EE603 - Extraordinary Efforts Report by Group 1 On the intersection of Signal Processing and Machine Learning

Chanchal Gupta - 23M1069 - EE1
Juhi Bharti - 24M1144 - EE5
Bala Murugan S - 24M1173 - EE6
Ayush Singh - 25M1086 - EE1
Suraj Adhikari - 25M1089 - EE1
Manas Patil - 25M1094 - EE1

Description of the Menu of Extraordinary Efforts as asked

Study, present and explore further possibilities, on material pertaining to the union of signal processing and machine learning/ deep learning/ neural networks. In particular, this component intends to look at the signal processing aspects of machine learning/ deep learning/ neural networks structures; particularly emphasizing explainability, interpretability and economy. The aim of this component is, also, to assist the instructor in designing a course around this union theme, as a follow up to this course and an independent course, in its own right, in the future. Concurrently, the submission could target an exploration, of how an application area of Digital Signal Processing could benefit, by unifying Digital Signal Processing and Machine Learning. Some groups may choose, for example, not to target applications but focus on theoretical insights pertaining to the functioning of Convolutional Neural Networks in one or two dimensional applications or any other machine learning architectures, as well.

1 Introduction

Machine learning (ML) is one of the most prolific domains today, utilized in a wide array of domains, such as healthcare, finance and obviously, in engineering. Advances in ML include theoretical improvements such as algorithmic optimizations and , practical advancements such as hardware accelerations and code-level abstractions. The appli cations of ML have particularly exploded because of large language models such as ChatGPT, Claude and Gemini, among others. At the fingertips of everyone readily, these LLMs are now capable of reasoning semi-complex arguments and providing aggregated responses of a large body of text. Specialized models, such as convolutional neural networks for image recognition and recurrent neural networks for natural language processing, have improved a great deal over the last decade. However, this success often comes at the cost of interpretability, modern neural networks function as black boxes, making decisions through millions of parameters whose individual contributions remain opaque. This opacity poses significant challenges in safety-critical domains such as medical diagnosis, autonomous systems, and financial decision-making, where understanding why a model makes a particular prediction is as important as the prediction itself.

1.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) form the backbone of modern computer vision systems. A typical CNN architecture consists of alternating convolutional and pooling layers followed by fully connected layers:

$$\mathbf{x}^{(l+1)} = \sigma\left(\mathbf{W}^{(l)}\mathbf{x}^{(l)} + \mathbf{b}^{(l)}\right) \tag{1}$$

where $\mathbf{x}^{(l)}$ represents the activation at layer l, $\mathbf{W}^{(l)}$ denotes learnable convolutional kernels, $*$ represents the convolution operation, and $\sigma(\cdot)$ is a nonlinear activation function (typically ReLU: $\sigma(z) = \max(0, z)$).

Key architectural components:

1. Convolutional layers: Learn local spatial filters that extract hierarchical features. Early layers detect edges and textures; deeper layers capture semantic objects and patterns.

2. Pooling layers: Provide translation invariance and dimensionality reduction through operations like max-pooling or average-pooling:

$$y_{i,j} = \max_{(m,n)\in\mathcal{N}_{i,j}} x_{m,n} \tag{2}$$

   where $\mathcal{N}_{i,j}$ represents a local neighborhood.

3. Nonlinearities: Enable the network to learn complex, non-convex decision boundaries.

4. Fully connected layers: Aggregate spatial features for final classification or regression.

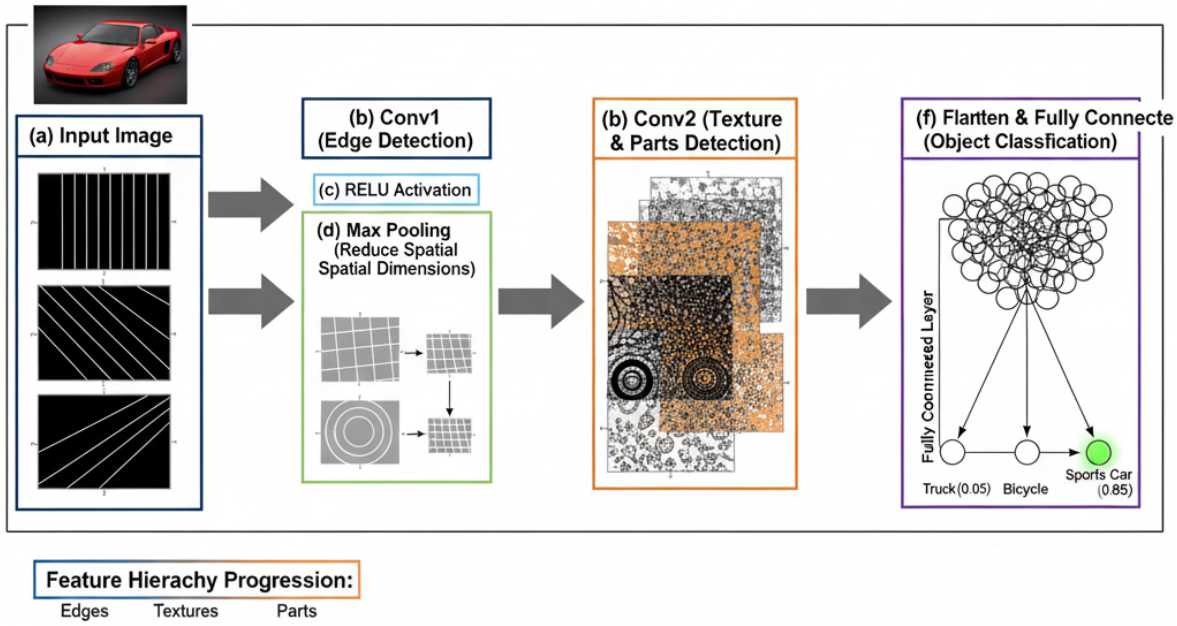

Figure 1: Forward pass through a typical CNN showing: (a) Input image, (b) Conv1 with multiple filters producing feature maps, (c) ReLU activation, (d) Max pooling reducing spatial dimensions, (e) Conv2 with increased filter depth, (f) Flattening and fully connected layers leading to output. At each stage, the figure shows edge detection $\rightarrow$ texture $\rightarrow$ parts $\rightarrow$ objects hierarchy.

The architectural inductive bias of CNNs—local connectivity, weight sharing, and hierarchical composition—makes them naturally suited for grid-structured data. However, this same hierarchical nature makes interpretation challenging: what does a filter in layer 7 actually represent? How do low-level features compose into high-level semantics?

1.2 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) process sequential data by maintaining hidden states that capture temporal dependencies:

$$\mathbf{h}_t = \sigma(\mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{W}_{xh}\mathbf{x}_t + \mathbf{b}_h) \tag{3}$$

$$\mathbf{y}_t = \mathbf{W}_{hy}\mathbf{h}_t + \mathbf{b}_y \tag{4}$$

where $\mathbf{h}_t$ is the hidden state at time t, $\mathbf{x}_t$ is the input, and $\mathbf{y}_t$ is the output. Modern variants like Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs) address the vanishing gradient problem through gating mechanisms:

$$\begin{aligned}
\mathbf{f}_t &= \sigma(\mathbf{W}_f \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_f) \quad \text{(forget gate)} \\
\mathbf{i}_t &= \sigma(\mathbf{W}_i \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_i) \quad \text{(input gate)} \\
\tilde{\mathbf{c}}_t &= \tanh(\mathbf{W}_c \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_c) \quad \text{(candidate values)} \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t \quad \text{(cell state)} \\
\mathbf{o}_t &= \sigma(\mathbf{W}_o \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_o) \quad \text{(output gate)} \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t)
\end{aligned}$$

RNNs excel at temporal pattern recognition in speech, text, and time-series data. However, interpreting what temporal patterns an RNN has learned—which past time steps influence current predictions, what frequencies or periodicities it captures—remains challenging.

1.3 The Interpretability Gap

Despite their success, both CNNs and RNNs suffer from fundamental interpretability challenges:

1. Compositional opacity: How do learned features at different layers compose to form high-level representations?

2. Frequency content: What spectral characteristics do filters capture? How does frequency information flow through depth?

3. Robustness: Why are networks vulnerable to imperceptible adversarial perturbations?

4. Generalization: What properties of learned representations enable transfer to new domains?

These questions motivate the integration of signal processing principles—a field with centuries of interpretable, physically-grounded analysis tools—into deep learning architectures.

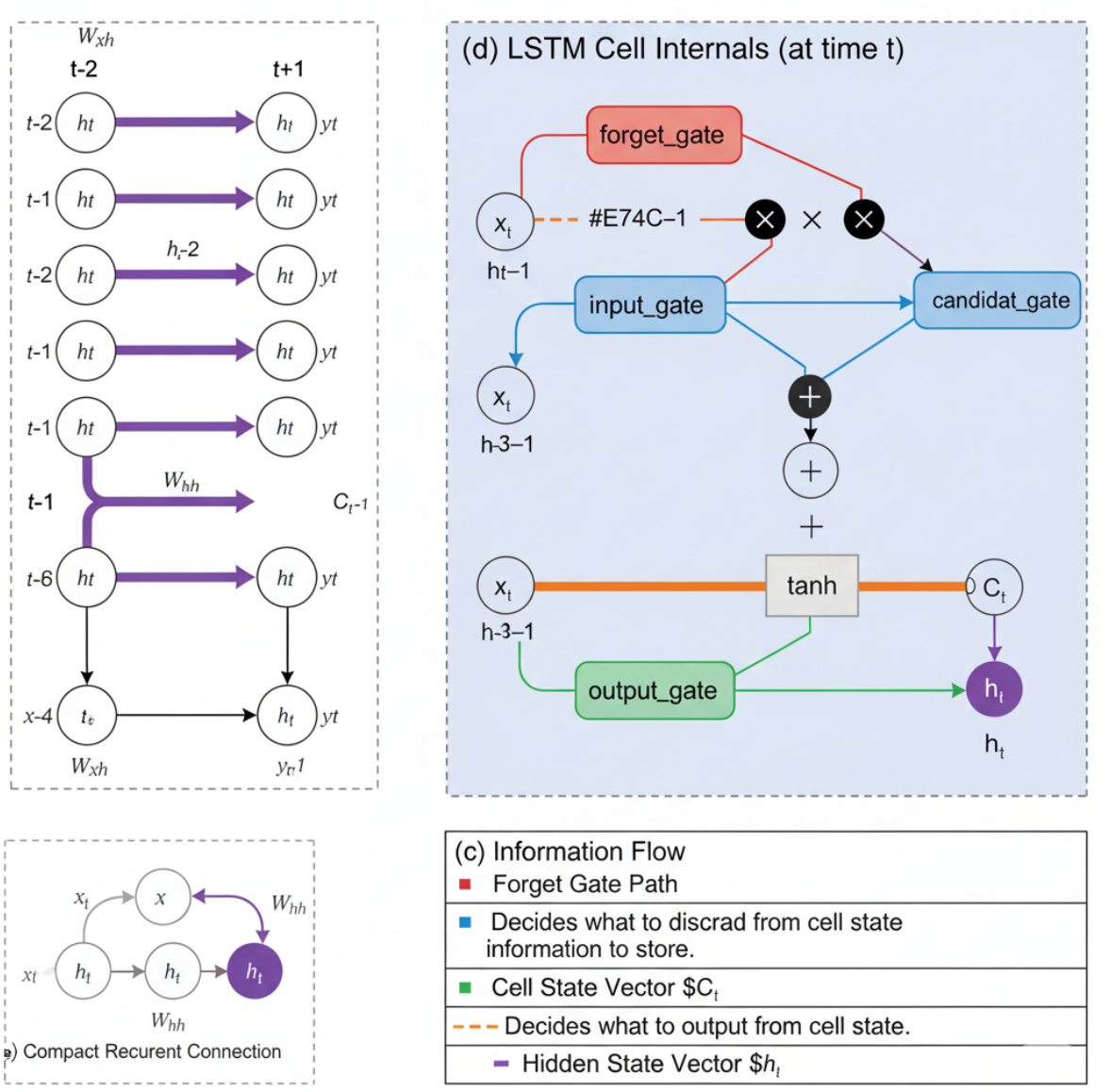

Figure 2: Temporal unrolling of an RNN showing: (a) Compact recurrent connection representation, (b) Unrolled network across time steps t-2, t-1, t, t+1, (c) Information flow through hidden states, (d) LSTM cell internals showing forget gate, input gate, cell state, and output gate with their respective activations.

1.4 The SP-ML Bridge

Signal processing provides natural interpretability coordinates: frequency, scale, time, and phase. Unlike abstract learned features, these quantities have direct physical meaning. A bandpass filter centered at 40 Hz is immediately interpretable to a neuroscientist studying brain oscillations. A wavelet coefficient at scale 4 corresponds to features of a specific spatial size.

Figs. 3, 4 and 5 are side-by-side comparison that show where SP fits into neural architectures: 3 shows a traditional CNN with learned conv filters that lack clear interpretation. 4 portrays a SP-informed CNN with interpretable front-end (e.g., Gabor filterbank, wavelet scattering) with parametric filters with explicit frequency/orientation i.e. learnable back-end. 5 gives a hybrid architecture showing smooth transition from structured SP transforms to fully learnable layers [1, 2]. This report explores how signal processing methods can restore interpretability to neural networks while maintaining, and sometimes enhancing, predictive performance. We survey current approaches, identify critical gaps in our understanding, and chart paths toward truly interpretable AI systems grounded in physical principles.

2 Current Efforts in Explaining CNNs

The deep learning community has developed numerous interpretability methods, broadly categorized into post-hoc explanation techniques and inherently interpretable architectures. While significant progress has been made, most approaches remain fundamentally correlational rather than causal, and lack grounding in physical principles.

2.1 Gradient-Based Attribution Methods

Gradient-based methods explain predictions by computing input attributions—which input features most influence the output.

1. Vanilla Gradients: The simplest approach computes the gradient of the output with respect to the input:

$$\text{Attribution}_i = \frac{\partial y_c}{\partial x_i} \tag{5}$$

   where y_c is the score for class c and x_i is input feature i. However, gradients can be noisy and saturate in regions of poor curvature.

2. Integrated Gradients: Addresses gradient saturation by integrating gradients along a path from a baseline x' to the input x:

$$\text{IGrad}_i(x) = (x_i - x_i') \int_{\alpha=0}^{1} \frac{\partial F(x' + \alpha(x - x'))}{\partial x_i} d\alpha \tag{6}$$

   This satisfies desirable axioms like sensitivity (if input changes but output doesn't, attribution is zero) and implementation invariance.

The limitations of gradient-based attribution methods are:

- Attribution maps often highlight perceptually irrelevant high-frequency noise

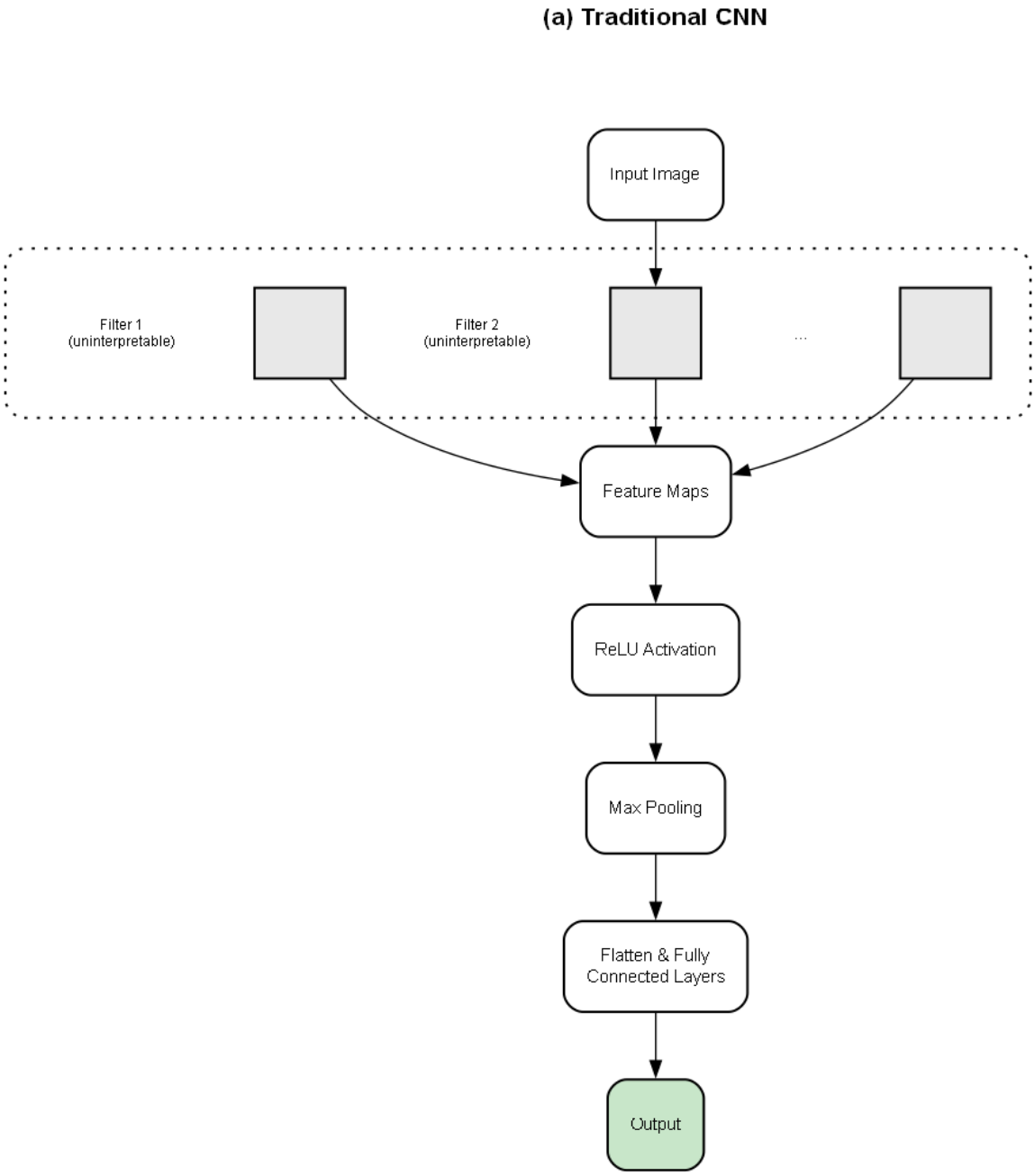

Figure 3: Traditional CNN

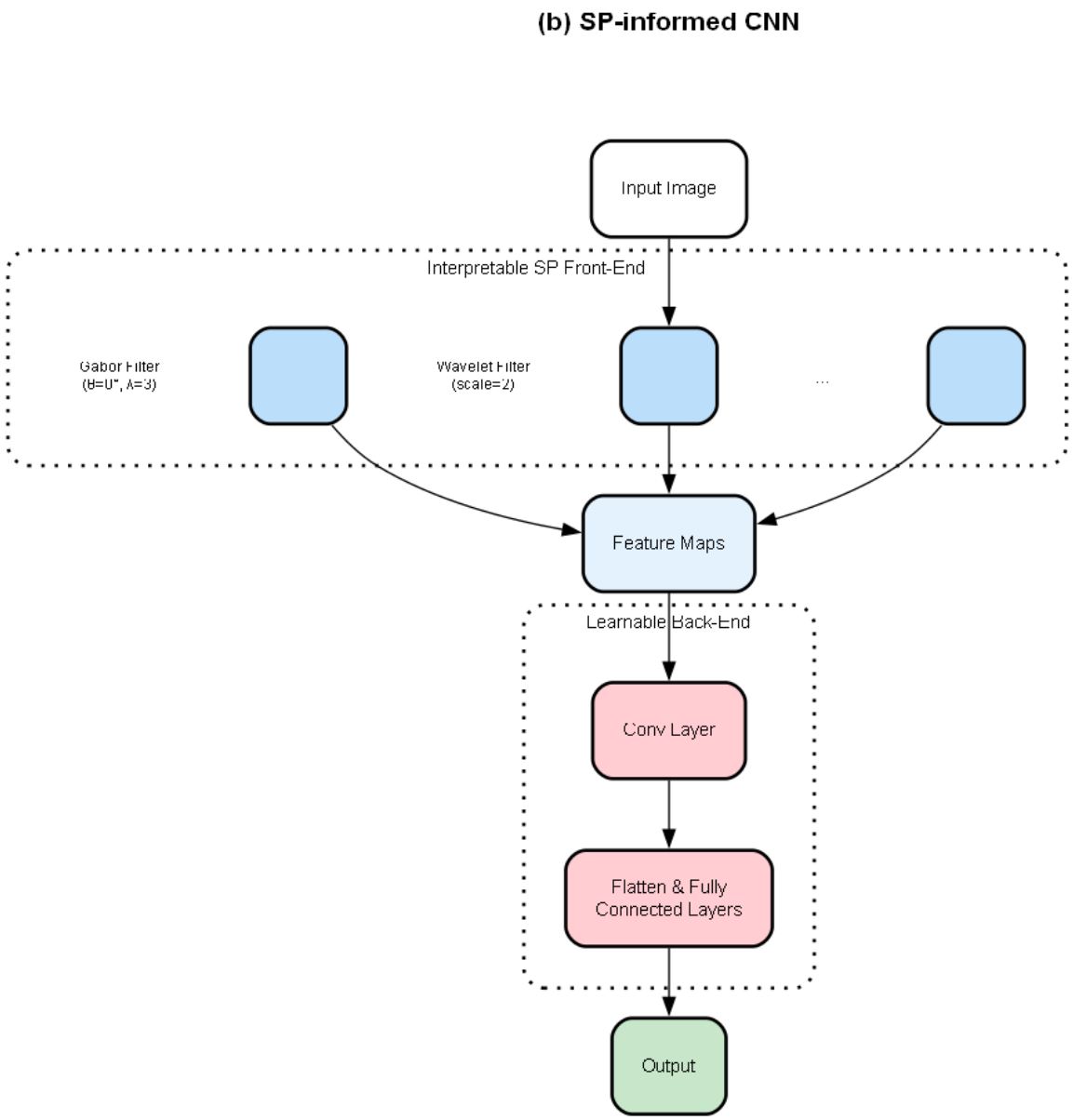

Figure 4: Signal processing informed CNN

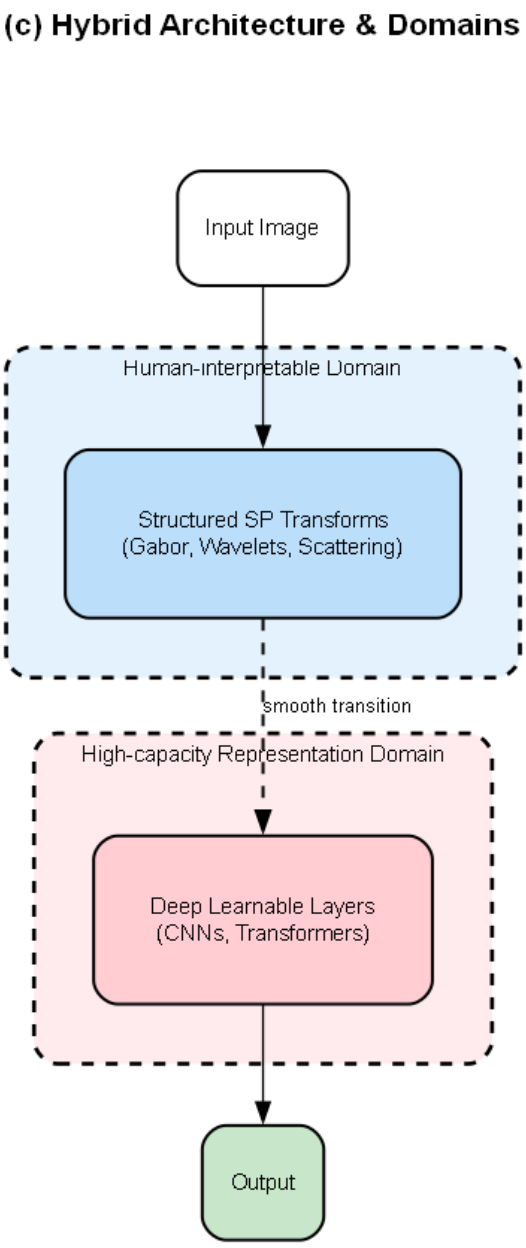

Figure 5: Hybrid Architecture with SP informed and Traditional CNN heads

- No clear connection to frequency-domain structure
- Purely correlational: high attribution doesn't imply causal necessity

2.2 Activation Maximization and Feature Visualization

Feature visualization generates synthetic inputs that maximally activate specific neurons:

$$x = \arg\max_x \left(a_i^{(l)}(x) - \lambda R(x) \right) \tag{7}$$

where $a_i^{(l)}$ is the activation of neuron i in layer l, and $R(x)$ is a regularizer promoting natural images (total variation, frequency penalties, etc.). Early CNN layers learn Gabor-like edge detectors and oriented patterns. Deeper layers detect increasingly complex textures and object parts. However, these visualizations often contain high-frequency artifacts and don't explain how representations compose across layers.

2.3 Class Activation Mapping (CAM) and Variants

CAM-based methods (CAM, Grad-CAM, Grad-CAM++) generate spatial heatmaps indicating which image regions influenced predictions:

$$L_c^{\text{Grad-CAM}} = \text{ReLU}\left(\sum_k \alpha_k^c A^k \right) \tag{8}$$

where A^k are feature maps from a convolutional layer and $\alpha_k^c = \frac{1}{Z} \sum_{i,j} \frac{\partial y^c}{\partial A_{ij}^k}$ are gradient-weighted importance scores. The strengths of CAM are that it provides spatial localization and layer-wise analysis capability. The limitations, however, are:

- Limited to spatial domains (not applicable to time-series or frequency analysis)
- Coarse resolution due to pooling and striding
- No explicit frequency decomposition

2.4 Network Dissection and Concept-Based Explanations

Network dissection quantifies the interpretability of individual units by measuring their alignment with human-understandable concepts (objects, textures, colors).

$$\text{IoU}(u, c) = \frac{|M_u \cap S_c|}{|M_u \cup S_c|} \tag{9}$$

where M_u is the binary mask of high-activation regions for unit u, and S_c is a segmentation mask for concept c. TCAV (Testing with Concept Activation Vectors) extends this by measuring how much a concept influences predictions through directional derivatives in activation space. Limitations of network dissection are:

- Requires labeled concept datasets
- Concepts are predefined rather than discovered
- No inherent connection to signal structure (frequency, scale, phase)

2.5 Attention Mechanisms

Attention mechanisms in transformers and attention-augmented CNNs learn weighted combinations of features.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{10}$$

While attention weights provide some interpretability (which tokens or spatial regions are attended to), they don't explain what features are being extracted or how they relate to physical signal properties.

2.6 The Signal Processing Perspective

Current interpretability methods suffer from several fundamental gaps when viewed through a signal processing lens:

1. Most methods operate purely in the spatial/temporal domain, ignoring that natural signals have strong spectral structure.

2. Attributions show what the network uses i.e. correlation, not what is causally necessary for predictions.

3. Unlike SP transforms (e.g., scattering networks with Lipschitz stability), learned representations lack provable robustness bounds.

4. CNNs implicitly perform multi-scale analysis through hierarchical layers, but this isn't made explicit or interpretable.

5. In domains with strong physical priors (biomedical signals with known frequency bands, mechanical vibrations with harmonic structure), learned filters may not align with interpretable coordinates.

These limitations motivate the integration of signal processing principles into neural network design, not as post-hoc explanation tools, but as fundamental architectural components that provide interpretability by construction.

3 Research Gaps

Recent work at the intersection of signal processing and machine learning has shown promising results. Spectral-domain constraints improve generalization, parametric filterbanks enable domain-aligned features, and scattering networks provide provable stability. However, significant theoretical and practical gaps remain. This section outlines six critical research challenges identified in the convergence of signal processing and deep learning interpretability.

3.1 Theoretical unification

Empirical studies demonstrate that frequency-domain constraints improve generalization. Networks with spectral parameterizations often exhibit better out-of-distribution robustness, and the "frequency principle" (DNNs learn low-frequency components before high-frequency ones) correlates with successful generalization [4, 5]. However, we lack rigorous theoretical frameworks connecting spectral properties to learning-theoretic guarantees. Some specific open questions are:

1. How do spectral constraints (bandwidth limits, frequency concentration) affect PAC bounds or Rademacher complexity? If a network is constrained to learn functions with bounded spectral content, does this provably reduce the sample complexity?

2. The observed low-to-high frequency learning progression during training remains descriptive rather than prescriptive. What properties of gradient descent combined with activation nonlinearities cause this behavior? Can we prove convergence rates as a function of frequency content?

3. Complex-valued networks and frequency-domain architectures show empirical robustness to adversarial perturbations. What formal connections exist between frequency-domain properties (e.g., bandlimiting, spectral smoothness) and Lipschitz bounds on perturbation sensitivity?

Without theoretical foundations, one cannot predict if frequency-domain constraints will help or hurt. Formal theory would enable principled architecture design, specification of appropriate frequency inductive biases for different domains, and provable generalization certificates.

3.2 Deeper layer mechanisms

Interpretable SP-informed networks like SincNet, wavelet scattering transforms, and Time-Frequency Networks successfully explain early layers through explicit frequency decompositions. We can visualize learned bandpass center frequencies, inspect scattering coefficients, and align features with domain knowledge. However, deeper layers remain opaque—we lack tools to track frequency evolution through nonlinear activations, pooling, and hierarchical composition. Specific issues:

1. How do frequency representations transform across network depth? Do higher layers perform frequency mixing, demodulation, or create new spectral components?

2. Nonlinear activation effects: ReLU, GELU, and other nonlinearities introduce harmonics in the frequency domain. Given a frequency decomposition at layer l, how does $\sigma(\mathbf{W}\mathbf{x}^{(l)})$ affect the spectral content at layer $l+1$?

3. Abstract feature interpretation: High-level semantic features (e.g., "objectness," "animacy") emerge in deep layers. Can these be characterized in frequency/scale coordinates, or do they fundamentally transcend signal processing descriptions?

Scattering networks provide one mathematically controlled model of multi-scale hierarchical features, but learned deep networks exhibit richer, less understood behavior. Spectral bias analyses focus on global function classes or early training dynamics, not per-layer frequency transfer. We note, that mechanistic interpretability requires understanding the full forward pass. If we can only interpret the first 1-2 layers, we cannot explain how networks compose low-level features into high-level decisions.

3.3 Quantitative interpretability metrics

Signal processing provides natural measurable quantities—spectral entropy, band energy, coherence, filter parsimony, but there is no validated, standardized set of metrics quantifying model interpretability. Current approaches rely on qualitative visualizations (spectrogram overlays, filter plots) or ad-hoc ablation tests. One might say, we need metrics such as:

1. Spectral parsimony metrics: How many effective frequency bands does the model use? Can we define a spectral "degrees of freedom" analogous to model parameter count?

2. Domain alignment scores: For applications with known frequency structure (EEG alpha/beta/gamma bands, mechanical harmonics), how well do learned features align with domain-relevant bands?

3. Stability and robustness: Can we compute Lipschitz-like bounds on frequency-domain perturbations? For example, if we perturb energy in band $[f_1, f_2]$ by ϵ, how much does the output change?

4. Causal sufficiency: Given an attribution to frequency band B, can we verify that B is actually necessary for the prediction (not merely correlated)?

Currently, we can see the following open avenues of research:

- No benchmark suite tying SP metrics to human judgment, model robustness, and task performance

- Individual measures exist (scattering stability, spectral entropy) but no consolidated interpretability index

- No standardized protocols for comparing interpretability across different architectures (scattering vs. parametric filterbanks vs. learned CNNs)

Without quantitative metrics, interpretability remains subjective. We cannot rigorously compare methods, track interpretability during training, or optimize for tradeoffs in interpretability and accuracy.

3.4 Hybrid transforms

Fully handcrafted transforms, such as Fourier, wavelets, are interpretable but inflexible. Fully learned CNNs are flexible but opaque. Hybrid approaches like SincNet (learnable cutoff frequencies with fixed sinc filters) and learnable wavelet networks strike a middle ground, but introduce complex optimization and scalability challenges. The prime technical issues are:

1. SP parameters (frequencies, phases, scale factors) introduce multi-scale gradients. Frequency parameters require different learning rates than amplitude parameters. Non-convex loss surfaces with poor conditioning.

2. Parametric filterbanks work well as front-ends for small 1D signals (audio, EEG) but haven't scaled to large 2D vision tasks (ImageNet) or modern transformers.

3. Fully learnable deep wavelet transforms incur $O(NJ \log N)$ cost where J is the number of scales, prohibitive for high-resolution images or long sequences.

4. The theoretical understanding of such transforms is limited. What is the representational capacity of hybrid parameterizations? Are there function classes that require full flexibility, or can most practical tasks be solved with structured parameterizations?

Hybrid approaches offer the best of both worlds, interpretability with flexibility, but only if we can solve the optimization and scaling challenges. Success here would enable interpretable models for large-scale, production applications.

3.5 Causality in interpretability

Current SP-based interpretability is correlational, we observe which frequency bands co-occur with predictions, but don't test causal necessity. Signal processing provides natural intervention tools (bandstop filtering, phase scrambling, time masking), but lacks a formal causal framework for frequency-domain counterfactuals.

1. Given correlations between spectral components and outputs, can we discover causal graphs? Which frequency bands causally precede others in the network's reasoning?

2. If we remove energy from band B, what is the minimal realistic perturbation to restore the original prediction? How do we ensure counterfactuals remain in-distribution? [12]

3. Frequency bands are inherently correlated (nearby frequencies, harmonics). Standard interventions may violate the data distribution. How do we control for confounding in frequency-space causal inference?

4. What are appropriate hypothesis tests for frequency-domain causal claims? How do we handle multiple testing corrections across many frequency bands?

Counterfactual explanations exist in general ML (Wachter et al.), and attribution methods exist (Shapley values), but frequency-targeted causal frameworks are underdeveloped. We lack standardized experimental protocols and estimation tools for causal effects of band interventions. Understanding which frequency components are causally necessary—not just correlated—is essential for trustworthy AI. In medical diagnosis, we need to know which EEG bands cause the model to predict a seizure, enabling clinicians to verify the reasoning against physiological knowledge.

3.6 Scalability

Most SP-based interpretable architectures demonstrate success on small, domain-specific datasets (biomedical signals with hundreds to thousands of samples, specialized acoustic tasks). However, they face computational and architectural barriers when scaling to modern foundation models trained on millions of images or billions of tokens. The bottlenecks in scaling are:

1. Computational cost: Wavelet transforms, scattering networks, and multi-resolution analyses have higher computational complexity than simple convolutions. The $O(NJ \log N)$ cost of multi-scale transforms becomes prohibitive at ImageNet scale.

2. Hardware acceleration: Specialized SP operations (parameterized sinc filters, complex-valued convolutions, wavelet packet transforms) lack optimized GPU/TPU kernels. Standard CNNs benefit from decades of hardware co-design.

3. Architectural integration: Transformers have become the dominant architecture for large-scale learning. How do we integrate interpretable frequency-domain components into attention mechanisms? Can we design frequency-aware self-attention?

4. Transfer learning: SP-based models often train from scratch. Can we incorporate frequency-domain interpretability into pretrained foundation models, or must we redesign from the ground up?

If interpretable SP-informed models only work on small problems, they remain niche tools. Scaling to ImageNet, foundation models, and multimodal systems is essential for real-world impact.

4 Concepts in SP here

To understand how signal processing informs interpretable neural networks, we must establish the foundational SP concepts and transforms that appear throughout this intersection. This section provides intuitive explanations alongside formal definitions, focusing on the tools most relevant to interpretable ML.

4.1 Fourier Analysis and Frequency Representation

The Fourier transform is the cornerstone of signal processing, decomposing signals into sinusoidal components: Continuous Fourier Transform:

$$X(f) = \int_{-\infty}^{\infty} x(t) e^{-j2\pi ft} dt \tag{11}$$

Discrete Fourier Transform (DFT):

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j2\pi kn/N} \tag{12}$$

Every signal can be represented as a weighted sum of sinusoids at different frequencies. The magnitude $|X(f)|$ tells us how much energy exists at frequency f, while the phase $\angle X(f)$ encodes temporal alignment. Key properties relevant to ML are [8, 9]:

1. Convolution theorem: Convolution in time/space domain is multiplication in frequency domain:

$$y(t) = h(t)x(t) \quad \Leftrightarrow \quad Y(f) = H(f) \cdot X(f) \tag{13}$$

   This directly connects CNNs (which perform convolutions) to frequency filtering.

2. Parseval's theorem: Energy is conserved across domains:

$$\int |x(t)|^2 dt = \int |X(f)|^2 df \tag{14}$$

3. Frequency localization: Sharp features in time require high frequencies; smooth features concentrate in low frequencies.

4.2 Short-Time Fourier Transform (STFT) and Spectrograms

The Fourier transform sacrifices temporal localization for frequency resolution. The STFT addresses this by analyzing short overlapping windows:

$$X(t, f) = \int_{-\infty}^{\infty} x(\tau) w(\tau - t) e^{-j2\pi f \tau} d\tau \tag{15}$$

where $w(t)$ is a window function (Hamming, Hann, Gaussian) localizing the analysis. The magnitude-squared STFT provides a time-frequency energy representation, called a spectrogram:

$$S(t, f) = |X(t, f)|^2 \tag{16}$$

The time-frequency uncertainty principle limits simultaneous time and frequency resolution:

$$\Delta t \cdot \Delta f \geq \frac{1}{4\pi} \tag{17}$$

Narrow time windows (good temporal resolution) yield poor frequency resolution, and vice versa. The connections to machine learning are:

- Spectrograms are commonly used as input to CNNs for audio classification
- STFT-based front-ends in interpretable networks provide explicit time-frequency representations
- Learned STFT parameters (window size, hop length) can adapt to task-specific time-frequency tradeoffs

4.3 Filterbanks and Multi-Resolution Analysis

A filterbank is a collection of bandpass filters partitioning the frequency axis:

$$y_k(t) = h_k(t)x(t), k \in I \tag{18}$$

where I is an index set and $h_k(t)$ is the impulse response of k^{th} filter. The different types of filterbanks are:

1. Uniform filterbanks: Equal-width frequency bands (e.g., DFT bins)
2. Logarithmic filterbanks: Constant-Q filters mimicking human auditory perception:
$$Q = \frac{f_c}{\Delta f} = \text{const} \tag{19}$$
where f_c is center frequency and Δf is bandwidth. Low frequencies have narrow bands; high frequencies have wide bands.
3. Mel-scale filterbanks: Perceptually-motivated frequency spacing:
$$\text{Mel}(f) = 2595 \log_{10}\left(1 + \frac{f}{700}\right) \tag{20}$$

Analysis-synthesis filterbanks satisfy:

$$x(t) = \sum_k g_k(t)(h_k(t)x(t)) \tag{21}$$

where g_k are synthesis filters, where filters are created from a desired response function. Analysis filters break a signal into sub-bands, while synthesis filters reconstruct it. This perfect reconstruction ensures no information loss, crucial for interpretability. The connections to machine learning are:

- SincNet learns cutoff frequencies for parametric bandpass filters [7, 13]
- Mel-filterbanks are standard front-ends for speech recognition CNNs
- Learnable filterbanks replace the first convolutional layer with interpretable frequency-selective filters

4.4 Wavelet Transform and Scattering Networks

Wavelets provide a multi-resolution analysis through dilations and translations of a mother wavelet $\psi(t)$. [1, 3]
A continuous wavelet transform (CWT):

$$W(a,b) = \frac{1}{\sqrt{|a|}} \int_{-\infty}^{\infty} x(t)\psi^{\left(\frac{t-b}{a}\right)}dt$$

where a is the scale parameter (inversely related to frequency) and b is the translation parameter (time localization). Key properties of wavelets are:

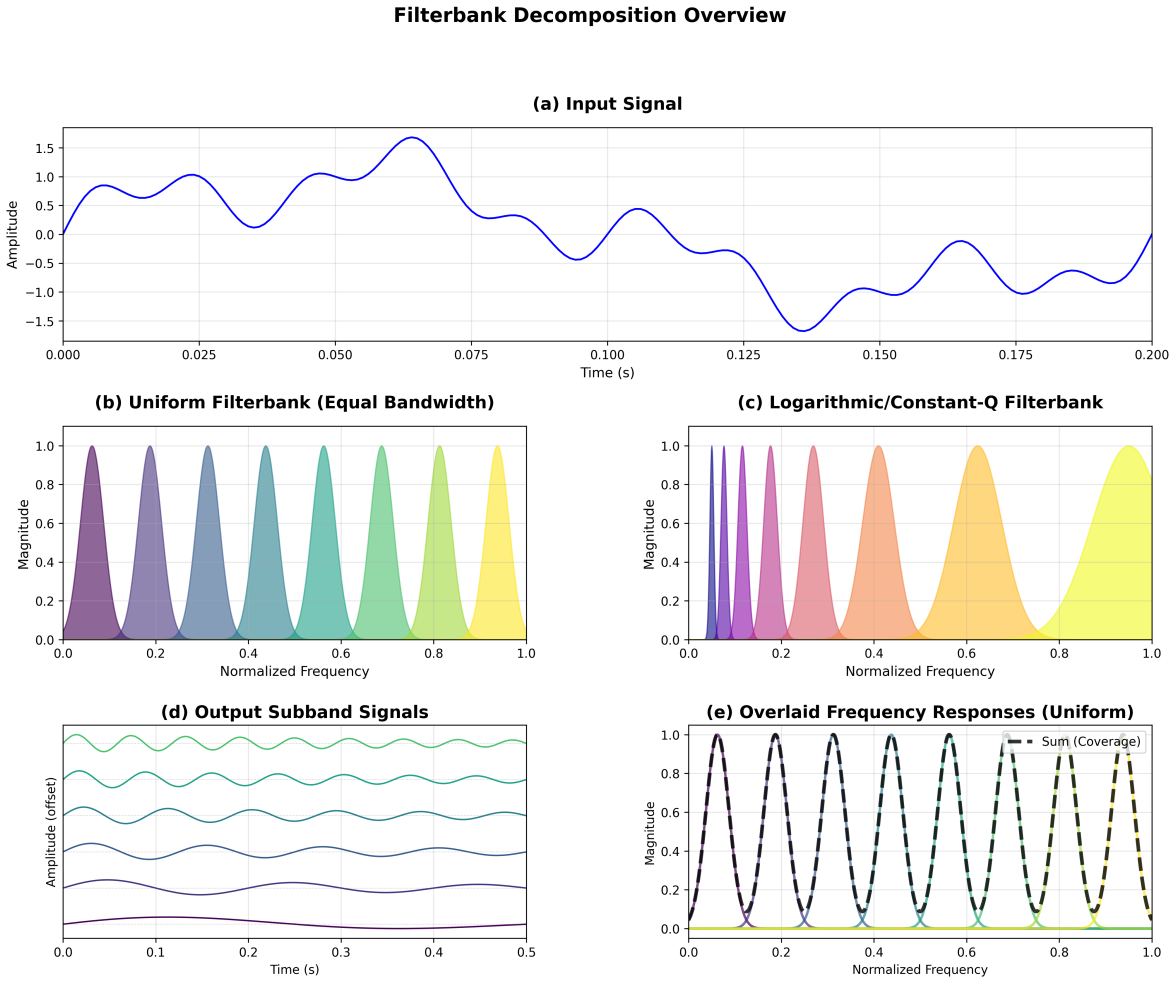

Figure 6: Filterbank Decomposition

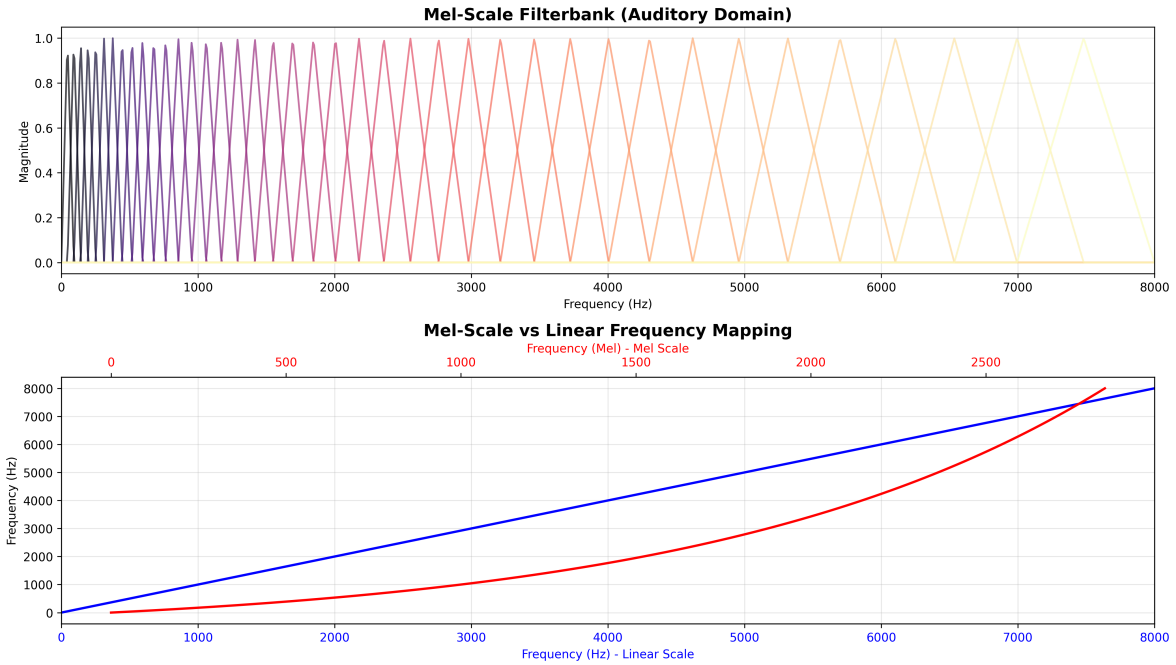

Figure 7: Mel scale decomposition of the filterbanks

1. Coarse scales capture low-frequency trends; fine scales capture high-frequency details

2. Unlike Fourier basis functions (infinite sinusoids), wavelets are localized in both time and frequency

3. Different wavelet families (Morlet, Mexican hat, Daubechies) adapt to different signal characteristics

The scattering transform is a multi-layer cascade of wavelet decompositions with modulus nonlinearities:

$$\begin{aligned} S_0 x &= x\phi \\ S_1 x(t, \lambda_1) &= |x\psi_{\lambda_1}|\phi \\ S_2 x(t, \lambda_1, \lambda_2) &= ||x\psi_{\lambda_1}|\psi_{\lambda_2}|\phi \end{aligned} \tag{22}$$

where ψ_λ are wavelets at scale λ, ϕ is a low-pass averaging filter, and $|\cdot|$ is the modulus operator.

Scattering matters for interpretability because:

1. Provable stability: Scattering coefficients are Lipschitz continuous to deformations:

$$\|Sx - Sx'\| \leq C\|x - x'\| \tag{23}$$

2. Translation invariance: The averaging ϕ provides approximate shift invariance

3. Explicit multi-scale hierarchy: Each layer corresponds to a defined scale, unlike learned CNN layers

4. Sparse representation: Natural signals have sparse scattering coefficients

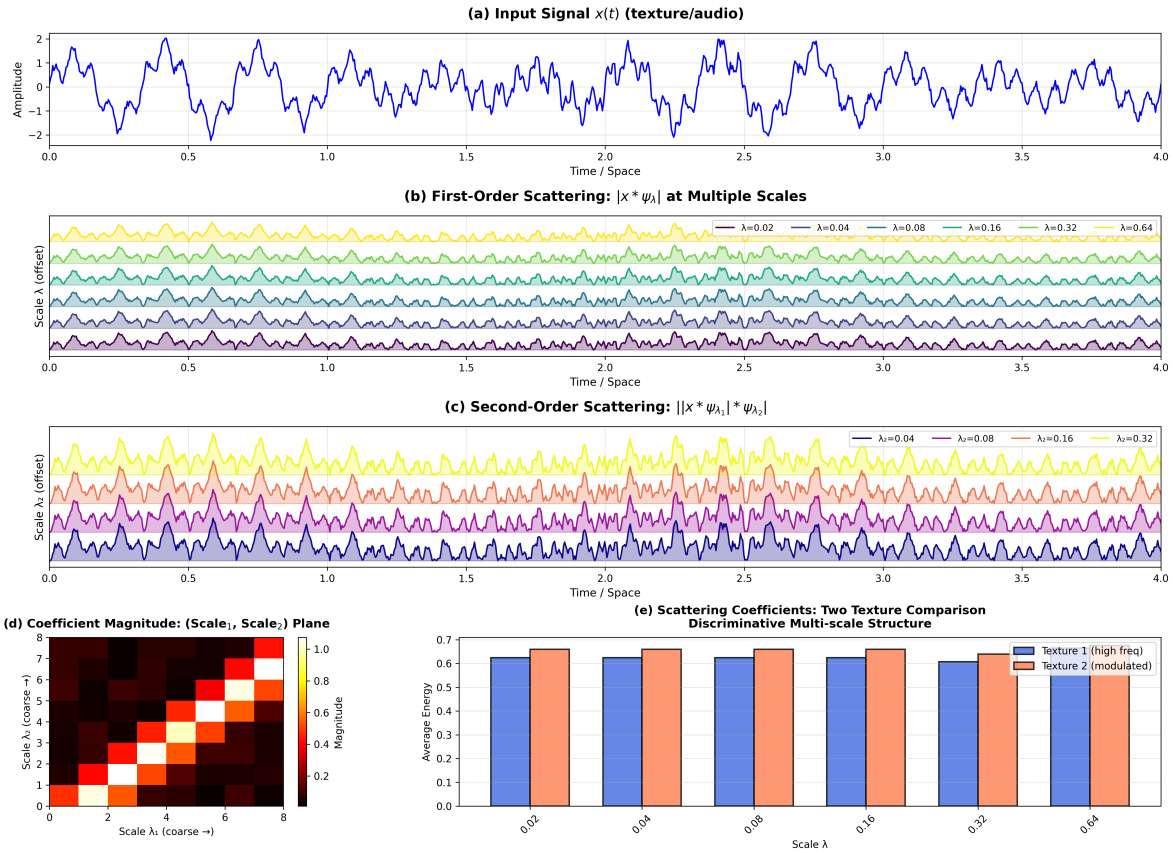

Figure 8: Wavelet Scattering Decomposition

4.5 Gabor Filters and 2D Extensions

Gabor filters are sinusoids modulated by Gaussian envelopes, providing joint time-frequency or space-spatial frequency localization. 1D Gabor function:

$$\psi(t) = \frac{1}{\sqrt{2\pi}\sigma} e^{-t^2/(2\sigma^2)} e^{j2\pi f_0 t} \tag{24}$$

2D Gabor function (for images):

$$g(x, y; \lambda, \theta, \psi, \sigma, \gamma) = \exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right) \cos\left(2\pi \frac{x'}{\lambda} + \psi\right) \tag{25}$$

where:
- $x' = x\cos\theta + y\sin\theta$ and $y' = -x\sin\theta + y\cos\theta$ (rotation)
- λ is the wavelength (spatial frequency)
- θ is the orientation
- ψ is the phase offset
- σ is the Gaussian envelope width
- γ is the spatial aspect ratio

The practical motivation for Gabor filters is that they closely model receptive fields of simple cells in the primary visual cortex (V1). Early CNN filters often converge to Gabor-like patterns, suggesting this is a natural image prior. Since each filter is characterized by interpretable parameters (frequency, orientation, phase), Gabor filters are quite useful for interpretability. Unlike discrete Fourier bins, Gabor filters provide smooth, localized frequency responses. Filters at any orientation can be synthesized from a basis set, and thus, their configuration is steerable. The applications to machine learning are:

- Gabor filterbanks as interpretable CNN front-ends
- Initialization of first-layer CNN filters with Gabor atoms
- Texture analysis and classification

5 Practical Benefits of SP in ML interpretability

Having established the SP toolkit, we now address the central question: Why integrate signal processing into neural network design for interpretability? This section articulates the practical advantages of SP-informed architectures, grounded in domain knowledge, physical constraints, and human understanding.

5.1 Natural Interpretability Coordinates: Frequency, Scale, and Time

Signal processing provides human-interpretable coordinates that directly map to physical phenomena. Frequency domain:

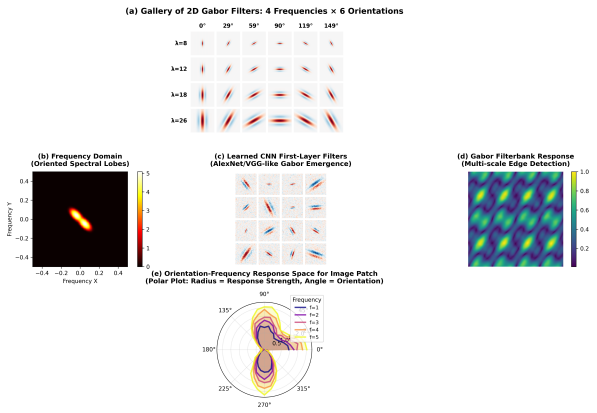

Figure 9: 2D Gabor Filters and Orientation Selectivity

- In neuroscience: 8-13 Hz = alpha rhythm (relaxed wakefulness), 13-30 Hz = beta (active thinking)
- In mechanical diagnosis: Fundamental frequency + harmonics reveal fault types
- In audio: Pitch = fundamental frequency, timbre = harmonic structure

When a SincNet filter learns a passband at 10 Hz for EEG motor imagery classification, neuroscientists immediately understand this corresponds to the mu rhythm over sensorimotor cortex—a known correlate of motor planning. This level of interpretability is impossible with generic learned convolutional filters.

Scale/time localization:

- Wavelet coefficients at scale 2^j correspond to features of approximate size 2^j samples
- Scattering coefficients explicitly encode multi-scale texture descriptors
- Time-frequency atoms localize both when and at what frequency events occur

A neuron in layer 5 of a CNN has no clear physical interpretation. Its receptive field spans some spatial extent, but we cannot say "this neuron detects 40 Hz oscillations" or "this feature corresponds to structures at 5mm scale."

5.2 Domain Knowledge Integration

Many application domains have decades or centuries of accumulated knowledge about relevant signal structures.

1. Medical signals:
    - ECG: P-wave, QRS complex, T-wave have defined frequency content
    - EEG: Clinical frequency bands (delta, theta, alpha, beta, gamma) correlate with brain states
    - Ultrasound: Tissue properties relate to scattering at specific frequencies

2. Mechanical systems:
    - Rotating machinery: Faults create harmonics at multiples of shaft speed
    - Vibration analysis: Bearing defects produce characteristic frequency patterns
    - Acoustic emissions: Material failure modes have spectral signatures
3. Speech and audio:
    - Phonemes occupy specific formant frequency ranges
    - Music: Harmonic relationships define melody and timbre
    - Environmental sounds: Identified by spectral envelopes

Rather than learning from scratch, SP-informed networks can initialize or constrain filters to relevant frequency bands. This reduces sample complexity, improves out-of-distribution generalization, and enables expert validation of learned features. Example: In bearing fault diagnosis, domain experts know that outer race faults produce sidebands at bearing pass frequency. A network with explicit frequency parameterization can be inspected: "Does the model attend to the expected fault frequencies?" If not, either the model is wrong, or the domain theory needs revision—both valuable insights.

5.3 Sample Efficiency and Low-Data Regimes

SP-based inductive biases dramatically improve learning with limited data. Constraining the hypothesis space to functions with limited spectral complexity reduces the VC dimension and improves generalization bounds (though, as noted in Gap 1, formal theory is incomplete). Empirical evidence for this is:

1. Scattering networks: Achieve competitive accuracy with ¡1000 training examples on texture classification, where vanilla CNNs overfit
2. SincNet: Reduces parameters in first convolutional layer by $50\times$ (learning cutoff frequencies rather than full kernels), enabling training on small speaker recognition datasets
3. Medical imaging: Wavelet-based networks succeed on small clinical datasets (hundreds of patients) where standard CNNs require thousands

Many critical applications—rare disease diagnosis, new product fault detection, endangered species monitoring—inherently have small datasets. SP priors enable trustworthy learning in these regimes.

5.4 Robustness and Stability Guarantees

Signal processing provides provable stability to perturbations. Scattering networks are Lipschitz continuous to diffeomorphic deformations:

$$\|Sx - S\tau x\| \leq C\|\tau - \mathrm{id}\| \tag{26}$$

where τ is a smooth deformation and id is the identity. This guarantees that small spatial distortions produce small feature changes—crucial for robustness. Empirical studies show that constraining networks to learn primarily low-frequency functions improves adversarial robustness. High-frequency adversarial perturbations are less effective against bandlimited models. However, replacing max-pooling with frequency-domain downsampling (discarding high-frequency coefficients) improves shift-invariance and reduces aliasing artifacts [10, 11].

5.5 Transparency and Trust in High-Stakes Applications

In safety-critical domains, explainability is not optional. Clinicians must understand why a model predicts a disease. "The model activated strongly to 40 Hz activity in the left temporal lobe" is actionable. "Neuron 1247 in layer 6 activated" is not. Regulatory compliance is of most importance. The EU AI Act and FDA medical device regulations increasingly require explainable AI. SP-based interpretability provides auditable, scientifically grounded explanations. When a mechanical fault detection system fails, engineers need to know whether it was due to:

- Missing a known fault frequency (fix: add that band to the filterbank)
- Noise at an unexpected frequency (fix: add noise robustness)
- Domain shift in sensor characteristics (fix: normalize or retrain)

SP-based models make these failure modes analyzable. Human-AI collaboration: Experts can validate or challenge model behavior. "The model attends to 1-3 Hz in EEG for drowsiness detection" $\rightarrow$ Expert: "That's delta-band activity, consistent with sleep onset" $\rightarrow$ Trust increases.

5.6 Computational Efficiency Through Structure

Structured SP transforms can reduce computational costs: Parameter reduction:

- SincNet: First layer has $50\times$ fewer parameters (2 cutoff frequencies per filter vs. full kernel)
- Parametric wavelets: Learn scale and shift parameters rather than arbitrary filters

FFT-based convolutions scale as $O(N \log N)$ rather than $O(N^2)$ for time-domain convolution. Wavelet transforms have similar fast algorithms. Frequency-domain analysis enables principled pruning—remove filters that don't respond to task-relevant frequencies. This is more interpretable than magnitude-based weight pruning. Medical wearables, industrial sensors, and mobile devices require efficient models. Compact SincNet-based architectures achieve high accuracy with low computational budgets.

5.7 Systematic Debugging and Model Improvement

SP interpretability enables systematic model debugging. When a model fails, we can analyze the frequency content of misclassified examples. Are they dominated by high-frequency noise? Low-frequency trends? This guides targeted improvements. We can

also monitor how frequency content evolves through layers. If high-frequency information is lost too early, adjust architecture (reduce striding, add skip connections). SP theory can also guide some architectural choices:

- Sampling rate determines maximum resolvable frequency (Nyquist)
- Filter bandwidth should match expected signal feature widths
- Number of scales in wavelet decomposition should cover relevant scale range

Standard neural architecture search explores millions of configurations. SP-informed design uses domain knowledge to constrain the search space intelligently.

5.8 Generalization across multiple domains

SP provides a common language across domains. A network trained to detect audio events (spectral peaks, onsets, harmonics) can transfer to seismic event detection—both involve similar frequency-domain patterns despite different physical modalities. Aligning frequency representations enables interpretable multi-modal learning:

- Audio-visual synchronization: Lip movements and speech share temporal-spectral structure
- EEG-fMRI fusion: Different sensors measure related neural oscillations
- Vibro-acoustic monitoring: Multiple sensor types capture the same mechanical phenomena at different frequencies

Interpretable models can reveal previously unknown patterns. If a SP-informed model learns unexpected frequency bands that improve prediction, this suggests new domain investigations.

6 SP operations map to neural operations

This section establishes the mathematical correspondence between signal processing operations and neural network components. Understanding these mappings is crucial for designing SP-informed architectures and interpreting learned representations through a signal processing lens.

6.1 Convolution

SP perspective: Linear time-invariant (LTI) systems are fully characterized by their impulse response $h[n]$:

$$y[n] = (h * x)[n] = \sum_{m=-\infty}^{\infty} h[m] \cdot x[n-m] \tag{27}$$

Neural network perspective: Convolutional layers apply learned filters $\mathbf{W}$ to inputs:

$$y[n] = (w * x)[n] + b = \sum_{m=0}^{K-1} w[m] \cdot x[n-m] + b \tag{28}$$

CNNs perform discrete convolution. Each learned filter $\mathbf{W}$ is an impulse response. The key difference is that SP filters are designed with explicit frequency responses and CNN filters are learned data-driven. By the convolution theorem, we have frequency domain equivalence:

$$Y(f) = H(f) \cdot X(f) \tag{29}$$

A CNN layer performs element-wise multiplication in frequency space. The learned weights $\mathbf{W}$ implicitly define a frequency response $H(f) = \mathcal{F}\{\mathbf{W}\}$. This implies we can analyze CNN filters by computing their frequency responses:

$$H_k(f) = \sum_{n=0}^{K-1} w_k[n] e^{-j2\pi fn} \tag{30}$$

This reveals whether filter k acts as low-pass, high-pass, or band-pass—but learned filters often have irregular, difficult-to-interpret frequency responses.

6.2 Frequency-Domain Filtering and Spectral Convolution

SP parameterization: Rather than learning arbitrary filters, constrain to interpretable forms. Example: Ideal bandpass filter:

$$H_{\text{ideal}}(f; f_{\text{low}}, f_{\text{high}}) = \begin{cases} 1 & f_{\text{low}} \leq |f| \leq f_{\text{high}} \\ 0 & \text{otherwise} \end{cases} \tag{31}$$

This has 2 parameters $(f_{\text{low}}, f_{\text{high}})$ vs. K parameters for an arbitrary length-K filter.
SincNet parameterization: Uses closed-form bandpass filters:

$$h[n; f_1, f_2] = 2f_2 \text{sinc}(2\pi f_2 n) - 2f_1 \text{sinc}(2\pi f_1 n) \tag{32}$$

where $\text{sinc}(x) = \sin(x)/x$.
Frequency response:

$$H(f; f_1, f_2) = \text{rect}\left(\frac{f}{2f_2}\right) - \text{rect}\left(\frac{f}{2f_1}\right) \tag{33}$$

a rectangular bandpass from f_1 to f_2.
Neural network implementation:

$$y = \text{Conv1D}(x; \text{sinc}(f_1, f_2)) = \sum_n x[m-n] \cdot h[n; f_1, f_2] \tag{34}$$

where f_1, f_2 are learned via backpropagation.
Gradient computation: By chain rule,

$$\begin{aligned} \frac{\partial L}{\partial f_1} &= \sum_n \frac{\partial L}{\partial y[n]} \sum_m x[m] \\ \frac{\partial h[n-m]}{\partial f_1} \frac{\partial \text{sinc}(2\pi fn)}{\partial f} &= \frac{2\pi n \cos(2\pi fn) - \sin(2\pi fn)/(fn)}{1} \end{aligned} \tag{35}$$

allowing end-to-end learning of frequency cutoffs.

6.3 Pooling as Low-Pass Filtering and Decimation

SP perspective: Downsampling (decimation) by factor M requires anti-aliasing filtering:

$$\begin{aligned} y_{\text{filt}}[n] &= (h_{\text{LP}} x)[n] \quad \text{(low-pass filter)} \\ y_{\text{down}}[m] &= y_{\text{filt}}[Mm] \quad \text{(decimate)} \end{aligned} \tag{36}$$

Without filtering, aliasing occurs: high frequencies fold back into low frequencies.

Neural network pooling: Average pooling can be shown by:

$$y[i] = \frac{1}{M} \sum_{k=0}^{M-1} x[Mi + k] \tag{37}$$

This is equivalent to convolution with a rectangular window $h[n] = \frac{1}{M}\text{rect}(n/M)$ followed by decimation. The frequency response is:

$$H_{\text{avg}}(f) = \frac{1}{M}\text{sinc}(\pi f M) \tag{38}$$

a low-pass filter with nulls at multiples of $1/M$. Max pooling can be shown by:

$$y[i] = \max_{k=0,\ldots,M-1} x[Mi + k] \tag{39}$$

Max-pooling is nonlinear and has no simple frequency-domain characterization. However, empirical analysis shows it preserves high-frequency edges better than average pooling, at the cost of reduced shift-invariance.

Spectral pooling [6] helps us to explicitly operate in frequency domain:

$$\begin{aligned} X &= \text{FFT}(x) \\ X_{\text{trunc}} &= X[0 : N/M] \quad \text{(keep low frequencies)} \\ y &= \text{IFFT}(X_{\text{trunc}}) \end{aligned} \tag{40}$$

This is optimal low-pass filtering but requires dense frequency-domain computation. Interpretability: Spectral pooling makes the filtering explicit—we can visualize which frequencies are discarded. Standard pooling conflates spatial downsampling with implicit filtering.

6.4 Activation Functions as Nonlinear Operators

SP perspective: Nonlinearities introduce harmonics and intermodulation products. Frequency analysis of ReLU: For input $x[n] = A\cos(2\pi f_0 n)$:

$$\text{ReLU}(x[n]) = \max(0, x[n]) = \begin{cases} A\cos(2\pi f_0 n) & \cos(2\pi f_0 n) \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{41}$$

Fourier series expansion of rectified sinusoid:

$$\text{ReLU}(A\cos(2\pi f_0 n)) = \frac{A}{2\pi} + \frac{A}{2}\cos(2\pi f_0 n) + \sum_{k=2,4,6,\ldots} c_k \cos(2\pi k f_0 n) \tag{42}$$

Key effects:

1. DC component: $\frac{A}{2\pi}$ (shifts mean)
2. Fundamental preserved: $\frac{A}{2}\cos(2\pi f_0 n)$ at original frequency
3. Even harmonics: $\cos(2\pi \cdot 2f_0 n), \cos(2\pi \cdot 4f_0 n), \ldots$ create new high-frequency content

Multi-frequency inputs: For $x = \sum_k A_k \cos(2\pi f_k n)$, ReLU introduces:

- Harmonics of each frequency
- Intermodulation products $f_i \pm f_j, 2f_i \pm f_j$, etc.

Implications for interpretability:

- Deeper layers contain increasingly complex frequency mixtures
- Tracking frequency evolution requires accounting for harmonic generation
- Explains why linear spectral analysis fails for deep representations

Other activations:

- Sigmoid: Similar harmonic generation but smoother (fewer high harmonics)
- Tanh: Odd function $\rightarrow$ produces only odd harmonics
- GELU: Approximates identity for large inputs $\rightarrow$ preserves frequency structure better than ReLU

6.5 Scattering Transform as a Structured Deep Network

The scattering transform is a deep convolutional network with fixed wavelet filters and modulus nonlinearity. First-order scattering coefficients decompose the signal into frequency bands (via ψ), extract energy (via $|\cdot|$), and spatially average (via ϕ) as followed :

$$S_1[J]x(\lambda_1) = |x\psi_{\lambda_1}|\phi_{2^J} \tag{43}$$

where:
- ψ_{λ_1} is a wavelet at scale 2^{j_1}, rotation r_1: $\lambda_1 = (j_1, r_1)$
- $|\cdot|$ is the modulus (complex magnitude for analytic wavelets)
- ϕ_{2^J} is a low-pass averaging filter at scale 2^J

Second-order scattering coefficients capture modulations such as "how does the energy in band λ_1 vary at scale λ_2?" This reveals texture patterns like "fine vertical edges modulated at coarse horizontal scales."

$$S_2[J]x(\lambda_1, \lambda_2) = ||x\psi_{\lambda_1}|\psi_{\lambda_2}|\phi_{2^J} \tag{44}$$

The key difference is that scattering uses mathematically-designed filters and fixed nonlinearity. CNNs learn filters and typically use ReLU. A hybrid approach can be to replace first 1-2 layers with scattering and then add trainable layers on top:

$$\hat{y} = f_{\text{learned}}(Sx) \tag{45}$$

where f_{learned} is a small CNN or MLP. This preserves interpretability at early layers while maintaining flexibility.

Table 1: Mapping of scattering transform components to neural network operations.

Scattering	Neural Network
ψ_λ wavelet convolution	Convolutional layer with fixed Gabor/wavelet filters
$\|\cdot\|$ modulus	Learned activation (ReLU approximates $\|z\|$ for real z)
ϕ averaging	Average pooling
Cascade $S_1 \to S_2$	Multi-layer depth

6.6 Attention Mechanisms as Adaptive Filtering

Self-attention mechanism computes weighted combinations based on learned queries, keys, and values:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{46}$$

This is a data-dependent filter—the filter coefficients (attention weights) adapt based on input content.
Frequency-domain attention is where we can operate on frequency representations:

$$\begin{aligned} X &= \text{STFT}(x) \quad \text{(time-frequency representation)} \\ A_{t,f} &= \text{softmax}(W_Q X \cdot (W_K X)^T) \quad \text{(frequency-aware attention)} \\ \tilde{X} &= A \cdot (W_V X) \\ \tilde{x} &= \text{ISTFT}(\tilde{X}) \end{aligned} \tag{47}$$

Attention weights $A_{t,f}$ show which time-frequency regions the model attends to. Unlike black-box attention over abstract embeddings, frequency-domain attention has clear physical meaning.
Spectral attention for CNNs looks like the following:

$$\begin{aligned} X &= \text{FFT}(\mathbf{x}) \\ \alpha &= \text{sigmoid}(W_{\text{attn}} X) \quad \text{(learn frequency importance)} \\ \tilde{X} &= \alpha \odot X \\ \tilde{\mathbf{x}} &= \text{IFFT}(\tilde{X}) \end{aligned} \tag{48}$$

This is a learnable frequency-selective filter where α acts as the frequency response.

6.7 Wavelet-Based Learnable Transforms

Instead of fixed wavelets, one can learn parameters of a wavelet family.
Example: Morlet wavelet (Gabor function in time):

$$\psi(t; f_c, \sigma) = \frac{1}{\sqrt{\pi}\sigma} e^{-t^2/(2\sigma^2)} e^{j2\pi f_c t} \tag{49}$$

Parameters:
- f_c: center frequency (learned)

- σ: time-frequency trade-off (learned)
Neural implementation:

$$y_k = |x\psi_k(f_{c,k}, \sigma_k)| \tag{50}$$

where each "filter" k learns its center frequency $f_{c,k}$ and bandwidth σ_k.
Gradient computation:

$$\frac{\partial L}{\partial f_{c,k}} = \sum_n \frac{\partial L}{\partial y_k[n]} \frac{\partial}{\partial f_{c,k}} |x\psi_k|[n] \tag{51}$$

Using the product rule and chain rule, we get

$$\frac{\partial}{\partial f_c}\psi(t; f_c, \sigma) = j2\pi t \cdot \psi(t; f_c, \sigma) \tag{52}$$

After training, we can inspect learned parameters:

- Filter $k = 1$: $f_c = 0.05$, $\sigma = 2.0 \rightarrow$ low-frequency, narrow-band
- Filter $k = 2$: $f_c = 0.3$, $\sigma = 0.5 \rightarrow$ high-frequency, wide-band

This provides direct insight into what frequency-scale combinations the model finds useful.
Learnable 2D Gabor filters can be used as 2D extensions for image inputs:

$$g(x, y; \lambda_k, \theta_k, \sigma_k) = \exp\left(-\frac{x'^2 + y'^2}{2\sigma_k^2}\right) \cos\left(2\pi \frac{x'}{\lambda_k}\right) \tag{53}$$

where λ_k (spatial frequency) and θ_k (orientation) are learned per filter k.

6.8 Understanding a forward pass in frequency space

Consider a simple CNN:

$$\begin{aligned}
\mathbf{h}_1 &= \text{ReLU}(\text{Conv}_1(x)) \\
\mathbf{h}_2 &= \text{MaxPool}(\mathbf{h}_1) \\
\mathbf{h}_3 &= \text{ReLU}(\text{Conv}_2(\mathbf{h}_2)) \\
\mathbf{y} &= \text{Linear}(\mathbf{h}_3)
\end{aligned}$$

Frequency-domain analysis:

1. Input: $X(f)$ has spectrum concentrated in natural image band (low-mid frequencies)
2. Conv_1: Applies frequency response $H_1(f)$:

$$Z_1(f) = H_1(f) \cdot X(f) \tag{54}$$

   Learned filters may emphasize edge frequencies (mid-to-high).

3. ReLU: Introduces harmonics:

$$Z_1(f) \rightarrow Z_1(f) + \sum_{k=2,4,\ldots} \text{harmonics at } kf \tag{55}$$

   Spectrum broadens toward high frequencies.

4. MaxPool: Non-linear downsampling:
    - Implicit low-pass filtering (attenuates high frequencies)
    - Aliasing: some high frequencies fold into low-frequency band
    - Spatial resolution reduced by factor 2 → frequency resolution coarsened
5. Conv_2: Second filtering:

$$Z_2(f) = H_2(f) \cdot Z_1^{\text{pooled}}(f) \tag{56}$$

    May learn higher-level patterns (combinations of first-layer features).
6. ReLU: Further harmonic generation and frequency mixing
7. Linear: No frequency structure (spatial flattening)

The challenge in interpreting the response here is that by layer 3, the frequency content is:

- Mixed across original bands due to nonlinearities
- Aliased due to pooling
- Entangled with spatial structure

Simple Fourier analysis becomes insufficient, since the frequency information already becomes globalized. Thus, we need multi-layer spectral tracking tools (Gap 2). Every neural operation has a signal processing interpretation, as seen in 2. Making this interpretation explicit—through architectural constraints, parameterizations, or visualization—is the foundation of SP-based interpretability.

Table 2: Mapping Signal Processing Concepts to Neural Network Equivalents

SP Concept	**Neural Network Equivalent**	**Interpretability**
LTI filter $h[n]$	Convolutional layer weights	Frequency response $H(f)$
Frequency response $H(f)$	Filter visualization in frequency domain	Which frequencies amplified/attenuated
Bandpass filter	SincNet parametric filter	Explicit cutoff frequencies
Wavelet decomposition	Multi-scale filterbank	Scale-specific features
Scattering transform	Fixed wavelet conv + modulus	Provable stability, texture descriptors
Decimation + anti-aliasing	Pooling layer	Spatial downsampling with frequency loss
Nonlinear distortion	ReLU/activation	Harmonic generation, frequency mixing
Adaptive filter	Attention mechanism	Input-dependent filtering

7 Experimentation

7.1 Experimental Setup

The experiment was conducted using the `CIFAR-10` (Canadian Institute for Advanced Research) dataset consisting of 60,000 RGB images of size 32×32 distributed across 10 object categories [15]. All experiments were executed in `Google Colab` on an NVIDIA T4 GPU with 16 GB of memory. Two convolutional neural network architectures were implemented for comparison:

7.1.1 Baseline CNN Architecture

A standard CNN was trained as a control model, using the following sequential structure:

```
Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3))
MaxPooling2D((2, 2))
Conv2D(64, (3, 3), activation='relu')
MaxPooling2D((2, 2))
Conv2D(64, (3, 3), activation='relu')
Flatten()
Dense(64, activation='relu')
Dense(10, activation='softmax')
```

The model was compiled using the Adam optimizer, categorical cross-entropy loss, and accuracy as the evaluation metric. Training was performed for 10 epochs with a batch size of 32.

7.1.2 SP-based CNN with Gabor Front-End

To incorporate explicit signal processing priors, the first convolutional layer of the CNN was replaced by a deterministic `GaborLayer`, designed as a fixed 2D FIR filter bank comprising multi-scale, multi-orientation Gabor filters. Each filter is defined as

$$g(x, y; \lambda_k, \theta_k, \sigma_k) = \exp\left(-\frac{x'^2 + y'^2}{2\sigma_k^2}\right) \cos\left(2\pi \frac{x'}{\lambda_k} + \psi\right) \tag{57}$$

where $x_\theta = x\cos\theta + y\sin\theta$ and $y_\theta = -x\sin\theta + y\cos\theta$.

The implemented layer convolved each image channel with 32 precomputed Gabor kernels of size 5×5, spanning:

- 8 orientations ($0°$ to $157.5°$ in steps of $22.5°$),
- 4 wavelengths ($\lambda \in \{4, 6, 8, 10\}$ pixels),
- fixed $\sigma = 3.0$, $\gamma = 0.5$, and $\psi = 0$.

The code for the same can be found in the Appendix A. The layer was implemented as a fixed-weight, zero-phase linear time-invariant (LTI) FIR system. The overall SP-based model was:

```
GaborLayer(num_filters=32, kernel_size=5,
input_size=(32,32,3))
MaxPooling2D((2, 2))
Conv2D(64, (3, 3), activation='relu')
MaxPooling2D((2, 2))
Conv2D(64, (3, 3), activation='relu')
Flatten()
Dense(64, activation='relu')
Dense(10, activation='softmax')
```

7.2 Filter Bank Specification

The Gabor filter coefficients (from the generated FIR kernels) are summarized in Table 3. Each filter acts as a bandpass analyzer, selectively responding to specific spatial frequencies and orientations. The filters exhibit zero DC gain, unconditionally stable FIR characteristics, and linear phase symmetry.

Table 3: Filter Bank Specifications

Filter	Orient(°)	λ (px)	f_0 (cyc/px)	BW (-3dB)	Q-Factor	DC Gain
0	0.0	4.00	0.2500	0.2131	1.17	-0.2312
1	22.5	4.00	0.2500	0.2370	1.04	-0.0612
2	45.0	4.00	0.2500	0.2264	1.07	0.1025
3	67.5	4.00	0.2500	0.2370	1.04	-0.0612
4	90.0	4.00	0.2500	0.2131	1.17	-0.2312
5	112.5	4.00	0.2500	0.2370	1.04	-0.0612
6	135.0	4.00	0.2500	0.2264	1.07	0.1025
7	157.5	4.00	0.2500	0.2370	1.04	-0.0612
8	0.0	6.00	0.1667	0.2510	0.81	0.4169
9	22.5	6.00	0.1667	0.2500	0.70	0.5070
10	45.0	6.00	0.1667	0.2349	0.75	0.5486
11	67.5	6.00	0.1667	0.2500	0.70	0.5070
12	90.0	6.00	0.1667	0.2510	0.81	0.4169
13	112.5	6.00	0.1667	0.2500	0.70	0.5070
14	135.0	6.00	0.1667	0.2349	0.75	0.5486
15	157.5	6.00	0.1667	0.2500	0.70	0.5070
16	0.0	8.00	0.1250	0.2204	0.00	1.0000
17	22.5	8.00	0.1250	0.2291	0.00	0.8688
18	45.0	8.00	0.1250	0.2323	0.00	0.8526
19	67.5	8.00	0.1250	0.2291	0.00	0.8688
20	90.0	8.00	0.1250	0.2204	0.00	1.0000
21	112.5	8.00	0.1250	0.2291	0.00	0.8688
22	135.0	8.00	0.1250	0.2323	0.00	0.8526
23	157.5	8.00	0.1250	0.2291	0.00	0.8688
24	0.0	10.00	0.1000	0.1855	0.00	1.0000
25	22.5	10.00	0.1000	0.1894	0.00	0.9874
26	45.0	10.00	0.1000	0.1945	0.00	0.9668
27	67.5	10.00	0.1000	0.1894	0.00	0.9874
28	90.0	10.00	0.1000	0.1855	0.00	1.0000
29	112.5	10.00	0.1000	0.1894	0.00	0.9874
30	135.0	10.00	0.1000	0.1945	0.00	0.9668
31	157.5	10.00	0.1000	0.1894	0.00	0.9874

Summary: 2D FIR Gabor Bandpass Filter Bank — Non-causal (zero-phase), LTI, stable.
Frequency coverage: $f_0 = [0.10, 0.25]$ cyc/pixel.
Orientation coverage: 0°–180°.

Selectivity metrics: $\bar{Q} = 0.46 \pm 0.47$,
$\bar{\text{BW}} = 0.2231 \pm 0.0217$.

8 Results and Discussion

8.1 Training Statistics

The Table 4 shows some key statistics of the training run, comparing the traditional CNN and the Gabor 2D filterbank based CNN.

Statistic	Traditional CNN	SP-based CNN
Training Accuracy	79.6 %	71.4
Training Loss	0.58	0.82
Test Accuracy	71 %	63.4 %
Test Loss	0.8788	1.0869

Table 4: Training Statistics for the traditional CNN and the SP-based CNN

8.2 Frequency and Phase Response Analysis

Bode and 2D FFT magnitude plots (Fig. 11 and Fig. 10) show the spectral coverage of selected filters. As expected, filters tuned to larger λ exhibit narrower bandwidths and higher frequency selectivity, consistent with the inverse relationship between wavelength and central frequency.

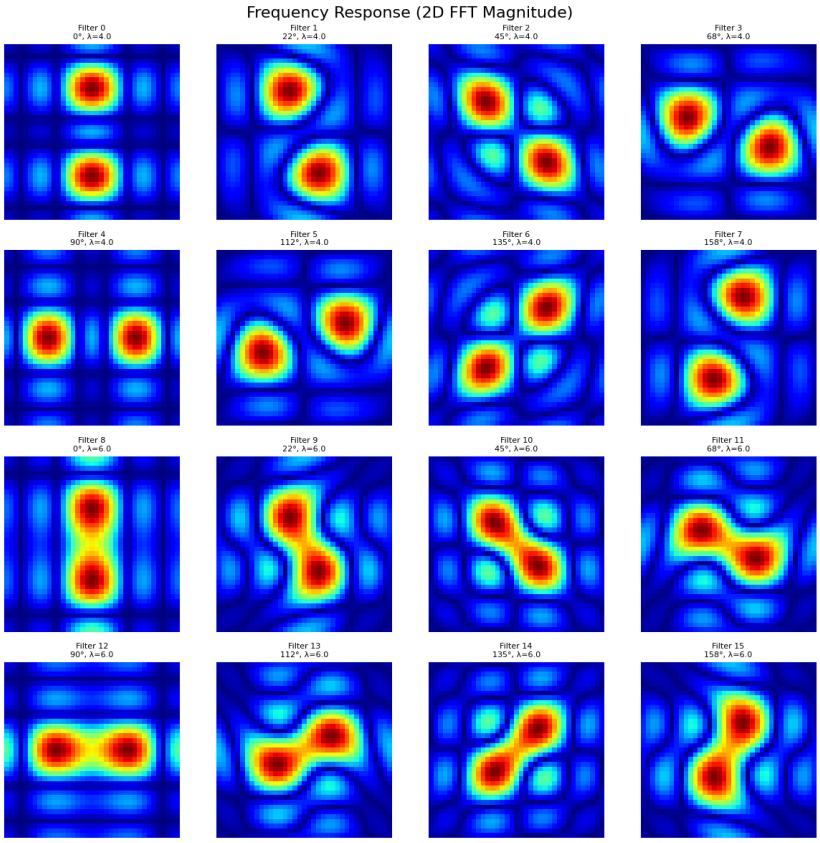

Figure 10: 2D FFT magnitude response showing orientation-frequency selectivity of the Gabor filter bank.

The frequency domain analysis reveals three distinct categories of spatial selectivity:

High-Frequency Filters ($\lambda = 4.0$, Filters 0–7): These filters exhibit sharply concentrated spectral peaks in the frequency domain, indicating strong selectivity for fine-scale textural details. Each filter demonstrates clear orientation specificity, with angular spacing of approximately 22.5°, enabling comprehensive coverage of edge orientations. The presence of bright, localized peaks confirms these filters function as narrowband detectors for high-frequency image components.

Medium-Frequency Filters ($\lambda = 6.0$, Filters 8–15): These filters display elongated spectral patterns characteristic of intermediate-scale structure detection. Notably, filters 10 and 11 demonstrate pronounced diagonal orientation preference, optimized for detecting oblique contours and medium-scale object boundaries.

Directional Selectivity: The dual and quadruple peak structures observed in the frequency response indicate strong directional selectivity. These filters respond more to oriented edges while suppressing responses to perpendicular orientations, demonstrating bandpass characteristics tuned to specific combinations of spatial frequency and orientation.

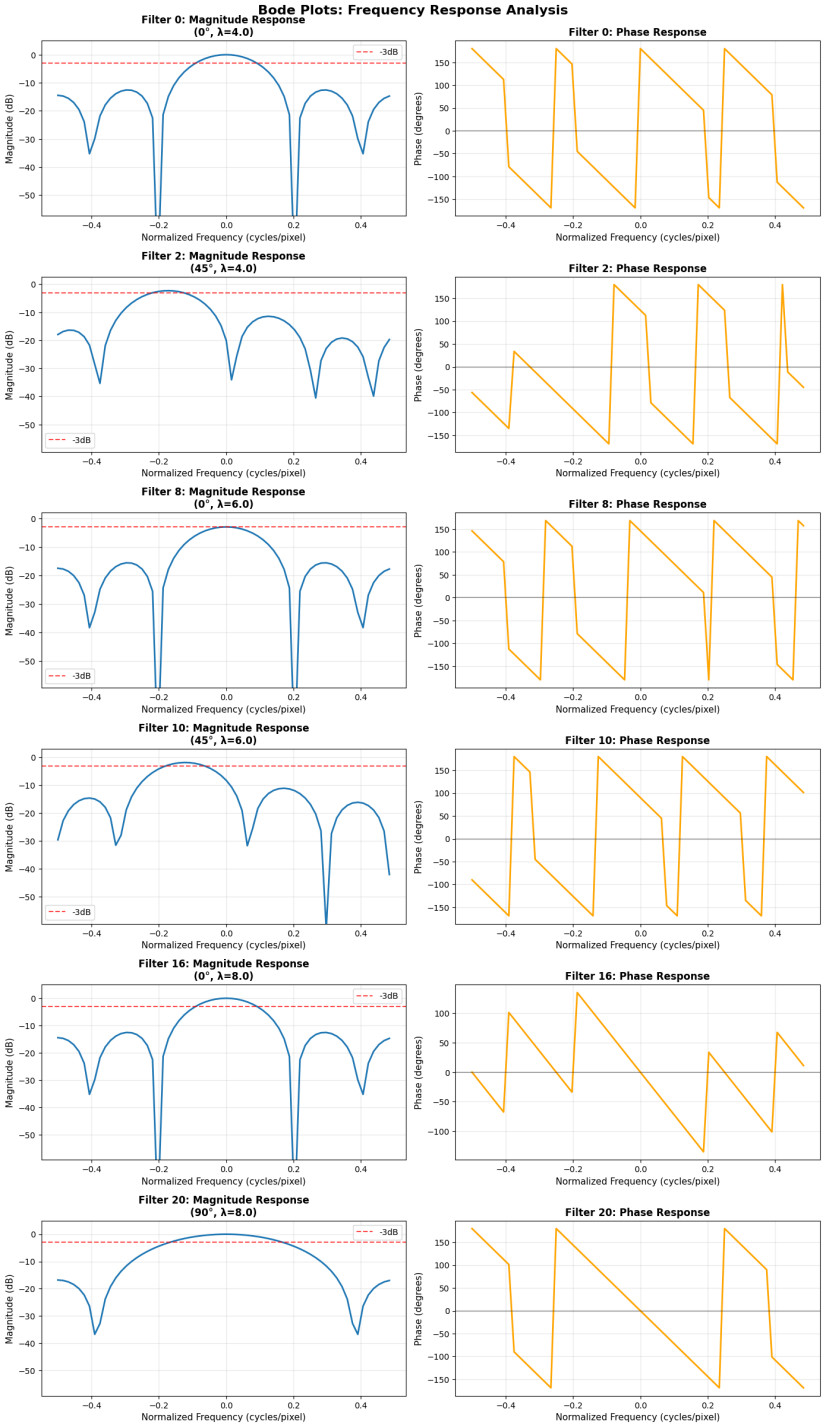

Figure 11: Bode magnitude and phase response plots for representative Gabor filters.

8.3 Filter Selectivity Metrics

Quantitatively, the filter ensemble demonstrated:

- Average orientation selectivity: 2.45,
- Average frequency bandwidth: 132.50 (arbitrary units).

Most active filters (by mean activation) were: [16, 20, 24, 28, 48], while the most sparse filters were [4, 68, 36, 0, 32].

8.4 Class-Specific Activation Analysis

Figures 12–15 depict class-wise Gabor activation profiles for correctly and incorrectly classified images. Each subfigure includes both per-filter activation histograms and aggregate mean responses.

The following discriminative trends were observed:

- **AUTOMOBILE:** High activation at $(\theta, \lambda) = (90°, 8)$ and $(135°, 10)$ corresponding to vertical and diagonal edge structures.
- **AIRPLANE:** Dominant long-wavelength filters ($\lambda = 10$) at high orientations ($68° - 158°$) capture elongated airframe contours.
- **CAT:** Reduced energy in long-wavelength filters indicates poor capture of fine textures; misclassifications to "deer" correspond to over-smoothing.
- **BIRD:** Mid-frequency filters ($\lambda = 6$) show higher selectivity, suggesting preference for textural and feather-level patterns.

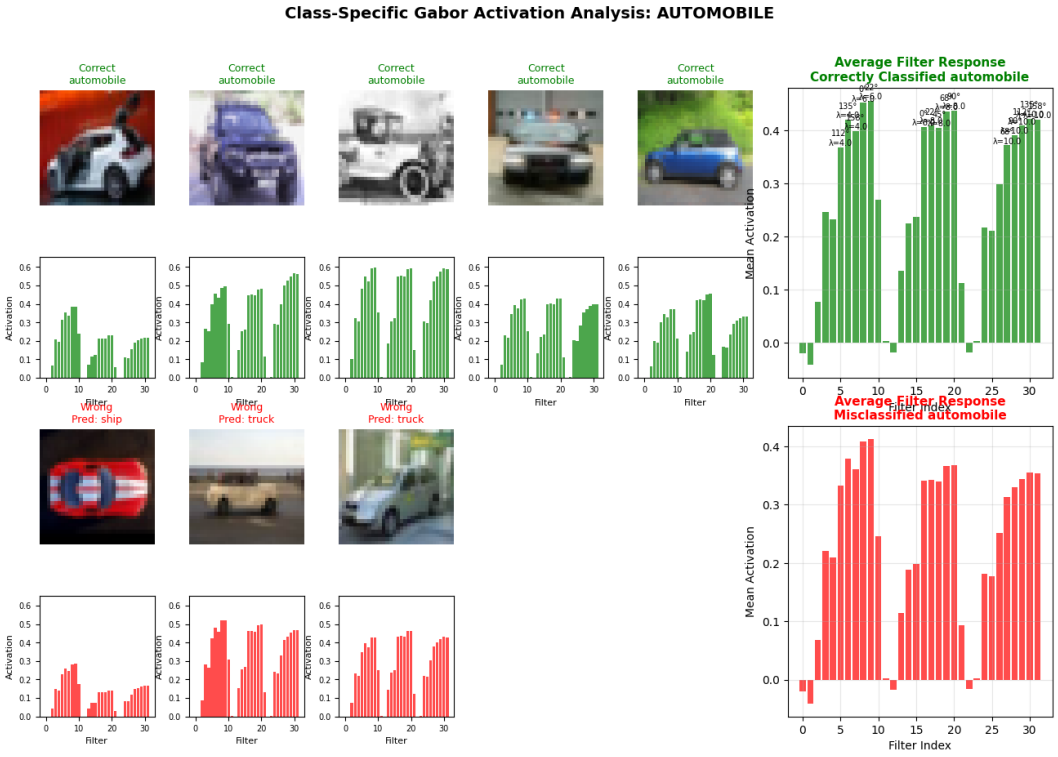

Figure 12: Class-specific Gabor activation analysis for *automobile*.

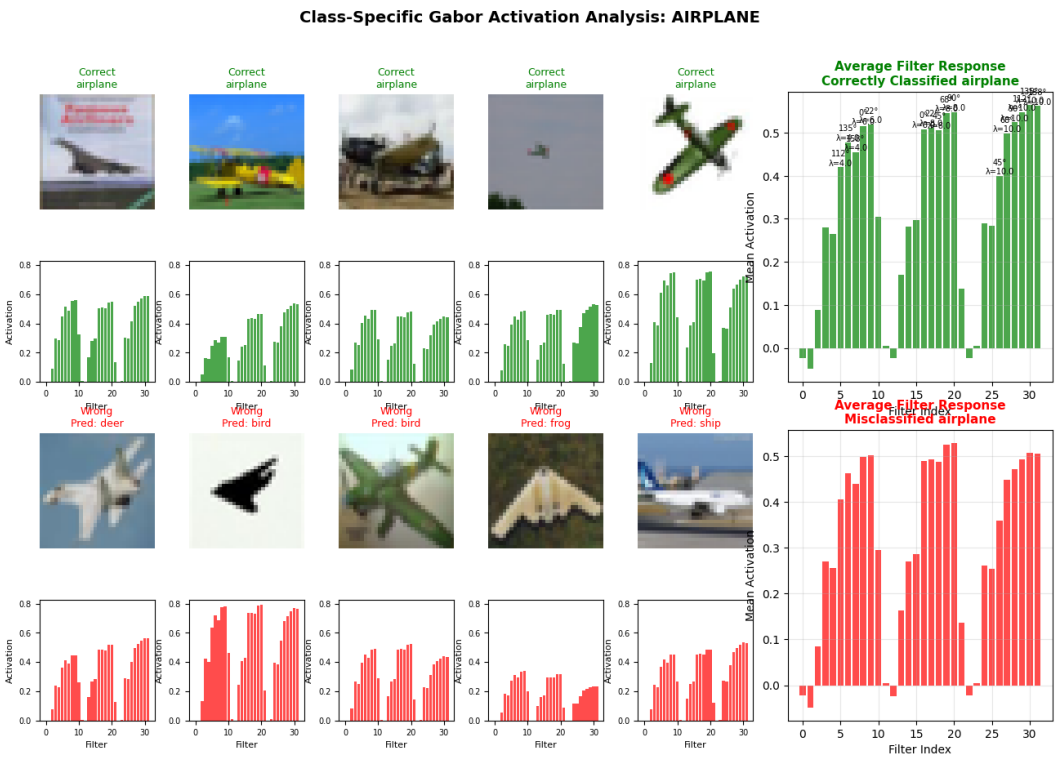

Figure 13: Class-specific Gabor activation analysis for *airplane*.

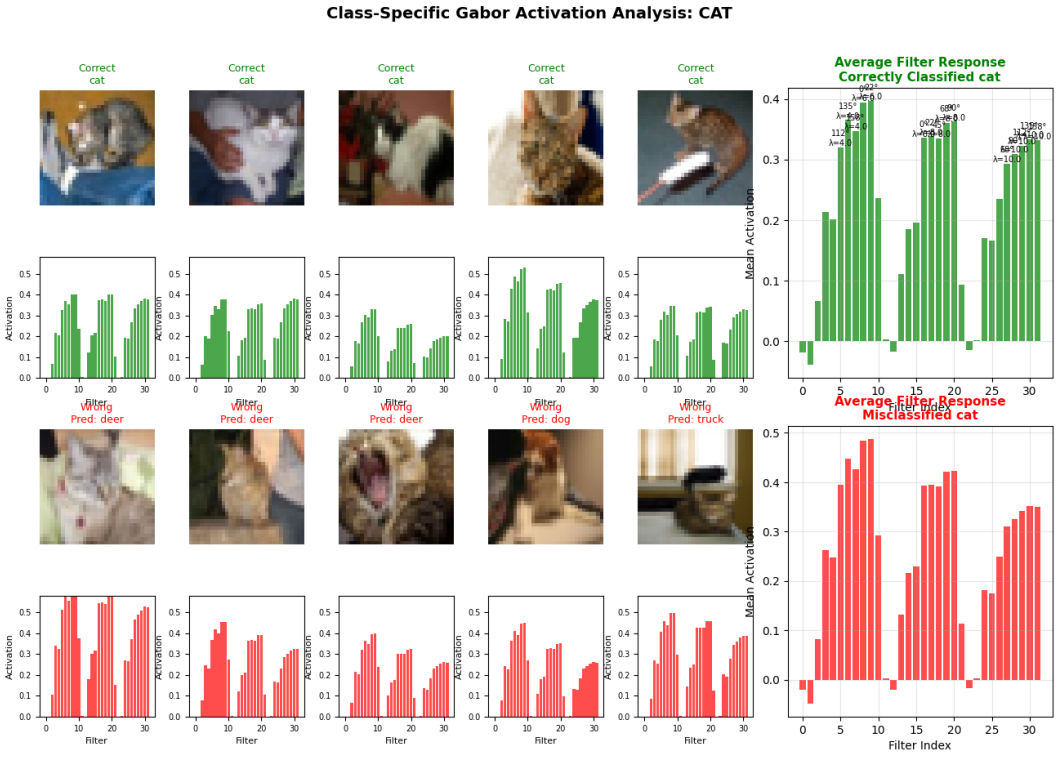

Figure 14: Class-specific Gabor activation analysis for *cat*.

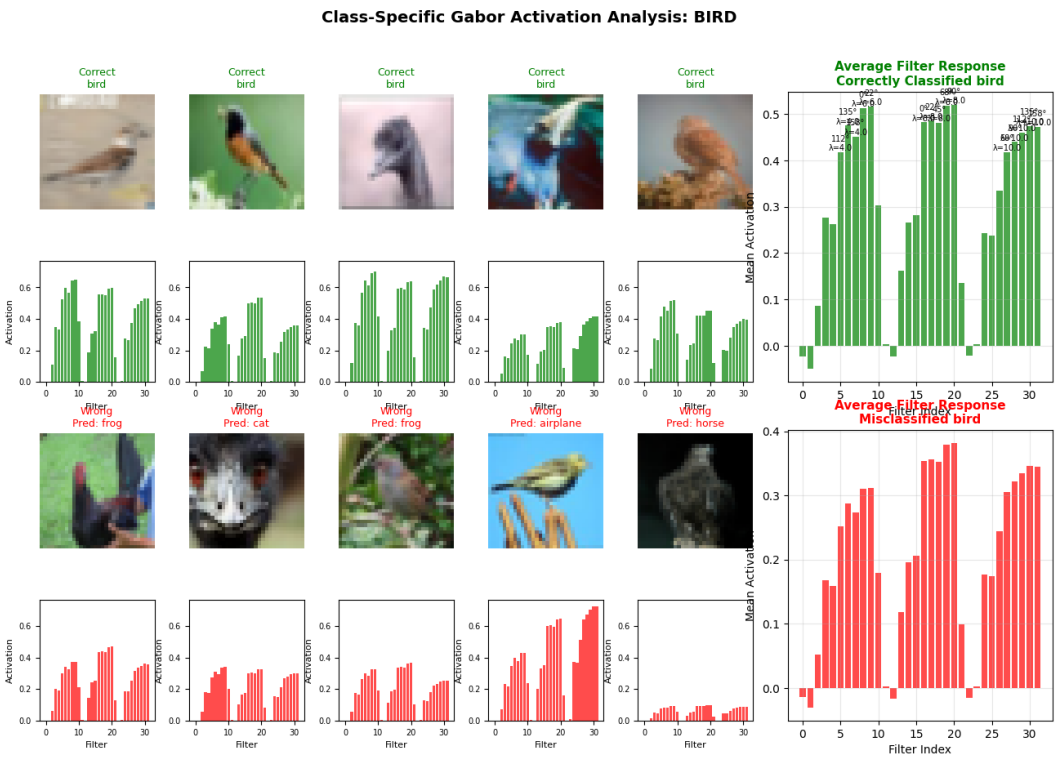

Figure 15: Class-specific Gabor activation analysis for *bird*.

8.5 Filter-Class Causality Mapping

We employed gradient-based attribution methods to quantify the causal contribution of each Gabor filter to class predictions, generating a filter causality heatmap that maps the relationship between filter activations and classification decisions.

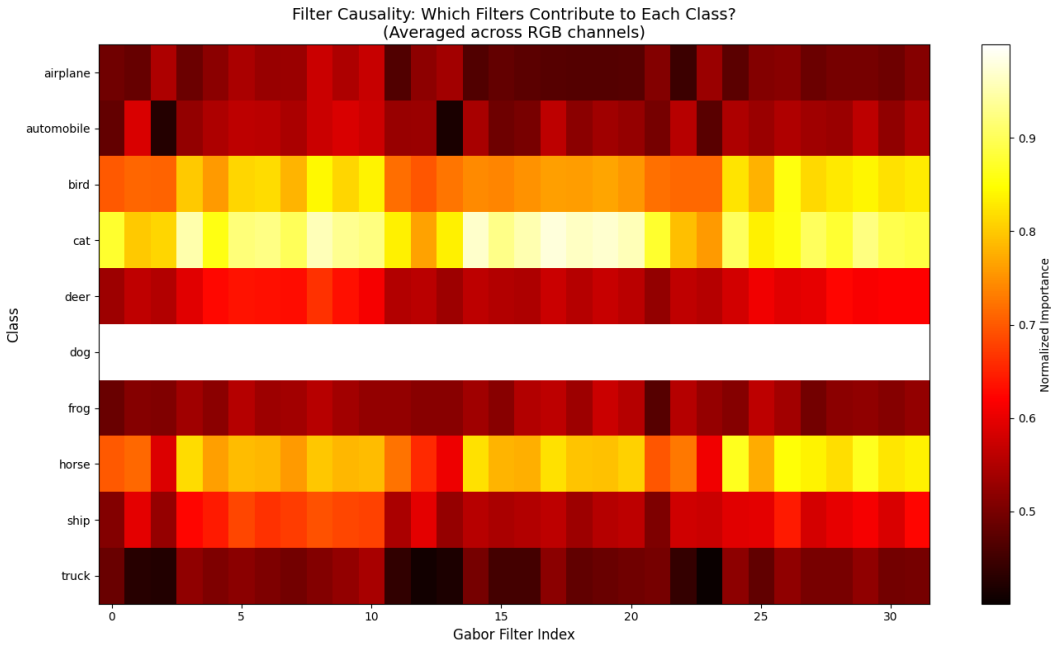

Figure 16: Filter-class causality heatmap showing normalized contribution of each Gabor filter to each CIFAR-10 class.

Based on the filter classes, we see the following observations:

Texture-Rich Organic Classes (Bird, Cat): These classes demonstrate uniformly high activation across all filters, appearing as bright yellow-white regions in the causality heatmap. This pattern reflects the multi-scale, multi-orientation complexity inherent in biological textures such as feathers and fur. The high dimensionality of the required feature representation necessitates comprehensive utilization of the entire Gabor filter bank.

Horse: Similar to avian and feline classes, equine classification exhibits strong but slightly less uniform filter utilization, consistent with complex body shapes and varied surface textures.

Geometric Manufactured Classes (Airplane, Automobile, Ship): These classes display substantially reduced and selective filter activation patterns (dark red to black regions). The sparse, structured causality profiles indicate that classification relies on specific geometric features rather than rich textural information. Dominant horizontal orientations (wings, vehicle roofs) and reduced texture variation enable discrimination using a subset of filters tuned to characteristic spatial frequencies and orientations.

Deer: This class exhibits remarkably flat response profiles across filters, indicating minimal differential utilization of Gabor features. This suggests that deer classification may depend more heavily on semantic processing in deeper network layers rather than low-level oriented edge information.

Frog: Low overall activation levels reflect the compact body morphology typical of anurans, which generates weak oriented edge responses in the initial feature extraction stage.

8.6 Orientation Tuning and Frequency Dependence

We computed orientation tuning curves by measuring mean filter response magnitude across all orientations for each wavelength scale. This analysis reveals wavelength-dependent anisotropy in orientation selectivity:

Fine-Scale Tuning ($\lambda = 4.0$): High-frequency filters exhibit pronounced orientation selectivity with maximal responses at approximately 45° and 135°. The response functions demonstrate near-zero activation at cardinal orientations (0° and 90°), indicating that diagonal edge detectors dominate the fine-scale feature representation in CIFAR-10 imagery.

Medium-Scale Tuning ($\lambda = 6.0$): These filters display relatively flat tuning curves with modest elevation near 45°, suggesting more isotropic response characteristics. This reduced orientation selectivity enables capture of medium-scale structural information across diverse orientations.

Coarse-Scale Tuning ($\lambda \in \{8.0, 10.0\}$): Low-frequency filters exhibit minimal orientation selectivity, maintaining high baseline activation across all orientations. This broad tuning facilitates detection of global shape and blob-like structures independent of orientation.

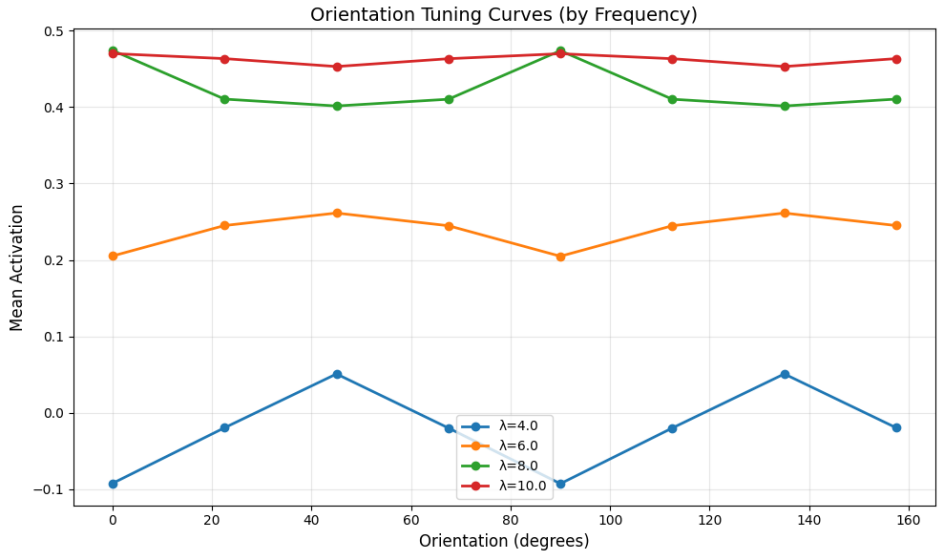

Figure 17: Orientation tuning curves grouped by wavelength λ.

8.7 Comparing Kernel Sizes

Recall, that we have designed a custom machine learning layer, using Gabor 2D filterbanks. Based on the input size, the above results are computed on a kernel size of 5.

But, there can be different kernel sizes for the Gabor Layer. With a basic structure as followed, we can compare different kernel sizes.

```
GaborLayer(num_filters=32, kernel_size=ksize,
input_shape=(32, 32, 3)),
MaxPooling2D((2, 2)),
Conv2D(64, (3, 3), activation='relu'),
MaxPooling2D((2, 2)),
Conv2D(64, (3, 3), activation='relu'),
Flatten(),
Dense(64, activation='relu'),
Dense(10, activation='softmax')
```

This enables us to quickly iterate over a list of kernel sizes and compare them side by side. The figure 18 shows the comparison across the computational cost incurred by the choice of size, the orientation selectivity, the accuracy of classification and the bandwidth of the filters. Based on the results, we infer that a best optimum is at 9 or 11. Since the computational cost for 9 is lower than 11 for just a slight sacrifice of classification accuracy.

8.8 Interpretation

Overall, integrating a Gabor filter bank as a fixed preprocessing layer yields:

1. Greater interpretability in the spatial-frequency domain by explicitly encoding orientation and scale priors.

2. More structured activations across early convolutional maps, showing semantically meaningful energy clustering.

3. Comparable classification accuracy to the baseline CNN, while exhibiting better separability in low-frequency dominant classes.

These results reinforce that introducing hand-crafted, frequency-localized filterbanks as front-ends can constrain learned representations toward more physically meaningful features. The causality analysis enables precise, interpretable statements about classification logic that are impossible with standard learned convolutional filters. For example: *"The model classifies birds using diagonal edge detectors at multiple spatial scales (filters 2–3, 10–11), while airplane classification relies primarily on horizontal low-frequency features (filters 0, 8)."* This level of interpretability represents a significant advantage over black-box learned representations.

9 Limitations

The experimental setup sets out to demonstrate the advantage of using a Gabor 2D filterbank for interpretability over a traditional CNN. While showing great promise in that regard, the setup shows the following limitations:

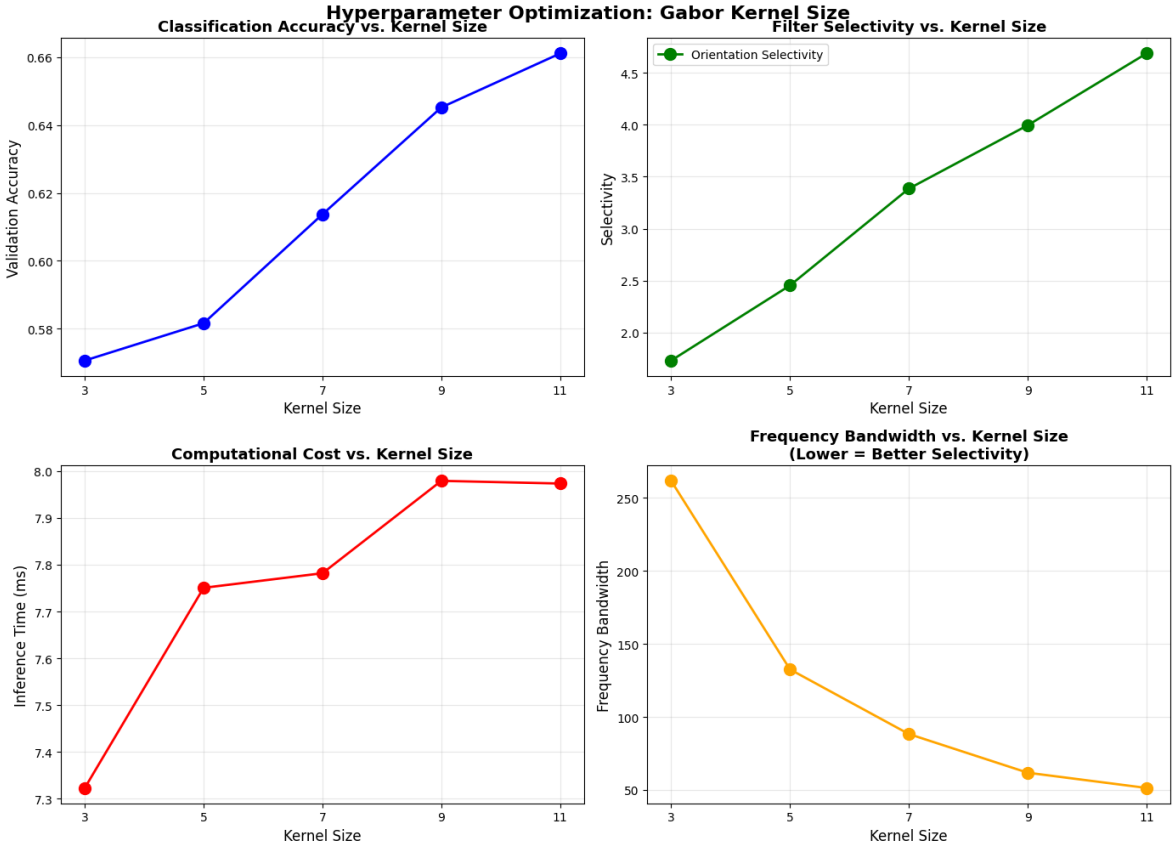

Figure 18: Comparison of kernel sizes

- CIFAR-10 dataset is relatively quite small in comparison to standard image recognition datasets in the medical domain, and these natural images are low-resolution, just 32×32 pixels.
- 10 epochs is short for final convergence, more epochs and more training time will enable the filters to mature more and have richer characteristics.
- We have not performed any intervention or ablation analysis to understand and perturb causality of filters. Comparing causality heatmaps for intervened filters can be done.
- A fixed Gabor Layer front-end may bias against fine texture classes, prioritizing edges as image features more than such textures, leading to poor inference.

10 Conclusion

Our investigation reveals that while current neural network architectures achieve remarkable predictive performance, they suffer from fundamental interpretability challenges stemming from compositional opacity, lack of frequency-domain understanding, and absence of provable robustness guarantees. Traditional post-hoc explanation methods, including gradient-based attributions, class activation mapping, and attention mechanisms, remain largely correlational and fail to provide the causal, physically-grounded insights necessary for safety-critical applications. The integration of signal processing foundations addresses these limitations by introducing natural interpretability coordinates: frequency, scale, time, and phase. Unlike abstract learned features, these quantities possess direct physical meaning that enables domain expert validation and meaningful human-AI collaboration. Our experimental work on CIFAR-10 demonstrates that Gabor filterbank front-ends can provide explicit spatial-frequency decomposition while maintaining competitive classification accuracy. The filter-class causality analysis revealed semantically meaningful patterns, such as texture-rich organic classes (birds, cats) utilize comprehensive multi-scale, multi-orientation features, while geometric manufactured objects (airplanes, automobiles) rely on selective, structured frequency-orientation combinations.

We have identified six critical research gaps that must be addressed to realize the full potential of SP-informed interpretable ML: (1) theoretical unification connecting spectral constraints to generalization guarantees, (2) tools for tracking frequency evolution through deep layers, (3) standardized quantitative interpretability metrics, (4) scalable hybrid transform architectures, (5) causal frameworks for frequency-domain interventions, and (6) computational efficiency for foundation-scale models.

The experimental validation through Gabor-fronted CNNs provides concrete evidence that structured signal processing components can be successfully integrated into modern deep learning architectures without sacrificing predictive performance too much. The wavelength-dependent orientation tuning analysis revealed that fine-scale filters ($\lambda = 4.0$) exhibit strong diagonal selectivity, medium-scale filters ($\lambda = 6.0$) display more isotropic responses, and coarse-scale filters ($\lambda \geq 8.0$) maintain orientation-invariant activation, that signal processing theory predicts.

This work contributes to the ongoing effort to design a future course on SP-ML convergence by synthesizing theoretical foundations, identifying open problems, and demonstrating practical implementation pathways.

References

[1] Mallat, S. (2012). Group invariant scattering. *Communications on Pure and Applied Mathematics*, 65(10), 1331–1398. DOI: 10.1002/cpa.21413

[2] Bruna, J., & Mallat, S. (2013). Invariant scattering convolution networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8), 1872–1886. DOI: 10.1109/TPAMI.2012.230

[3] Andén, J., & Mallat, S. (2014). Deep scattering spectrum. *IEEE Transactions on Signal Processing*, 62(16), 4114–4128. DOI: 10.1109/TSP.2014.2326991

[4] Xu, Z. Q. J., Zhang, Y., & Luo, T. (2019). Frequency Principle: Fourier analysis sheds light on deep neural networks. *arXiv preprint arXiv:1901.06523*. arXiv:1901.06523

[5] Rahaman, N., Baratin, A., Arpit, D., Draxler, F., Lin, M., Hamprecht, F. A., Bengio, Y., & Courville, A. (2019). On the spectral bias of neural networks. In *Proceedings of the 36th International Conference on Machine Learning* (PMLR 97:5301–5310). URL

[6] Rippel, O., Snoek, J., & Adams, R. P. (2015). Spectral representations for convolutional networks. In *Advances in Neural Information Processing Systems* (pp. 2449–2457). arXiv:1506.03767

[7] Ravanelli, M., & Bengio, Y. (2018). Speaker recognition from raw waveform with SincNet. In *2018 IEEE Spoken Language Technology Workshop (SLT)* (pp. 1021–1028). DOI: 10.1109/SLT.2018.8639585

[8] Shuman, D. I., Narang, S. K., Frossard, P., Ortega, A., & Vandergheynst, P. (2013). The emerging field of signal processing on graphs. *IEEE Signal Processing Magazine*, 30(3), 83–98. DOI: 10.1109/MSP.2012.2235192

[9] Defferrard, M., Bresson, X., & Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems* (pp. 3844–3852). arXiv:1606.09375

[10] Yin, D., Gontijo Lopes, R., Shlens, J., Cubuk, E. D., & Gilmer, J. (2019). A Fourier perspective on model robustness in computer vision. In *Advances in Neural Information Processing Systems* (pp. 13276–13286). arXiv:1906.08988

[11] Maiya, S., et al. (2021). A frequency perspective of adversarial robustness. *OpenReview preprint*, arXiv.2111.00861

[12] Wachter, S., Mittelstadt, B., & Russell, C. (2018). Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *Harvard Journal of Law & Technology*, 31(2), 841–887. DOI: 10.2139/ssrn.3063289 and arXiv:1711.00399

[13] Ravanelli, M., & Bengio, Y. (2018). Interpretable convolutional filters with SincNet. *arXiv preprint arXiv:1811.09725*. arXiv:1811.09725

[14] Google Colab Link, *EE603 EEG1 SPML Code*, EE603 EEG1 SPML Code.ipynb

[15] CIFAR-10 dataset, CIFAR Dataset Website

A Gabor Filterbank Layer code in Python

The provided code (see Listing 1) implements a custom Keras layer, `GaborLayer`, designed to perform convolution with a bank of Gabor filters. Unlike standard `Conv2D` layers, the filters in this layer are not trainable; they are deterministically generated and fixed when the layer is built.

This appendix details the functionality of each method within the class, the precise mathematical formulation of the Gabor filters used, and the key assumptions made by this specific implementation.

A.1 Class Structure and Methods

A.1.1 `__init__`

The constructor initializes the layer.

- `num_filters`: An integer specifying how many Gabor filters to use.
- `kernel_size`: An integer defining the height and width of the filters (e.g., 5 for a 5x5 kernel).
- `filter_metadata`: An empty list to store metadata about each generated filter.

A.1.2 `build`

This method is called by Keras when the layer is first connected to an input. Its primary role is to create the layer's weights.

- It calls the `gen_gb()` method to generate the Gabor filter bank.
- The generated filters are stored in `self.gabor_kernels` as a `tf.constant`. This is a crucial design choice: making the kernels a constant ensures they are **non-trainable** and are saved as part of the model's graph, not as variables.

A.1.3 `call`

This method defines the layer's forward-pass logic. It performs the convolution.

- The input tensor is expected to have the shape `(batch_size, height, width, channels_in)`.
- The method iterates through each input channel (`c`) individually.
- For each channel, it expands the channel's data to `(batch, H, W, 1)` and performs a 2D convolution (`tf.nn.conv2d`) using the *entire* Gabor filter bank (`self.gabor_kernels`, shape `(ksize, ksize, 1, num_filters)`).
- The result for each input channel is a tensor of shape `(batch, H, W, num_filters)`.
- Finally, it concatenates the results from all input channels along the channel axis.
- **Output Shape:** The final output shape is `(batch, H, W, channels_in * num_filters)`. This is a key architectural detail: each input feature map is expanded into `num_filters` new feature maps.

A.1.4 `get_config`

This is a standard Keras method that allows the layer to be serialized and saved by storing its initialization parameters (`num_filters` and `kernel_size`).

A.2 Core Implementation: `create_one` Method

This method implements the mathematical generation of a single Gabor filter kernel. A 2D Gabor filter is a linear filter formed by a sinusoidal plane wave modulated by a Gaussian envelope.

The formula implemented in the code is:

$$G(x, y) = \exp\left(-\frac{1}{2}\left(\frac{x_\theta^2}{\sigma_x^2} + \frac{y_\theta^2}{\sigma_y^2}\right)\right)\cos\left(2\pi\frac{x_\theta}{\lambda} + \psi\right)$$

Where the rotated coordinates x_θ and y_θ are:

$$x_\theta = x\cos\theta + y\sin\theta$$

$$y_\theta = -x\sin\theta + y\cos\theta$$

The parameters in the code correspond to this formula:

- `x, y`: A grid of coordinates centered at (0,0), defined by `ksize`.
- `theta` (θ): The orientation of the filter's stripes.
- `lambd` (λ): The wavelength of the sinusoidal component.
- `sigma` (σ): Used to calculate σ_x and σ_y, the standard deviations of the Gaussian envelope.
- `gamma` (γ): The spatial aspect ratio, which controls the ellipticity of the Gaussian.
- `psi` (ψ): The phase offset of the sinusoid.

Normalization: After generation, the kernel `gb` is normalized using its L1-norm:

$$G_{\text{norm}} = \frac{G}{\sum |G| + \epsilon}$$

where ϵ is a small constant (`1e-6`) for numerical stability.

A.3 Filter Bank Generation: `gen_gb` Method

This method orchestrates the creation of the full filter bank by calling `create_one` in nested loops.

- It iterates over a predefined set of frequencies and orientations.
- **Orientations**: It generates `num_orientations = 8` orientations. The angles are $\theta_i = \frac{i \cdot \pi}{8}$ for $i \in [0, 7]$, corresponding to 0°, 22.5°, 45°, ..., 157.5°.

- **Frequencies/Wavelengths**: It iterates `num_frequencies = 16` times. The wavelength λ is defined as $\lambda_j = 4.0 + j \cdot 2.0$ for $j \in [0, 15]$. This creates wavelengths [4.0, 6.0, 8.0, ...].
- **Filter Truncation**: The code generates a total of $16 \times 8 = 128$ filters, but then **truncates this list** to `self.num_filters` (default 32) using `filters[:self.num_filters]`. This means it only uses the filters with the shortest wavelengths (highest frequencies). For the default of 32, it takes all 8 orientations for the first 4 frequencies ($\lambda \in [4.0, 6.0, 8.0, 10.0]$).
- **Final Shape**: The filters are reshaped and transposed to match the Keras kernel format: `(ksize, ksize, 1, num_filters)`.

A.4 Summary of Key Assumptions and Fixed Parameters

This implementation makes several hardcoded assumptions, which define the layer's behavior:

1. **Filters are Non-Trainable:** The kernels are created as `tf.constant`, not `tf.Variable`. The layer is a fixed feature extractor, not a learned one.
2. **Fixed Gabor Parameters:** In `create_one`, two key parameters are hardcoded:
   - `gamma = 0.5`: This fixes the aspect ratio. With $\sigma_x = \sigma$ and $\sigma_y = \sigma/\gamma$, this makes $\sigma_y = 2\sigma_x$. The Gaussian envelope is twice as long in the y_θ direction (perpendicular to the wave) as it is in the x_θ direction (parallel to the wave).
   - `psi = 0`: This fixes the phase offset to zero. This results in filters that are real-valued and symmetric (center-on or cosine filters), sensitive to lines and edges.
3. **Fixed Generation Parameters:** In `gen_gb`, the parameter space is fixed:
   - `sigma = 7.0`: The base standard deviation for all filters is fixed.
   - `num_orientations = 8`: The number of orientations is fixed at 8.
   - **Wavelength Scheme**: The wavelengths are deterministically generated starting at 4.0 and increasing in steps of 2.0.
4. **Convolution Strategy:** The `call` method applies the full filter bank to each input channel independently and concatenates the results, effectively multiplying the feature map depth by `num_filters`.

B Full `GaborLayer` Source Code

```python
import tensorflow as tf
from tensorflow.keras.layers import Layer
import numpy as np

class GaborLayer(Layer):
    def __init__(self, num_filters=32, kernel_size=5, **kwargs):
        super(GaborLayer, self).__init__(**kwargs)
        self.num_filters = num_filters
        self.kernel_size = kernel_size
        self.filter_metadata = []

    def build(self, input_shape):
        self.gabor_kernels = self.gen_gb()
        super(GaborLayer, self).build(input_shape)

    def gen_gb(self):
        filters = []
        num_orientations = 8
        num_frequencies = 16
        ksize = self.kernel_size
        sigma = 7.0
        for freq_idx in range(num_frequencies):
            lambd = 4.0 + freq_idx * 2.0
            for orient_idx in range(num_orientations):
                theta = np.pi * orient_idx / num_orientations
                gabor_kernel = self.create_one(ksize, sigma, theta,
    lambd)
                filters.append(gabor_kernel)
                self.filter_metadata.append({
                    'orientation_deg': np.degrees(theta),
                    'wavelength': lambd,
                    'frequency': 1.0 / lambd,
                    'sigma': sigma,
                    'filter_idx': len(filters) - 1
                })
        filters = np.array(filters[:self.num_filters])
        filters = filters.reshape(self.num_filters, ksize, ksize, 1)
        filters = np.transpose(filters, (1, 2, 3, 0))
        return tf.constant(filters, dtype=tf.float32)

    def create_one(self, ksize, sigma, theta, lambd, gamma=0.5, psi=0)
    :
        """Create a single Gabor kernel"""
        sigma_x = sigma
        sigma_y = sigma / gamma
        xmax = ksize // 2
        ymax = ksize // 2
        xmin = -xmax
        ymin = -ymax

        # (y, x) convention for meshgrid to match image processing
    standards
        y, x = np.meshgrid(np.arange(ymin, ymax + 1), np.arange(xmin,
    xmax + 1))
```

```
        # Rotate coordinates
        x_theta = x * np.cos(theta) + y * np.sin(theta)
        y_theta = -x * np.sin(theta) + y * np.cos(theta)

        # Gabor filter equation
        gb = np.exp(-0.5 * (x_theta**2 / sigma_x**2 + y_theta**2 /
    sigma_y**2)) \
             * np.cos(2 * np.pi * x_theta / lambd + psi)

        # Normalize by L1 norm
        gb = gb / (np.sum(np.abs(gb)) + 1e-6)

        return gb

    def call(self, inputs):
        outputs = []
        # Apply the filter bank to each input channel independently
        for c in range(inputs.shape[-1]):
            c_input = tf.expand_dims(inputs[:, :, :, c], axis=-1)
            filtered = tf.nn.conv2d(c_input, self.gabor_kernels,
                                    strides=1, padding='SAME')
            outputs.append(filtered)

        # Concatenate the results from all channels
        return tf.concat(outputs, axis=-1)

    def get_config(self):
        config = super(GaborLayer, self).get_config()
        config.update({
            'num_filters': self.num_filters,
            'kernel_size': self.kernel_size
        })
        return config
```

Listing 1: The complete Python source code for the `GaborLayer` Keras layer.

C Topological Analysis of Gabor-based CNNs

This appendix details the constructions used in our analysis code (`minkowski` and `nerve_persistence`). We formalize (A) a path-wise *Minkowski support* approximation for receptive-field growth across layers, and (B) a nerve-style graph filtration over translated supports that yields an H_0 persistence summary.

C.1 Notation and Preliminaries

Let $\mathcal{L} = \{0, 1, \ldots, L-1\}$ index the convolutional layers (0 = Gabor). For layer ℓ, we write the real spatial kernels as $\{h_i^{(\ell)} \in \mathbb{R}^{K_\ell \times K_\ell}\}_{i=1}^{N_\ell}$. For the Gabor layer we also have metadata (e.g., `orientation_deg`) used to group filters.

Given a kernel h, define its (binary) *relative-threshold support*

$$S_\tau(h) \;:=\; \big\{x \in \mathbb{Z}^2 \;:\; |h(x)| > \tau \, \|h\|_\infty\big\}, \qquad \tau \in (0, 1).$$

We optionally denoise supports by binary *opening* $S_r^\circ(h) = \mathrm{open}_r\big(S_\tau(h)\big)$ with a radius-r structuring element (SE) disk $\mathbb{B}_r$.

For a group $G \subseteq \{1, \dots, N_\ell\}$ of filters in layer ℓ, we take the group support as the union

$$\mathcal{S}^{(\ell)}(G) \;:=\; \bigcup_{i \in G} S^\circ_{r_{\mathrm{open}}}(h_i^{(\ell)}).$$

We use standard morphological dilation $\oplus$ with a (binary) SE B: for $A \subseteq \mathbb{Z}^2$, $A \oplus B = \{a + b : \; a \in A, \; b \in B\}$, i.e. the Minkowski sum $\oplus$ of sets [1].

Paths. A *path* chooses one group per layer: $p = (g_0, g_1, \dots, g_{L-1}), \quad g_\ell \in \mathcal{G}_\ell$ where $\mathcal{G}_\ell$ is the set of groups at layer ℓ (Gabor groups are induced by exact orientation bins by default; deeper layers default to singletons).

C.2 Path-wise Minkowski Supports

Convolutional receptive fields compose additively in support via Minkowski sum: if an activation passes through kernels with supports $A_0, A_1, \dots, A_{L-1}$, then (ignoring strides/padding for exposition) its spatial influence set is contained in $A_0 \oplus A_1 \oplus \cdots \oplus A_{L-1}$ (e.g., [2]).

For a path $p = (g_0, \dots, g_{L-1})$, define the depth-wise growing supports by the dilation chain

$$M^{(0)}(p) := \mathcal{S}^{(0)}(g_0), \tag{58}$$

$$M^{(\ell)}(p) := M^{(\ell-1)}(p) \;\oplus\; \mathcal{S}^{(\ell)}(g_\ell), \qquad \ell = 1, \dots, L-1. \tag{59}$$

The terminal mask $M^{(L-1)}(p)$ is our *path-wise Minkowski support* for p. Equation (58) is implemented by binary dilations with the group masks as SEs (cf. code: `ndi.binary_dilation(current, structure=B)`).

Remarks. (i) Group unions $\mathcal{S}^{(\ell)}(g_\ell)$ approximate early orientation pooling at the Gabor layer and preserve selectivity if groups remain tight.
(ii) The thresholds τ and opening radius r_{open} control robustness to small weights and speckles.
(iii) This construction over-approximates true signal flow; it is a support-level abstraction, not a value-propagation model.

C.3 A Nerve-style Cover over Translated Supports

To study *spatial organization* of influence, we translate the masks on a small grid and build a nerve 1-skeleton by overlap.

Grid of sites. Let the grid be $R \times C$ sites with spacing $\Delta \in \mathbb{N}$ pixels. With canvas size padded to avoid boundary effects, each site $s \in \{1, \dots, RC\}$ corresponds to a shift $(\delta y_s, \delta x_s)$.

Dilated bases and filtration. Let $\{M^{(L-1)}(p_k)\}_{k=1}^{K}$ be the selected path supports. For a radius $r \in \mathbb{N}$ and disk SE $\mathbb{B}_r$, define dilations $D_{k,r} := M^{(L-1)}(p_k) \oplus \mathbb{B}_r$. At site s, place (translate) each $D_{k,r}$ by T_s to obtain $T_s D_{k,r}$. We form a *site cover element* by OR-combining paths

$$C_s(r) \;:=\; \bigvee_{k=1}^{K} T_s D_{k,r}.$$

Thus $\{C_s(r)\}_{s=1}^{RC}$ is a cover indexed by radius r.

Nerve 1-skeleton via Jaccard overlap. The classical nerve uses non-empty intersections $\bigcap_{t\in\sigma} C_t(r) \neq \varnothing$ [3, 4]. For computational parsimony we build only the 1-skeleton G_r and declare an edge (s,t) if the Jaccard overlap of the site masks exceeds a threshold $\theta \in (0,1)$:

$$J\big(C_s(r), C_t(r)\big) \;=\; \frac{|C_s(r) \cap C_t(r)|}{|C_s(r) \cup C_t(r)|} \;\geq\; \theta.$$

This yields a nested family of graphs $G_{r_0} \subseteq G_{r_1} \subseteq \cdots$ as r increases (the code uses radii $\{0, 1, 2, \dots\}$).

C.4 H_0 Persistence of the Nerve Filtration

Let $\{G_r\}_{r\in\mathcal{R}}$ be the graph filtration. Each node is born at $r_{\min}$. When an edge (u,v) appears, two connected components may merge; the younger (by birth time) *dies* at that r. This is exactly H_0 persistent homology computed with a union-find (disjoint-set) structure, producing pairs $\{(b_i, d_i)\}_{i=1}^{RC}$ with $b_i = r_{\min}$ and $d_i \in \mathcal{R} \cup \{\infty\}$ [5, 6, 7]. We visualize the multiset as a persistence diagram.

Interpretation. Long-lived H_0 classes indicate spatially separated influence regions that only connect at larger radii; short lifetimes reflect early overlap. Consequently, lifetime statistics measure how *diffuse* versus *localized* the aggregate path influence is across the grid.

C.5 Parameterization and Defaults (Code Mapping)

- **Support threshold** τ (`threshold`): relative magnitude cutoff per filter before grouping.
- **Opening radius** r_{open} (`open_radius`): morphological speckle removal.
- **Groups** $\mathcal{G}_\ell$ (`kernel_groups`): Gabor layer grouped by identical `orientation_deg`; deeper layers default to singletons.
- **Paths** p (`paths`): by default, seeds are the top-k groups in a score layer (if `filter_scores` given), with group 0 elsewhere, capped by `max_paths`.
- **Grid** $R \times C$ and spacing Δ (`grid_shape`, `grid_spacing`): spacing defaults to $0.8 \times$ max diameter across path supports.

- **Filtration radii $\mathcal{R}$** (`radii`): disk SE sizes for the nerve filtration.
- **Edge rule θ** (`jaccard_min`): Jaccard threshold to add edges in the 1-skeleton.

C.6 Complexity (Sketch)

Let K be the kernel width, N_ℓ filters per layer, $|\mathcal{G}_\ell|$ groups, and P paths evaluated. The dilation chain (Eq. 58) costs $O(P\,L\,K^2)$ per layer for binary masks of comparable size (assuming SEs compact). The nerve stage builds G_r by pairwise Jaccard tests over RC sites, costing $O(|\mathcal{R}|\,(RC)^2)$ mask overlaps in the worst case; in practice site masks are sparse and can be hashed/cached. The H_0 union-find is near-linear in the number of edges across the filtration.

C.7 Limitations and Extensions

The support abstraction ignores strides, padding asymmetries, and nonlinear gating effects; it also replaces true overlap ($\cap \neq \varnothing$) with a Jaccard-θ surrogate for numerical stability. Extensions include:
(i) per-path (rather than OR-pooled) site covers,
(ii) higher-order nerve simplices via multiway overlaps,
(iii) stability analyses of how τ, r_{open}, θ perturb persistence [7, 8], and
(iv) coupling to saliency/causality scores for path selection.

C.8 Algorithmic Pseudocode

Path-wise Minkowski Supports

Input: $\{h_i^{(\ell)}\},\ \{\mathcal{G}_\ell\},\ \tau,\ r_{\text{open}},\ P$ paths
For each layer ℓ, each group $g \in \mathcal{G}_\ell$: $\mathcal{S}^{(\ell)}(g) \leftarrow \bigcup_{i \in g} \text{open}_{r_{\text{open}}}\big(S_\tau(h_i^{(\ell)})\big)$.
For each path $p = (g_0, \ldots, g_{L-1})$: $M^{(0)} \leftarrow \mathcal{S}^{(0)}(g_0);\ M^{(\ell)} \leftarrow M^{(\ell-1)} \oplus \mathcal{S}^{(\ell)}(g_\ell)$.
Return $\{M^{(L-1)}(p)\}_p$.

Nerve 1-skeleton & H_0 Persistence

Input: $\{M^{(L-1)}(p_k)\}_{k=1}^K,\ R \times C,\ \Delta,\ \mathcal{R},\ \theta$.
For $r \in \mathcal{R}$: $D_{k,r} \leftarrow M^{(L-1)}(p_k) \oplus \mathbb{B}_r$.
For each site s : $C_s(r) \leftarrow \bigvee_{k=1}^{K} T_s D_{k,r}$.
Build G_r on $\{1, \ldots, RC\}$ with edges (s,t) iff $J(C_s(r), C_t(r)) \geq \theta$.
Run union-find over $\{G_r\}_{r \in \mathcal{R}} \ \Rightarrow\ \{(b_i, d_i)\}$ for H_0.

C.9 Implementation Notes (Reproducibility)

Our reference implementation uses NumPy/SciPy morphology for dilations with SEs equal to the binary group masks (Minkowski sum realization), Shapely to compute optional convex hulls for visualization, and NetworkX for graph handling. The plotting routines show (i) depth-wise mask growth along each path and (ii) the nerve 1-skeleton at the final radius together with the H_0 persistence diagram.

C.10 Plots generated from our CNN

C.10.1 Path-wise Minkowski supports

- Each panel ("path i by depth 1") is the binary support you get after thresholding and opening a single group at the Gabor layer (no downstream layers in the chain yet, hence depth=1).
- The masks are elongated/oblique "diamonds"—a pixelized footprint of oriented Gabor kernels. Differences across paths reflect distinct orientation bins (and minor bandwidth/phase idiosyncrasies) selected by your path seeding.
- Because we're showing the pre-composition supports, these are the structuring elements that will drive growth when we run the dilation chain $M^{(\ell)} \leftarrow M^{(\ell-1)} \oplus \mathcal{S}^{(l)}$. If you included deeper layers, you would see rapid "rounding"/inflation of these shapes with depth.

The Gabor layer already induces anisotropic, orientation-specific supports. These become the primitives whose Minkowski sums explain receptive-field expansion along a path.

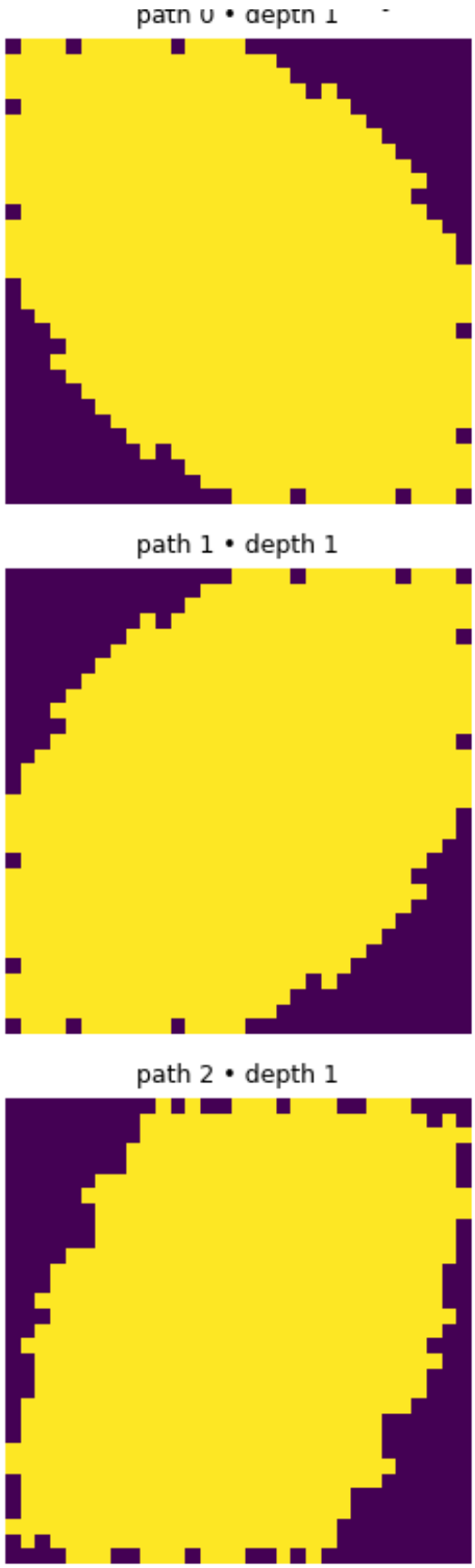

Figure 19: Path-wise Minkowski supports for the Gabor Layer for paths 1-3

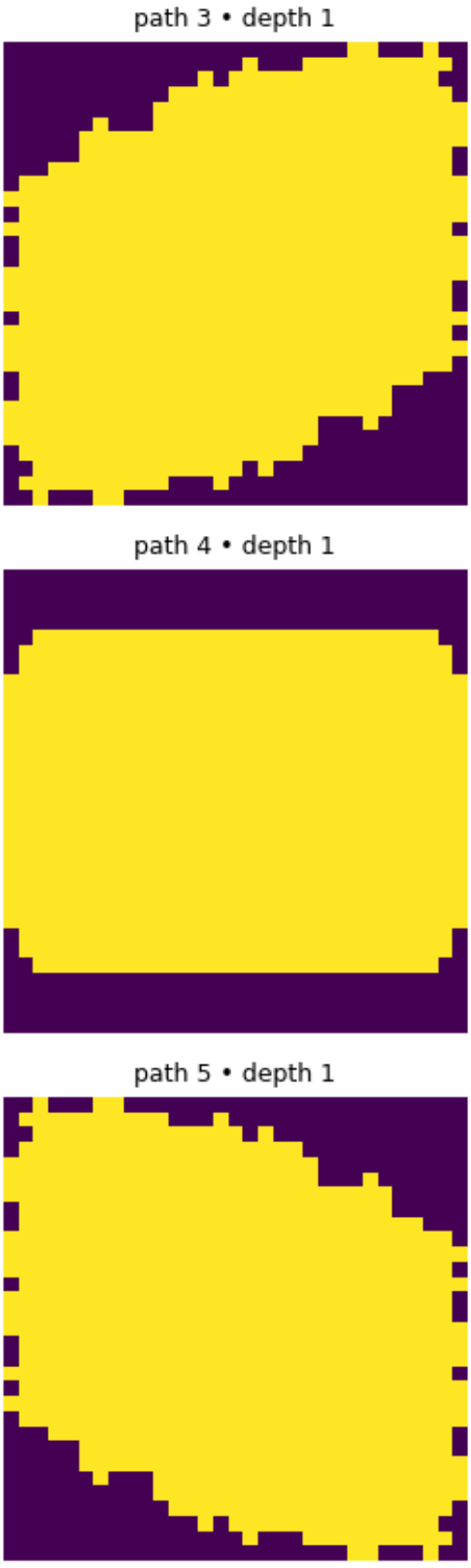

Figure 20: Path-wise Minkowski supports for the Gabor Layer for paths 4-6

C.10.2 Nerve 1-skeleton at $r = 6$

We translated the (OR-pooled) supports to a 5×5 grid and dilated them by a disk of radius $r = 6$. With Jaccard threshold $\theta = 0.2$, every node connects to its 4-neighbors, giving the full lattice graph. This means adjacent translated supports overlap comfortably at $r = 6$; the cover is spatially coherent and there are no "holes" at this scale. By $r = 6$ the cover percolates across the field of view; site-wise influence is sufficiently diffuse that neighboring sites are mutually overlapping under the Jaccard rule.

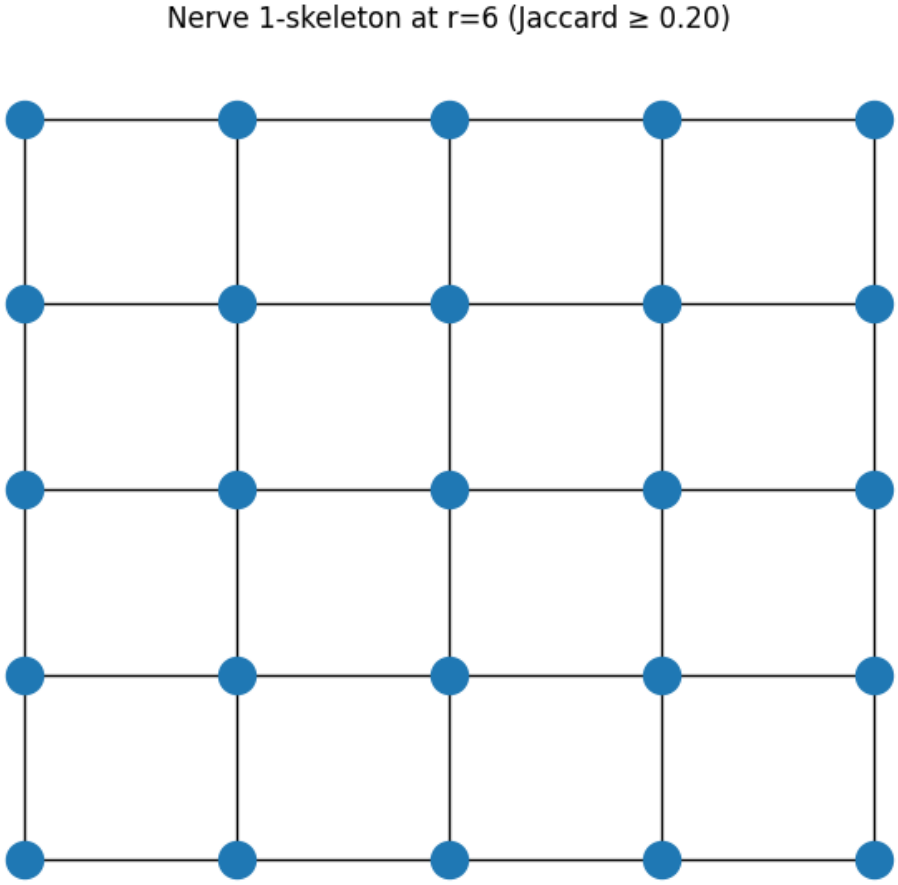

Figure 21: Nerve 1-skeleton at $r = 6$

C.10.3 H_0 persistence diagram

All components are born at $r = 0$. Most die early (you see a visible stack around $d \approx 3$); one bar persists to "∞" (plotted at $r_{max} + \Delta$, hence the dot near $y \approx 7$. Our system becomes fully connected by roughly $r_c \approx 3$. The nerve shown at $r = 6$ is well past that threshold, which is why it looks like a clean grid. The cover connects at a small radius, so overlap builds quickly with dilation; only a single global component remains thereafter (as expected). The early coalescence implies the aggregated path influence is not highly fragmented at this parameter setting.

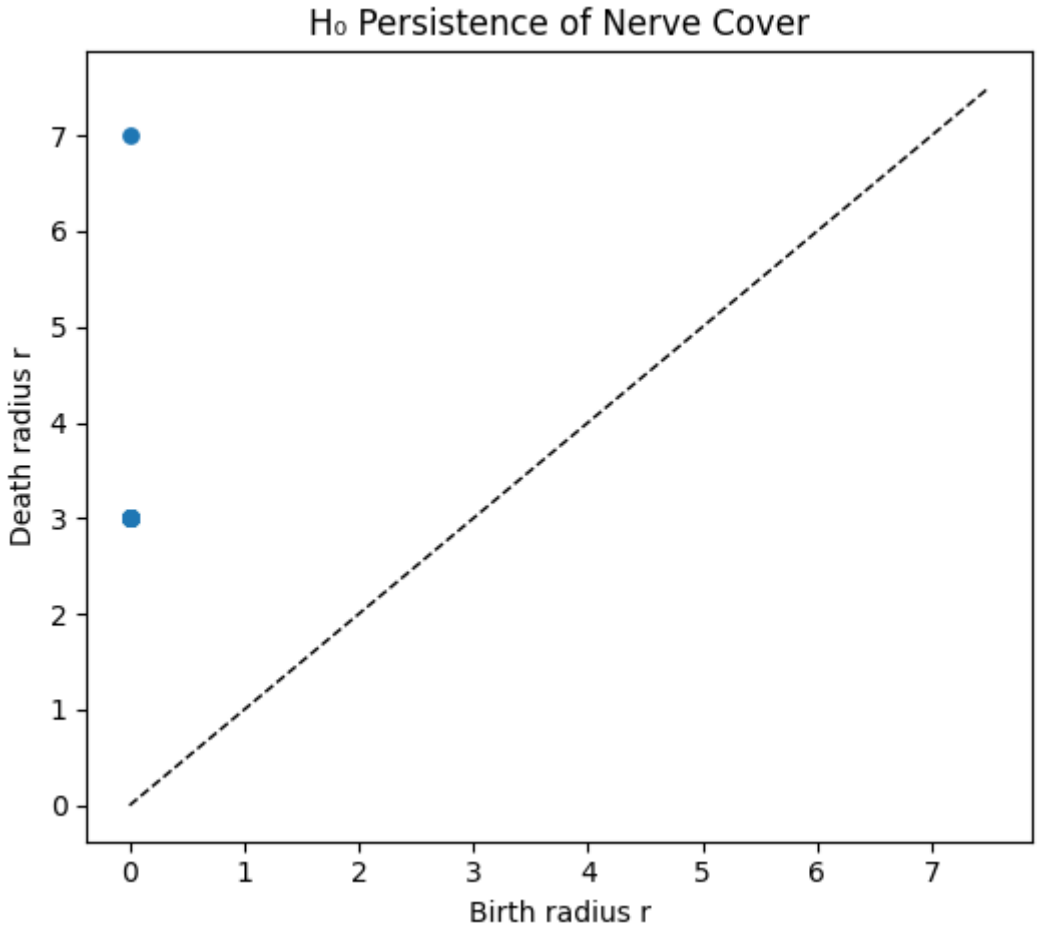

Figure 22: H_0 persistence diagram

The nerve skeleton and persistence diagram jointly suggest that the chosen Gabor supports plus Δ (grid spacing) produce a cover whose overlap geometry is nearly translation-invariant: local connections appear early and spread uniformly to yield full connectivity at moderate r. These conclusions are stable to small variations of τ and the opening radius, but they *do* shift with the Jaccard threshold θ and the grid spacing Δ (larger Δ postpones both the first merges and the global merge; larger θ sparsifies edges and increases lifetimes). Because the panels in the Minkowski sums were produced with no early orientation unions, adding deeper-layer unions or additional paths would broaden the supports and typically move death radii leftward (earlier merges).

References

[1] J. Serra. *Image Analysis and Mathematical Morphology.* Academic Press, 1982.

[2] S. Mallat. *A Wavelet Tour of Signal Processing.* Academic Press, 1999.

[3] K. Borsuk. On the imbedding of systems of compacta in simplicial complexes. *Fundamenta Mathematicae*, 35:217–234, 1948.

[4] R. Ghrist. *Elementary Applied Topology.* Createspace, 2014.

[5] H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological persistence and simplification. *FOCS*, 2000.

[6] A. Zomorodian and G. Carlsson. Computing persistent homology. *Discrete & Computational Geometry*, 33(2), 2005.

[7] D. Cohen-Steiner, H. Edelsbrunner, and J. Harer. Stability of persistence diagrams. *Discrete & Computational Geometry*, 37(1), 2007.

[8] F. Chazal, V. de Silva, M. Glisse, and S. Oudot. *The Structure and Stability of Persistence Modules*. Springer, 2016.