WAVEKERNEL: CONTINUOUS KERNEL SYNTHESIS VIA OPTICAL WAVE PROPAGATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Continuous-time models aim to learn discretization-consistent signal representations, with their expressive properties significantly influenced by the functional parameterization of local temporal operators. Prior approaches largely rely on predefined basis functions or neural networks, leaving the effective parameterization of continuous kernels an important open challenge. We propose WaveKernel, a continuous convolution operator that synthesizes kernels by evaluating the optical field generated by a slab-type diffractive model at continuous spatial positions. Learnable amplitude–phase modulation units jointly shape the spatial profile of the optical field, thereby defining a continuous kernel expanded in a basis of optical impulse responses. By introducing a structured inductive bias derived from optical wave propagation and mapping relative temporal coordinates onto continuous spatial positions for kernel evaluation, the proposed parameterization allows the same learned coefficients to be evaluated across different sampling rates. Integrated into a shared time-series architecture, WaveKernel improves seven-rate mean accuracy over spline-based continuous convolution on three sleep-staging benchmarks, while maintaining stable performance across the evaluated sampling rates.

1 INTRODUCTION

Neural operators learn mappings between infinite-dimensional function spaces rather than transformations bound to specific numerical grids, offering a powerful framework for discretization-consistent learning (Kovachki et al., 2023; Lu et al., 2021). This perspective has been widely adopted in scientific machine learning, where underlying physical fields are intrinsically continuous yet observed only through discrete sensors (Li et al., 2020; 2021). However, many existing neural operators focus on global function-to-function mappings, whereas sequential and spatiotemporal problems often require local operators defined over continuous spatial or temporal coordinates (Li et al., 2021; Lu et al., 2021; Brandstetter et al., 2022). Continuous convolution addresses this by defining a kernel $k(\tau)$ over continuous domains rather than fixed discrete taps (Romero et al., 2022b;a; Fey et al., 2018). Despite recent advances, the effective parameterization of continuous kernels remains an important open challenge.

For local continuous convolution, existing parameterizations fall broadly into two paradigms. Fixed-basis methods represent continuous kernels using predefined basis functions, such as splines or Fourier series (Fey et al., 2018; Worrall et al., 2017; Weiler et al., 2018). Neural parameterizations instead learn coordinate-to-weight mappings using neural networks (Wu et al., 2019; Romero et al., 2022b;a). The two paradigms impose distinct inductive biases: fixed bases encode explicit and interpretable structural priors but confine kernels to prescribed function families, whereas neural parameterizations afford greater flexibility at the cost of less explicit, architecture-dependent priors and greater optimization sensitivity (Sitzmann et al., 2020; Tancik et al., 2020; Rahaman et al., 2019). The choice of parameterization therefore shapes kernel expressivity, inductive bias, and parameter efficiency.

The central challenge, then, lies not in continuity alone, but in the structure governing how kernel values vary with coordinates. Wave propagation naturally redistributes a field across space, while superposition linearly combines contributions from multiple propagation paths (Shastri et al., 2021; Zhang et al., 2021). Building on these properties, we propose WaveKernel, a continuous kernel

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.

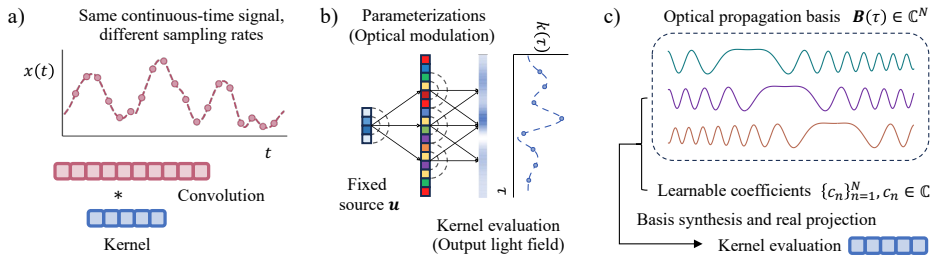


Figure 1: WaveKernel overview. a) A continuous-time signal is observed on different sampling grids. b) Fixed slab propagation and trainable optical modulation generate coordinate-dependent propagation responses for continuous kernel synthesis. c) Formulation of the continuous kernel as a learned linear superposition of optical impulse responses.

parameterization whose fixed kernel responses are derived from diffraction physics and propagation geometry. Concretely, the continuous kernel is constructed by sampling the output optical field generated by a slab-type diffractive model at continuous spatial positions, while a finite set of learnable kernel expansion coefficients is represented by trainable complex-amplitude modulation between two simulated diffraction stages (Zhu et al., 2022; Fu et al., 2023). The resulting continuous kernel is expressed as a linear superposition of these optical propagation responses. For time-series learning, relative temporal coordinates are mapped onto continuous spatial positions, allowing the corresponding optical responses to define kernel values at arbitrary temporal coordinates. Because the kernel is defined continuously through propagation responses, the same learned coefficients can be evaluated across different sampling grids without retraining.

Physiological signals, including electroencephalography (EEG), electrooculography (EOG), and electrocardiography (ECG), exhibit strong non-stationarity, localized temporal structure, and heterogeneous acquisition settings, making robust modeling across temporal resolutions particularly important (Jiang et al., 2024; Eldele et al., 2021; Lawhern et al., 2018). We integrate WaveKernel into NeurOCNN (Kumar et al., 2026) by replacing its spline-based continuous convolution kernel. Across five folds, WaveKernel improves the average accuracy across seven sampling rates on three sleep benchmarks by 1.57–3.66 percentage points, while achieving comparable performance on ECG with a 0.37-point deficit. Across sampling rates, WaveKernel exhibits stability comparable to spline, while noise experiments reveal a distinct robustness trade-off under strong broadband corruption.

Our main contribution is WaveKernel, a continuous-kernel parameterization that synthesizes kernels as learned combinations of fixed optical propagation responses, enabling the same learned coefficients to be evaluated across different temporal sampling grids. We validate this formulation through numerical simulation with task performance, unseen-rate generalization, ablations, and noise robustness experiments.

2 Related Work

Neural Operators Neural operator learning provides a framework for learning mappings between function spaces in a discretization-consistent manner, allowing the same learned parameters to be evaluated across different sampling grids (Kovachki et al., 2023; Bartolucci et al., 2023). The Fourier neural operator parameterizes integral transformations in Fourier space (Li et al., 2021), while DeepONet combines an encoding of the input function with a representation of the output coordinates (Lu et al., 2021). Subsequent architectures have explored a range of extensions, including multiscale backbones (Rahman et al., 2023), localized wavelet representations (Gupta et al., 2021), and Laplace-domain formulations (Cao et al., 2024). Building on these developments, neural operators have been widely applied in scientific computing, particularly for surrogate modeling of partial differential equations (PDEs) (Kovachki et al., 2023).

Continuous-Time Signal Learning Conventional time-series models typically operate on discretized sequences defined over predefined temporal grids, leaving their representations dependent on the sampling and tokenization used during training (Wu et al., 2023; Nie et al., 2023; Goswami et al., 2024). Continuous-kernel convolutions offer an alternative formulation by defining temporal filters as functions of continuous coordinates, allowing the same kernel to be evaluated across different sampling resolutions (Romero et al., 2022b;a). More recently, NeurOCNN extends the discretization-consistent perspective of operator learning to physiological sequence modeling through spline-parameterized continuous convolutions, enabling decoding across varying sampling rates (Kumar et al., 2026). Its spline-based formulation highlights kernel parameterization as a key design choice in continuous convolution operators.

Continuous Kernel Parameterizations Continuous convolution defines kernel values as functions of relative coordinates, allowing the same kernel to be evaluated across different sampling grids (Schütt et al., 2017). Prescribed functional structures introduce inductive biases, such as B-spline bases on irregular geometric domains (Fey et al., 2018) and circular harmonics for rotational equivariance (Worrall et al., 2017). Neural kernel generators instead learn coordinate-to-weight mappings using multilayer perceptrons, for local geometric kernels in PointConv (Wu et al., 2019) and continuous temporal kernels in CKConv (Romero et al., 2022b). FlexConv further learns kernel support and controls kernel bandwidth to enable evaluation across varying resolutions (Romero et al., 2022a). Our formulation follows a finite basis expansion, where the basis functions are determined by diffraction physics and propagation geometry, while task-specific adaptation is captured by learnable complex coefficients. This ties the representable kernel family to a physical propagation model while keeping the learned parameters independent of the sampling grid.

Photonic Neural Computing The inherent parallelism and high propagation speed of light make photonic systems attractive for high-throughput and low-latency computation, motivating optical implementations of neural network operations (Wetzstein et al., 2020; Shastri et al., 2021). Coherent nanophotonic neural networks realize trainable linear transformations through programmable interferometric circuits (Shen et al., 2017). Diffractive neural networks instead exploit wave propagation between spatially modulating layers for optical inference (Lin et al., 2018). This principle has been extended to silicon photonic platforms using on-chip diffractive units and slab propagation (Fu et al., 2023). Hybrid integrated architectures further combine diffractive Fourier-transform cells with programmable Mach–Zehnder interferometers for convolution and related transformations (Zhu et al., 2022). Optical propagation naturally supports spatial feature mixing and continuous aggregation, while wave interference and tunable modulation provide flexible mechanisms for representation encoding (Xu et al., 2024). Building on these physical primitives, our work leverages propagation-defined optical responses as continuous basis functions for kernel synthesis, linking photonic wave dynamics with temporal modeling across discretizations.

3 METHOD

This section details the methodology of WaveKernel. We begin by formulating the continuous-time convolution problem considered in this work (Section 3.1). Next, Section 3.2 introduces the optical kernel synthesis mechanism, where deterministic propagation provides a fixed continuous basis, and low-dimensional complex coefficients linearly combine these basis functions into task-specific kernels. Section 3.3 describes kernel evaluation and numerical calibration across different sampling grids. Finally, we present the integration of the WaveKernel operator into the time-series learning architecture (Section 3.4).

3.1 PROBLEM FORMULATION

Given multichannel time series observed at potentially different sampling rates, we consider a local temporal operator whose kernel is defined in continuous physical time and independent of any particular sampling grid. For an input signal $\mathbf{x}(t)$, the continuous local operator is formulated as

$$(\mathcal{K}\mathbf{x})_o(t) = \sum_{i=1}^{C_{\text{in}}} \int_{\mathcal{I}_D} k_{o,i}(\tau) x_i(t-\tau)\, d\tau, \tag{1}$$

where C_{in} is the number of input channels, $\mathcal{I}_D = [-D/2, D/2]$ denotes the temporal support of duration D seconds, and $k_{o,i} : \mathcal{I}_D \to \mathbb{R}$ is the continuous kernel associated with input–output channel pair (o, i) (Fey et al., 2018).

The kernel $k_{o,i}(\tau)$ is defined over continuous temporal coordinates rather than discrete sample indices. Consequently, changing the sampling rate alters its numerical evaluation grid while leaving the underlying parameterization unchanged. The primary modeling challenge is therefore to construct a compact and learnable family of continuous functions for $k_{o,i}(\tau)$.

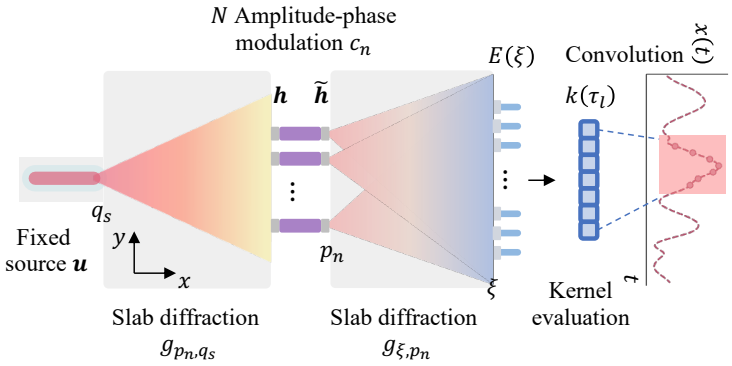


Figure 2: Continuous kernel synthesis in WaveKernel. A fixed optical propagation model maps the source excitation to latent modes, which are modulated by trainable amplitude–phase coefficients and propagated again to form the continuous field $E(\xi)$. Mapping spatial coordinate ξ to temporal offset τ yields the continuous kernel $k(\tau)$.

3.2 Optical Kernel Synthesis

Integrated diffractive photonic systems exploit optical propagation in slab waveguides to realize dense linear transformations through the interference of spatially distributed optical fields (Zhu et al., 2022). Motivated by this physical mechanism, we construct the continuous kernel basis using two stages of deterministic slab propagation separated by a low-dimensional complex modulation stage. A fixed optical excitation is first propagated to a set of latent positions, where mode-wise complex coefficients provide task-specific adaptation. The modulated field then undergoes a second passive propagation, producing a coordinate-dependent optical response from which the continuous kernel is synthesized.

Based on the scalar Huygens–Fresnel principle, we define the complex propagation coefficient from a source point q to an observation point p in the slab waveguide as

$$g_{p,q} = \frac{1 + \cos\theta_{p,q}}{2r_{p,q}} \exp\left(\mathrm{j}\frac{2\pi n_{\text{slab}}}{\lambda} r_{p,q}\right), \tag{2}$$

where $r_{p,q} = \sqrt{(x_p - x_q)^2 + (y_p - y_q)^2}$ is the propagation distance, $\cos\theta_{p,q} = (x_p - x_q)/r_{p,q}$ represents the obliquity factor, λ is the operating wavelength, n_{slab} is the effective refractive index of the slab, and $\mathrm{j} = \sqrt{-1}$ denotes the imaginary unit (Fu et al., 2023). The magnitude of $g_{p,q}$ characterizes the geometric attenuation of the optical field, while its phase encodes the accumulated optical phase through the slab.

Let $\{q_s\}_{s=1}^{S}$ denote the fixed source positions, $\{p_n\}_{n=1}^{N}$ the latent optical positions, and ξ a continuous transverse coordinate on the output plane. The first diffraction stage defines the propagation matrix $\mathbf{W}_1 \in \mathbb{C}^{N\times S}$ with entries $[\mathbf{W}_1]_{n,s} = g_{p_n,q_s}$, which maps the source excitation to the latent optical field,

$$\mathbf{h} = \mathbf{W}_1\mathbf{u}. \tag{3}$$

Here, $\mathbf{u} \in \mathbb{C}^S$ is the fixed source-excitation vector, and h_n, the n-th element of $\mathbf{h}$, represents the complex optical field at latent position p_n.

Task-specific adaptation is introduced by applying a learnable complex coefficient $c_n \in \mathbb{C}$ to each latent mode. The modulated latent field is

$$\widetilde{h}_n = c_n h_n. \tag{4}$$

The modulated latent field then undergoes a second diffraction stage. At a continuous transverse coordinate ξ on the output plane, the resulting complex optical field is

$$E(\xi) = \sum_{n=1}^{N} g_{\xi,p_n}\, \widetilde{h}_n. \tag{5}$$

To associate the optical field with a continuous temporal kernel, we map the relative temporal coordinate $\tau \in \mathcal{I}_D$ to the transverse coordinate as $\xi = \gamma\tau$, where $\gamma = L_\xi/D$ and L_ξ denotes the transverse span corresponding to the kernel duration D. Evaluating the optical field along this mapping yields

$$E(\tau) = \sum_{n=1}^{N} g_{\gamma\tau,p_n}\, \widetilde{h}_n. \tag{6}$$

The corresponding signed real-valued convolution kernel is obtained through real-quadrature readout,

$$k(\tau) = \mathrm{Re}\{E(\tau)\}\,. \tag{7}$$

The above propagation process can be equivalently expressed as a continuous basis expansion. Specifically, we define the response associated with the n-th latent mode as

$$B_n(\tau) = g_{\gamma\tau,p_n} h_n. \tag{8}$$

Since $\widetilde{h}_n = c_n h_n$, the continuous optical field can be rewritten as

$$E(\tau) = \sum_{n=1}^{N} c_n B_n(\tau). \tag{9}$$

Its real part gives the convolution kernel in Eq. (7).

Here, $\{B_n(\tau)\}_{n=1}^{N}$ forms a fixed continuous basis determined by the optical propagation geometry and source excitation, while $\{c_n\}_{n=1}^{N}$ specifies its task-specific combination. Figure 6 visualizes example response magnitudes and phases. Consequently, the synthesized kernels lie in the real span of the real and imaginary parts of the propagation responses, with dimension at most $2N$ per channel pair. The implemented coefficient constraints select a subset of this span; neither linear independence nor approximation universality is assumed. A learned layer-level readout phase can be absorbed into the coefficients under the amplitude-phase formulation (Appendix A.2).

3.3 Continuous Kernel Evaluation

The kernel in Eq. (7) is defined in physical time and is therefore independent of any particular sampling grid. For a uniformly sampled signal with sampling rate f_s, the number of kernel samples is determined by the physical duration D as

$$K(f_s) = \max\{\mathrm{round}(Df_s), 1\}. \tag{10}$$

The corresponding temporal coordinates are

$$\tau_\ell = \frac{\ell - (K-1)/2}{f_s}, \qquad \ell = 0, \ldots, K-1, \tag{11}$$

with sampling interval $\Delta t = 1/f_s$.

The discrete convolution weights used in our experiments are

$$w_{o,i,\ell}^{(f_s)} = \rho_{\mathrm{conv}}(f_s)\, k_{o,i}(\tau_\ell), \tag{12}$$

where $\rho_{\mathrm{conv}}(f_s)$ is a deterministic sampled-basis normalization factor defined in Appendix A.3. Changing the sampling rate therefore changes the number and locations of kernel samples without altering the learned coefficients $\{c_{o,i,n}\}$ or the underlying continuous kernel parameterization.

Before sampled-basis normalization, the continuous operator in Eq. (1) admits the quadrature approximation

$$(\mathcal{K}\mathbf{x})_o(t) \approx \Delta t \sum_{i=1}^{C_{\mathrm{in}}} \sum_{\ell=0}^{K-1} k_{o,i}(\tau_\ell) x_i(t-\tau_\ell). \tag{13}$$

The underlying continuous kernel parameterization and learned coefficients remain unchanged across sampling rates. The implemented operator additionally depends on the evaluation grid, quadrature interval, and sampled-basis normalization. All reported experiments use this normalization; cross-rate robustness is evaluated empirically. We assume uniformly sampled inputs, with sampling rates allowed to differ between training and evaluation. Normalization and discrete alignment conventions are detailed in Appendix A.3.

3.4 Model Architecture

We integrate the proposed continuous local operator into a function-to-label architecture for time-series classification. Given an input signal $\mathbf{x}$ observed at sampling rate f_s, the complete model is expressed as

$$\hat{\mathbf{y}} = \mathcal{G}_\Theta(\mathbf{x}; f_s) = \mathcal{H}_\psi\left(\mathcal{B}_\theta(\mathbf{x}; f_s)\right), \tag{14}$$

where $\hat{\mathbf{y}} \in \mathbb{R}^{C_{\mathrm{cls}}}$ denotes the classification output, $\mathcal{B}_\theta$ is the continuous local-operator backbone, and $\mathcal{H}_\psi$ is the downstream prediction head. Here, C_{cls} is the number of target classes, θ and ψ denote the trainable parameters of the backbone and prediction head, respectively, and $\Theta = (\theta, \psi)$.

The backbone consists of four local operator layers with channel widths $C_{\mathrm{in}} \to 64 \to 128 \to 64 \to 32$. The first three layers are followed by GELU activation and one-group GroupNorm, while the final layer is followed by GroupNorm. Each local operator uses the continuous kernel defined in Section 3.2, with its temporal support specified in physical time and its discrete evaluation grid determined by the input sampling rate as described in Section 3.3.

The backbone produces a time-indexed feature sequence. Following NeurOCNN (Kumar et al., 2026), physical-time segments are projected onto a fixed Fourier basis to form 32-dimensional tokens, which are processed by attention and a Transformer encoder. Learned-query attention pooling then aggregates the token sequence, followed by an MLP classifier. These downstream components are shared by the WaveKernel, spline, and CKConv variants, providing a common backbone and head for comparing kernel parameterizations. Implementation details are provided in Appendix B.3.

4 Experiments

In this section, we evaluate WaveKernel on four physiological time-series benchmarks spanning EEG, EOG, and ECG modalities: SleepEOG (EOG sleep staging) (Kemp et al., 2000), ISRUC and HMC (EEG sleep staging) (Alvarez-Estevez & Rijsman, 2021; Khalighi et al., 2016), and PhysioNet/CinC 2017 (ECG rhythm classification) (Clifford et al., 2017). We compare its performance against LocalNO (Liu-Schiaffini et al., 2024) and U-Sleep (Perslev et al., 2021) as strong representative baselines from the neural-operator and non-operator model families (Li et al., 2021), respectively, along with spline-based NeurOCNN and an MLP-parameterized CKConv variant as continuous-time counterparts. We further examine robustness to unseen sampling rates and conduct controlled ablations to study the effects of key WaveKernel design choices. Dataset preprocessing and model configurations are detailed in Appendix B.

4.1 Physiological Benchmark Performance

Table 1 reports accuracy averaged over seven sampling rates within each fold, followed by the mean and standard deviation across five folds. WaveKernel improves over spline by 1.95, 1.57, and 3.66 percentage points on ISRUC, HMC, and SleepEOG, respectively, with consistent gains across all five folds. On ECG, spline performs slightly better by 0.37 points. Appendix C reports detailed per-rate comparisons among these methods.

Table 1: Five-fold accuracy (%), reported as mean ± standard deviation over folds after averaging across 80, 100, 128, 200, 256, 512, and 1024 Hz. Bold/underline indicate best/second-best results.

Method	Params	ISRUC	HMC	Sleep-EDF (EOG)	ECG
NeurOCNN + Spline	319.59K	74.71 ± 1.30	73.41 ± 0.87	<u>55.78 ± 1.30</u>	**73.41 ± 0.58**
NeurOCNN + WaveKernel	338.15K	**76.66 ± 1.18**	<u>74.98 ± 1.43</u>	**59.44 ± 1.44**	<u>73.04 ± 0.16</u>
NeurOCNN + CKConv	677.25K	56.47 ± 7.75	64.39 ± 3.79	35.90 ± 4.25	58.83 ± 2.60
LocalNO	57K	72.55 ± 1.44	69.19 ± 1.05	53.83 ± 2.00	–
U-Sleep	3.71M	<u>75.93 ± 1.99</u>	**75.90 ± 1.67**	53.51 ± 0.89	–

Compared with the task-specific U-Sleep baseline, WaveKernel performs better on ISRUC and SleepEOG, while U-Sleep is slightly stronger on HMC. Overall, these results indicate that WaveKernel provides a competitive continuous-kernel parameterization across physiological benchmarks, with particularly consistent improvements on sleep-staging tasks.

4.2 Discretization Robustness

Figure 3 shows that WaveKernel and spline maintain nearly flat accuracy profiles across the evaluated sampling rates, indicating comparable cross-rate robustness. Tables 3 and 4 report the rate-wise standard deviation and worst-case accuracy drop, respectively.

Importantly, WaveKernel preserves this robustness while achieving higher accuracy on three of the four benchmarks (Table 1). This suggests that its advantage over spline is more closely associated with its optical propagation-based kernel parameterization than with improved robustness to temporal discretization.

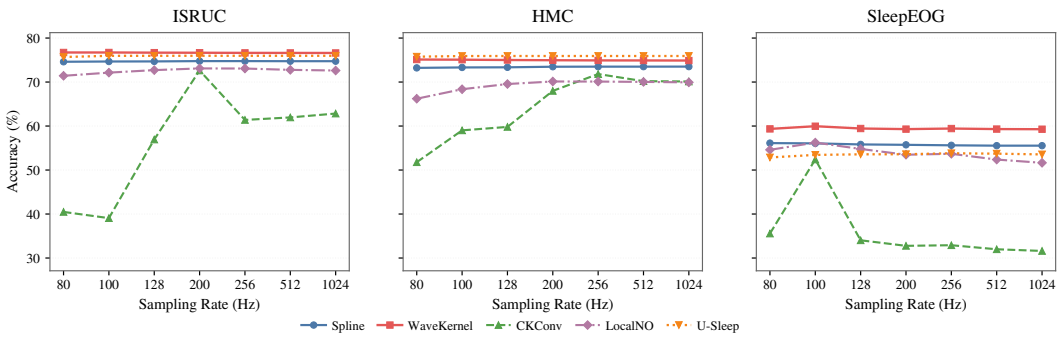


Figure 3: Test accuracy across sampling rates on ISRUC, HMC, and SleepEOG. Markers denote five-fold means, with training rates of 200, 256, and 100 Hz, respectively.

4.3 Ablation Studies

Retrained geometry sweeps (Figure 4) reveal a broad effective region rather than a single optimal configuration. Intermediate output spans of 12–24 μm yield similar accuracy, whereas performance decreases at the tested extremes. Increasing the second propagation distance reduces accuracy overall, while latent spacing exhibits a weaker nonmonotonic effect. These results suggest that the propagation geometry acts as a meaningful inductive bias.

Figure 5 further examines latent dimension and coefficient constraints. Phase-adaptive variants outperform amplitude-only adaptation across all tested dimensions, while the difference between phase-only and amplitude–phase adaptation is comparatively small. For amplitude–phase adaptation, accuracy increases from 58.40% at $N = 2$ to 59.89% at $N = 8$ and 60.04% at $N = 16$, indicating diminishing returns with increasing latent dimension. These experiments are parameterization rather than equal-budget ablations, since changing N affects both the coefficient count and latent-array extent (Appendix B.6).

4.4 Noise Robustness

On SleepEOG (Figure 7), WaveKernel's higher accuracy under powerline interference and baseline drift largely reflects its clean-accuracy advantage. Strong broadband corruption reveals a different trade-off: at 10 dB white noise, WaveKernel drops to 42.08% accuracy compared with 54.01% for spline, while EMG-like noise produces a similar but less pronounced pattern. These results indicate that WaveKernel's clean-accuracy gains do not translate into uniformly improved noise robustness.

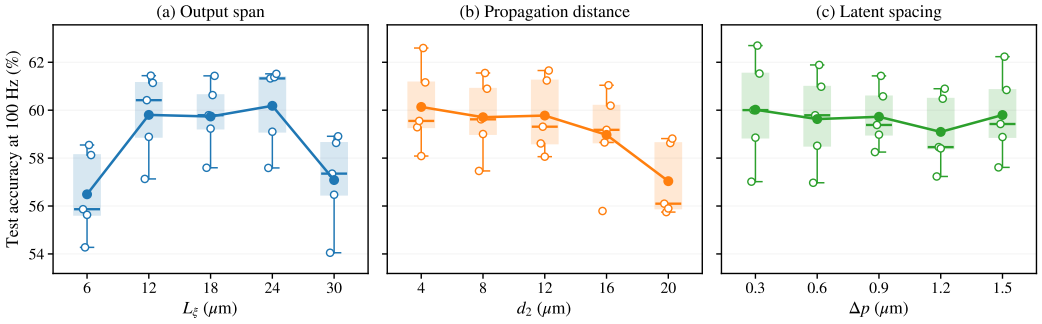


Figure 4: Geometry ablations on SleepEOG at 100 Hz with eight latent modes and amplitude–phase coefficients. (a) Output span L_ξ, (b) second-stage propagation distance d_2, and (c) latent pitch Δp. Hollow points denote five-fold accuracies, boxes show quartiles and medians, and curves connect mean accuracies. Each configuration is trained separately.

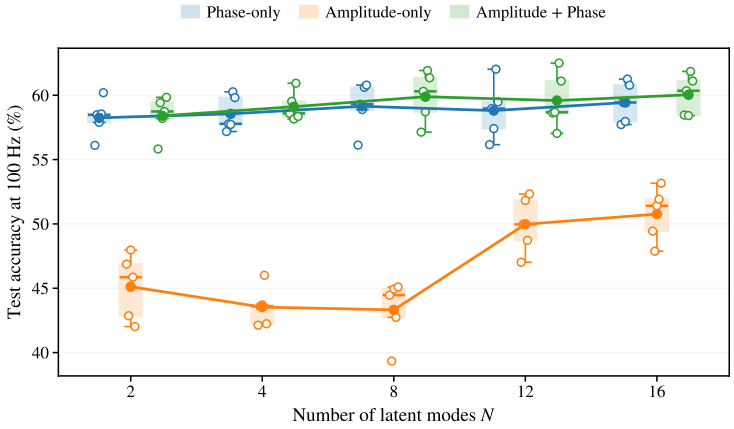


Figure 5: Latent-dimension and coefficient ablations on SleepEOG at 100 Hz. Hollow points denote five-fold accuracies, boxes show quartiles and medians, and curves connect fold means. Each configuration is trained separately.

5 Discussion

Propagation-defined function space. WaveKernel differs from spline-based continuous convolution primarily in the function family used to parameterize the kernel. The spline baseline constructs a natural cubic interpolant from values defined at uniformly spaced knots, whereas WaveKernel represents the kernel using smooth propagation responses whose amplitude and phase vary jointly with coordinate. Writing $B_n = u_n + \mathrm{j}v_n$ and $c_n = a_n \exp(\mathrm{j}\phi_n)$ gives $\mathrm{Re}(c_n B_n) = a_n(u_n \cos\phi_n - v_n \sin\phi_n)$, showing that phase adaptation selects combinations of two real response components per mode. Interference among responses with different path-length profiles can therefore generate coordinated oscillatory structure across the kernel support. This provides a plausible

inductive bias for physiological rhythms and transient patterns and is consistent with the strong performance of phase-adaptive variants and the diminishing gains observed as N increases.

Discretization and perturbation robustness. Both WaveKernel and spline define kernels continuously and reevaluate them on the target sampling grid rather than learning a separate discrete filter for each resolution. Their similar cross-rate behavior therefore suggests that discretization robustness is largely associated with continuous kernel evaluation, while their difference in predictive accuracy is more closely related to the underlying kernel parameterization. Continuity alone does not imply invariant predictions, since finite-grid evaluation, normalization, nonlinear processing, and downstream pooling also affect the realized model. The noise experiments further distinguish cross-rate stability from robustness to strong broadband corruption.

Scope of the interpretation. The consistent gains on the sleep benchmarks support propagation-based kernel synthesis as a useful parameterization in these settings, while the small ECG deficit and noise trade-offs limit broader claims. Optimization, effective capacity, coefficient constraints, geometry, and dataset-specific structure remain possible contributors to the observed differences. The present results concern the functional parameterization induced by simulated wave propagation.

6 Conclusion

We introduced WaveKernel, a continuous convolution operator that synthesizes kernels from fixed optical propagation responses with learned complex coefficients. Across physiological benchmarks, WaveKernel improves over spline-based continuous convolution on all three sleep-classification datasets while achieving comparable performance on ECG. It maintains stable performance when evaluated directly on unseen sampling grids without retraining, while coefficient and geometry ablations highlight the role of the propagation-defined function family. These results suggest that physical wave propagation provides a useful inductive bias for continuous kernel parameterization and offers a promising direction for discretization-consistent physiological signal modeling.

Limitations and Future Work

Our experiments rely on numerical optical propagation rather than fabricated photonic hardware. The simulation does not capture fabrication tolerances, phase drift, device noise, nonuniform loss, or calibration errors, and therefore does not establish robustness or efficiency in a physical implementation. Digital evaluation also incurs the cost of basis computation and sampled convolution, so no hardware speed or energy advantage is claimed.

Evaluation is limited to physiological classification, a finite set of baselines, and uniformly sampled signals. Broader tasks, capacity-matched comparisons, additional training seeds, and stronger robustness studies are needed to assess generality. Future work includes physical-device validation, learned or partially optimized propagation geometry, native irregular-time processing, and theoretical analysis of approximation properties, stability, and discretization error.

Reproducibility Statement

The propagation model, coefficient constraints, and numerical calibration are specified in Appendix 6. Dataset preprocessing, grouped data splits, training settings, and model configurations are described in Appendix B. Sampling-rate and noise evaluation protocols are given in Appendices B.4 and B.5, with per-rate results in Appendix C.

References

Diego Alvarez-Estevez and Roselyne Rijsman. Haaglanden medisch centrum sleep staging database (version 1.0.1). *PhysioNet*, 2021.

Francesca Bartolucci, Emmanuel De Bézenac, Bogdan Raonic, Roberto Molinaro, Siddhartha Mishra, and Rima Alaifari. Representation equivalent neural operators: a framework for alias-free operator learning. *Advances in Neural Information Processing Systems*, 36:69661–69672, 2023.

Johannes Brandstetter, Daniel E. Worrall, and Max Welling. Message passing neural PDE solvers. In *International Conference on Learning Representations*, 2022.

Qianying Cao, Somdatta Goswami, and George Em Karniadakis. Laplace neural operator for solving differential equations. *Nature Machine Intelligence*, 6(6):631–640, 2024.

Gari D Clifford, Chengyu Liu, Benjamin Moody, Li-wei H Lehman, Ikaro Silva, Qiao Li, Alistair E Johnson, and Roger G Mark. Af classification from a short single lead ecg recording: The physionet/computing in cardiology challenge 2017. In *2017 computing in cardiology (CinC)*, pp. 1–4. IEEE, 2017.

Emadeldeen Eldele, Zhenghua Chen, Chengyu Liu, Min Wu, Chee-Keong Kwoh, Xiaoli Li, and Cuntai Guan. An attention-based deep learning approach for sleep stage classification with single-channel eeg. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 29:809–818, 2021.

Matthias Fey, Jan Eric Lenssen, Frank Weichert, and Heinrich Müller. SplineCNN: Fast geometric deep learning with continuous B-spline kernels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

Tingzhao Fu, Yubin Zang, Yuyao Huang, Zhenmin Du, Honghao Huang, Chengyang Hu, Minghua Chen, Sigang Yang, and Hongwei Chen. Photonic machine learning with on-chip diffractive optics. *Nature Communications*, 14:70, 2023.

Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. Moment: A family of open time-series foundation models. In *International Conference on Machine Learning*, 2024.

Gaurav Gupta, Xiongye Xiao, and Paul Bogdan. Multiwavelet-based operator learning for differential equations. *Advances in neural information processing systems*, 34:24048–24062, 2021.

Wei-Bang Jiang, Liming Zhao, and Bao-Liang Lu. Large brain model for learning generic representations with tremendous eeg data in bci. In *International Conference on Learning Representations*, volume 2024, pp. 16405–16426, 2024.

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.

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.

Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to pdes. *Journal of Machine Learning Research*, 24(89):1–97, 2023.

Daya Kumar, Uday Devulapalli, Aarat Satsangi, and Apurva Narayan. NeurOCNN: A neural-operator-based model for physiological time series. In *International Conference on Machine Learning*, 2026.

Vernon J Lawhern, Amelia J Solon, Nicholas R Waytowich, Stephen M Gordon, Chou P Hung, and Brent J Lance. Eegnet: a compact convolutional neural network for eeg-based brain–computer interfaces. *Journal of neural engineering*, 15(5):056013, 2018.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Andrew Stuart, Kaushik Bhattacharya, and Anima Anandkumar. Multipole graph neural operator for parametric partial differential equations. In *Advances in neural information processing systems*, volume 33, pp. 6755–6766, 2020.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021.

Xing Lin, Yair Rivenson, Nezih T. Yardimci, Muhammed Veli, Yi Luo, Mona Jarrahi, and Aydogan Ozcan. All-optical machine learning using diffractive deep neural networks. *Science*, 361(6406): 1004–1008, 2018.

Miguel Liu-Schiaffini, Julius Berner, Boris Bonev, Thorsten Kurth, Kamyar Azizzadenesheli, and Anima Anandkumar. Neural operators with localized integral and differential kernels. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 32576–32594. PMLR, 2024.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature machine intelligence*, 3(3):218–229, 2021.

Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *The Eleventh International Conference on Learning Representations*, 2023.

Mathias Perslev, Sune Darkner, Lykke Kempfner, Miki Nikolic, Poul Jørgen Jennum, and Christian Igel. U-sleep: resilient high-frequency sleep staging. *NPJ digital medicine*, 4(1):72, 2021.

Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred A. Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *International Conference on Machine Learning*, volume 97, pp. 5301–5310, 2019.

Md Ashiqur Rahman, Zachary E Ross, and Kamyar Azizzadenesheli. U-NO: U-shaped neural operators. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856.

David W. Romero, Robert-Jan Bruintjes, Erik J. Bekkers, Jakub M. Tomczak, Mark Hoogendoorn, and Jan C. van Gemert. FlexConv: Continuous kernel convolutions with differentiable kernel sizes. In *International Conference on Learning Representations*, 2022a.

David W. Romero, Anna Kuzina, Erik J. Bekkers, Jakub M. Tomczak, and Mark Hoogendoorn. CKConv: Continuous kernel convolution for sequential data. In *International Conference on Learning Representations*, 2022b.

Kristof Schütt, Pieter-Jan Kindermans, Huziel Enoc Sauceda Felix, Stefan Chmiela, Alexandre Tkatchenko, and Klaus-Robert Müller. Schnet: A continuous-filter convolutional neural network for modeling quantum interactions. *Advances in neural information processing systems*, 30, 2017.

Bhavin J Shastri, Alexander N Tait, Thomas Ferreira de Lima, Wolfram HP Pernice, Harish Bhaskaran, C David Wright, and Paul R Prucnal. Photonics for artificial intelligence and neuromorphic computing. *Nature Photonics*, 15(2):102–114, 2021.

Yichen Shen, Nicholas C. Harris, Scott Skirlo, Mihika Prabhu, Tom Baehr-Jones, Michael Hochberg, Xin Sun, Shijie Zhao, Hugo Larochelle, Dirk Englund, and Marin Soljačić. Deep learning with coherent nanophotonic circuits. *Nature Photonics*, 11:441–446, 2017.

Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *Advances in Neural Information Processing Systems*, volume 33, pp. 7462–7473, 2020.

Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Maurice Weiler, Mario Geiger, Max Welling, Wouter Boomsma, and Taco S. Cohen. 3d steerable CNNs: Learning rotationally equivariant features in volumetric data. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Gordon Wetzstein, Aydogan Ozcan, Sylvain Gigan, Shanhui Fan, Dirk Englund, Marin Soljačić, Cornelia Denz, David AB Miller, and Demetri Psaltis. Inference in artificial intelligence with deep optics and photonics. *Nature*, 588(7836):39–47, 2020.

Daniel E. Worrall, Stephan J. Garbin, Daniyar Turmukhambetov, and Gabriel J. Brostow. Harmonic networks: Deep translation and rotation equivariance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5028–5037, 2017.

Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. Timesnet: Temporal 2d-variation modeling for general time series analysis. In *The Eleventh International Conference on Learning Representations*, 2023.

Wenxuan Wu, Zhongang Qi, and Li Fuxin. PointConv: Deep convolutional networks on 3d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

Zhihao Xu, Tiankuang Zhou, Muzhou Ma, ChenChen Deng, Qionghai Dai, and Lu Fang. Large-scale photonic chiplet taichi empowers 160-tops/w artificial general intelligence. *Science*, 384 (6692):202–209, 2024.

Hui Zhang, Mile Gu, Xu Dong Jiang, Jayne Thompson, Hong Cai, Stefano Paesani, Raffaele Santagati, Anthony Laing, Yi Zhang, Man-Hong Yung, et al. An optical neural chip for implementing complex-valued neural network. *Nature communications*, 12(1):457, 2021.

H. H. Zhu, J. Zou, H. Zhang, Y. Z. Shi, S. B. Luo, N. Wang, H. Cai, L. X. Wan, B. Wang, X. D. Jiang, J. Thompson, X. S. Luo, X. H. Zhou, L. M. Xiao, W. Huang, L. Patrick, M. Gu, L. C. Kwek, and A. Q. Liu. Space-efficient optical computing with an integrated chip diffractive neural network. *Nature Communications*, 13:1044, 2022.

A WaveKernel Parameterization and Numerical Evaluation

A.1 Propagation Geometry and Optical Basis

Source and latent positions are placed symmetrically around the transverse origin. For a latent array containing N positions with pitch Δp,

$$y_n^{\mathrm{lat}} = \left(n - \frac{N+1}{2}\right)\Delta p, \qquad n = 1, \dots, N. \tag{15}$$

The same construction defines y_s^{src} using the source count S and source pitch.

Unless otherwise specified, the fixed source uses uniform normalized excitation,

$$u_s = \frac{1}{\sqrt{S}}, \qquad s = 1, \dots, S. \tag{16}$$

For a set of kernel coordinates $\{\tau_\ell\}_{\ell=0}^{K-1}$, the sampled complex basis can be written compactly as

$$[\mathbf{B}_{\mathrm{raw}}]_{\ell n} = g_{\gamma\tau_\ell, p_n} h_n, \qquad \mathbf{B}_{\mathrm{raw}} = \mathbf{W}_2 \,\mathrm{diag}(\mathbf{h}). \tag{17}$$

Here, $\mathbf{W}_2 \in \mathbb{C}^{K\times N}$ has entries $[\mathbf{W}_2]_{\ell n} = g_{\gamma\tau_\ell, p_n}$. Because the geometry and excitation are fixed, sampled basis matrices may be cached for repeated evaluations on the same temporal grid.

Default optical parameters. The default numerical configuration uses $\lambda = 1.55\,\mu\mathrm{m}$, $n_{\mathrm{slab}} = 3.48$, $d_1 = d_2 = 8\,\mu\mathrm{m}$, source spacing $1.0\,\mu\mathrm{m}$, latent spacing $0.6\,\mu\mathrm{m}$, and optical span $L_\xi = 12\,\mu\mathrm{m}$. These values define the numerical propagation basis and should not be interpreted as an experimentally calibrated fabricated device.

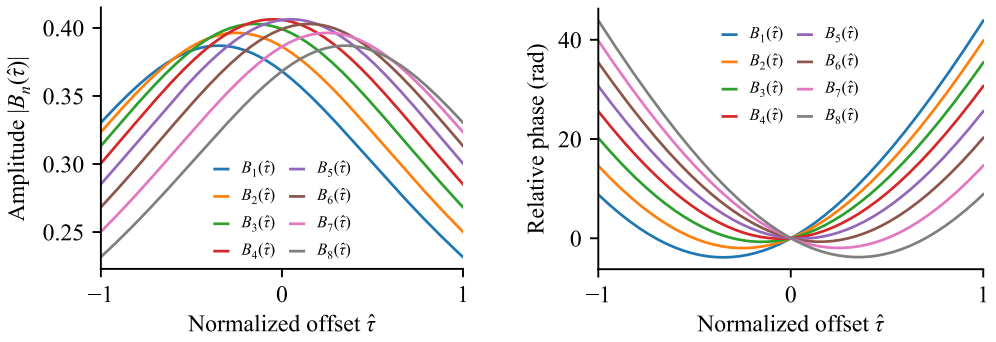


Figure 6: Magnitude (left) and relative phase (right) of eight complex propagation responses over normalized temporal offset $\hat{\tau} = 2\tau/D$. Magnitudes use the model's amplitude scaling without per-mode normalization; phases are unwrapped and referenced to $\hat{\tau} = 0$. Learned coefficients combine these responses into task-specific real-valued kernels.

A.2 Geometry of Complex Adaptation

The complex basis provides a useful interpretation of the phase-constrained kernel family. Writing

$$B_n(\tau) = u_n(\tau) + \mathrm{j}v_n(\tau), \tag{18}$$

the real contribution of latent mode n becomes

$$a_n \left[u_n(\tau)\cos(\phi_n - \varphi_{\mathrm{rd}}) - v_n(\tau)\sin(\phi_n - \varphi_{\mathrm{rd}})\right]. \tag{19}$$

For phase-only adaptation, the coefficients applied to (u_n, v_n) therefore satisfy

$$\alpha_n^2 + \beta_n^2 = 1, \qquad \alpha_n = \cos(\phi_n - \varphi_{\mathrm{rd}}), \quad \beta_n = -\sin(\phi_n - \varphi_{\mathrm{rd}}), \tag{20}$$

and lie on a unit circle. Amplitude adaptation permits the radius to vary, enlarging the accessible function family without changing the physical basis.

When latent phases are trainable, the shared readout phase φ_{rd} can be absorbed into a common shift of all ϕ_n, adding an optimization variable without enlarging the kernel family. Amplitude–phase adaptation learns $a_n = 2\,\mathrm{sigmoid}(r_n) \in (0, 2)$ and ϕ_n; phase-only fixes $a_n = 1$, whereas amplitude-only fixes the latent phases at zero. All variants retain a learned shared readout phase and fixed unit gain. Thus amplitude-only still permits a common quadrature rotation, but not independent phase adaptation across modes. Amplitudes start at one, trainable latent phases are uniform on $[0, 2\pi)$, and the shared readout phase starts at zero.

A.3 Numerical Evaluation and Calibration

Convolution-basis calibration. For WaveKernel, let $\mathbf{B}_{\mathrm{raw}} \in \mathbb{C}^{K\times N}$ denote the sampled complex basis at sampling rate f_s. Sampled-basis normalization computes

$$R_{\mathrm{conv}}(f_s) = \left(\frac{1}{KN}\sum_{\ell=0}^{K-1}\sum_{n=1}^{N} |[\mathbf{B}_{\mathrm{raw}}]_{\ell n}|^2\right)^{1/2}, \tag{21}$$

and applies

$$\rho_{\mathrm{conv}}(f_s) = \frac{1}{\sqrt{N}\max\{R_{\mathrm{conv}}(f_s), 10^{-8}\}}. \tag{22}$$

The sampled basis used to construct the numerical kernel is therefore

$$\mathbf{B}_{\mathrm{conv}}^{(f_s)} = \rho_{\mathrm{conv}}(f_s)\mathbf{B}_{\mathrm{raw}}, \tag{23}$$

and the corresponding sampled kernel can be written as

$$w_{o,i,\ell}^{(f_s)} = \rho_{\mathrm{conv}}(f_s)k_{o,i}(\tau_\ell) \tag{24}$$

when $k_{o,i}$ denotes the underlying unnormalized continuous kernel.

Although $\rho_{\mathrm{conv}}(f_s)$ is deterministic and parameter-free, it generally depends on the sampled grid. The reported experiments use this normalization, so cross-rate results establish empirical robustness of the calibrated implementation rather than identical scaling across grids.

Discrete convolution. Let $x_i[j]$ denote the sampled input and $p_{\mathrm{L}} = \lfloor (K-1)/2 \rfloor$ the left padding width. The implemented one-dimensional cross-correlation is

$$y_o[j] = \Delta t \sum_{i=1}^{C_{\mathrm{in}}} \sum_{\ell=0}^{K-1} w_{o,i,\ell}^{(f_s)}\, x_i[j + \ell - p_{\mathrm{L}}], \tag{25}$$

with out-of-range indices handled by SAME padding. Here, $w_{o,i,\ell}^{(f_s)}$ includes the sampled-basis normalization in Eq. (12).

The kernel coordinates follow Eq. (11). Their relation to the integer input offsets is

$$\frac{\ell - p_{\mathrm{L}}}{f_s} = \tau_\ell + \delta, \qquad \delta = \frac{(K-1)/2 - p_{\mathrm{L}}}{f_s}.$$

Thus $\delta = 0$ for odd-length kernels and $\delta = 1/(2f_s)$ for even-length kernels. Relative to Eq. (1), cross-correlation uses a reflected-kernel convention, with the above alignment offset for even-length kernels.

The implementation assumes uniformly sampled sequences with a common sampling rate within each batch.

A.4 Photonic Interpretation and Parameterization Cost

The first slab-propagation stage transforms fixed excitation into latent optical modes; learned complex modulation is followed by a second slab-propagation stage and real-quadrature readout. This correspondence defines the kernel family, which is evaluated by numerical simulation rather than fabricated photonic hardware. Physical realization would additionally require signal encoding, detection, calibration, and control; no measured energy, latency, or chip-area advantage is claimed.

For one convolution layer, amplitude–phase adaptation uses $2NC_{\mathrm{in}}C_{\mathrm{out}}$ trainable scalars plus one shared readout phase. Phase-only and amplitude-only each use $NC_{\mathrm{in}}C_{\mathrm{out}} + 1$ scalars. Geometry and readout gain are fixed. Spline uses $PC_{\mathrm{in}}C_{\mathrm{out}}$ control values for P knots. At $N = 8$ and $P = 15$, these budgets are close but not equal; a low-dimensional kernel parameterization does not imply a reduction in model-wide parameters.

Fixed propagation responses can be precomputed and cached for each evaluation grid. Digital evaluation still incurs kernel synthesis and sampled convolution costs, without an established runtime advantage.

B Experimental Settings

We describe dataset preprocessing, grouped partitions, training, and the configurations used to compare continuous-kernel parameterizations.

B.1 Datasets and Preprocessing

SleepEOG. SleepEOG uses horizontal EOG from Sleep-EDF (Kemp et al., 2000), with five-class sleep staging on 30-second epochs at 100 Hz. Each epoch receives a fourth-order 35 Hz Butterworth low-pass filter applied forward and backward. All nights from one subject remain in the same partition.

ISRUC. ISRUC Subgroup I comprises 100 subjects and uses C3–A2 and C4–A1 EEG at 200 Hz (Khalighi et al., 2016). Signals are low-pass filtered at 35 Hz with an FIR filter and normalized per recording and channel by subtracting the median and dividing by the IQR, floored at 10^{-8}.

Table 2: Dataset configurations. Rates are native/training sampling rates. Sleep labels are W, N1, N2, N3, and REM; ECG labels are normal rhythm, atrial fibrillation, other rhythm, and noise. Preprocessing details are given below.

Dataset	Modality/channels	Hz	Seconds	Classes	Main preprocessing	Split unit
SleepEOG	Horizontal EOG	100	30	5	35 Hz low-pass	Subject
ISRUC I	EEG: C3–A2, C4–A1	200	30	5	35 Hz low-pass; median/IQR	Subject
HMC	EEG: C4–M1, C3–M2	256	30	5	35 Hz low-pass; median/IQR	Subject
CinC 2017	Single-lead ECG	300	8	4	Millivolt calibration; no additional filtering	Recording

Annotation-aligned 30-second epochs use semantic W/N1/N2/N3/REM labels; unknown labels, partial or out-of-bounds epochs, and epochs whose maximum absolute normalized amplitude exceeds 20 are excluded.

HMC. HMC uses C4–M1 and C3–M2 EEG at 256 Hz (Alvarez-Estevez & Rijsman, 2021), with 35 Hz low-pass filtering and per-recording, per-channel median/IQR normalization (IQR floor 10^{-8}). Complete 30-second windows are aligned to scoring annotations; N4 is merged into N3, and unlabeled epochs, terminal partial windows, and epochs with maximum absolute normalized amplitude above 20 are excluded. The dataset contains 151 processed recordings and 129,320 retained epochs.

ECG. PhysioNet/CinC 2017 uses four recording labels: normal rhythm, atrial fibrillation, other rhythm, and noise (Clifford et al., 2017). Waveforms are calibrated to millivolts using the recording gain and baseline, then divided into complete nonoverlapping 8-second frames at 300 Hz; shorter recordings and terminal partial frames are excluded. Frames inherit their recording label, with no additional filtering, normalization, clipping, or artifact rejection. Accuracy is measured per frame, with all frames from a recording grouped in one partition.

B.2 Data Splits and Training

Evaluation uses five-fold stratified group cross-validation with seed 3. Groups are subjects for SleepEOG, ISRUC, and HMC, and recordings for ECG. Within each outer training partition, a group shuffle split reserves 10% of the training groups for validation, using seed $3 + 1000 + k$ for fold $k \in \{1, \ldots, 5\}$. Resampling follows partition assignment, so different-rate views of an example cannot cross training, validation, and test partitions.

Each fold initializes a new model and trains for 20 epochs with batch size 64, AdamW, learning rate 10^{-3}, and unweighted cross-entropy loss, without a learning-rate scheduler or early stopping. The checkpoint with the highest validation accuracy at the training rate is selected, retaining the earliest checkpoint in a tie. Test rates and noise levels do not influence checkpoint selection. Random generators are seeded and deterministic computation is requested; reported deviations measure variation across folds and their trained models, not independent training seeds.

Experiments use PyTorch, NumPy, SciPy, scikit-learn, MNE, and `torchcubicspline`. The LocalNO environment uses PyTorch 2.4.1 with CUDA 12.1 on an NVIDIA A100-SXM4-80GB, neuraloperator 2.0.0, and torch-harmonics 0.7.4.

B.3 Model and Baseline Configurations

Shared backbone and head. WaveKernel, spline, and CKConv use the same NeurOCNN backbone, downstream head, and training protocol. The four convolution layers have widths $C_{\text{in}} \to 64 \to 128 \to 64 \to 32$, kernel duration $D = 0.5$ s, SAME padding, and unit stride, with activation and normalization as in Section 3.4. We retain NeurOCNN's Fourier projection and Transformer classification head: 50 Fourier functions per segment, 5-second segments for sleep data and 2-

second segments for ECG, and two Transformer layers of width 32, feed-forward width 128, and dropout 0.2.

WaveKernel. WaveKernel uses $N = 8$ latent modes with amplitude–phase adaptation, one shared readout phase per layer, fixed unit gain, and one uniformly excited source ($S = 1$). The four layers share the fixed geometry in Appendix A.1 and have separate channel-pair coefficients. Sampled-basis normalization follows Appendix A.3.

Spline. The spline baseline learns $P = 15$ control values per channel pair, initialized with standard-normal draws, and constructs a natural cubic interpolant at uniformly spaced knots on $[-0.5, 0.5]$. Evaluation uses an endpoint-spanning normalized grid, with the rounded length $\max\{\mathrm{round}(Df_s), 1\}$ increased by one when even. Both spline and WaveKernel apply $1/f_s$ quadrature scaling, but their endpoint-spanning and centered-grid conventions can yield slightly different support and alignment; the comparison does not isolate basis shape from these differences.

CKConv. The CKConv kernel MLP has hidden width 32, sine activation with $\omega_0 = 30$, weight dropout 0.1, bias, and no kernel-network normalization. It is evaluated on $[-1, 1]$ with $\max\{\mathrm{round}(Df_s), 1\}$ coordinates. The $1/f_s$ factor scales both convolution and bias, which may affect cross-rate behavior. Results concern this configuration rather than an exhaustive tuning study.

LocalNO and U-Sleep. LocalNO first lifts the input to a hidden width of 128 using a pointwise convolution, followed by four LocalNO layers with 150 Fourier modes and no DISCO layers. The resulting features are projected to width 32 and passed to a Fourier pooling attention head with two attention heads. U-Sleep uses an encoder–decoder of depth 12, with an initial filter count of 5, a complexity factor of 2, kernel size 9, ELU activations, batch normalization, max pooling, and nearest-neighbor upsampling. Comparisons with these baselines are restricted to the three sleep datasets.

B.4 Evaluation Across Sampling Rates

Models train and select checkpoints at 100 Hz (SleepEOG), 200 Hz (ISRUC), 256 Hz (HMC), or 300 Hz (ECG). The common reporting set is

$$\mathcal{F} = \{80, 100, 128, 200, 256, 512, 1024\}\ \mathrm{Hz}.$$

The additional 300 Hz ECG evaluation supplies its training-rate reference. No retraining or fine-tuning occurs at evaluation rates.

SleepEOG and ECG inputs are resampled by linear interpolation; ISRUC and HMC use polyphase resampling with anti-alias filtering. Continuous-kernel models reevaluate their kernels at the target sampling rate. U-Sleep instead resamples the target-rate inputs back to its training grid. Upsampling does not add measured signal information.

Let $A_k(f)$ denote accuracy in percent for fold k at rate f. Table 1 averages $\bar{A}_k = |\mathcal{F}|^{-1} \sum_{f \in \mathcal{F}} A_k(f)$ across folds. The worst-case nonnegative drop is

$$\Delta_k = \max\left\{0, \max_{f \in \mathcal{F}} \left[A_k(f_{\mathrm{train}}) - A_k(f)\right]\right\}. \tag{26}$$

Table 4 reports its mean and population standard deviation over five folds, in pp. Accuracy is example-weighted within folds, with equal weighting of folds in the final summaries.

B.5 Noise Robustness Protocol

The noise experiment uses a separately trained set of five clean-trained fold models. Within each fold, the selected checkpoint and test examples are fixed across all noise types and SNRs, and the corresponding clean performance is used as the reference. The models are trained on clean SleepEOG and evaluated on 30-second test windows at 100 Hz. Noise is added after preprocessing and resampling, with no subsequent denoising or retraining. Four synthetic noise types are generated independently per example and channel: independent standard-normal samples; powerline

Table 3: Rate-wise standard deviation of accuracy over the seven-rate reporting set, summarized as mean ± population standard deviation over five folds (pp; lower is better). Dashes indicate unavailable results.

Method	ISRUC	HMC	SleepEOG	ECG
WaveKernel	0.08 ± 0.05	0.11 ± 0.07	0.35 ± 0.16	0.10 ± 0.05
Spline	0.07 ± 0.03	0.11 ± 0.02	0.26 ± 0.12	0.16 ± 0.05
CKConv	12.10 ± 3.49	7.91 ± 5.54	6.92 ± 2.04	3.60 ± 2.18
LocalNO	0.70 ± 0.49	1.39 ± 1.01	1.87 ± 1.24	–
U-Sleep	0.17 ± 0.10	0.08 ± 0.05	0.70 ± 0.33	–

Table 4: Worst-case accuracy drop relative to the training sampling rate over the seven-rate reporting set, summarized as mean ± population standard deviation over five folds (pp; lower is better). Training rates are 200/256/100/300 Hz for ISRUC/HMC/SleepEOG/ECG. Dashes indicate unavailable results.

Method	ISRUC	HMC	SleepEOG	ECG
WaveKernel	0.11 ± 0.05	0.06 ± 0.03	0.90 ± 0.63	0.16 ± 0.11
Spline	0.17 ± 0.07	0.28 ± 0.07	0.57 ± 0.34	0.29 ± 0.15
CKConv	36.68 ± 9.87	23.44 ± 16.29	21.23 ± 5.92	10.53 ± 6.55
LocalNO	1.79 ± 1.41	3.90 ± 2.74	5.24 ± 3.56	–
U-Sleep	0.39 ± 0.33	0.20 ± 0.16	1.30 ± 0.97	–

sinusoids at 50, 100, and 150 Hz with independent uniform phases, retaining only frequencies at or below Nyquist; a sum of three sinusoids whose frequencies are uniform on $[0.05, 0.5]$ Hz and phases uniform on $[0, 2\pi)$; and Gaussian noise Fourier-masked to $[20, \min(60, 0.95 f_s/2)]$ Hz and inverse-transformed. At 100 Hz, the latter band is 20–47.5 Hz and powerline interference contains only the 50 Hz Nyquist component. This synthetic setup should not be interpreted as a complete model of physiological artifacts.

For signal x and generated noise η, their powers P_x and P_η are temporal means of squared samples, without additional centering. The implemented scaling is

$$x_{\text{noisy}} = x + \eta \sqrt{\frac{P_x}{10^{\text{SNR}/10}(P_\eta + 10^{-8})}}, \qquad \text{SNR} = 10 \log_{10}(P_x / P_{\text{noise}}) \text{ dB}. \tag{27}$$

Scaling is per example and channel. Evaluation uses SNRs of 30, 25, 20, 15, and 10 dB, reporting means and population standard deviations across five folds. Clean-to-noisy drops use the corresponding checkpoint's clean accuracy.

B.6 Ablation Configuration

Ablations use the SleepEOG protocol, train at 100 Hz, and report the 100 Hz test accuracy for five folds. The coefficient sweep uses $N \in \{2, 4, 8, 12, 16\}$, with phase-only, amplitude-only, and amplitude–phase adaptation. Geometry remains at $L_\xi = 12$ μm, $d_2 = 8$ μm, and latent pitch 0.6 μm. Geometry sweeps instead fix $N = 8$ with amplitude–phase coefficients, varying one of $L_\xi \in \{6, 12, 18, 24, 30\}$ μm, $d_2 \in \{4, 8, 12, 16, 20\}$ μm, or latent pitch in $\{0.3, 0.6, 0.9, 1.2, 1.5\}$ μm. Every setting is trained separately under the same training protocol.

The amplitude constraint and learned shared readout phase are described in Appendix A.2. Changing N changes coefficient count, centered-array extent, and the deterministic $1/\sqrt{N}$ basis scale. The sweep therefore does not isolate dimension from every other aspect of the representation. Each geometry sweep trains its reference configuration separately. The box plots use native-rate fold values, not the seven-rate averages in Table 1.

C Supplementary Sampling-Rate Results

Tables 5–8 compare WaveKernel, spline, CKConv, LocalNO, and U-Sleep at individual sampling rates, where results are available. Each entry is the mean ± population standard deviation over five

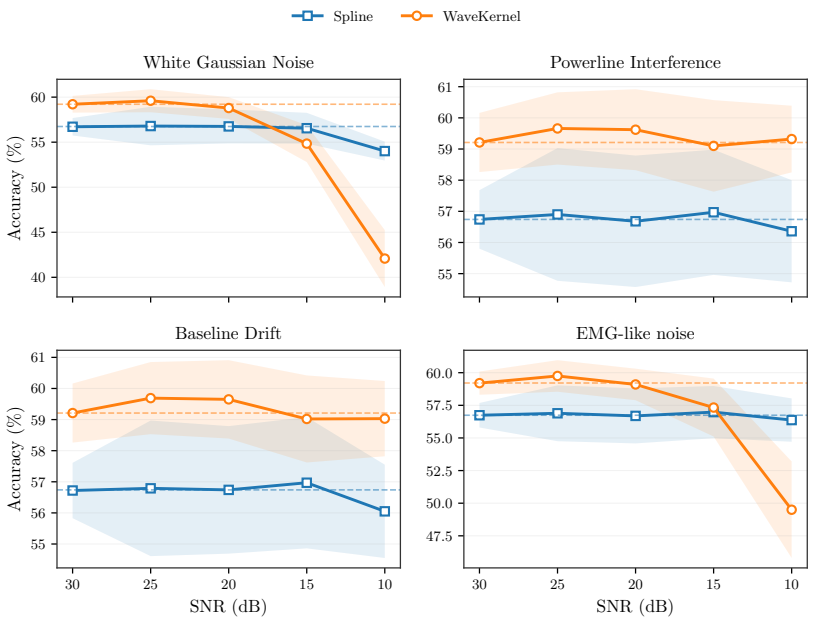


Figure 7: SleepEOG test accuracy at 100 Hz under white Gaussian noise, powerline interference, baseline drift, and EMG-like noise. Curves show five-fold means at SNRs of 30–10 dB; shaded bands indicate one standard deviation across folds. Dashed lines show each method's clean reference from the 30 dB run, not the separate clean reference for every SNR. Noise is applied only at test time; higher accuracy is better.

folds. The training-rate column is marked by a dagger. The seven-rate average in Table 1 weights rates equally within each fold; the additional 300 Hz ECG column is its training-rate reference and is excluded from that average. These are the same trained models evaluated on resampled inputs, not separately trained models for each column.

Table 5: ISRUC test accuracy (%) by sampling rate (Hz); higher is better. † denotes training rate. U-Sleep resamples inputs back to that rate.

Method	80	100	128	200†	256	512	1024
WaveKernel	76.72 ± 1.21	76.72 ± 1.21	76.68 ± 1.21	76.66 ± 1.17	76.63 ± 1.15	76.62 ± 1.15	76.61 ± 1.18
Spline	74.61 ± 1.24	74.67 ± 1.27	74.69 ± 1.28	74.76 ± 1.30	74.76 ± 1.31	74.74 ± 1.35	74.74 ± 1.38
CKConv	40.48 ± 14.76	39.07 ± 9.88	56.94 ± 8.46	72.63 ± 1.80	61.39 ± 7.65	61.96 ± 9.86	62.84 ± 8.37
LocalNO	71.42 ± 1.89	72.14 ± 1.89	72.71 ± 1.69	73.10 ± 1.38	73.07 ± 1.41	72.77 ± 1.31	72.62 ± 1.33
U-Sleep	75.69 ± 1.62	75.96 ± 2.05	75.96 ± 2.06	75.97 ± 2.05	75.97 ± 2.06	75.97 ± 2.06	75.97 ± 2.06

Table 6: HMC test accuracy (%) by sampling rate (Hz); higher is better. † denotes training rate. U-Sleep resamples inputs back to that rate.

Method	80	100	128	200	256†	512	1024
WaveKernel	75.12 ± 1.39	75.08 ± 1.42	75.01 ± 1.44	74.96 ± 1.45	74.92 ± 1.44	74.91 ± 1.45	74.89 ± 1.46
Spline	73.22 ± 0.91	73.32 ± 0.88	73.35 ± 0.86	73.48 ± 0.87	73.50 ± 0.85	73.50 ± 0.87	73.51 ± 0.86
CKConv	51.78 ± 16.16	59.02 ± 6.98	59.80 ± 9.35	67.99 ± 1.94	71.82 ± 1.78	70.20 ± 1.75	70.13 ± 1.62
LocalNO	66.21 ± 1.90	68.37 ± 1.33	69.55 ± 1.03	70.13 ± 1.28	70.11 ± 1.45	70.03 ± 1.63	69.93 ± 1.73
U-Sleep	75.76 ± 1.76	75.91 ± 1.64	75.92 ± 1.66	75.92 ± 1.65	75.92 ± 1.65	75.92 ± 1.65	75.92 ± 1.65

Fold-level consistency. For the seven-rate averages, WaveKernel exceeds spline in all five matched fold indices on ISRUC, HMC, and SleepEOG. The fold-wise differences range from 1.43 to 2.51 pp, 0.36 to 3.42 pp, and 1.04 to 5.79 pp, respectively. On ECG they range from -1.10 to $+0.63$ pp, with four folds favoring spline. These descriptive comparisons do not constitute a significance test.

Table 7: SleepEOG test accuracy (%) by sampling rate (Hz); higher is better. † denotes training rate. U-Sleep resamples inputs back to that rate.

Method	80	100†	128	200	256	512	1024
WaveKernel	59.35 ± 1.48	59.96 ± 1.57	59.44 ± 1.53	59.30 ± 1.36	59.43 ± 1.43	59.31 ± 1.48	59.28 ± 1.49
Spline	56.12 ± 1.41	56.06 ± 1.22	55.82 ± 1.37	55.73 ± 1.38	55.62 ± 1.31	55.54 ± 1.25	55.54 ± 1.26
CKConv	35.57 ± 3.06	52.40 ± 0.95	34.03 ± 4.31	32.77 ± 5.51	32.91 ± 5.63	31.99 ± 5.93	31.61 ± 6.02
LocalNO	54.59 ± 0.92	56.24 ± 0.99	54.79 ± 1.03	53.48 ± 2.27	53.69 ± 2.44	52.38 ± 3.77	51.64 ± 4.56
U-Sleep	52.86 ± 1.42	53.44 ± 1.37	53.57 ± 0.98	53.59 ± 0.86	53.81 ± 0.95	53.72 ± 1.14	53.55 ± 1.18

Table 8: ECG frame-level test accuracy (%) by sampling rate (Hz); higher is better. † denotes the 300 Hz training rate, which is excluded from the seven-rate main-table average.

Method	80	100	128	200	256	300†	512	1024
WaveKernel	73.02 ± 0.17	73.02 ± 0.16	73.10 ± 0.16	73.01 ± 0.20	73.05 ± 0.23	73.05 ± 0.23	73.05 ± 0.22	73.02 ± 0.20
Spline	73.30 ± 0.67	73.38 ± 0.70	73.41 ± 0.60	73.47 ± 0.57	73.47 ± 0.54	73.44 ± 0.53	73.44 ± 0.52	73.36 ± 0.59
CKConv	51.76 ± 6.42	56.59 ± 4.73	59.57 ± 1.93	60.89 ± 3.17	61.61 ± 2.41	61.85 ± 2.35	61.58 ± 2.58	59.82 ± 2.32

D AI USE STATEMENT

We use LLMs to aid or polish writing, support retrieval and discovery, assist research ideation or execution, and draft sections of the paper.