

Spoken Script

Explainable AI for Sleep Staging with PhysioEx
UGent — Explainable & Trustworthy AI (Use Cases)

Guido Gagliardi

June 24, 2026 — Target: 20 minutes

Slide 0 — Title Page

[0:00–0:15]

Good morning everyone. My name is Guido Gagliardi, I'm a researcher at ESAT-STADIUS, KU Leuven. Today I'll show you a concrete use case of explainable AI, applied to the problem of automatic sleep staging from EEG signals. Everything I'll show you is built on top of PhysioEx, an open-source library that we develop.

Slide 1 — What is PhysioEx?

[0:15–1:30]

So what is PhysioEx? It's a Python library for sleep staging. You install it with a single pip command, and you get access to twelve public sleep datasets — all through a uniform API — thirteen pretrained models hosted on HuggingFace, and a suite of explainability methods built in.

▷ Point to the bullet list.

The idea is that you can go from raw data to explained predictions in just a few lines of code. Today we'll walk through this full pipeline: load data, run a model, and then ask the model *why* it made its prediction.

▷ Point to the logo and GitHub URL.

It's all open-source. The link is at the bottom if you want to follow along.

Slide 2 — The Sleep Staging Problem

[1:30–3:00]

Let me give you a quick primer on the problem. In clinical sleep studies, a patient sleeps for a whole night wearing EEG electrodes. The resulting signal — which can be eight hours long — gets divided into thirty-second chunks called epochs. A trained clinician then assigns each epoch one of five stages: Wake, N1, N2, N3, or REM.

▷ Point to the table.

Each stage has characteristic patterns. The one we'll focus on today is **N3** — **deep sleep**. It's dominated by large, slow **delta waves** in the 0.5 to 4 Hz range. These are the easiest to recognize visually, but also the most interesting from an explainability perspective because we know exactly what frequency signature to expect.

▷ Point to the bottom bullets.

A clinician scores about a thousand epochs per night, and inter-rater agreement is only around 82%. So there's clear motivation to automate this — but we also need to make sure the

model is actually using the right features, not exploiting artifacts. That's where explainability comes in.

Slide 3 — Data: Two Representations

[3:00–4:00]

In PhysioEx, we support two main input representations. On the left, you see the raw waveform — that's the EEG signal sampled at 100 Hz, so each thirty-second epoch is a vector of 3000 numbers. On the right, we have the spectrogram representation — the same signal but after a short-time Fourier transform, giving us a 2D time-frequency matrix: 29 time steps by 129 frequency bins.

▷ Point to the code snippets.

Notice the API is the same — you just change the `pipelines` argument. The raw representation goes to TinySleepNet, and the spectrogram goes to SeqSleepNet.

The key idea here is that the choice of representation determines what the model can “see.” A raw waveform preserves all temporal detail — exact waveform morphology, amplitude — but frequency information is only implicit. A spectrogram trades some temporal resolution for explicit frequency content. This distinction between representations will become crucial when we try to explain the models.

Slide 4 — Raw EEG — N3 (Deep Sleep)

[4:00–5:30]

Here's an actual N3 epoch from the Sleep-EDF dataset. On the left, the raw waveform: you can see these large, slow oscillations — amplitudes reaching 80 microvolts, clearly slower than the higher-frequency activity earlier in the recording.

▷ Point to the PSD panel on the right.

On the right, we computed the power spectral density using Welch's method. The colored bands show the standard EEG frequency bands: delta, theta, alpha, sigma, and beta. You can clearly see that the power is overwhelmingly concentrated in the delta band — the green shaded region between 0.5 and 4 Hz. That's exactly what we expect for N3 deep sleep.

This is important because it gives us a **ground truth** to compare against: when we later explain what the model sees, we should find attribution in this same frequency range.

Slide 5 — Spectrogram — N3 (Deep Sleep)

[5:30–7:00]

Now the same sleep stage, but from the MASS dataset in spectrogram form. On the left, the bright yellow band at the bottom of the spectrogram corresponds to the high delta power — you can see it's consistent across the entire 30-second epoch.

▷ Point to the mean spectrum on the right.

The right panel shows the time-averaged power per frequency bin. The delta band — the lowest frequencies — has more than 40 dB more power than the beta band. This confirms what we saw in the raw PSD: the spectrogram captures the same physics, but now the frequency information is *explicit* — it's literally one axis of the image.

This will matter for explainability: if the model's attribution highlights the bottom rows of the spectrogram, we can directly say “the model is looking at delta.”

Slide 6 — Sequence-to-Sequence Architecture

[7:00–8:30]

Before we run the models, let me quickly explain the architecture they share. Both TinySleepNet and SeqSleepNet are **sequence-to-sequence** models. They take twenty consecutive epochs as input — that’s ten minutes of sleep — and predict a sleep stage for each one simultaneously.

▷ Point to the TikZ diagram.

The architecture has two stages. First, a **per-epoch encoder** processes each epoch independently: a CNN for raw signals, or a filterbank plus recurrent network for spectrograms. Then, a **sequence model** — an LSTM or GRU — looks at all twenty encoded epochs together to capture temporal context. This is important because sleep stages don’t change randomly — there are transition rules, and neighboring epochs provide context.

Slide 7 — Pretrained Models in PhysioEx

[8:30–9:15]

Now, how do we actually use these models? This is the code slide. In PhysioEx, loading a pretrained model is a single function call: `load_from_pretrained` with the model name and the device. We have thirteen pretrained models on HuggingFace — they’re downloaded and cached automatically the first time you use them.

▷ Point to the `load_from_pretrained` lines.

Inference is equally simple: you call the model on a tensor of shape batch, L epochs, channels, and the input features. The output is always the same: batch, L, five — one five-class probability distribution per epoch. Both models share this exact interface, so you can swap them freely.

Let’s see what happens when we run them.

Slide 8 — TinySleepNet

[9:15–10:15]

Let’s run TinySleepNet on our Sleep-EDF sample. This model was published by Supratak and Guo in 2020. It uses four convolutional layers to go from 3000 raw samples to a 2048-dimensional feature vector, then an LSTM for sequence context, and finally a linear classifier.

▷ Point to the hypnogram.

On the left you see the true labels on top and the model’s predictions below. The model gets most epochs right — the long N2 stretch is perfectly matched, and it correctly identifies the N3 epochs at the end.

▷ Point to the barplot.

On the right, the barplot shows the model’s confidence for one specific N3 epoch: 78% for N3, with 21% residual probability on N2. It’s correct, but not perfectly confident. The black border marks the true label.

Slide 9 — SeqSleepNet

[10:15–11:00]

Now SeqSleepNet on the MASS spectrogram data. This model by Phan et al. uses a learnable filterbank instead of a fixed one — meaning it learns during training which frequency combinations are most discriminative for sleep staging. This is followed by a bidirectional LSTM with attention for per-epoch encoding, and a BiGRU for sequence context.

▷ Point to the hypnogram and barplot.

This sample from MASS has eighteen consecutive N3 epochs, and the model gets them all right. In fact, it only makes one error — confusing N2 with N3 at the very end of the sequence, where the transition happens. The barplot shows 99.4% confidence on our target N3 epoch — the model is extremely sure. Compare that to TinySleepNet’s 78% — the spectrogram model is significantly more confident on this stage.

Slide 10 — Both predict N3... but why?

[11:00–11:30]

(Pause for emphasis.)

So both models correctly classify N3. TinySleepNet at 78%, SeqSleepNet at 99.4%. But *why*? Which parts of the input are driving these predictions? And can we validate the explanation against what we know from sleep physiology — that N3 should be about delta waves?

This is where Integrated Gradients comes in.

Slide 11 — IG Intuition

[11:30–13:00]

Integrated Gradients is a post-hoc attribution method. The idea is intuitive: imagine you start from a **baseline** — silence, or the noise floor — and gradually morph it into the actual signal. At each point along this path, you measure: which features are the model most sensitive to right now? Then you accumulate these sensitivities over the entire path.

▷ Point to the enumerated steps.

The result is an attribution map with the same shape as the input. Positive values — shown in red — mean that feature pushes *toward* the predicted class. Negative values — blue — push away from it.

▷ Point to the Completeness box.

A key property is **completeness**: the attributions sum exactly to the difference between the model’s output on the actual input and on the baseline. Nothing is lost or double-counted. This is a formal guarantee that gradient saliency methods don’t provide.

Slide 12 — IG Formulation

[13:00–14:15]

Here’s the formal definition. The attribution for feature i is the difference between the input and baseline value, multiplied by the integral of the gradient along the straight-line path from baseline to input.

▷ Point to the integral formula.

In practice, we approximate this with a Riemann sum using 64 steps. That means 64 forward and backward passes through the model — it’s computationally more expensive than a single gradient, but the theoretical guarantees are worth it.

▷ Point to the axioms at the bottom.

Two key axioms: **Sensitivity** — if changing a feature changes the output, it gets nonzero attribution. And **Implementation Invariance** — two networks with identical input-output behavior get identical attributions, regardless of their internal architecture. Vanilla gradients violate both of these.

Slide 13 — Computing IG Attributions

[14:15–15:00]

Here’s how you actually compute this in PhysioEx. The first step is to define a score function — a wrapper that takes a single epoch as input and returns the model’s confidence for the predicted class. This is necessary because the model expects a full sequence of twenty epochs, but we want to attribute a single epoch.

▷ Point to the `score_fn` definition.

Then you create an `IntegratedGradients` object with 64 Riemann steps. And finally, you call it on the epoch tensor. The attribution has the exact same shape as the input — 3000 samples for raw, or 29 by 129 for spectrograms.

▷ Point to the baseline options.

Notice the two baseline options at the bottom: for raw EEG, the default baseline is zeros — silence. For spectrograms, we pass the noise floor explicitly.

Slide 14 — IG on TinySleepNet

[15:00–16:00]

Now let’s see the attributions. Top panel is the raw EEG, bottom panel is the Integrated Gradients attribution, normalized to the range minus one to one.

▷ Point to the attribution peaks.

You can see that the positive attributions — the red peaks — concentrate on the large slow oscillations. That makes sense: those are the delta waves that define N3. The model is clearly focusing on the right features.

But here’s the limitation: we’re looking at attribution *in the time domain*. We can see *where in time* the model looks, but we can’t directly confirm that it’s responding to the delta *frequency* specifically. The slow oscillations *look like* delta waves, sure — but without a frequency decomposition of the attribution itself, we can’t be absolutely certain. Maybe the model is picking up on amplitude, or waveform shape, or some other temporal feature that happens to correlate with delta. This is a fundamental limitation of post-hoc attribution on raw time-domain signals — the attribution lives in the same space as the input, and that space doesn’t have an explicit frequency axis.

Slide 15 — IG on SeqSleepNet

[16:00–17:45]

Now the same analysis on SeqSleepNet, but with the spectrogram input. This is where things get really interesting.

▷ Point to the left panel (spectrogram + IG).

On the left, you see the original spectrogram and the IG attribution heatmap. The dashed horizontal lines mark the EEG band boundaries. Look at where the red — positive attribution — concentrates: it’s all in the bottom rows, below the 4 Hz line. That’s the delta band.

▷ Point to the right panel (attribution vs PSD overlay).

On the right, we overlaid the mean attribution per frequency bin on top of the power spectrum. The red curve — attribution — peaks sharply in the delta range, exactly where the blue curve — power — also peaks. This is a quantitative confirmation: the model is using delta power to classify N3.

This is the key advantage of the spectrogram representation for explainability. Because frequency is an explicit axis, we can directly validate that the model’s explanation matches the known physiology of deep sleep.

Slide 16 — The Baseline Choice

[17:45–18:45]

One important detail I should mention is the baseline choice. For TinySleepNet, we used zeros — literal silence. For SeqSleepNet, we used the **noise floor**: the mean power in frequency bins above 40 Hz.

▷ Point to the table.

Why? Because in a spectrogram, zero dB doesn’t mean silence — it means unit power. The EEG signal is bandpass-filtered between 0.3 and 40 Hz, so everything above 40 Hz is pure recording noise. Taking the mean of those bins gives us a physically meaningful “silence” — around minus 35 dB. The attributions then tell us: what does the model see *above the noise*?

There are other valid choices — random noise, the mean training signal, or even a contrastive baseline from a different sleep stage. Each gives a different “question” to the model. The completeness axiom guarantees that regardless of the baseline choice, the attributions will always sum exactly to the output difference.

Slide 17 — Wrap-Up

[18:45–19:30]

Let me summarize. First, we examined N3 deep sleep in both raw and spectrogram representations, and confirmed delta dominance via power spectral analysis. Second, we ran two sequence-to-sequence models that correctly classify N3 at high confidence. Third, we applied Integrated Gradients and found that the attributions match known sleep physiology — the models are using the right features.

▷ Point to the Key Insight box.

The main takeaway is that **input representation matters** — not just for model performance, but for how interpretable the explanations are. Raw signals give temporal attribution; spectrograms give frequency attribution that we can directly validate against clinical band definitions.

▷ Point to the “also supports” box.

PhysioEx also includes other explainability methods — SpectralGradients for direct frequency-band attribution, concept-based explanations, and support for foundation models — but those are topics for another talk.

Slide 18 — Thank You

[19:30–20:00]

Thank you very much for your attention. All the code, the pretrained models, and the exact notebook I’ve been showing you today are available on GitHub. PhysioEx is on PyPI, so you can install it in one line and start experimenting with your own data. I’m happy to take any questions.

(Open for Q&A. Likely questions: “Why not SHAP?” (IG has completeness, no sampling), “How do you choose the number of steps?” (64 is standard, convergence plot in the paper), “Does this work for other stages?” (yes, but N3 is the clearest example).)