Especificación de Interfaz de Entrada/Salida (I/O) - Sistema de Predictores Universales

Arquitectura de Sistemas

February 18, 2026

Contents

1 Resumen Ejecutivo

Este documento define la interfaz abstracta de Entrada/Salida (I/O) para el Sistema de Predictores Universales, independiente de su implementación específica (Python/JAX, C++, Rust, FPGA). Describe los vectores de configuración necesarios para instanciar el sistema, el flujo de datos en tiempo de ejecución y la estructura de las señales de salida y telemetría.

2 Vector de Configuración (Hyper-Inputs)

El sistema se inicializa mediante un vector de configuración Λ que define la topología y sensibilidad de los módulos. Estos parámetros suelen ser estáticos durante la sesión de operación o calibrados por un meta-optimizador externo.

Parámetro	**Símbolo**	**Descripción Funcional**
Regularización Entrópica	ϵ	Suavizado del transporte de masa en el Orquestador JKO (Sinkhorn).
Tasa de Aprendizaje	τ	Velocidad de adaptación de los pesos ρ ante gradientes de energía.
Profundidad de Firma	L	Grado de truncueo de la Log-Signature (Kernel D - Topológico).
Memoria WTMM	N_{buf}	Tamaño de la ventana deslizante para estimar singularidades.
Cono de Besov	C_{besov}	Radio de influencia para el seguimiento de máximos en ondelettes.
Umbral de Hölder	H_{min}	Valor crítico de regularidad que activa el *Circuit Breaker*.
Umbral CUSUM	h	Nivel de desviación acumulada que dispara el reinicio de pesos.
Slack CUSUM	k	Tolerancia a la deriva (”ruido blanco”) permitida sin acumular error.
Memoria de Volatilidad	α	Tasa de decaimiento (EMA) para estimar la varianza del error.

Table 1: Vector de Hiperparámetros Λ

3 Flujo de Entrada (Data Injection)

3.1 1. Fase de Calibración (Bootstrapping)

Estado inicial requerido antes de la operación secuencial.

Entrada: Historia $\mathcal{H} = \{y_{-T}, \dots, y_0\}$

- **Estructura:** Serie temporal de vectores $\mathbf{y} \in \mathbb{R}^d$ o escalares $y \in \mathbb{R}$.
- **Propósito:**
  - Inicialización de núcleos dependientes de historia (ej. parámetros de Lévy).
  - Estabilización de los pesos iniciales ρ_0 del orquestador.
  - Llenado del búfer de singularidad para el módulo WTMM.

3.2 2. Fase Operativa (Online Stream)

Ciclo de actualización paso a paso en tiempo t.

Entrada del Paso t: Tupla $(y_t, y_{target}, \tau_{epoch})$

- **Marca de Tiempo (τ_{epoch}):** Timestamp absoluto (Unix Nanoseconds). Obligatorio para sincronización y cálculo de latencia en la política de abandono.
- **Observación Actual (y_t):**
  - El nuevo dato disponible en t.
  - Se utiliza para alimentar los núcleos (K_A, K_B, K_C, K_D) y generar las predicciones para $t+1$.
  - **Gestión de Errores de Dominio:** Si $|y_t| > 20\sigma$ (respecto a la normalización histórica), el dato se clasifica como *Outlier Catastrófico*. El sistema debe descartar la entrada, mantener el estado inercial y emitir una alerta crítica de validación, protegiendo los núcleos de divergencia numérica.

- **Detección de Señales Congeladas (Frozen Signal Validation):** Si el stream inyecta exactamente el mismo valor durante $N_{freeze} \geq 5$ pasos consecutivos, el sistema debe:
  1. Calcular la varianza de los últimos N_{freeze} valores: $\text{Var}([y_{t-4}, y_{t-3}, y_{t-2}, y_{t-1}, y_t]) = 0$
  2. Identificar esto como falla de sensor o corrupción de fuente de datos
  3. Emitir `FrozenSignalAlarmEvent` con timestamp del evento
  4. **Impacto matemático:** El exponente de Hölder en Rama D requiere variabilidad: $H_t = \lim_{s \to 0} \frac{\log |\gamma(t+s) - \gamma(t)|}{\log s}$. Con señal congelada, el numerador es cero, causando singularidades o indeterminaciones $(0/\log s)$. Esto invalida el espectro multifractal. El sistema debe:
     * Congelar la Rama Topológica (Kernel D) en último valor válido
     * NO actualizar pesos del orquestador (mantener inercia)
     * Activar modo de inferencia degradada
     * Continuar predicciones usando Ramas A, B, C solamente
  5. **Recuperación:** Una vez que se detecte varianza $> 0.1 \times \text{Var}_{histrica}$ durante 2 pasos consecutivos, liberar el bloqueo de Rama D y reanudar normalidad.

- **Malla de Inferencia (Anti-Aliasing Input):**
  - **Frecuencia de Muestreo vs Escalas (N_{buf}):** Para garantizar la estabilidad del análisis WTMM (Singularidades del Kernel D), la frecuencia de inyección de datos debe mantener una densidad suficiente respecto a las escalas wavelet más finas.
  - *Restricción:* Se impone una **Frecuencia Mínima de Inyección** (Nyquist soft-limit) dependiente de C_{besov}. Si la densidad de eventos cae por debajo de este umbral, el espectro multifractal colapsará, y el sistema debe congelar la actualización de la rama topológica.

- **Objetivo de Validación (y_{target}):**
  - El valor "real" correspondiente a la predicción generada en el paso anterior $(t-1)$.
  - Generalmente $y_{target} \equiv y_t$ (en predicción one-step-ahead causal).
  - Se utiliza para calcular el error $e_t = y_{target} - \hat{y}_{t|t-1}$ y calcular el gradiente de energía ∇E para el transporte JKO.

- **Métrica de Abandono (Staleness Policy):**
  - **Tiempo de Vida (TTL):** Parámetro Δ_{max}.
  - **Comportamiento ante Violación:** Si el retraso de y_{target} excede Δ_{max}, la actualización JKO se cancela.
  - **Señal de Integridad:** El sistema debe emitir una bandera persistente de *Inferencia Degradada* ("Stale Weights"). Esto alerta al ejecutor de que, aunque la predicción $\hat{y}$ sigue generándose, los pesos ρ tienen inercia obsoleta y el riesgo ya no está siendo optimizado geométricamente.

4 Políticas de Seguridad en la Capa de I/O (Credenciales)

El diseño del Predictor Estocástico exige operaciones de ingesta de alta frecuencia contra infraestructuras de mercado externas (ej. WebSockets institucionales, brokers o APIs REST). El acceso a estas infraestructuras introduce vectores de vulnerabilidad crítica.

Inyección de Entorno Segura

Queda categóricamente prohibido el *hardcoding* (codificación en duro) de tokens, claves API, secretos de bases de datos o credenciales de conexión en cualquier módulo del código fuente futuro (ej. `io/streams.py`). Toda implementación **debe** aplicar el patrón de Inyección de Entorno, leyendo estas credenciales en tiempo de ejecución desde variables del sistema operativo o archivos locales `.env`.

Exclusión de Control de Versiones

El repositorio de implementación resultante debe incluir reglas explícitas de exclusión en su sistema de control de versiones (ej. forzar `*.env` en el archivo `.gitignore`) para garantizar que ningún secreto sea expuesto inadvertidamente.

5 Salidas del Sistema (System Outputs)

5.1 1. Señal de Control (Prediction Signal)

La salida primaria para la toma de decisiones.
Salida: $\hat{y}_{t+1}$

- **Descripción:** Estimación del valor esperado del proceso para el siguiente instante.
- **Malla de Inferencia (Output Quantization):**
  - La salida $\hat{y}_{t+1}$ se entrega en el espacio normalizado (Z-Score) consistente con la entrada y_t.
  - El Actor/Ejecutor es responsable de aplicar la transformación inversa (des-normalización) utilizando las estadísticas de ventana rodante si se requiere un precio absoluto.
- **Composición:** Combinación convexa de los núcleos base: $\hat{y}_{t+1} = \sum_{i \in \{A,B,C,D\}} \rho_i^{(t)} \cdot K_i(y_t)$.

5.2 2. Indicadores de Estado (State Telemetry)

Variables latentes que describen la "salud" y el régimen del mercado.

- **Estado de Riesgo ($\mathbb{S}_{risk}$):**
  - **Exponente de Hölder local (H_t):** Medida de regularidad puntual. $H_t < 0.5$ indica antipersistencia/ruido; $H_t < H_{min}$ indica crash/shock inminente.
  - **Curtosis Empírica (κ_t):** Cuarto momento estandarizado de los residuos de predicción sobre ventana móvil:
    $$\kappa_t = \frac{E[(e_t - \mu_e)^4]}{(\sigma_e)^4}$$
    donde $e_t = y_{target} - \hat{y}_{t|t-1}$ son los residuos de predicción.
    - **Propósito:** Validar la idoneidad del umbral CUSUM adaptativo. Valores $\kappa_t > 3$ indican distribución leptocúrtica (colas pesadas), lo cual activa el ajuste logarítmico del umbral $h_t = k \cdot \sigma \cdot (1 + \ln(\kappa_t/3))$.
    - **Interpretación:** $\kappa_t \approx 3$ (Gaussiano), $\kappa_t \in [5, 10]$ (régimen de volatilidad financiera estándar), $\kappa_t > 15$ (régimen de crisis con eventos extremos frecuentes).
    - **Alerta:** Si $\kappa_t > 20$ de forma persistente, emitir advertencia de posible falla en el modelo de residuos o presencia de outliers sistemáticos no detectados.
  - **Entropía del Predictor DGM (H_{DGM}):** Entropía diferencial de la distribución de valores de la función valor neuronal $V_\theta(t, x)$ sobre el dominio espacial:
    $$H_{DGM} = -\int p_V(v) \log p_V(v)\, dv$$
    donde $p_V(v)$ es la densidad empírica de valores de V_θ evaluados en una malla de puntos del dominio.
    - **Propósito:** Monitorizar la salud de la Rama B (solución HJB mediante Deep Galerkin Method) y detectar colapso de modo donde la red neuronal predice una solución constante o degenerada.
    - **Umbral de Colapso:** Se debe comparar contra la entropía de la condición terminal: $H_{DGM} \geq \gamma \cdot H[g]$ con $\gamma \in [0.5, 1.0]$. Si la desigualdad se viola persistentemente (más de 10 pasos consecutivos), emitir `ModeDegradationAlert`.
    - **Acción Correctiva:** Reducir el peso de la Rama B en el orquestador ($\rho_B \to 0$) y priorizar ramas alternativas hasta que se re-entrene la red DGM o se reinicialicen sus pesos.
    - **Nota:** Este indicador solo es relevante si la Rama B está activa ($\rho_B > 0.05$). Para sistemas que no utilizan DGM, este campo puede omitirse o reportarse como `NaN`.
  - **Estadístico CUSUM (G^+):** Nivel de desajuste estructural acumulado.
  - **Distancia al Colapso ($h_t - G^+$):** Margen de seguridad restante antes de un reinicio forzado del modelo. Nota: h_t es ahora dinámico y depende de σ_t y κ_t.

  - **Energía Libre Residual ($\mathcal{F}$):** Valor instantáneo del funcional de JKO. Monitoriza si el modelo está "atrapado" en un mínimo local estable o si la regularización entrópica ϵ es demasiado alta, diluyendo excesivamente la capacidad predictiva del transporte de masa.

- **Estado del Orquestador (ρ):**
  - **Vector de Pesos:** $[\rho_A, \rho_B, \rho_C, \rho_D]$ tal que $\sum \rho = 1$.
  - Indica qué "física" domina actualmente el mercado (Saltos vs Difusión vs Memoria vs Topología).

- **Health-Check Estocástico:**
  - **Convergencia Sinkhorn (Bool):** Indica si el algoritmo de transporte de masa convergió dentro del número máximo de iteraciones.
  - *True:* Distancia Wasserstein exacta. *False:* Aproximación sub-óptima (alerta de precisión numérica).

- **Flags de Operación (Modo y Circuit Breakers):**
  - **Modo de Operación Base:**
    * *Standard (MSE):* Operación normal bajo supuestos Gaussianos locales.
    * *Robust (Huber):* Operación defensiva activada por singularidades ($H_t < H_{min}$) o alta volatilidad.
  - **Modo de Inferencia Degradada (Degraded Inference Mode):** Flag booleano crítico para monitoreo de calidad temporal:
    * **Condición de Activación:** Se activa cuando el Time-To-Live (TTL) de la señal y_{target} excede el umbral máximo permitido:

$$\text{TTL}(y_{target}) = t_{current} - t_{signal} > \Delta_{max}$$

    * **Implicaciones Operacionales:**
      1. La actualización del transporte JKO se suspende inmediatamente
      2. Los pesos ρ se congelan en su último valor válido (modo inercial)
      3. Las predicciones $\hat{y}_{t+1}$ continúan generándose, pero son sub-óptimas pues no reflejan el estado real del mercado
      4. El riesgo ya NO está siendo optimizado geométricamente
    * **Señalización al Ejecutor:** Este flag debe alertar explícitamente que:
      · Las predicciones actuales tienen *confianza degradada*
      · Los pesos son obsoletos (stale weights)
      · Se recomienda reducir exposición o operar en modo conservador
      · El sistema opera en "modo supervivencia" hasta que se restablezca el flujo de datos frescos
    * **Recuperación:** El flag se desactiva automáticamente cuando se recibe una señal fresca con $\text{TTL}(y_{target}) < 0.8 \cdot \Delta_{max}$ (umbral con histéresis para evitar oscilaciones). En ese momento se reanuda el transporte JKO y se emite `NormalOperationRestoredEvent`.
  - **Emergency Mode (Singularity Fallback):** Flag que indica si se activó el modo de emergencia por singularidad crítica ($H_t < H_{min}$), forzando $w_D \to 1.0$ y cambiando a métrica de Huber.
  - **Regime Change Detected:** Flag que indica si CUSUM detectó un cambio de régimen en el último paso, con reinicio de entropía a distribución uniforme.

6 Diagrama de I/O Abstracto

$$\boxed{\text{Entorno}} \xrightarrow{(y_t, y_{target})} \boxed{\text{Predictor Universal}(\Lambda)} \xrightarrow{(\hat{y}_{t+1}, \mathbb{S}_{risk}, \rho)} \boxed{\text{Actor / Ejecutor}}$$

6.1 Ciclo de Proceso Interno

1. **Ingesta:** Recibir y_t. Actualizar historial local.
2. **Análisis de Singularidad:** Calcular H_t usando WTMM sobre ventana reciente.
3. **Control de Calidad (CUSUM):**
   - Calcular error e_t usando y_{target} y la predicción almacenada $\hat{y}_{t|t-1}$.
   - Actualizar acumulador de deriva G^+.
   - Si $G^+ > h$ o $H_t < H_{min} \rightarrow$ Emitir señal de reinicio/alerta.
4. **Transporte (JKO):**
   - Calcular gradiente de energía ∇E basado en e_t.
   - Transportar masa de probabilidad $\rho_{t-1} \rightarrow \rho_t$ (Sinkhorn).
5. **Proyección:**
   - Ejecutar núcleos $K_i(y_t)$ para obtener componentes.
   - Agregar componentes usando nuevos pesos ρ_t para obtener $\hat{y}_{t+1}$.

7 Persistencia (Snapshotting)

Para garantizar la continuidad operativa, el sistema debe ser capaz de serializar su estado interno completo Σ_t en cualquier instante t.

$$\Sigma_t = \{\rho_t, G_t^+, \sigma_{ema}^2, \kappa_t, H_{DGM}, \text{Flags}, \text{Búfer}_{WTMM}, \text{KernelsState}\}$$

donde:

- κ_t: Curtosis empírica móvil de los errores de predicción (window size = 252).
- H_{DGM}: Entropía diferencial del predictor DGM (para detección de mode collapse).
- `Flags`: Estructura de flags booleanos que incluye `DegradedInferenceMode`, `EmergencyMode`, y `RegimeChangeDetected`.

La estructura `KernelsState` se debe segmentar en sub-bloques independientes (K-Blocks) para permitir actualizaciones modulares o parciales:

$$\text{KernelsState} = \{S_A(\text{Lévy}), S_B(\text{PDE}), S_C(\text{Memoria}), S_D(\text{Topología})\}$$

La operación de restauración $Load(\Sigma_t)$ debe permitir reanudar el flujo en $t+1$ sin necesidad de re-calibración sobre la historia $\mathcal{H}$. La correcta restauración de κ_t y H_{DGM} es crítica para preservar la capacidad de detección de anomalías y mode collapse tras un reinicio.

7.1 Protocolo de Snapshotting Atómico y Verificado

Se exige el uso de formatos de serialización binaria (ej. Protocol Buffers, MessagePack) en lugar de texto (JSON/XML) para el almacenamiento de Σ_t. Esto minimiza la latencia de I/O en operaciones de "Hot-Start" críticas.

- **Integridad Obligatoria (Mandatory Checksum):** Dado que se utilizan formatos binarios densos, un error de un solo bit en el estado de las matrices de los núcleos o el búfer WTMM podría provocar un colapso del sistema o comportamiento indefinido. Por tanto, la estructura Σ_t debe incluir un hash de validación robusto (ej. SHA-256 o CRC32c) al final del bloque.
- **Validación Pre-Inyección:** La rutina de restauración $Load(\Sigma_t)$ debe recalcular y validar este hash *antes* de inyectar el estado en la memoria operativa. Si la validación falla, el snapshot debe descartarse y el sistema debe reiniciarse en modo "Cold-Start" (recarga de historia).