interactive visualization, rapid comparison among algorithms, and portability across devices and operating systems. However, browser deployment requires re-implementing signal processing primitives and model inference in JavaScript and addressing local-file security constraints (e.g., `file://` CORS restrictions).

This work documents the purpose and methodology behind Lucidify EEG Viewer, including (i) EDF ingest and display calculations, (ii) feature-based staging with two dataset-specific Lucidify models, (iii) a YASA-compatible staging pipeline implemented in pure JavaScript via exported LightGBM model parameters, and (iv) reference overlay and agreement metrics for validation against ground truth.

2. System overview and architecture

Lucidify EEG Viewer is a static web application composed of modular JavaScript files loaded in `index.html`. The core modules are:

- **EDF parsing:** `edf_parser.js` (exposes `window.LucidifyParseEdf`)
- **Visualization:** `renderers.js` (exposes `window.LucidifyDrawWaveform`, `window.LucidifyDrawSpectrogram`, `window.LucidifyResizeCanvasToDisplaySize`)
- **Sleep staging (Lucidify models):** `sleep_local_bundle.js` and `sleep_models_embedded.js` (exposes `window.LucidifySleepStage`)
- **Hypnogram rendering:** `hypnogram_renderer.js` (exposes step-style drawing and overlay rendering)
- **Reference hypnogram parsing and metrics:** `hypnogram_reference.js` (exposes `window.HYPNO_REF`)
- **YASA staging (pure JS):** `yasa/*` including DSP primitives, feature extraction, LightGBM inference, and orchestration

`main.js` binds UI controls, manages shared view state (time window, zoom/pan), merges channels as needed for staging, triggers rendering, and coordinates model selection and reference overlays.

3. Data sources and reference annotations

3.1 PhysioNet Sleep-EDF Expanded

PhysioNet Sleep-EDF Expanded provides EDF PSG recordings and corresponding manually scored hypnograms distributed as EDF+ annotations files (commonly `*-Hypnogram.edf`). PhysioNet+1 The hypnogram annotations use EDF+ "EDF Annotations" channels containing time-stamped annotation lists (TAL). edfplus.info+1

3.2 OpenNeuro BOAS (ds005555)

OpenNeuro BOAS is a BIDS dataset containing simultaneous PSG and wearable EEG ("headband") recordings across 128 nights. OpenNeuro+1 In BIDS, event timing is represented in `*_events.tsv` using `onset` and `duration` (seconds relative to recording start) and may also include sample-based columns depending on modality conventions. BIDS Specification+2BIDS+2

In practice, BOAS events files may contain multiple stage columns (e.g., human-scored vs automated). For validation, the Viewer supports loading an events file and selecting the most appropriate stage column (e.g., preferring a human-scored column when present).

4. EDF ingestion and preprocessing

4.1 EDF/EDF+ parsing

EDF files encode a fixed ASCII header followed by interleaved signal samples per data record; EDF+ extends EDF with optional annotation streams stored as signals. PhysioNet+1 The Viewer's EDF parsing module produces an internal "Recording" object with:

- channel metadata (name/label, sample rate, physical dimension string),
- sample arrays, and
- total duration.

4.2 Unit handling

For sleep staging, consistent units are critical. The Viewer distinguishes:

- **Lucidify LR models:** signals are normalized to volts prior to feature extraction when needed (e.g., µV → V).

- **YASA pipeline:** signals are expected in microvolts, consistent with YASA guidance; therefore the Viewer preserves native units when running YASA and passes along the channel's physical dimension metadata. (Any conversion policy should be explicitly documented per acquisition device.)

5. Visualization methods

5.1 Waveform display

Waveforms are drawn into an HTML canvas for the current time window with interactive zoom and pan. The view window is shared with the spectrogram and hypnogram controls and is updated by slider input and pan gestures.

Implementation note: the precise downsampling/decimation strategy for drawing (e.g., min–max envelope per pixel column vs fixed-rate decimation) and any filtering applied for display should be verified against `renderers.js` (not included among the uploaded files here).

5.2 Spectrogram display

The spectrogram view displays a time–frequency representation over the same time window. The UI supports constraining the displayed maximum frequency (up to Nyquist based on the selected channel's sampling rate).

Implementation note: the exact STFT/Welch parameters (window length in samples/seconds, overlap, FFT length, PSD scaling, log compression) should be verified against `renderers.js`. The YASA DSP utilities (Section 6) implement Welch PSD with Hamming windows and median aggregation, but the *display spectrogram* may use distinct settings.

5.3 Hypnogram rendering

Hypnograms are rendered as a step function across epochs using `hypnogram_renderer.js`, with a fixed stage order (top→bottom): **W, REM, N1, N2, N3**. The renderer maps epochs evenly across the drawable x-range and places labeled grid lines at stage levels. A separate overlay renderer draws a second step function using an alternate style (e.g., translucent yellow) without duplicating axes.

6. Sleep staging pipelines

6.1 Lucidify logistic-regression models (PhysioNet-trained and BOAS-trained)

The Viewer supports two Lucidify models serialized as JSON (`model_physio.json`, `model_boas.json`). Each model contains:

- `feature_order`: the canonical ordering of features,
- `means`, `stds`: per-feature standardization constants,
- `W`, `b`: weight matrix and bias vector for a multi-class linear classifier,
- label metadata mapping class indices to **W, N1, N2, N3, REM**.

Inference follows the standard pipeline:

1. Extract epoch-wise features (30 s epochs),
2. Standardize: z=(x−μ)/σz = (x - \mu) / \sigmaz=(x−μ)/σ,
3. Compute logits: ℓ=Wz+b\ell = W z + bℓ=Wz+b,
4. Convert to probabilities via softmax,
5. Select argmax label per epoch.

6.2 Optional temporal smoothing (HMM/Viterbi)

The Viewer includes an optional hidden Markov model (HMM)-style smoother implemented by Viterbi decoding in log space using a hand-tuned transition matrix and an initial prior favoring wake at recording onset. This reduces rapid, unlikely stage flips at the cost of potentially “over-smoothing” genuine transitions. This smoothing is applied to Lucidify model probabilities; it can be enabled/disabled interactively.

6.3 YASA staging in the browser (JavaScript port)

YASA is an open-source sleep analysis library providing automated sleep staging via a LightGBM classifier and a specific feature set, with a commonly cited reference implementation. GitHub+2eLife+2

6.3.1 Rationale for a pure-JS port

Running YASA entirely in the browser enables:

- offline staging on private recordings,
- identical behavior across platforms without Python dependencies,
- direct comparison of YASA outputs with dataset-specific models under identical visualization settings.

6.3.2 Model export strategy

The Python YASA classifier is distributed as a serialized object (e.g., `joblib`). Browser deployment requires exporting model structure into a JSON representation and implementing inference. In this system, the LightGBM booster is exported into a JSON "dump" and embedded into a JavaScript file (`yasa_model_dump_embedded.js`). Embedding avoids `fetch()` failures under `file://` contexts due to browser CORS restrictions.

6.3.3 DSP and feature extraction in JavaScript

The YASA port implements required DSP primitives in `yasa_dsp.js`, including:

- resampling/downsampling utilities targeting **100 Hz**,
- bandpass filtering **0.4–30 Hz** implemented via RBJ-style biquad high/low-pass sections (a pragmatic approximation to the reference filtering pipeline),
- Welch PSD computation using Hamming windows and **median** aggregation,
- bandpower integration helpers.

Quirks / deviations vs reference YASA:

- **Filtering implementation:** the port uses RBJ biquads rather than SciPy's canonical IIR/FIR design; frequency response and phase behavior may differ slightly.
- **Resampling implementation:** linear interpolation / simple decimation strategies differ from YASA's typical polyphase resampling; this can change high-frequency content and some features.
- **Browser numeric constraints:** computations are performed in JavaScript `Number` (IEEE-754 double) and typed arrays; results should be validated against Python outputs on representative signals.

- **Channel/modalities:** the selected exported classifier may be EEG-only rather than EEG+EOG+EMG; performance and calibration can differ across montages and populations.

These differences should be documented explicitly and validated quantitatively for any research claims, especially across datasets and acquisition devices.

7. Reference hypnogram overlay and agreement metrics

The Viewer supports importing a reference hypnogram and overlaying it on the predicted hypnogram for the current time window:

- **EDF+ reference hypnogram:** parsed from "EDF Annotations" channels and decoded from TAL blocks (EDF+ specification). edfplus.info+1

- **BIDS `events.tsv` reference:** parsed using `onset` and `duration` fields, with optional use of sample-based columns depending on dataset conventions. BIDS Specification+1

The overlay is time-aligned by converting the current view start time to a starting epoch index and slicing the reference stage array to match the number of predicted epochs displayed. Epoch-wise agreement is then computed over overlapping epochs excluding null/unknown reference labels:

- **Accuracy:** fraction of matching epochs

- **Cohen's κ:** agreement corrected for chance

These metrics are printed to the browser console for transparency and can be logged alongside screenshots for audit trails.

8. Practical considerations and limitations

1. **Domain shift:** Models trained on different datasets (PSG vs wearable, different montages and filtering) can yield qualitatively different hypnograms on the same recording. Switching among models and comparing against reference overlays is recommended when ground truth is available.

2. **Clinical use:** The tool is designed for research and inspection. Any clinical use requires formal validation on intended populations/devices, documented performance, and appropriate regulatory review.

3. **Reproducibility:** Browser execution can vary slightly across engines due to floating-point and implementation details. Pinning the application version and recording hash, and exporting intermediate features/probabilities when needed, improves reproducibility.

4. **Visualization parameterization:** Waveform and spectrogram display settings should be explicitly versioned (window sizes, overlap, scaling) to ensure that figures are interpretable and reproducible. (Verify exact settings in `renderers.js`.)

9. Software components (local file references)

- `index.html` — UI layout and script load order
- `style.css` — layout styling (canvas sizing, flex rules)
- `main.js` — UI binding, view state, model selection, staging orchestration, overlay wiring
- `hypnogram_renderer.js` — hypnogram step renderer + overlay renderer
- `hypnogram_reference.js` — reference parsing + agreement metrics (loaded but not included in the uploaded set here)
- `edf_parser.js`, `renderers.js` — EDF parsing and waveform/spectrogram rendering (loaded but not included in the uploaded set here)
- `model_physio.json`, `model_boas.json` — Lucidify staging model weights and normalization constants
- `yasa/yasa_model_dump_embedded.js` — embedded LightGBM dump for browser inference
- `yasa/yasa_dsp.js`, `yasa/yasa_features.js`, `yasa/yasa_lgbm.js`, `yasa/yasa_staging.js`, `yasa/yasa_sleep_stage.js` — YASA JS port (DSP, features, inference, orchestration)

References (web links in code block)

```
[1] PhysioNet Sleep-EDF Database Expanded (sleep-edfx v1.0.0):
```

    https://www.physionet.org/content/sleep-edfx/1.0.0/
    DOI: https://doi.org/10.13026/C2X676

[2] OpenNeuro BOAS dataset (ds005555, versioned releases):
    https://openneuro.org/datasets/ds005555/versions/1.1.0

[3] BIDS specification: events files (events.tsv):

https://bids-specification.readthedocs.io/en/stable/modality-agnostic-files/events.html

[4] Vallat, R. & Walker, M.P. (2021). An open-source, high-performance tool for automated sleep staging. eLife.
    https://elifesciences.org/articles/70092
    DOI: https://doi.org/10.7554/eLife.70092

[5] YASA project repository (documentation and licensing):
    https://github.com/raphaelvallat/yasa
    https://yasa-sleep.org/generated/yasa.SleepStaging.html

[6] EDF+ specification (annotations / TAL):
    https://www.edfplus.info/specs/edfplus.html