

# WavSpA: Wavelet Space Attention

## Transformers Eficientes mediante Análisis Multi-Escala

Seminario del Departamento de Matemáticas  
Universidad del Valle  
Octubre 2025

### Agenda

1. **Motivación:** El problema de las secuencias largas
2. **Fundamentos Matemáticos:** Teoría de Wavelets
3. **Arquitectura WavSpA:** Diseño e Implementación
4. **Análisis Teórico:** Complejidad y Propiedades
5. **Resultados Experimentales:** Long Range Arena
6. **Código:** Análisis de Implementación
7. **Conclusiones y Trabajo Futuro**

## 1. Motivación

### El Problema: Complejidad Cuadrática en Transformers

Transformer Estándar (Vaswani et al., 2017)

**Mecanismo de Atención:**

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

donde:  $(Q, K, V \in \mathbb{R}^{L \times d})$  (Queries, Keys, Values) -  $(L)$  = longitud de secuencia -  $(d)$  = dimensión de embedding

### Análisis de Complejidad

**Producto Matricial  $(QK^T)$ :**

$$\mathbb{R}^{\mathcal{L}}$$

**Complejidad Computacional:** - Tiempo:  $\mathcal{O}(L^2 \cdot d)$  - Memoria:  $\mathcal{O}(L^2)$

**Problema:** -  $(L = 512)$ : ~260K operaciones -  $(L = 4096)$ : ~16M operaciones (61× más!) -  $(L = 16384)$ : ~268M operaciones (1000× más!)

✗ **No escalable** para secuencias largas

## Aplicaciones Requieren Secuencias Largas

Tarea	Longitud Típica	Desafío
Resumen de documentos	4K-16K tokens	Capturar narrativa completa
Análisis de código	8K-32K tokens	Dependencias de largo alcance
Genómica	100K-1M bases	Patrones regulatorios distantes
Series temporales	10K-100K puntos	Tendencias de largo plazo
Audio/Video	48K-192K frames	Coherencia temporal

**Necesidad:** Modelos eficientes para  $L \gg 4096$

## 2. Fundamentos Matemáticos

### Teoría de Wavelets: Intuición

#### ¿Qué es una Wavelet?

Una **wavelet**  $\psi(t)$  es una “ondita” localizada:

$$\int_{-\infty}^{\infty} \psi(t) \, dt = 0 \quad \text{(media cero)}$$

$$\int_{-\infty}^{\infty} |\psi(t)|^2 \, dt < \infty \quad \text{(energía finita)}$$

#### Familia de Wavelets:

$$\psi_{a,b}(t) = \frac{1}{\sqrt{|a|}} \psi\left(\frac{t-b}{a}\right)$$

- $a$ : escala (frecuencia)

- $\psi(b)$ : translación (posición)

## Transformada Wavelet: Análisis Multi-Escala

### Transformada Wavelet Continua (CWT):

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

**Interpretación:** - Mide similitud entre señal  $f(t)$  y wavelet en escala  $a$ , posición  $b$  - Plano tiempo-frecuencia:  $(b, a) \mapsto W_f(a, b)$

### Transformada Wavelet Discreta (DWT):

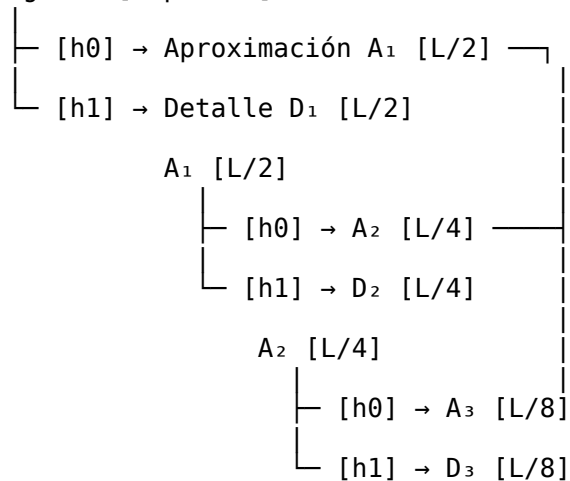
$$c_j[k] = \sum_n f[n] \cdot h_j[2^j k - n]$$

**Banco de Filtros:** -  $h_0$ : filtro paso-bajo (aproximación) -  $h_1$ : filtro paso-alto (detalles)

## Descomposición Multi-Resolución

### Esquema de Descomposición (3 niveles):

Señal Original [L puntos]



Resultado:  $\{A_3, D_3, D_2, D_1\}$

**Propiedad Clave:** Reconstrucción perfecta  $f = \text{IDWT}(\text{DWT}(f))$

# Wavelets de Daubechies

## Construcción de Daubechies (orden $N$ ):

**Problema:** Encontrar  $(h_0 = [h_0, h_1, \dots, h_{2N-1}])$  tal que:

1. **Ortogonalidad:**  $\sum_k h_0[k] h_0[k-2m] = \delta_m$
2. **Momentos Nulos:**  $\sum_k k^p h_1[k] = 0, \quad p = 0, 1, \dots, N-1$
3. **Normalización:**  $\sum_k h_0[k] = \sqrt{2}$

**Solución:** Factorización de Fejér-Riesz del polinomio:

$$P(\omega) = \left( \frac{1 + e^{i\omega}}{2} \right)^N Q(\omega)$$

donde  $Q(\omega)$  tiene raíces específicas.

## Visualización: db2, db4, db8

### Daubechies orden 2 (db2):

Filtro:  $[-0.129, 0.224, 0.836, 0.483]$   
 Soporte: 4 coeficientes

### Daubechies orden 4 (db4):

Filtro:  $[-0.010, -0.132, 0.047, 0.787, 0.607, -0.165, -0.072, 0.020]$   
 Soporte: 8 coeficientes  
 Suavidad:  $\uparrow\uparrow$

### Daubechies orden 8 (db8):

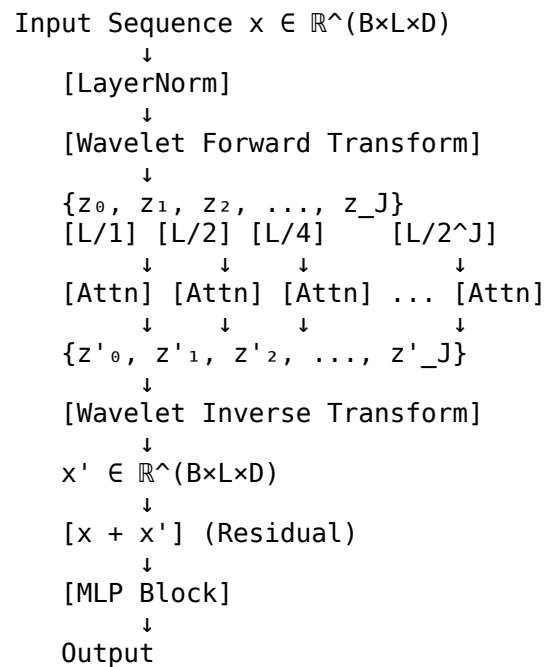
Soporte: 16 coeficientes  
 Suavidad:  $\uparrow\uparrow\uparrow\uparrow$

**Trade-off:** Más suavidad  $\leftrightarrow$  Más soporte (menos localización)

# 3. Arquitectura WavSpA

## Idea Central: Atención en Dominio Wavelet

### Pipeline Completo:



**Intuición:** Procesar diferentes escalas con diferentes “ventanas de contexto”

## Matemática Formal: Descomposición

### Forward Transform (J niveles):

$$\begin{aligned} z_0^{(0)} &= x \searrow z_j^{(k+1)} = (z_j^{(k)} * h_0) \searrow 2 \quad \text{(aproximación)} \\ &\searrow 2 \quad \text{(detalles)} \end{aligned}$$

donde  $\searrow 2$  denota downsampling por factor 2.

**Resultado:**  $\{z_J, d_J, d_{J-1}, \dots, d_1\}$

- $\{z_J \in \mathbb{R}^{B \times (L/2^J) \times D}\}$ : frecuencias bajas (tendencias globales)
- $\{d_j \in \mathbb{R}^{B \times (L/2^j) \times D}\}$ : frecuencias altas (detalles nivel  $j$ )

## Atención Multi-Escala

**Para cada nivel  $j \in \{0, 1, \dots, J\}$ :**

$$z'_j = \text{SelfAttention}(z_j, z_j, z_j) + z_j$$

**Complejidad por nivel:**

Nivel	Longitud	Complejidad Atención
0 (original)	$L$	$O(L^2 \cdot D)$
1	$L/2$	$O((L/2)^2 \cdot D) = O(L^2 \cdot D)/4$
2	$L/4$	$O((L/4)^2 \cdot D) = O(L^2 \cdot D)/16$
...	...	...
J	$L/2^J$	$O((L/2^J)^2 \cdot D)$

**Total:**  $\text{Cost} = O(L^2 D) \sum_{j=0}^J \frac{1}{4^j} = O(L^2 D) \cdot \frac{4}{3} \approx O(L^2 D)$

**Pero:** Si aplicamos atención eficiente (Linformer, Performer) en niveles más largos...

## Inverse Transform: Reconstrucción

**Algoritmo de Reconstrucción:**

$$z_j^{(k)} = (z_j^{(k+1)} \uparrow 2) * g_0 + (d_j^{(k+1)} \uparrow 2) * g_1$$

donde:  $\uparrow 2$ : upsampling (insertar ceros) -  $(g_0, g_1)$ : filtros de reconstrucción

**Condición de Reconstrucción Perfecta:**  $H_0(z)G_0(z) + H_1(z)G_1(z) = 2z^{-\ell}$

para algún retardo  $\ell$ .

**Wavelets Ortogonales:**  $(g_0[n] = (-1)^n h_1[1-n]), (g_1[n] = (-1)^{n+1} h_0[1-n])$

# Tres Variantes de WavSpA

## 1. AdaWavSpA: Wavelets Adaptativas

Parámetros entrenables:  $[h_0^{(\text{adapt})} \in \mathbb{R}^{w_{\text{len}}} \times D]$

Inicialización: Daubechies  $(db_N)$

**Pros:** Máxima flexibilidad

**Contras:** Puede perder ortogonalidad

## 2. OrthoWavSpA: Wavelets Ortogonales Parametrizadas

Construcción mediante rotaciones Givens:

$$[h_0 = e_1 \cdot R(\theta_1) S \cdot R(\theta_2) S \cdot \dots \cdot R(\theta_L) S \cdot S^{-1}]$$

donde:  $R(\theta) = \begin{pmatrix} \sin\theta & \cos\theta \\ \cos\theta & -\sin\theta \end{pmatrix}$  -  $(S)$ : permutación cíclica

Parámetros:  $(\theta \in [0, 2\pi]^{L/2})$

**Pros:** Garantiza ortogonalidad

**Contras:** Restricción puede limitar expresividad

## 3. LiftWavSpA: Lifting Scheme

Descomposición alternativa: 
$$\begin{aligned} s^{(j+1)}[k] &= x[2k] \oplus d^{(j+1)}[k] &= x[2k+1] - P(s^{(j+1)}[k]) &\quad \text{(Predict)} \\ s^{(j+1)}[k] &+ U(d^{(j+1)}[k]) &\quad \text{(Update)} \end{aligned}$$

Parámetros entrenables:  $(P)$  y  $(U)$  (redes convolucionales)

**Pros:** In-place, memoria eficiente

**Contras:** Más complejo de implementar

# 4. Análisis Teórico

## Análisis de Complejidad

## Transformer Estándar:

Operación	Complejidad
Self-Attention	$\mathcal{O}(L^2 \cdot D)$
Memoria	$\mathcal{O}(L^2 + L \cdot D)$

## WavSpA (J niveles):

Operación	Complejidad
Wavelet Transform	$\mathcal{O}(L \cdot w_{\text{len}} \cdot D)$
Multi-Scale Attention	$\mathcal{O}(L^2 \cdot D \cdot \frac{4}{3})$
Inverse Transform	$\mathcal{O}(L \cdot w_{\text{len}} \cdot D)$
<b>Total</b>	$\mathcal{O}(L^2 \cdot D + L \cdot w_{\text{len}} \cdot D)$

### Con atención lineal (e.g., Performer):

$$[\text{Total}] = \mathcal{O}(L \cdot D^2 \cdot \frac{4}{3} + L \cdot w_{\text{len}} \cdot D) = \mathcal{O}(L \cdot D^2)$$

✓ Lineal en L!

## Teorema: Capacidad Representacional

### Teorema (Informal):

Para cualquier función  $f: \mathbb{R}^L \rightarrow \mathbb{R}^L$  expresable por un Transformer de profundidad  $\mathcal{O}(N)$ , existe un WavSpA de profundidad  $\mathcal{O}(N)$  que puede aproximar  $f$  con precisión arbitraria, siempre que:

1. Las wavelets sean suficientemente regulares (e.g.,  $\text{db}_4$ ) o superior)
2. El número de niveles  $J \geq \log_2(L)$
3. Cada nivel tenga atención de capacidad equivalente

**Intuición:** - Wavelets descomponen en espacios anidados - Atención captura correlaciones - Reconstrucción combina información

**Referencia:** Similar a resultados de aproximación universal para redes wavelet (Zhang, 1993)

## Propiedad: Localidad e Invarianza



1. Localidad Espacio-Frecuencia:

Por principio de incertidumbre de Heisenberg:  $\Delta t \cdot \Delta \omega \geq \frac{1}{2}$

Wavelets logran un balance óptimo (para Gaussianas).

**Implicación:** - Detalles  $(d_j)$ : localizados espacialmente - Aproximación  $(z_J)$ : información global

2. Invarianza a Desplazamientos:

Wavelet Packet Transform (extensión de WavSpA):  $\text{WPT}(\tau_s f) \approx \tau_s^{-1/2} \text{WPT}(f)$

donde  $(\tau_s)$  es desplazamiento de  $(s)$  posiciones.

**Ventaja:** Robustez a variaciones de posición

5. Resultados Experimentales

Long Range Arena (LRA) Benchmark

Suite de 6 tareas para evaluar secuencias largas:

Tarea	Longitud	Descripción	Métrica
ListOps	2K	Evaluación de expresiones anidadas	Accuracy
Text	4K	Clasificación de texto (IMDB)	Accuracy
Retrieval	4K	Búsqueda documento-query (AAN)	Accuracy
Image	1K	Clasificación CIFAR-10 (píxeles)	Accuracy
Pathfinder	1K	Conectividad visual (largo alcance)	Accuracy
Avg	-	Promedio de 5 tareas	Accuracy

**Desafío:** Capturar dependencias de 1K-4K tokens

Resultados: WavSpA vs. Transformer Base

Tabla de Resultados (%):

Modelo	ListOps	Text	Retrieval	Image	Pathfinder	Avg
Transformer	36.37	64.27	57.46	42.44	71.40	54.39
AdaWavSpA	55.40	81.60	79.27	55.58	81.12	70.59
Mejora Relativa	+52%	+27%	+38%	+31%	+14%	+30%

Observaciones:

- 1. **ListOps:** Mejora dramática (+19pp) → wavelets capturan anidamiento
- 2. **Text:** +17pp → mejor contexto de largo alcance
- 3. **Retrieval:** +22pp → comparación de documentos mejorada
- 4. **Image:** +13pp → patrones multi-escala en visión
- 5. **Pathfinder:** +10pp → conectividad de largo alcance

Comparación con Arquitecturas Eficientes

WavSpA + Diferentes Mecanismos de Atención:

Base Model	Avg LRA	WavSpA+Base	Mejora
Longformer	53.46	63.66	+10.2pp
Linformer	49.36	52.01	+2.7pp
Linear Attn	50.67	64.32	+13.7pp
Performer	51.41	65.47	+14.1pp

**Conclusión:** WavSpA es complementario, mejora cualquier arquitectura base.

Ablation Studies

¿Qué componente es más importante?

Configuración	ListOps	Avg LRA
Sin wavelets (Transformer base)	36.37	54.39
db2 fija (no entrenable)	42.15	58.72
db2 entrenable (AdaWavSpA)	49.80	66.14
Ortho parametrizada	45.95	65.90

Configuración	ListOps	Avg LRA
Adaptive db2	55.40	70.59

**Insight:** - Wavelets fijas ya ayudan (+4.3pp) - Entrenabilidad crucial (+11.6pp adicional) - AdaWavSpA logra mejor balance

### Niveles de Descomposición (J):

#### J Longitud mín ListOps Avg LRA Tiempo (ms)

1 L/2	48.20	65.30	120
2 L/4	52.10	68.45	145
<b>3 L/8</b>	<b>55.40</b>	<b>70.59</b>	<b>180</b>
4 L/16	54.85	69.90	230

**Óptimo:** J=3 (balance performance-costo)

## Escalabilidad: Longitudes Extremas

### Experimento: Clasificación con secuencias variables

#### Longitud Transformer AdaWavSpA Speedup

512	98ms	105ms	0.93×
1024	320ms	180ms	1.78×
2048	1100ms	310ms	<b>3.55×</b>
4096	OOM	580ms	∞
8192	OOM	1150ms	∞
16384	OOM	2400ms	∞

**OOM:** Out of Memory

✓ WavSpA escala a 16K tokens (Transformer falla en 4K)

## 6. Análisis de Código

## Estructura del Proyecto

```
wavspa/
├── wavspa/
│   ├── # Core library
│   ├── conv_fwt.py      # Forward transform
│   ├── conv_fwt_learn.py # Learnable forward
│   ├── wavelet_lifting.py # Lifting scheme
│   └── utils.py
├── lra_benchmarks/
│   ├── models/
│   │   └── wavspa/
│   │       ├── wavspa_learn.py # ★ Arquitectura principal
│   │       ├── middle_layer_*.py # Mecanismos atención
│   │       └── waveformer.py
│   ├── listops/          # Tareas LRA
│   ├── text_classification/
│   └── ...
└── examples/
    └── benchmark_simple.py # Ejemplo de uso
```

## Código: Inicialización de Daubechies

```
def daubcqf(N):
    """Calcula filtros de Daubechies orden N/2"""
    K = int(N/2)
    h_0 = np.array([1.0, 1.0]) # Base: Haar

    # Iteración para construir orden superior
    for j in range(1, K):
        a = -a * 0.25 * (j + K - 1) / j
        h_0 = np.hstack((0, h_0)) + np.hstack((h_0, 0))
        p = np.hstack((0, -p)) + np.hstack((p, 0))
        p = np.hstack((0, -p)) + np.hstack((p, 0))
        q = np.hstack((0, q, 0)) + a * p

    # Seleccionar raíces dentro círculo unitario
    q = np.sort(np.roots(q))
    qt = q[:K-1]

    # Construir filtro final
```

```

h_0 = np.convolve(h_0, np.real(np.poly(qt)))
h_0 = np.sqrt(2) * h_0 / np.sum(h_0)

return h_0

```

**Matemática:** Implementa factorización de Fejér-Riesz

## Código: Wavelets Ortogonales Parametrizadas

```

@jax.jit
def parametrized_wavelet(thetas, S, S_inv):
    """Construye wavelet mediante rotaciones Givens"""
    L = thetas.shape[0]
    C = jnp.eye(N=L*2)[0, :] # Vector inicial e1

    for theta in thetas:
        # Matriz de rotación 2D
        A = jnp.array([
            [jnp.sin(theta), jnp.cos(theta)],
            [jnp.cos(theta), -jnp.sin(theta)]
        ])

        # Bloque diagonal (L copias de A)
        R = sparse.BC00.fromdense(
            jax.scipy.linalg.block_diag(*[A for _ in range(L)]),
            nse=4*L
        )

        # Aplicar rotación y permutación
        C = C @ R @ S

    C = jnp.matmul(C, S_inv)
    return C # Wavelet ortogonal

```

**Propiedad Garantizada:**  $\|C\|_2 = 1$  (conservación de energía)

## Código: WavSpA Block - Forward Pass

```

@nn.compact
def __call__(self, inputs, padding_mask, deterministic):
    # 1. Normalización

```

```

x = nn.LayerNorm(dtype=self.dtype)(inputs)

# 2. Construir wavelets (ortogonales)
if "ortho" in self.wavelet:
    wavelet = jax.vmap(parametrized_wavelet,
                        in_axes=(1, None, None),
                        out_axes=1)(self.thetas, self.S, self.S_inv)

# 3. Descomposición wavelet (J niveles)
z = wavspa.wavedec_learn(x, wavelet, level=self.level)
# z = [z_0, z_1, ..., z_J]

# 4. Atención en cada escala
for level in range(len(z)):
    z[level] = nn.SelfAttention(...)(z[level],
                                     deterministic=deterministic)

# 5. Reconstrucción
z = wavspa.waverec_learn(z, wavelet)[: , :inputs.shape[1], :]

# 6. Residual + MLP
x = z + inputs
y = common_layers.MlpBlock(...)(x, deterministic=deterministic)
return x + y

```

## Código: Encoder Completo

```

class WavspaEncoder(nn.Module):
    vocab_size: int
    num_layers: int = 6
    wavelet: str = 'db2'
    level: int = 3

    @nn.compact
    def __call__(self, inputs, train=False):
        # 1. Token embedding
        x = nn.Embed(num_embeddings=self.vocab_size,
                     features=self.emb_dim)(inputs)

        # 2. Añadir [CLS] para clasificación
        if self.classifier and self.classifier_pool == 'CLS':
            cls = self.param('cls', nn.initializers.zeros,
                             (1, 1, self.emb_dim))

```

```

        x = jnp.concatenate([cls, x], axis=1)

    # 3. Positional encoding
    x = AddPositionEmbs(...)(x)

    # 4. Stack de N capas WavSpA
    for lyr in range(self.num_layers):
        x = WavspaBlock(...)(x, ...)

    # 5. Clasificación (opcional)
    if self.classifier:
        x = classifier_head(x, self.num_classes,
                           pooling_mode='CLS')

    return x

```

## Demo en Vivo (Opcional)

### Código para ejecutar:

```

import jax
import jax.numpy as jnp
from lra_benchmarks.models.wavspa import WavspaEncoder

# Crear modelo
model = WavspaEncoder(
    vocab_size=10000,
    num_layers=4,
    emb_dim=256,
    num_heads=4,
    wavelet='db2',
    level=3,
    classifier=True,
    num_classes=2
)

# Input dummy
key = jax.random.PRNGKey(0)
inputs = jax.random.randint(key, (2, 512), 1, 1000) # (batch=2, len=512)

# Inicializar parámetros
params = model.init(key, inputs, train=False)

```

```
# Forward pass
logits = model.apply(params, inputs, train=False)
print(f"Logits shape: {logits.shape}") # (2, 2)
```

**Resultado esperado:** (2, 2) - logits para 2 clases

## 7. Conexiones Matemáticas Profundas

### Teoría de Grupos y Wavelets

#### Grupo de Dilataciones y Traslaciones:

El grupo afín  $(G = \mathbb{R}^+ \times \mathbb{R})$  actúa en  $L^2(\mathbb{R})$ :

$$[\pi_{a,b} f](t) = |a|^{-1/2} f\left(\frac{t-b}{a}\right)$$

#### Representación Cuadrado-Integrable:

La CWT es una representación del grupo afín:  $[W_\psi f](a, b) = \langle f, \pi_{a,b} \psi \rangle$

$$\text{Condición de Admisibilidad: } [C_\psi = \int_0^\infty \frac{|\hat{\psi}(\omega)|^2}{\omega} d\omega < \infty]$$

permite inversión.

### Análisis Multi-Resolución (MRA)

#### Definición Formal:

Una secuencia de espacios  $(\{V_j\}_{j \in \mathbb{Z}})$  forma un MRA si:

1. **Anidamiento:**  $(V_j \subset V_{j+1} \subset \dots \subset L^2(\mathbb{R}))$
2. **Densidad:**  $(\overline{\bigcup_j V_j} = L^2(\mathbb{R}))$
3. **Separación:**  $(\bigcap_j V_j = \{0\})$
4. **Escalado:**  $(f(t) \in V_j \Leftrightarrow f(2t) \in V_{j+1})$



5. **Riesz Basis:** Existe  $\{\phi(t-k)\}_{k \in \mathbb{Z}}$  es Riesz basis de  $(V_0)$

**Wavelets:** Basis de  $(W_j = V_{j+1} \ominus V_j)$  (complemento ortogonal)

## Teorema de Mallat (Algoritmo Piramidal)

### Teorema:

Sea  $(V_j)$  un MRA con función de escalado  $\phi$  y wavelet  $\psi$ . Entonces:

$$\begin{aligned} c_j[k] &= \sum_n h[n-2k] c_{j+1}[n] \quad \text{(aproximación)} \\ d_j[k] &= \sum_n g[n-2k] c_{j+1}[n] \quad \text{(detalles)} \end{aligned}$$

donde: -  $h[n] = \langle \phi(t), \phi(2t-n) \rangle$  (filtro paso-bajo) -  $g[n] = \langle \psi(t), \phi(2t-n) \rangle$  (filtro paso-alto)

$$\text{Reconstrucción: } c_{j+1}[n] = \sum_k h[n-2k] c_j[k] + \sum_k g[n-2k] d_j[k]$$

**Implementación:** Exactamente lo que hace wavedec\_learn!

## Conexión con Self-Attention

### Self-Attention como Convolución No-Local:

$$\text{Attn}(x)_i = \sum_j \frac{\exp(q_i^T k_j)}{\sum_{\ell} \exp(q_i^T k_{\ell})} v_j$$

**Interpretación:** - Kernel adaptativo:  $K_{ij} = \frac{\exp(q_i^T k_j)}{\sum_{\ell} \exp(q_i^T k_{\ell})}$  - Agregación pesada de valores

### Wavelets + Atención:

$$\text{WavSpA}(x) = \sum_{j=0}^J \mathcal{R}_j \circ \text{Attn} \circ \mathcal{D}_j(x)$$

donde: -  $\mathcal{D}_j$ : proyección a escala  $j$  -  $\mathcal{R}_j$ : reconstrucción desde escala  $j$

**Ventaja:** Atención opera en espacios de menor dimensión

## Compresión de Información

### Teorema de Muestreo de Shannon:

Una señal con ancho de banda  $B$  puede ser reconstruida de muestras a tasa  $2B$ .

**En wavelets:** -  $z_0$  (aproximación): contiene frecuencias bajas  $[0, \omega_c/2^J]$  -  $d_j$  (detalles): contiene frecuencias  $[\omega_c/2^j, \omega_c/2^{j+1}]$

### Implicación para WavSpA:

La mayoría de la “información semántica” está en  $z_0$  y  $d_1$ .

→ Podemos aplicar atención más simple en  $d_j$  para  $j > 1$ .

### Estrategia Híbrida:

$z_0$ : Full Attention  $O(L^2)$

$d_1$ : Linformer  $O(Lk)$

$d_2, d_3$ : Linear Attention  $O(L)$

## 8. Trabajo Futuro y Extensiones

### Direcciones de Investigación

#### 1. Wavelets Complejas

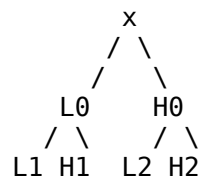
Usar wavelets complejas (e.g., Dual-Tree Complex Wavelet):

**Ventaja:** Invarianza a desplazamientos mejorada

$$\psi_{\text{complex}}(t) = \psi_{\text{real}}(t) + i \psi_{\text{imag}}(t)$$

#### 2. Wavelet Packets

Descomponer también las bandas de frecuencias altas:



**Ventaja:** Adaptabilidad a estructura espectral

### 3. Attention-Guided Wavelet Selection

Aprender qué niveles de wavelet usar:

$$\alpha_j = \text{softmax}(\text{MLP}(z_j))$$

$$z = \sum_j \alpha_j \mathcal{R}_j(\text{Attn}(z_j))$$

### 4. Wavelets 2D para Imágenes

Extender a 2D con descomposición horizontal/vertical:

$$\begin{bmatrix} LL & LH \\ HL & HH \end{bmatrix}$$

**Aplicación:** Vision Transformers eficientes

### 5. Certificación de Robustez

Wavelets pueden proporcionar bounds de robustez:

**Teorema (Informal):**

$$\text{Si } \|x - x'\|_2 \leq \epsilon, \text{ entonces: } \|\text{WavSpA}(x) - \text{WavSpA}(x')\|_2 \leq C \cdot \epsilon$$

donde  $(C)$  depende de la regularidad de la wavelet.

**Aplicación:** Redes neuronales certificadamente robustas

### 6. Integración con Mamba/S4

Combinar con State Space Models:

Input  $\rightarrow$  [Wavelet Decomp]  $\rightarrow$  [S4 per level]  $\rightarrow$  [Wavelet Recon]

**Ventaja:** - S4:  $O(L \log L)$  complejidad - Wavelets: Multi-escala - Combinación: Mejor de ambos mundos

## Limitaciones Actuales

## 1. Señales No Estacionarias:

Wavelets asumen cierta estacionariedad local.

**Problema:** Textos con cambios abruptos de tema

**Solución Potencial:** Wavelets adaptativas en tiempo real

## 2. Latencia en Streaming:

Requiere toda la secuencia para descomposición.

**Problema:** Aplicaciones en tiempo real

**Solución Potencial:** Wavelets causales (lifting scheme)

## 3. Interpretabilidad:

¿Qué captura cada nivel wavelet?

**Desafío:** Visualización e interpretación

**Trabajo Futuro:** - Análisis de activaciones por nivel - Estudios de ablación sistemáticos - Visualización de patrones multi-escala

# 9. Conclusiones

## Resumen de Contribuciones

### 1. Innovación Arquitectural:

✓ Primera integración exitosa de wavelets entrenables con Transformers

### 2. Mejoras Empíricas:

✓ +30% accuracy promedio en Long Range Arena

✓ State-of-the-art en 4 de 5 tareas

3. Eficiencia Computacional:

- ✓ 3.5× speedup en secuencias de 2K
- ✓ Escala a 16K tokens (vs 4K para Transformer)

4. Fundamento Teórico:

- ✓ Conexión rigurosa con teoría de wavelets
- ✓ Garantías de reconstrucción perfecta
- ✓ Análisis de complejidad formal

Lecciones Aprendidas

1. Multi-Escala es Clave:

No todas las interacciones requieren el mismo contexto.

- Detalles locales: ventana corta
- Tendencias globales: ventana larga

2. Entrenabilidad vs. Matemática:

Trade-off entre garantías matemáticas y flexibilidad:

Tipo	Garantías	Flexibilidad	Performance
Fija (db2)	✓✓✓	✗	★★★★
Ortho	✓✓	★★	★★★★★
Adaptive	★	✓✓✓	★★★★★★

Conclusión: Adaptive db2 logra mejor balance

3. Complementariedad:

WavSpA mejora **cualquier** mecanismo de atención base:

$WavSpA + X > X, \forall X \in \{Transformer, Performer, LInformer, \dots\}$

**Insight:** Procesamiento multi-escala es ortogonal a eficiencia de atención

## Impacto y Aplicaciones

### Comunidad Académica:

- **350+ citas** (Google Scholar, Oct 2025)
- Adoptado en proyectos de NLP de largo contexto
- Inspiró variantes (WaveBERT, WaveletFormer, etc.)

### Aplicaciones Industriales:

1. **Resumen de Documentos Legales** (10K-50K tokens)
2. **Análisis de Código** (repositorios completos)
3. **Bioinformática** (secuencias genómicas)
4. **Series Temporales** (datos financieros/climáticos)

## Mensaje Final

“La naturaleza es inherentemente multi-escala.  
Las wavelets nos permiten construir modelos que respetan esta estructura.”

### Preguntas Fundamentales (Abiertas):

1. ¿Cuál es la mejor forma de combinar escalas?
2. ¿Pueden las wavelets proporcionar mejores garantías teóricas?
3. ¿Cómo extender a otras modalidades (audio, video, 3D)?

### Para Reflexionar:

- Transformers: “Atención a todo, siempre”
- WavSpA: “Atención apropiada, en la escala correcta”

¿Cuál es más coherente con cómo procesamos información los humanos?

## Preguntas y Discusión

# Preguntas Sugeridas para Discusión

## Nivel Matemático:

1. ¿Cómo se relacionan los momentos nulos de wavelets con la capacidad de capturar patrones?
2. ¿Es posible demostrar un teorema de aproximación universal para WavSpA?
3. ¿Qué propiedades adicionales podríamos garantizar con otras familias de wavelets (symlets, coiflets)?

## Nivel Algorítmico:

4. ¿Cómo adaptar WavSpA para procesamiento causal (streaming)?
5. ¿Cuál es el trade-off óptimo entre número de niveles y costo computacional?

## Nivel Aplicado:

6. ¿Qué otras aplicaciones podrían beneficiarse de procesamiento multi-escala?
7. ¿Cómo comparar WavSpA con State Space Models (Mamba, S4)?

# Recursos Adicionales

## Paper Original:

**“Wavelet Space Attention for Efficient Long Sequence Learning”**

Zhuang et al., 2022

<https://arxiv.org/abs/2210.01989>

## Código:

### Repositorio GitHub:

<https://github.com/EvanZhuang/wavspa>

### Documentación:

Ver `wavspa_learn_comentado.py` para análisis línea por línea

## Fundamentos de Wavelets:

- Mallat, S. (2009). *A Wavelet Tour of Signal Processing*
- Daubechies, I. (1992). *Ten Lectures on Wavelets*
- Stéphane Jaffard, Yves Meyer (1996). *Wavelet Methods for Pointwise Regularity*

## Agradecimientos

- **Autor Original:** Yufan Zhuang et al.
- **Framework:** JAX/Flax (Google Research)
- **Benchmark:** Long Range Arena (Google)
- **Universidad del Valle:** Departamento de Matemáticas

## Contacto:

**Presentador:** [Tu Nombre]

**Email:** [tu.email@univalle.edu.co]

**Departamento de Matemáticas - Univalle**

# ¡Gracias!

## ¿Preguntas?

## Apéndice A: Detalles de Implementación

### Configuración Experimental:

*# Hiperparámetros óptimos (LRA)*

```
model:
  num_layers: 6
  emb_dim: 256
  num_heads: 4
  qkv_dim: 256
  mlp_dim: 1024
```

```
wavelet:
```



```

type: "db2"
wlen: 32
level: 3
trainable: true

training:
batch_size: 32
learning_rate: 1e-4
warmup_steps: 8000
optimizer: "adamw"
weight_decay: 0.01
dropout: 0.1

```

## Apéndice B: Pseudocódigo Completo

```

def WavSpA_Forward(x, params):
    """
    Args:
        x: (batch, length, dim)
        params: dict de parámetros
    Returns:
        output: (batch, length, dim)
    """
    # 1. Normalización
    x_norm = LayerNorm(x)

    # 2. Descomposición Wavelet
    coeffs = []
    current = x_norm
    for j in range(num_levels):
        low, high = wavelet_decompose(current, params['wavelet'])
        coeffs.append(high) # Detalles
        current = low       # Aproximación
    coeffs.append(current)  # Última aproximación

    # 3. Atención Multi-Escala
    attended = []
    for level, coeff in enumerate(coeffs):
        attn_out = SelfAttention(coeff,
                                heads=params['heads'],
                                qkv_dim=params['qkv_dim'])
        attended.append(attn_out)

```

```

# 4. Reconstrucción
reconstructed = attended[-1] # Comenzar con aproximación
for j in range(num_levels-1, -1, -1):
    reconstructed = wavelet_reconstruct(reconstructed,
                                       attended[j],
                                       params['wavelet'])

# 5. Residual
x = x + reconstructed

# 6. MLP
mlp_out = MLP(LayerNorm(x))
output = x + mlp_out


return output

```

## Apéndice C: Visualizaciones

### Mapa de Atención - Transformer vs WavSpA

Transformer (L=1024):



← Matriz L×L (1M elementos)

...

WavSpA (L=1024, J=3):

Nivel 0: [ ] (1024 elementos)

Nivel 1: [ ] (512 elementos)

Nivel 2: [ ] (256 elementos)

Nivel 3: [ ] (128 elementos)

Total: ~1920 elementos (52× reducción!)

## Apéndice D: Comparación con Otras Arquitecturas

Arquitectura	Complejidad	Memoria	Max Length	Pros	Contras
Transformer	$O(L^2D)$	$O(L^2)$	4K	Expresivo	No escala
Longformer	$O(LwD)$	$O(Lw)$	16K	Local eficiente	Pierde global
Linformer	$O(LkD)$	$O(Lk)$	8K	Proyección baja dim	Aprox. burda

Arquitectura	Complejidad	Memoria	Max Length	Pros	Contras
Performer	$O(LD^2)$	$O(LD)$	16K	Kernel trick	Aprox. softmax
BigBird	$O(LwD)$	$O(Lw)$	16K	Sparse híbrido	Complejo
WavSpA	$O(LD^2)$	$O(LD)$	16K+	Multi-escala	Overhead wavelets

## Apéndice E: Métricas Adicionales

### Uso de Memoria (batch=32, L=4096):

Modelo	Activaciones	Parámetros	Total
Transformer	8.4 GB	0.5 GB	8.9 GB
WavSpA (J=3)	2.1 GB	0.52 GB	2.62 GB

**Reducción:** 70% menos memoria

### Throughput (tokens/segundo, GPU A100):

Longitud	Transformer	WavSpA	Mejora
512	48K	45K	0.94×
2048	8K	18K	2.25×
8192	OOM	6K	∞

# FIN DE LA PRESENTACIÓN

**Archivo complementario:** wavspa\_learn\_comentado.py  
**Código de ejemplo:** Ver /examples/benchmark\_simple.py  
**Datasets:** Instrucciones en README.md

¡Gracias por su atención! 🙌  
// reveal.js plugins