

# Project JANUS

A Neuromorphic Architecture for Autonomous Trading

## Technical Specification & Theoretical Foundations

*Brain-Inspired Multi-Region Architecture Integrating Neuro-Symbolic Reasoning, Complementary Learning Systems, and Engineered Heterogeneity*

### Unified Documentation

This document consolidates all theoretical foundations and technical specifications of Project JANUS:

1. **Theoretical Foundations** — Crowding resistance, neuroscience framework, and dual-process architecture
2. **Forward Service** — Real-time decision-making and execution
3. **Backward Service** — Memory consolidation and learning
4. **Neuromorphic Architecture** — Brain-region mapping
5. **Rust Implementation** — Production deployment guide
6. **Limitations & Open Problems** — Honest engagement with challenges
7. **Validation Framework** — Experimental design and benchmarks

### Author

Jordan Smith

### Date

February 22, 2026

*"The god of beginnings and transitions, looking simultaneously to the future and the past."*

## Abstract

Project JANUS is a neuromorphic trading architecture that maps ten functionally distinct brain regions—basal ganglia, hippocampus, cerebellum, thalamus, hypothalamus, prefrontal cortex, amygdala, visual cortex, neocortex, and central nervous system—to specialized trading subsystems within a unified autonomous agent. Grounded in the Doya–Hassabis framework of neuroscience-inspired AI (Doya, 1999; Hassabis et al., 2017), JANUS implements a dual-process architecture separating real-time perception and action (System 1, Forward Service) from offline memory consolidation and schema formation (System 2, Backward Service), following Complementary Learning Systems theory (James L McClelland, Bruce L McNaughton, and O'Reilly, 1995; Kumaran, Hassabis, and James L. McClelland, 2016).

A central design objective is *crowding resistance*: the mitigation of co-impact costs (Bucci, Mastromatteo, et al., 2020) and strategy crowding risk (Khandani and Lo, 2011; Stein, 2009) through engineered heterogeneity across deployments. Each JANUS instance is parameterized to produce idiosyncratic order flow, reducing cross-instance correlation and increasing effective market capacity (DeMiguel, Martin-Utrera, and Uppal, 2021; Wagner, 2011). The system integrates neuro-symbolic reasoning via Logic Tensor Networks (Badreddine et al., 2022) for differentiable constraint satisfaction, video vision transformers for visual pattern recognition (Arnab et al., 2021), Opponent Actor Learning for basal ganglia decision-making (Collins and Michael J Frank, 2014; Jaskir and Michael J. Frank, 2023), and a three-timescale memory hierarchy inspired by hippocampal sharp-wave ripple replay (M. A. Wilson and Bruce L. McNaughton, 1994; Masset et al., 2025).

No prior published system maps multiple brain regions to distinct trading subsystems. This paper presents the theoretical foundations, mathematical specifications, neuromorphic architecture, and Rust implementation of JANUS, along with a candid assessment of limitations and a proposed validation framework.

**Keywords:** neuromorphic computing, algorithmic trading, neuro-symbolic AI, strategy crowding, co-impact, complementary learning systems, reinforcement learning, Rust

# Contents

<b>Abstract</b>	<b>2</b>
<b>I Main Architecture</b>	<b>8</b>
I The Epistemological Transition to Quant 4.0	8
II The Doya–Hassabis Framework: Neuroscience-Inspired Architecture	9
II.1 Functional Brain-Region Decomposition . . . . .	9
II.2 From Neuroscience to Architecture . . . . .	9
II.3 Complementary Learning Systems . . . . .	10
III The Dual-Process Architecture	10
<b>IV Resilience to Strategy Crowding and Co-Impact</b>	<b>11</b>
IV.1 Co-Impact and Algorithmic Herding . . . . .	11
IV.2 Vulnerability in Current Systems . . . . .	12
IV.3 JANUS's Defense: Engineered Heterogeneity . . . . .	12
<b>II Forward Service (Janus Bifrons)</b>	<b>13</b>
<b>1 Visual Pattern Recognition: DiffGAF and Vision Models</b>	<b>14</b>
1.1 Mathematical Foundation: Gramian Angular Fields . . . . .	14
1.1.1 Input Preprocessing . . . . .	14
1.1.2 Step 1: Normalization . . . . .	14
1.1.3 Step 2: Polar Coordinate Transformation . . . . .	15
1.1.4 Step 3: Gramian Field Generation . . . . .	15
1.1.5 Differentiable GAF (DiffGAF) . . . . .	15
1.2 3D Spatiotemporal Manifolds: GAF Video . . . . .	15
1.2.1 Sliding Window GAF Video Generation . . . . .	15
1.3 Vision Model Architecture . . . . .	16
1.3.1 DiffGAF-LSTM . . . . .	16
1.3.2 Video Vision Transformer (ViViT) . . . . .	16
<b>2 Ensemble Regime Detection</b>	<b>17</b>
2.1 Detection Methods . . . . .	17
2.1.1 Hidden Markov Model (HMM) . . . . .	17
2.1.2 Statistical Methods . . . . .	17

2.1.3	Technical Methods . . . . .	17
2.2	Regime Categories . . . . .	17
<b>3</b>	<b>Logic Tensor Networks: Symbolic Reasoning Engine</b>	<b>18</b>
3.1	Mathematical Foundation . . . . .	18
3.1.1	Grounding Function . . . . .	18
3.1.2	Predicate Grounding . . . . .	18
3.2	Łukasiewicz T-Norm Operations . . . . .	19
3.2.1	Conjunction (AND) . . . . .	19
3.2.2	Disjunction (OR) . . . . .	19
3.2.3	Negation (NOT) . . . . .	19
3.2.4	Implication (IF-THEN) . . . . .	19
3.2.5	Bi-Implication (Equivalence) . . . . .	19
3.2.6	Linguistic Hedges . . . . .	20
3.3	Knowledge Base Formulation . . . . .	20
3.3.1	Wash Sale Constraint . . . . .	20
3.3.2	Almgren-Chriss Risk Constraint . . . . .	20
3.3.3	Additional Constraint Categories . . . . .	20
3.4	Logical Loss Function . . . . .	21
3.4.1	Satisfiability Aggregation . . . . .	21
3.4.2	Logical Loss . . . . .	21
<b>4</b>	<b>Multimodal Fusion: Specialized Gated Attention</b>	<b>21</b>
4.1	Input Modalities . . . . .	21
4.2	Gated Cross-Attention Mechanism . . . . .	22
4.2.1	Attention Computation . . . . .	22
4.2.2	Gating Mechanism . . . . .	22
<b>5</b>	<b>Decision Engine: Basal Ganglia Pathways</b>	<b>22</b>
5.1	Praxeological Motor: Dual Pathways . . . . .	22
5.1.1	Direct Pathway (Go Signal) . . . . .	22
5.1.2	Indirect Pathway (No-Go Signal) . . . . .	23
5.2	Action Selection . . . . .	23
5.3	Brain Wiring Pipeline . . . . .	23
5.4	Cerebellar Forward Model (Wolpert, Miall, and Kawato, 1998; Sokolov, Miall, and Ivry, 2017) . . . . .	24
5.4.1	Market Impact Prediction . . . . .	24
5.4.2	Execution Error Correction . . . . .	24

<b>III Backward Service (Janus Consivius)</b>	<b>25</b>
<b>1 Memory Hierarchy: Three-Timescale Architecture</b>	<b>25</b>
1.1 Short-Term Memory (Hippocampus) . . . . .	26
1.1.1 Episodic Buffer . . . . .	26
1.1.2 Pattern Separation . . . . .	26
1.1.3 Spatial Mapping . . . . .	26
1.2 Medium-Term Consolidation (SWR Simulator) . . . . .	26
1.2.1 Replay Prioritization . . . . .	26
1.2.2 Sampling Probability . . . . .	27
1.2.3 Importance Sampling Correction . . . . .	27
1.2.4 Efficient Sampling via Sum Tree . . . . .	27
1.2.5 Sleep-Phase Consolidation . . . . .	27
1.3 Long-Term Memory (Neocortex) . . . . .	27
1.3.1 Schema Representation . . . . .	27
1.3.2 Recall-Gated Consolidation . . . . .	28
<b>2 UMAP Visualization: Cognitive Dashboard</b>	<b>28</b>
2.1 AlignedUMAP for Schema Formation . . . . .	28
2.1.1 Objective Function . . . . .	29
2.2 Parametric UMAP for Real-Time Monitoring . . . . .	29
<b>3 Persistence Layer</b>	<b>29</b>
3.1 Primary Storage: PostgreSQL and Redis . . . . .	29
3.2 Vector Storage: Qdrant . . . . .	30
<b>IV Neuromorphic Architecture</b>	<b>31</b>
<b>1 Neuromorphic Design Philosophy</b>	<b>31</b>
1.1 Why Brain-Inspired Architecture? . . . . .	31
1.2 Neuroscience-to-Trading Mapping . . . . .	32
<b>2 Brain Region Architectures</b>	<b>32</b>
2.1 Visual Cortex: Pattern Recognition . . . . .	33
2.2 Cortex: Strategic Planning & Long-term Memory . . . . .	33
2.2.1 Trading Implementation . . . . .	33
2.3 Hippocampus: Episodic Memory & Experience Replay . . . . .	33
2.3.1 Trading Implementation . . . . .	33
2.4 Thalamus: Attentional Gating & Modality Fusion . . . . .	34
2.4.1 Trading Implementation . . . . .	34

2.5 Hypothalamus: Homeostatic Regulation . . . . .	35
2.5.1 Trading Implementation . . . . .	35
2.6 Basal Ganglia: Action Selection & Reinforcement Learning . . . . .	35
2.7 Prefrontal Cortex: Logic, Planning & Compliance . . . . .	36
2.7.1 Trading Implementation . . . . .	36
2.8 Amygdala: Fear, Threat Detection & Circuit Breakers . . . . .	37
2.8.1 Trading Implementation . . . . .	37
2.9 Cerebellum: Motor Control & Execution . . . . .	38
2.9.1 Trading Implementation . . . . .	38
2.10 Integration: Inter-Region Coordination . . . . .	39
<b>V Rust Implementation</b>	<b>40</b>
<b>1 Architectural Overview</b>	<b>40</b>
1.1 The Rust-Only Philosophy . . . . .	40
1.2 Component Diagram . . . . .	41
<b>2 Machine Learning Framework Strategy</b>	<b>41</b>
2.1 Framework Stack . . . . .	42
2.2 Rust-Native ML Architecture . . . . .	42
<b>3 Forward Service: Rust Implementation</b>	<b>42</b>
3.1 Performance Requirements . . . . .	43
3.2 Core Data Structures . . . . .	43
3.3 GAF Transformation Algorithm . . . . .	43
3.4 LTN Constraint Evaluation . . . . .	44
3.5 Async Service Architecture . . . . .	44
<b>4 Backward Service: Batch Processing</b>	<b>45</b>
4.1 Prioritized Experience Replay . . . . .	45
4.2 Schema Consolidation Algorithm . . . . .	46
<b>5 Execution Service</b>	<b>46</b>
5.1 Multi-Exchange Support . . . . .	46
5.2 Service Interface . . . . .	47
5.3 Execution Algorithms . . . . .	47
<b>6 CNS Service: System Health Monitoring</b>	<b>48</b>
6.1 Preflight Validation . . . . .	48
6.2 Runtime Monitoring . . . . .	48

<b>7 Trading Strategies</b>	<b>49</b>
7.1 Trend-Following Strategies . . . . .	49
7.2 Mean-Reversion Strategies . . . . .	50
7.3 Strategy Gating and Affinity . . . . .	50
<b>8 Neuromorphic Module: Brain-Region Implementations</b>	<b>50</b>
8.1 Key Implementation Details . . . . .	51
8.2 Supporting Crates . . . . .	54
<b>9 Deployment Architecture</b>	<b>55</b>
9.1 Service Orchestration . . . . .	55
 <b>VI Limitations and Open Problems</b>	 <b>58</b>
<b>1 The Brain Metaphor Critique</b>	<b>58</b>
<b>2 Integration Complexity</b>	<b>58</b>
<b>3 Non-Stationarity and Regime Shifts</b>	<b>59</b>
<b>4 Backtest Overfitting Risk</b>	<b>59</b>
<b>5 Crowding Resistance Is Not Crowding Immunity</b>	<b>59</b>
<b>6 LTN Scalability</b>	<b>60</b>
<b>7 Adversarial Robustness</b>	<b>60</b>
 <b>VII Validation Framework and Experimental Design</b>	 <b>61</b>
<b>1 Multi-Agent LOB Simulation Protocol</b>	<b>61</b>
<b>2 Walk-Forward Backtesting</b>	<b>62</b>
<b>3 Ablation Studies</b>	<b>62</b>
<b>4 Benchmark Comparisons</b>	<b>62</b>

## Part I

# Main Architecture

## I The Epistemological Transition to Quant 4.0

The trajectory of algorithmic trading has historically been defined by a tension between interpretability and capability. We are currently witnessing a phase transition from the “black box” empiricism of deep learning (LeCun, Bengio, and Hinton, 2015) toward a new paradigm of Neuro-Symbolic integration (Marra et al., 2024; A. d. Garcez and Lamb, 2023). Project JANUS stands at the vanguard of this transition, termed **Quant 4.0**. This architecture reimagines the financial agent as a biological entity—one that perceives, reasons, remembers, and fears.

The intellectual foundation rests on three convergent programs: (i) neuroscience-inspired AI (Hassabis et al., 2017), which argues that understanding the computational principles of the brain remains the richest source of algorithmic innovation; (ii) the Adaptive Markets Hypothesis (Lo, 2004; Lo, 2017), which replaces the Efficient Markets assumption with an evolutionary framework where market agents adapt, compete, and are selected; and (iii) neuro-symbolic reasoning (A. d. Garcez and Lamb, 2023; Badreddine et al., 2022), which bridges the gap between statistical pattern recognition and logical constraint satisfaction.

### Historical Evolution of Quantitative Finance

- **Quant 1.0 (1980s–1990s):** Era of heuristics and expert systems with high interpretability but extreme rigidity
- **Quant 2.0 (1990s–2000s):** Statistical rigour through mean reversion, cointegration, and factor models (Fama and French, 1993)
- **Quant 3.0 (2010s–Present):** Deep learning hegemony with LSTMs, Transformers (Vaswani et al., 2017), and deep reinforcement learning (Sutton and Barto, 2018)
- **Quant 4.0 (JANUS):** Neuro-Symbolic AI achieving adaptability of deep learning with reliability of rule-based systems, grounded in neuroscience

## II The Doya–Hassabis Framework: Neuroscience-Inspired Architecture

JANUS’s multi-region architecture applies functional brain-region decomposition to trading system design, grounded in the Doya–Hassabis framework of neuroscience-inspired AI (Doya, 1999; Hassabis et al., 2017).

### II.1 Functional Brain-Region Decomposition

Doya (1999) proposed the canonical functional decomposition of brain regions by learning type:

- **Cerebellum** → supervised learning (forward models, error correction)
- **Basal ganglia** → reinforcement learning (reward-driven action selection)
- **Cerebral cortex** → unsupervised learning (schema formation, pattern discovery)

Doya (2002) extended this framework to show that neuromodulators (dopamine, serotonin, norepinephrine, acetylcholine) regulate meta-parameters of learning—learning rate, discount factor, exploration–exploitation balance—providing a biologically grounded mechanism for adaptive parameter control. Caliggiore et al. (2019) formalized the “super-learning hypothesis,” demonstrating that the integration of learning processes *across* cortex, cerebellum, and basal ganglia produces capabilities exceeding any single region.

### II.2 From Neuroscience to Architecture

Hassabis et al. (2017) articulated the programmatic statement for neuroscience-inspired AI: the goal is *functional* inspiration, not biological simulation. JANUS adopts this position, with brain-region mappings in Part IV justified by functional analogy—each region implements a computational principle (e.g., supervised error correction, reinforcement-driven action selection, episodic memory replay) that addresses a specific trading requirement.

This approach is further validated by Yamakawa (2021), who proposed the “whole brain architecture” approach as a systematic method for developing artificial general intelligence by referencing brain structure, and by Caliggiore et al. (2017), who established consensus on the functional interactions between cerebellum, basal ganglia, and cortex.

## II.3 Complementary Learning Systems

The separation of JANUS into Forward (System 1) and Backward (System 2) services directly implements Complementary Learning Systems (CLS) theory (James L McClelland, Bruce L McNaughton, and O'Reilly, 1995). Kumaran, Hassabis, and James L. McClelland (2016) updated CLS theory ("CLS 2.0"), establishing that intelligent agents require both a fast-learning system (hippocampus, for rapid encoding of specific experiences) and a slow-learning system (neocortex, for gradual extraction of statistical regularities). JANUS maps these to:

- **Forward Service (hippocampal analogue):** Rapid encoding of each trading episode with episodic specificity
- **Backward Service (neocortical analogue):** Slow consolidation of episodic memories into generalized market schemas through sharp-wave ripple replay (M. A. Wilson and Bruce L. McNaughton, 1994; Buzsáki, 2015)

Masset et al. (2025) recently demonstrated that dopaminergic neurons encode reward prediction errors at diverse temporal discount factors, providing direct biological validation for JANUS's three-timescale memory architecture (short-term hippocampal buffer, medium-term SWR consolidation, long-term neocortical schemas).

## III The Dual-Process Architecture

JANUS separates real-time perception and action from offline memory consolidation, implementing Dual-Process Theory (Kahneman, 2011; J. S. B. Evans, 2008; J. S. B. T. Evans and Stanovich, 2013) as a concrete engineering design. The system divides into two complementary services:

Service	Persona	Cognitive Role	Biological Analogue
Forward Service	Janus Bifrons	Perception & Action (System 1)	Basal Ganglia & Thalamus
Backward Service	Janus Consivius	Memory & Learning (System 2)	Hippocampus & Neocortex

This separation allows JANUS to optimize for latency on the hot path (Forward Service) while reserving heavy computational resources for consolidation and schema formation on the cold path (Backward Service), implementing CLS theory (James L McClelland, Bruce L McNaughton, and O'Reilly, 1995; Kumaran, Hassabis, and James L. McClelland, 2016).

In addition to the core Forward and Backward services, the production system deploys three supporting services: an **Execution Service** for multi-exchange order routing, a **Data Service** for centralized market data management, and a **CNS (Central Nervous System) Service** for system-wide health monitoring and preflight validation (Section 6).

**Note:** The detailed mathematical specifications for each component are presented in Parts II–V below, followed by limitations (Part VI) and validation framework (Part VII).

## IV Resilience to Strategy Crowding and Co-Impact

Strategy crowding—the convergence of multiple algorithmic trading systems on similar positions—poses systemic risk through amplified co-impact costs and correlated drawdowns. JANUS’s multi-region, heterogeneous architecture is designed to resist these effects.

### IV.1 Co-Impact and Algorithmic Herding

The *square-root impact law*—stating that the price impact of a metaorder scales as  $\Delta p \sim \sigma \sqrt{Q/V}$ , where  $Q$  is order volume and  $V$  is daily volume—is one of the most robust empirical regularities in market microstructure (Tóth et al., 2011; Bouchaud et al., 2018; Bucci, Benzaquen, et al., 2019). Bucci, Mastromatteo, et al. (2020) extended this framework to show how *simultaneous* institutional metaorders interact through net order flow: when  $N$  agents each trade  $Q/N$ , the market responds to aggregate flow without distinguishing individual metaorders. Critically, co-impact introduces a finite intercept  $I_0$  that grows with sign correlation among agents—meaning correlated (crowded) trading amplifies impact costs beyond what single-agent models predict.

The canonical empirical demonstration is the August 2007 quant meltdown (Khandani and Lo, 2011), where quantitative equity funds that had developed models independently suffered coordinated losses because their strategies had converged on similar factor exposures. Stein (2009) formalized this as a coordination problem: individual traders cannot observe how many others pursue the same strategy, creating negative externalities through crowded-trade effects and leverage decisions.

Subsequent empirical work has confirmed crowding’s destructive potential. Kralingen et al. (2021) showed that market clustering—measured against a maximum-entropy null model—has a *causal* effect on tail risk, even after controlling for standard risk drivers. Brown, Howard, and Lundblad (2022) demonstrated that crowded stocks outperform non-crowded stocks on average but with substantially elevated tail risk. Lou and Polk (2022) proposed “comomentum” (high-frequency abnormal return correlation among momentum stocks) as a crowding measure, finding sharp reversals when co-

momentum is high.

## IV.2 Vulnerability in Current Systems

Contemporary algorithmic trading systems are particularly susceptible to crowding for three reasons:

1. **Shared foundation models:** As firms adopt similar pre-trained architectures, output homogeneity increases regardless of nominal strategy differences
2. **Alternative risk premia (ARP) crowding:** Baltas (2019) classifies ARP into *divergence premia* (e.g., momentum—self-reinforcing, no fundamental anchor) and *convergence premia* (e.g., value—self-correcting), showing divergence premia underperform following crowded periods. Volpati et al. (2020) identifies significant and increasing crowding in Fama–French factors
3. **AI crowd effects:** Stillman and Baggott (2024) demonstrates that groups of homogeneous neuro-symbolic traders in virtual markets produce *price suppression*, directly illustrating the risks of AI crowd homogeneity

## IV.3 JANUS's Defense: Engineered Heterogeneity

JANUS addresses crowding through *engineered heterogeneity*: each deployment is configured to produce idiosyncratic order flow that minimizes cross-instance correlation. This is grounded in two key theoretical results:

- Wagner (2011): Rational agents should choose heterogeneous portfolios and forgo diversification benefits to avoid joint liquidation risk. Portfolio heterogeneity directly reduces systemic risk.
- DeMiguel, Martin-Utrera, and Uppal (2021): “Trading diversification”—institutions exploiting different characteristics—increases capacity by 45%, optimal investment by 43%, and profits by 22%.

The heterogeneity is implemented across multiple brain regions:

**Important caveat:** The literature supports heterogeneity as a mechanism for crowding *resistance*, not crowding *immunity*. The 2007 quant quake demonstrated that independently developed strategies can converge on similar factor exposures despite nominal diversity (Khandani and Lo, 2011). Evolutionary dynamics research shows that crowding effects can emerge even in heterogeneous populations when memory sizes are small. We therefore claim crowding resistance, not elimination, and note that empirical validation via multi-agent simulation is required (see Part VII).

Table 1: Sources of engineered heterogeneity across brain regions

Brain Region	Heterogeneity Mechanism	Effect on Order Flow
Thalamus	Modality attention weights randomized at initialization	Different signal weighting per instance
Hypothalamus	Risk setpoints drawn from individualized distributions	Different position sizing and risk tolerance
Basal Ganglia	Dopamine sensitivity and discount factors varied	Different action selection thresholds
Prefrontal	LTN axiom weights and hedge parameters varied	Different constraint priorities
Hippocampus	Replay priorities and consolidation schedules varied	Different learning from the same experience
Cerebellum	Execution timing models calibrated to different horizons	Different order placement patterns

## Part II

# Forward Service (Janus Bifrons)

## Abstract

JANUS Forward represents the “wake state” of the JANUS trading system, responsible for all real-time decision-making during market hours. This service combines:

- **Visual Pattern Recognition** using Gramian Angular Fields (GAF) (Wang and Oates, 2015) with LSTM and Video Vision Transformer (ViViT) (Arnab et al., 2021) temporal modeling
- **Symbolic Reasoning** via Logic Tensor Networks (LTN) (Badreddine et al., 2022) for constraint satisfaction
- **Multimodal Fusion** integrating time series, visual, order book, and sentiment data through specialized fusion engines
- **Dual-Pathway Decision Making** inspired by basal ganglia architecture (Collins and Michael J Frank, 2014)
- **Ensemble Regime Detection** combining Hidden Markov Models, statistical, and technical methods for market state identification

The Forward service operates on a hot path with strict latency requirements, implementing a six-stage neural pipeline (regime → hypothalamus → amygdala → gating → correlation → execution) that routes signals through brain regions in real time.

FPGA acceleration (Marino et al., 2023; Vemeko, 2023) is planned for nanosecond-level latency in future high-frequency trading applications.

# 1 Visual Pattern Recognition: DiffGAF and Vision Models

The visual subsystem transforms time series data into spatiotemporal images, enabling the system to “see” market patterns that traditional numerical methods miss. This approach is grounded in the work of Wang and Oates (2015), who demonstrated that imaging time series significantly improves classification and imputation tasks by exposing temporal correlations to the inductive biases of convolutional neural networks.

## 1.1 Mathematical Foundation: Gramian Angular Fields

Time series are encoded into polar coordinates and projected onto Gramian matrices, creating 2D representations that preserve temporal correlations (Wang and Oates, 2015). Research by Chen and Tsai (2020) has validated that GAF-based encodings of financial candlestick data substantially outperform raw time-series inputs in classification tasks by capturing multi-scale temporal structures.

### 1.1.1 Input Preprocessing

Given raw market data  $X = \{x_1, x_2, \dots, x_T\}$  where  $x_t \in \mathbb{R}^D$  (multi-feature time series), we first apply feature selection to extract  $F$  relevant features.

### 1.1.2 Step 1: Normalization

For inference, we apply min-max normalization to the domain  $[-1, 1]$ , followed by Piecewise Aggregate Approximation (PAA) for temporal resizing:

$$\tilde{x}_t = 2 \cdot \frac{x_t - x_{\min}}{x_{\max} - x_{\min}} - 1 \quad (1)$$

ensuring the subsequent  $\arccos$  operation is well-defined with  $\tilde{x}_t \in [-1, 1]$ .

For the training pipeline, we employ learnable affine transformations with domain constraints:

$$\tilde{x}_t = \tanh \left( \gamma \odot \frac{x_t - \mu}{\sigma} + \beta \right) \quad (2)$$

where  $\gamma, \beta \in \mathbb{R}^F$  are learned parameters, and  $\mu, \sigma$  are running statistics. The  $\tanh$  function guarantees  $\tilde{x}_t \in (-1, 1)$ .

### 1.1.3 Step 2: Polar Coordinate Transformation

Map normalized values to angular space:

$$\phi_t = \arccos(\tilde{x}_t) \in [0, \pi] \quad (3)$$

$$r_t = \frac{t}{T} \quad (\text{normalized timestamp}) \quad (4)$$

### 1.1.4 Step 3: Gramian Field Generation

Construct the Gramian Angular Summation Field (GASF):

$$\mathbf{G}_{ij} = \cos(\phi_i + \phi_j) = \tilde{x}_i \tilde{x}_j - \sqrt{1 - \tilde{x}_i^2} \sqrt{1 - \tilde{x}_j^2} \quad (5)$$

Or the Gramian Angular Difference Field (GADF), which JANUS employs to encode velocity of price changes as visual textures:

$$\mathbf{G}_{ij} = \sin(\phi_i - \phi_j) = \sqrt{1 - \tilde{x}_i^2} \tilde{x}_j - \tilde{x}_i \sqrt{1 - \tilde{x}_j^2} \quad (6)$$

This transformation allows the system to visually perceive volatility regimes and microstructure dynamics (Wang and Oates, 2015).

### 1.1.5 Differentiable GAF (DiffGAF)

The Rust implementation provides a fully differentiable GAF engine with analytically computed Jacobians for end-to-end gradient flow. The key derivative through the polar mapping is:

$$\frac{d\phi_k}{d\tilde{x}_k} = \frac{-1}{\sqrt{1 - \tilde{x}_k^2}} \quad (7)$$

which enables backpropagation through the entire GAF encoding pipeline. Numerical gradient verification tests confirm correctness within  $10^{-4}$  tolerance.

## 1.2 3D Spatiotemporal Manifolds: GAF Video

To capture temporal dynamics, we generate a sequence of GAF frames using sliding windows.

### 1.2.1 Sliding Window GAF Video Generation

Given a time series of length  $T$ , window size  $W$ , and stride  $S$ :

1. Extract windows:  $X_k = \{x_{(k-1)S+1}, \dots, x_{(k-1)S+W}\}$  for  $k = 1, \dots, N$
2. Generate GAF for each window:  $\mathbf{G}_k = \text{GAF}(X_k) \in \mathbb{R}^{W \times W}$

3. Stack into video:  $\mathbf{V} = [\mathbf{G}_1, \mathbf{G}_2, \dots, \mathbf{G}_N] \in \mathbb{R}^{N \times W \times W}$

The data ingestion pipeline handles windowed buffering and streaming through dedicated preprocessing modules.

## 1.3 Vision Model Architecture

### 1.3.1 DiffGAF-LSTM

The DiffGAF-LSTM vision model pairs the DiffGAF encoding with an LSTM temporal model. GAF frames are encoded and fed sequentially into an LSTM network that captures inter-frame temporal dependencies. This architecture provides robust performance while maintaining the differentiability required for end-to-end training.

### 1.3.2 Video Vision Transformer (ViViT)

The ViViT model uses a factorized spatiotemporal transformer (Arnab et al., 2021). Unlike standard Vision Transformers (Dosovitskiy et al., 2020) which process static images, ViViT factorizes attention across both space (price/volume levels of the limit order book) and time (sequences of LOB snapshots), enabling the system to track dynamic microstructure events.

**Patch Embedding:** Divide each frame  $\mathbf{G}_k$  into non-overlapping patches:

$$\mathbf{P}_k = \text{Reshape}(\mathbf{G}_k) \in \mathbb{R}^{P \times (p^2)} \quad (8)$$

where  $P = (W/p)^2$  is the number of patches per frame.

**Spatial Attention:** Apply self-attention within each frame:

$$\mathbf{Z}_k^{(l)} = \text{MSA}(\text{LN}(\mathbf{Z}_k^{(l-1)})) + \mathbf{Z}_k^{(l-1)} \quad (9)$$

**Temporal Attention:** Apply attention across frames:

$$\mathbf{H}^{(l)} = \text{MSA}(\text{LN}([\mathbf{Z}_1^{(l)}, \dots, \mathbf{Z}_N^{(l)}])) \quad (10)$$

Both the DiffGAF-LSTM and ViViT architectures are implemented natively in Rust, with the system selecting the appropriate model based on configuration and available resources. The ViViT model gracefully degrades to the DiffGAF-LSTM model when computational constraints require it.

## 2 Ensemble Regime Detection

Market regime identification is a critical upstream component that conditions all downstream decision-making. JANUS employs an ensemble approach combining multiple detection methods to robustly classify market states.

### 2.1 Detection Methods

#### 2.1.1 Hidden Markov Model (HMM)

A Gaussian HMM models latent regime states with observable market features:

$$P(\mathbf{x}_t \mid z_t = k) = \mathcal{N}(\mathbf{x}_t; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \quad (11)$$

where  $z_t \in \{1, \dots, K\}$  is the latent regime state and the transition matrix  $\mathbf{A}_{ij} = P(z_{t+1} = j \mid z_t = i)$  captures regime persistence and switching dynamics.

#### 2.1.2 Statistical Methods

Complementing the HMM, JANUS applies rolling statistical tests including variance ratio tests for mean-reversion detection, Hurst exponent estimation for trend persistence, and distributional tests for fat-tail identification.

#### 2.1.3 Technical Methods

Classical technical analysis signals (trend strength indicators, volatility measures, momentum oscillators) are fused with the statistical methods to provide corroborating evidence for regime classification.

## 2.2 Regime Categories

The ensemble classifier identifies seven market regimes, each mapped to distinct trading behavior profiles:

The detected regime feeds into the Hypothalamus module for homeostatic regulation and the Basal Ganglia for dopamine-modulated action selection, forming the first stage of the brain wiring pipeline.

Regime	Characteristics	System Behavior
Bull	Sustained upward trend, expanding volume	Amplify Direct (Go) pathway
Bear	Sustained downward trend, risk-off	Amplify Indirect (No-Go) pathway
Ranging	Low directional conviction, bounded action	Favor mean-reversion strategies
Crisis	Extreme volatility, correlation breakdown	Engage Amygdala circuit breakers
Recovery	Post-crisis normalization, lower volatility	Gradually release risk constraints
Bubble	Parabolic acceleration, euphoric sentiment	Heighten caution via Hypothalamus
Deflation	Sustained contraction, liquidity withdrawal	Reduce position sizing

### 3 Logic Tensor Networks: Symbolic Reasoning Engine

LTNs bridge neural networks and first-order logic, enabling differentiable constraint satisfaction (Badreddine et al., 2022; Serafini and A. S. d. Garcez, 2016). This neuro-symbolic approach allows JANUS to enforce regulatory and risk constraints directly within gradient descent optimization, a capability absent in pure deep learning systems (Marra et al., 2024; A. d. Garcez and Lamb, 2023). No prior work applies LTN to financial decision-making; this represents a novel contribution of JANUS.

#### 3.1 Mathematical Foundation

The central innovation of LTNs (Badreddine et al., 2022) is the ability to make Boolean logic differentiable using Real Logic, specifically Łukasiewicz t-norms.

##### 3.1.1 Grounding Function

Map logical constants to real vectors:

$$\mathcal{G} : \mathcal{C} \rightarrow \mathbb{R}^d \tag{12}$$

##### 3.1.2 Predicate Grounding

A predicate  $P(x)$  is grounded as a neural network  $f_\theta : \mathbb{R}^d \rightarrow [0, 1]$ , where truth values range continuously from 0 to 1 rather than being discrete binary values.

## 3.2 Łukasiewicz T-Norm Operations

Following Badreddine et al. (2022), we employ fuzzy logic operators based on Łukasiewicz t-norms that are differentiable and thus compatible with backpropagation.

### 3.2.1 Conjunction (AND)

For training, we use Product Logic to ensure smooth gradients:

$$u \wedge v = u \cdot v \quad (13)$$

For inference/evaluation, standard Łukasiewicz logic is used:

$$u \wedge v = \max(0, u + v - 1) \quad (14)$$

### 3.2.2 Disjunction (OR)

$$u \vee v = \min(1, u + v) \quad (15)$$

### 3.2.3 Negation (NOT)

$$\neg u = 1 - u \quad (16)$$

### 3.2.4 Implication (IF-THEN)

For training (Product Logic):

$$u \Rightarrow v = 1 - u + u \cdot v \quad (17)$$

For inference (Łukasiewicz Logic):

$$u \Rightarrow v = \min(1, 1 - u + v) \quad (18)$$

### 3.2.5 Bi-Implication (Equivalence)

$$u \Leftrightarrow v = 1 - |u - v| \quad (19)$$

### 3.2.6 Linguistic Hedges

The implementation extends classical fuzzy logic with linguistic hedge operators for nuanced truth-value modification:

$$\text{very}(x) = x^2 \quad (20)$$

$$\text{somewhat}(x) = \sqrt{x} \quad (21)$$

$$\text{slightly}(x) = \sqrt{x} - x \quad (22)$$

$$\text{extremely}(x) = x^3 \quad (23)$$

These hedges allow more expressive constraint formulation (e.g., “very risky” vs. “somewhat risky” conditions).

## 3.3 Knowledge Base Formulation

The knowledge base  $\mathcal{KB}$  encodes regulatory constraints and risk management rules as logical predicates. The implementation provides a comprehensive compliance engine covering multiple constraint categories.

### 3.3.1 Wash Sale Constraint

The Wash Sale Rule (Internal Revenue Service, 2024)—a critical regulatory constraint for active traders—prevents claiming tax losses on securities sold and repurchased within 30 days. The implementation enforces the full 30-day window both before and after the sale, tracks loss sales per symbol, computes disallowed loss amounts, and blocks violating trades:

$$\forall t : \text{Sell}(t) \wedge \text{Buy}(t') \wedge |t - t'| < 30 \Rightarrow \neg \text{TaxLoss}(t) \quad (24)$$

### 3.3.2 Almgren-Chriss Risk Constraint

Following the optimal execution framework of Almgren and Chriss (2001), we constrain market impact relative to volatility:

$$\forall \text{order} : \text{Execute}(\text{order}) \Rightarrow \text{Slippage}(\text{order}) < \lambda \cdot \text{Volatility} \quad (25)$$

This ensures trades remain within the efficient frontier between expected cost and risk.

### 3.3.3 Additional Constraint Categories

Beyond the foundational constraints above, the production knowledge base includes:

- **Position Limits:** Maximum exposure per asset and aggregate portfolio

- **Capital Allocation:** Constraints on capital deployment across strategies
- **Risk Limits:** Value-at-Risk, maximum drawdown, and volatility thresholds
- **Proprietary Firm Rules:** Compliance with specific prop trading firm requirements (daily loss limits, trailing drawdown, consistency rules)

## 3.4 Logical Loss Function

### 3.4.1 Satisfiability Aggregation

$$\text{SAT}(\mathcal{KB}) = \text{p-mean}_{i=1}^{|\mathcal{KB}|}(\phi_i) \quad (26)$$

The evaluation context maintains EMA-smoothed satisfaction scores with per-constraint violation tracking for monitoring and diagnostics.

### 3.4.2 Logical Loss

$$\mathcal{L}_{\text{logic}} = 1 - \text{SAT}(\mathcal{KB}) \quad (27)$$

## 4 Multimodal Fusion: Specialized Gated Attention

JANUS integrates multiple data modalities through gated cross-attention (Alayrac et al., 2022), allowing the system to dynamically weight inputs based on their predictive uncertainty. This is the machine-learning analogue of thalamic attentional gating in biological systems (Halassa and Kastner, 2017; George et al., 2025).

### 4.1 Input Modalities

The production system processes four specialized data streams through dedicated fusion engines:

- **Order Book:** Full limit order book depth, bid-ask dynamics, and microstructure features
- **Price:** Multi-timeframe price action including OHLCV and derived indicators, with Chronos foundation model forecasting (Ansari et al., 2024b; Ansari et al., 2024a)
- **Volume:** Volume profile analysis, volume-weighted metrics, and participation rates
- **Sentiment:** BERT/FinBERT embeddings (Hugging Face, 2024) from news feeds (NewsAPI, CryptoPanic) and social sentiment signals, with Qdrant-backed similarity search for regime-aware aggregation

Additionally, the system integrates supplementary data sources including weather data (via OpenWeatherMap) and space weather/celestial data for correlation analysis with commodity and energy markets.

## 4.2 Gated Cross-Attention Mechanism

### 4.2.1 Attention Computation

Following the attention mechanism of Vaswani et al. (2017), with support for causal masking and multi-head configuration:

$$\text{Attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V} \quad (28)$$

### 4.2.2 Gating Mechanism

The gating function suppresses noise and amplifies signal, similar to thalamic regulation (Halassa and Kastner, 2017):

$$g = \text{sigmoid}(\mathbf{W}_g[\mathbf{v}; \mathbf{t}; \mathbf{s}] + \mathbf{b}_g) \quad (29)$$

Each modality-specific fusion engine applies this gating independently, allowing the system to dynamically suppress noisy order book data during low-liquidity periods while amplifying sentiment signals during news-driven regimes.

## 5 Decision Engine: Basal Ganglia Pathways

Action selection in JANUS is mediated by a model of the Basal Ganglia, comprising Direct (“Go”) and Indirect (“No-Go”) pathways (Collins and Michael J Frank, 2014; Foster, Morris, and Dayan, 2013). This architecture, known as Opponent Actor Learning (OpAL), creates dynamic risk tolerance that adapts to market conditions.

### 5.1 Praxeological Motor: Dual Pathways

#### 5.1.1 Direct Pathway (Go Signal)

Encodes the benefits of an action, amplified by dopamine during high-confidence regimes (Schultz, Dayan, and Montague, 1997):

$$Q_\theta^+(s, a) = \mathbb{E}[\text{Reward} \mid s, a] \quad (30)$$

The implementation includes dopamine sensitivity parameters, learning rates for value updates, decay rates, and threshold-based action release.

### 5.1.2 Indirect Pathway (No-Go Signal)

Encodes the costs and risks, amplified during uncertainty (Collins and Michael J Frank, 2014):

$$Q_\theta^-(s, a) = \mathbb{E}[\text{Risk} \mid s, a] \quad (31)$$

The implementation provides risk assessment, inhibition generation, and caution scoring.

## 5.2 Action Selection

The final action is determined by the competition between pathways:

$$\mathbf{a}_t = \text{softmax}(\mathbf{d}_{\text{direct}} - \lambda \cdot \mathbf{d}_{\text{indirect}}) \quad (32)$$

where  $\lambda > 0$  is the inhibition weight parameter, and each pathway is computed as:

$$\mathbf{d}_{\text{direct}} = \text{ReLU}(\mathbf{W}_{\text{direct}} \mathbf{h} + \mathbf{b}_{\text{direct}}) \quad (33)$$

$$\mathbf{d}_{\text{indirect}} = \text{ReLU}(\mathbf{W}_{\text{indirect}} \mathbf{h} + \mathbf{b}_{\text{indirect}}) \quad (34)$$

where  $\mathbf{h}$  is the fused state representation from the Thalamus.

An actor-critic framework ties both pathways together, with decision confidence scoring determining whether the action is released for execution.

## 5.3 Brain Wiring Pipeline

In production, decisions traverse a six-stage pipeline that chains brain regions together:

1. **Regime Detection:** Ensemble classifier identifies the current market state
2. **Hypothalamus:** Adjusts position sizing and risk appetite based on regime and portfolio homeostasis
3. **Amygdala:** Evaluates threat signals; may trigger circuit breakers before further processing
4. **Gating:** Thalamic attention gates filter and weight modality signals
5. **Correlation:** Cross-asset correlation tracking across monitored pairs informs diversification
6. **Execution:** Basal ganglia action selection routes to the Execution Service

This pipeline is operationally richer than the abstract dual-pathway model, reflecting the biological reality that action selection involves coordination across multiple brain regions.

## 5.4 Cerebellar Forward Model (Wolpert, Miall, and Kawato, 1998; Sokolov, Miall, and Ivry, 2017)

The Cerebellar module simulates market dynamics to predict execution outcomes before committing to a trade.

### 5.4.1 Market Impact Prediction

Following the Almgren-Chriss framework (Almgren and Chriss, 2001; Markwick, 2023), the model predicts slippage and volatility. To detect predatory environments, JANUS employs the VPIN (Volume-Synchronized Probability of Informed Trading) metric (Easley, Lopez de Prado, and O'Hara, 2011; Easley, De Prado, and O'Hara, 2012), which serves as a proxy for “Flow Toxicity”—the probability that the counterparty has superior information. High VPIN levels often precede flash crashes and feed into the Amygdala circuit for threat detection.

$$\hat{p}_{t+1} = f_{\text{cerebellum}}(\mathbf{s}_t, \mathbf{a}_t) \quad (35)$$

### 5.4.2 Execution Error Correction

The Cerebellum also provides closed-loop error correction for execution quality, implemented through a PID controller:

$$u(t) = K_p \cdot e(t) + K_i \cdot \int_0^t e(\tau) d\tau + K_d \cdot \frac{de(t)}{dt} \quad (36)$$

where  $e(t)$  is the deviation between predicted and realized execution cost. Adaptive correction and feedback loops continuously refine the forward model’s predictions based on observed outcomes.

## Part III

# Backward Service (Janus Consivius)

## Abstract

JANUS Backward represents the “sleep state” of the system, responsible for memory consolidation, schema formation, and learning from accumulated experience. This service implements the Complementary Learning Systems (CLS) theory (James L McClelland, Bruce L McNaughton, and O'Reilly, 1995), which posits that intelligent agents require two learning systems: a fast-learning hippocampus for episodic details and a slow-learning neocortex for statistical generalization. This service implements:

- **Three-Timescale Memory Hierarchy** (Hippocampus → SWR → Neocortex) following CLS architecture (James L McClelland, Bruce L McNaughton, and O'Reilly, 1995)
- **Sharp-Wave Ripple Simulation** for prioritized experience replay (Buzsáki, 2015; Schaul et al., 2015)
- **Schema Formation** via feature-range matching with UMAP-based visualization (McInnes, Healy, and Melville, 2018b; McInnes, Healy, and Melville, 2018a)
- **Recall-Gated Consolidation** ensuring only successful patterns are promoted (Michael J Frank, Loughry, and O'Reilly, 2006)
- **Signal Persistence and Analytics** via PostgreSQL repositories

The Backward service runs on a cold path during off-market hours, performing computationally intensive operations to distill daily experiences into long-term knowledge, effectively replicating the biological process of memory consolidation during sleep (Buzsáki, 2015).

## 1 Memory Hierarchy: Three-Timescale Architecture

The three-tier memory architecture directly implements the Complementary Learning Systems theory (James L McClelland, Bruce L McNaughton, and O'Reilly, 1995), preventing catastrophic forgetting while enabling rapid learning of new market patterns.

## 1.1 Short-Term Memory (Hippocampus)

The hippocampal buffer stores episodic memories of individual trading events, enabling fast learning without interfering with consolidated knowledge (James L McClelland, Bruce L McNaughton, and O'Reilly, 1995).

### 1.1.1 Episodic Buffer

Stores raw experiences during trading, mirroring the role of biological hippocampus in episodic memory formation:

$$\mathcal{D}_{\text{hippo}} = \{(s_t, a_t, r_t, s_{t+1}, \mathbf{c}_t, \mathbf{e}_t)\}_{t=1}^T \quad (37)$$

where  $\mathbf{c}_t$  contains contextual metadata (volatility, spreads, volume) and  $\mathbf{e}_t$  contains emotional tags (fear level, confidence, surprise) that bias consolidation priority.

### 1.1.2 Pattern Separation

Uses random projections to ensure diverse encoding:

$$\mathbf{h}_t = \tanh(\mathbf{W}_{\text{rand}} \cdot [s_t; a_t; \mathbf{c}_t]) \quad (38)$$

### 1.1.3 Spatial Mapping

Experiences are organized into a spatial map that preserves topological relationships between market states, analogous to hippocampal place cells. This enables efficient retrieval of contextually similar experiences during replay.

## 1.2 Medium-Term Consolidation (SWR Simulator)

Consolidation occurs during Sharp-Wave Ripples (SWRs)—high-frequency oscillations that replay compressed sequences of neural activity (Buzsáki, 2015). JANUS mimics this using Prioritized Experience Replay (Schaul et al., 2015), modified to incorporate surprise, emotion, and logical violations.

### 1.2.1 Replay Prioritization

During “sleep” (post-market hours), experiences are replayed in priority order. Biological research shows that replay is biased towards salient and novel events (Joo and L. M. Frank, 2023; Mattar and Nathaniel D. Daw, 2018), which JANUS replicates through composite priority scoring. This design is directly inspired by hippocampal sharp-wave ripple replay (M. A. Wilson and Bruce L. McNaughton, 1994; Buzsáki,

2015; Buzsáki, 1989), which the deep RL community adopted as “experience replay” (Mnih et al., 2015; Schaul et al., 2015). Compute TD-error based priority:

$$p_i = |\delta_i| + \epsilon \quad (39)$$

where  $\delta_i = r_i + \gamma \max_{a'} Q(s_{i+1}, a') - Q(s_i, a_i)$  and  $\epsilon = 10^{-6}$  ensures numerical stability.

### 1.2.2 Sampling Probability

$$P(i) = \frac{p_i^\alpha}{\sum_j p_j^\alpha} \quad (40)$$

where  $\alpha \in [0, 1]$  controls prioritization strength (default  $\alpha = 0.6$ ).

### 1.2.3 Importance Sampling Correction

$$w_i = \left( \frac{1}{N \cdot P(i)} \right)^\beta \quad (41)$$

where  $\beta$  is annealed from  $0.4 \rightarrow 1.0$  during training via a configurable increment parameter, fully correcting bias at convergence.

### 1.2.4 Efficient Sampling via Sum Tree

The replay buffer uses a sum tree data structure for  $\mathcal{O}(\log n)$  sampling, enabling efficient prioritized replay even with large buffer sizes.

### 1.2.5 Sleep-Phase Consolidation

The SWR simulator progresses through biologically inspired phases:

$$\text{Phase} \in \{\text{Awake} \rightarrow \text{Light} \rightarrow \text{Deep} \rightarrow \text{Integration} \rightarrow \text{Transition}\} \quad (42)$$

Each phase adjusts replay parameters (compression ratio, replay rate, consolidation threshold), mirroring the empirical finding that different sleep stages serve distinct memory functions.

## 1.3 Long-Term Memory (Neocortex)

### 1.3.1 Schema Representation

Schemas represent learned market regime prototypes. The production implementation uses feature-range matching with weighted multi-criteria scoring rather than pure

centroid-based clustering:

$$\text{match}(\mathbf{x}, \mathcal{S}_k) = \frac{1}{|\mathcal{F}|} \sum_{f \in \mathcal{F}} w_f \cdot \mathbb{1}[x_f \in \text{range}_f^{(k)}] \quad (43)$$

where  $\mathcal{F}$  is the feature set,  $w_f$  is the feature weight, and  $\text{range}_f^{(k)}$  is the acceptable range for feature  $f$  in schema  $k$ . This approach is deterministic and interpretable, providing explicit decision boundaries for each regime.

The system supports seven regime schemas (Bull, Bear, Ranging, Crisis, Recovery, Bubble, Deflation), each with a Markov transition matrix for regime prediction:

$$P(\mathcal{S}_{t+1} = j \mid \mathcal{S}_t = i) = \mathbf{T}_{ij} \quad (44)$$

For embedding-based operations (similarity search, schema formation from new experiences), centroid-based representations remain available:

$$\mathbf{z}_k = \frac{1}{|\mathcal{C}_k|} \sum_{i \in \mathcal{C}_k} \mathbf{h}_i \quad (45)$$

### 1.3.2 Recall-Gated Consolidation

Only update schemas from successfully recalled experiences:

$$\mathbf{z}_k \leftarrow \mathbf{z}_k + \eta \cdot \mathbb{1}[\text{recall\_success}] \cdot (\mathbf{h}_{\text{new}} - \mathbf{z}_k) \quad (46)$$

## 2 UMAP Visualization: Cognitive Dashboard

To visualize and manage schema structures, JANUS employs Uniform Manifold Approximation and Projection (UMAP) (McInnes, Healy, and Melville, 2018b), which preserves both local and global structure better than alternatives like t-SNE.

### 2.1 AlignedUMAP for Schema Formation

Track how internal representations evolve over time using AlignedUMAP (McInnes, Healy, and Melville, 2018b; McInnes, Healy, and Melville, 2018a), which aligns manifolds across different time steps to monitor representational drift. Maintains consistent embeddings across sleep cycles.

### 2.1.1 Objective Function

The full UMAP loss includes both attraction and repulsion terms:

$$\mathcal{L}_{\text{UMAP}} = \sum_{i \neq j} [w_{ij} \log(q_{ij}) + (1 - w_{ij}) \log(1 - q_{ij})] \quad (47)$$

where  $q_{ij} = (1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}$ .

**Note:** In practice, the repulsion term  $(1 - w_{ij})$  is approximated via *negative sampling* to achieve  $\mathcal{O}(N)$  complexity. For each positive edge, we sample  $k = 5$  random negative pairs.

## 2.2 Parametric UMAP for Real-Time Monitoring

Train a neural network to project new experiences:

$$\mathbf{y}_{\text{new}} = f_{\theta}(\mathbf{h}_{\text{new}}) \quad (48)$$

The parametric UMAP implementation uses a multi-layer encoder network trained on the fuzzy simplicial set graph, with configurable distance metrics (Euclidean, Cosine, Manhattan), smooth  $k$ -nearest-neighbor bandwidth estimation, and a Qdrant bridge for persistent storage of projected embeddings. A drift detection module computes embedding-space displacement between reference and live data, classifying severity (Low/Moderate/High/Critical) for regime monitoring. Regime cluster analysis provides per-cluster statistics (centroid, intra-cluster distances, standard deviation) and inter-cluster separation metrics.

## 3 Persistence Layer

### 3.1 Primary Storage: PostgreSQL and Redis

The production system uses PostgreSQL for ACID-compliant storage of trading records and Redis for low-latency operational state:

- **Signal Repository:** Persists generated trading signals with full metadata
- **Portfolio Repository:** Tracks positions, P&L, and capital allocation
- **Performance Repository:** Stores performance analytics for backward analysis
- **Redis:** Walk-forward optimizer parameters, kill switch state, hot-reloadable configuration

PostgreSQL provides the ACID guarantees essential for financial records, while Redis enables sub-millisecond reads for time-critical operational state.

### 3.2 Vector Storage: Qdrant

For schema similarity search, JANUS integrates Qdrant (Qdrant, 2024), a high-performance vector similarity search engine. Schemas are stored with L2-normalized centroid vectors for cosine similarity search:

$$\mathcal{N}_k = \arg \max_k \text{cosine}(\mathbf{h}_t, \mathbf{z}_k) \quad (49)$$

The vector database layer complements the relational storage, serving the specific use case of nearest-neighbor retrieval for regime pattern matching.

## Part IV

# Neuromorphic Architecture

## Abstract

This document maps the computational components of Project JANUS to specific brain regions, following the Doya–Hassabis framework established in Section II. The neuromorphic approach is grounded in functional brain-region decomposition (Doya, 1999; Hassabis et al., 2017; Caligiore et al., 2019) and provides:

- **Modular Design** with clear functional boundaries mirroring brain organization (Yamakawa, 2021)
- **Biological Validation** of architectural decisions based on empirical neuroscience (James L McClelland, Bruce L McNaughton, and O'Reilly, 1995; Kumaran, Hassabis, and James L. McClelland, 2016)
- **Emergent Intelligence** through brain-inspired interactions and allostatic regulation (Sterling, 2012)

**Positioning relative to existing work:** The intersection of neuromorphic computing and finance is extremely sparse. Mohan et al. (2025) apply spiking neural networks to cross-market portfolio optimization but use no brain-region mapping. Bi and Calhoun (2025) propose a “Financial Connectome” using brain connectivity analysis methods on market data, but this applies neuroscience *analysis tools* to markets rather than neuroscience *architectural principles* to trading systems. Ezinwoke and Rhodes (2025) apply SNNs to HFT price prediction without multi-region architecture. **No published system maps multiple brain regions to distinct trading subsystems.** JANUS is the first.

## 1 Neuromorphic Design Philosophy

### 1.1 Why Brain-Inspired Architecture?

The brain efficiently solves problems similar to trading (Nathaniel D Daw et al., 2006), demonstrating capabilities that map directly to trading challenges:

- Pattern recognition under uncertainty (visual cortex and hippocampus (Buzsáki, 2015))
- Fast decision-making with delayed rewards (basal ganglia (Collins and Michael J Frank, 2014))

- Continual learning without catastrophic forgetting (complementary learning systems (James L McClelland, Bruce L McNaughton, and O'Reilly, 1995))
- Multi-timescale memory consolidation (hippocampus to neocortex (Buzsáki, 2015))
- Homeostatic regulation under varying conditions (hypothalamic control (Sterling, 2012))

## 1.2 Neuroscience-to-Trading Mapping

This mapping is grounded in empirical neuroscience and cognitive modeling (Michael J Frank, Loughry, and O'Reilly, 2006; Collins and Michael J Frank, 2014; Foster, Morris, and Dayan, 2013). JANUS implements ten brain regions, each serving a distinct functional role:

Brain Region	Biological Function	Trading Function
Visual Cortex	Pattern recognition	GAF/ViT chart analysis (Wang and Oates, 2015; Arnab et al., 2021)
Hippocampus	Episodic memory (Buzsáki, 2015)	Experience replay buffer (Schaul et al., 2015)
Prefrontal Cortex	Logic and planning (Michael J Frank, Loughry, and O'Reilly, 2006)	LTN constraint checking (Badreddine et al., 2022)
Basal Ganglia	Action selection (Collins and Michael J Frank, 2014)	Buy/sell/hold decisions
Cerebellum	Motor prediction (Wolpert, Miall, and Kawato, 1998)	Market impact forecasting (Almgren and Chriss, 2001)
Amygdala	Threat detection (LeDoux, 2000)	Risk circuit breakers
Thalamus	Attentional gating (Halassa and Kastner, 2017)	Modality fusion and signal routing
Hypothalamus	Homeostatic regulation (Sterling, 2012)	Position sizing and risk appetite
Cortex	Strategic planning	Regime schemas and hierarchical RL
Integration	Inter-region coordination	Service bridges and data pipeline

## 2 Brain Region Architectures

The following sections detail how each brain region's computational principles are implemented in JANUS.

## 2.1 Visual Cortex: Pattern Recognition

The visual cortex processes market data as images through the DiffGAF pipeline (Section 1), with submodules for GAF encoding (GASF/GADF), vision model inference (DiffGAF-LSTM and ViViT), data ingestion and buffering, parametric UMAP (McInnes, Healy, and Melville, 2018b) with neural encoder projection, drift detection (Low/Moderate/High/Critical severity classification), and Qdrant-backed regime cluster persistence for real-time representation monitoring.

## 2.2 Cortex: Strategic Planning & Long-term Memory

The neocortex implements slow, statistical learning of market schemas (James L McClelland, Bruce L McNaughton, and O'Reilly, 1995).

### 2.2.1 Trading Implementation

**Component:** Neocortical Schema Network with Hierarchical RL Manager

- Schema prototypes stored with feature-range matching and confidence scoring
- Seven market regime templates (Bull, Bear, Ranging, Crisis, Recovery, Bubble, Deflation) with Markov transition matrices
- Declarative memory and long-term knowledge base for persistent market insights
- Hierarchical RL manager for multi-level strategic planning
- Slow consolidation during sleep cycles

## 2.3 Hippocampus: Episodic Memory & Experience Replay

The hippocampus provides fast learning and episodic memory storage, with consolidation via Sharp-Wave Ripples (Buzsáki, 2015; Joo and L. M. Frank, 2023).

### 2.3.1 Trading Implementation

**Component:** Episodic Buffer + SWR Replay

- Fixed-size circular buffer storing recent trades
- Sparse encoding via random projections
- Emotional tagging for consolidation priority
- Trade episode and market event recording

- Spatial mapping of experience relationships
- Prioritized replay with sum-tree sampling during training
- Multi-phase sleep consolidation (Awake → Light → Deep → Integration → Transition)
- **Diffusion-Based Synthetic Data:** Regime-conditional DDPM (Coletta et al., 2024) generates synthetic market sequences for training data augmentation, with configurable noise schedules (linear, cosine, quadratic), Min-SNR loss weighting, EMA model averaging, and automated quality assessment comparing synthetic vs. real feature distributions and autocorrelation structure

## 2.4 Thalamus: Attentional Gating & Modality Fusion

The Thalamus functions as the “gatekeeper” of perception in JANUS, regulating the flow of visual and numerical data into the decision engine (Halassa and Kastner, 2017; George et al., 2025).

### 2.4.1 Trading Implementation

**Component:** Multi-Head Cross-Attention with Modality-Specific Fusion

- Cross-attention with causal masking and residual connections
- Saliency computation and attentional focus control
- Gating mechanism for signal suppression and amplification
- Specialized fusion engines: order book, price, volume, and sentiment
- External data source integration (news, weather, celestial/space weather)
- Signal routing to downstream brain regions
- **Chronos Time Series Forecasting:** ONNX-based inference pipeline (Ansari et al., 2024b; Ansari et al., 2024a) with quantile-based tokenization, configurable presets (T5-Tiny/Small/Base), and confidence-interval forecast output
- **BERT Sentiment Analysis:** FinBERT/DistilBERT sentiment embeddings (Hugging Face, 2024) running natively in Rust via Candle, producing [CLS] embeddings stored in Qdrant for regime-aware sentiment aggregation

The Thalamic Reticular Nucleus provides attentional gating, enabling JANUS to focus computational resources on the most informative market data streams. Wilson-Cowan mean-field models (H. R. Wilson and Cowan, 1972; Y. Zhang et al., 2024) implement oscillatory attention dynamics, modeling coupled excitatory-inhibitory neural populations via ODEs with Hilbert-transform-based amplitude/phase estimation, bifurcation analysis, neuromodulator-driven regime switching, and fixed-point stability classification.

## 2.5 Hypothalamus: Homeostatic Regulation

The Hypothalamus implements allostatic regulation (Sterling, 2012), maintaining the system's internal balance across varying market conditions.

### 2.5.1 Trading Implementation

**Component:** Adaptive Position Sizing and Risk Appetite Control

- **Position Sizing:** Dynamically adjusts position sizes based on regime, portfolio heat, and recent performance using Kelly Criterion-inspired scaling
- **Homeostasis:** Monitors portfolio-level vital signs (exposure, drawdown, correlation) and applies corrective adjustments to maintain target ranges
- **Energy Management:** Tracks “metabolic” state of the trading system—capital utilization, margin usage, and recovery capacity—and modulates aggression accordingly
- **Risk Appetite:** Integrates regime signals from the ensemble detector with internal portfolio state to produce a single risk appetite scalar that modulates all downstream position sizing

The Hypothalamus sits at the second stage of the brain wiring pipeline, translating raw regime detection into calibrated risk parameters before signals reach the Amygdala and downstream modules.

## 2.6 Basal Ganglia: Action Selection & Reinforcement Learning

The basal ganglia implements Opponent Actor Learning (OpAL) (Collins and Michael J Frank, 2014; Jaskir and Michael J. Frank, 2023), balancing Go (Direct) and No-Go (Indirect) pathways modulated by dopamine (Schultz, Dayan, and Montague, 1997). Jaskir and Michael J. Frank (2023) extended OpAL with dynamic dopamine modulation,

normalized prediction errors, and uncertainty-adjusted learning rates (OpAL\*), demonstrating robust advantages in sparse reward environments and large action spaces—conditions directly analogous to financial markets. High dopamine (bull market/high confidence) amplifies the Direct pathway; low dopamine (bear market/uncertainty) amplifies the Indirect pathway. This creates dynamic risk tolerance that adapts to market volatility, implementing allostatic regulation (Sterling, 2012) rather than simple homeostatic feedback. The Hypothalamus module (Section 2.5) translates regime detection into calibrated risk parameters that modulate dopaminergic signaling throughout the decision pipeline. **No prior work applies OpAL or any basal ganglia computational model to financial decision-making.** See Section 5 for the complete mathematical formulation.

## 2.7 Prefrontal Cortex: Logic, Planning & Compliance

The prefrontal cortex provides working memory gating and logical reasoning capabilities (Michael J Frank, Loughry, and O'Reilly, 2006).

### 2.7.1 Trading Implementation

**Component:** Logic Tensor Network + Conscience Module

- Łukasiewicz fuzzy logic with linguistic hedges
- Predicate grounding and constraint satisfaction
- Wash sale rule enforcement with full 30-day window tracking
- Position limits and risk limit constraints
- Proprietary firm rule compliance (daily loss limits, trailing drawdown, consistency)
- Strategic planning with goal decomposition, subgoal generation, plan synthesis, and contingency planning
- **Quantum-Inspired Portfolio Optimization** (Mugel, Lizaso, and Orus, 2022): QAOA (Quantum Approximate Optimization Algorithm) simulator for combinatorial asset selection via QUBO formulation, VQE (Variational Quantum Eigensolver) for continuous weight optimization using parameterized circuits, and simulated quantum annealing for escaping local minima in non-convex portfolio landscapes — complementing classical Mean-Variance, Risk Parity, and Black-Litterman optimizers

## 2.8 Amygdala: Fear, Threat Detection & Circuit Breakers

The amygdala provides rapid threat detection and fear learning (LeDoux, 2000), with connections to substantia nigra enabling fear extinction (Monfils et al., 2009). The amygdala's role in encoding emotional significance and modulating memory consolidation (Caligiore et al., 2019) maps directly to JANUS's circuit-breaker and fear-network architecture.

### 2.8.1 Trading Implementation

**Component:** Multi-Layer Threat Detection and Safety System

**Anomaly Detection** uses multiple complementary methods: Z-score deviation, isolation forest scoring, moving average deviation, percentile outliers, and multivariate scoring. Threats are classified into severity levels (None → Low → Medium → High → Critical).

**Mahalanobis Distance:**

$$D_M(\mathbf{s}_t) = \sqrt{(\mathbf{s}_t - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{s}_t - \boldsymbol{\mu})} \quad (50)$$

where  $\boldsymbol{\mu}$  is the historical mean state and  $\boldsymbol{\Sigma}$  is the covariance matrix.

**Circuit Breaker Condition:**

$$\text{Trigger} = \begin{cases} 1 & \text{if } D_M(\mathbf{s}_t) > \tau_{\text{danger}} \\ 0 & \text{otherwise} \end{cases} \quad (51)$$

where  $\tau_{\text{danger}}$  is calibrated to a false-positive rate (e.g.,  $\tau = 5$  for  $p < 0.001$ ).

**Additional Threat Signals:**

- Sudden volatility spike:  $\sigma_t > 3 \cdot \sigma_{\text{baseline}}$
- Drawdown threshold: cumulative loss  $> L_{\text{max}}$
- Liquidity crisis: bid-ask spread  $> 10 \times$  normal
- Regime shift detection
- Correlation breakdown across monitored pairs
- Black swan event detection
- Flash crash detection via VPIN

**VPIN Flow Toxicity:** The VPIN calculator (Easley, Lopez de Prado, and O'Hara, 2011; Easley, De Prado, and O'Hara, 2012) uses volume bucket aggregation, bulk

volume classification, and rolling window computation to produce a toxicity score. High and critical thresholds trigger graduated responses from position reduction to full kill switch activation.

#### Production Circuit Breakers:

- **Kill Switch:** Dual-layer design with in-process `AtomicBool` for zero-latency local halt and Redis-backed distributed coordination across services
- **Position Freeze:** Prevents new position entry while allowing exits
- **Safe Mode:** Reduces system to minimal-risk operation
- **Cancel All:** Emergency cancellation of all pending orders

**Fear Extinction:** The fear learning system implements extinction mechanisms (Monfils et al., 2009; Caligiore et al., 2019), allowing the system to “unlearn” fear when threats have passed. This prevents the agent from becoming permanently paralyzed by a single traumatic market event (e.g., flash crash) while maintaining protective circuit breakers for genuine systemic risks.

## 2.9 Cerebellum: Motor Control & Execution

The cerebellum provides forward models for motor prediction, adapted here for market impact forecasting (Almgren and Chriss, 2001).

### 2.9.1 Trading Implementation

**Component:** Forward Model for Market Impact with Error Correction

- **Almgren-Chriss Model:** Full optimal execution with permanent/temporary impact coefficients, risk aversion parameter, and optimal trajectory calculation
- **VPIN Integration:** Volume-Synchronized Probability of Informed Trading for flow toxicity detection
- **Forward Models:** Adverse selection detection, Smith predictor for latency compensation, order latency estimation, and fill probability prediction
- **Error Correction:** PID controller, feedback loops, and adaptive correction for continuous execution quality improvement
- **LOB Simulator** (Fu, Pakkanen, and Cont, 2024): Full limit order book simulator with price-time priority matching engine, support for Limit, Market, IOC, FOK,

Post-Only, Iceberg, Stop, and Stop-Limit order types, self-trade prevention, maker/taker fee computation, L2/L3 snapshots, VWAP and weighted mid-price calculations, queue position estimation, configurable tick/lot sizing, and Almgren-Chriss market impact modeling — enabling parallel synthetic training data generation for reinforcement learning

Price movement from order execution:

$$\Delta p = f_{\text{cerebellum}}(\text{order\_size}, \text{liquidity}, \text{volatility}) \quad (52)$$

## 2.10 Integration: Inter-Region Coordination

The Integration module provides the “white matter” connecting brain regions, handling service bridges, cross-region coordination, and the data pipeline that routes information between the Forward, Backward, Execution, Data, and CNS services.

## Part V

# Rust Implementation

## Abstract

This document provides production-ready Rust implementation specifications for Project JANUS. The choice of Rust is strategic, prioritizing memory safety and concurrency (Matsakis and Klock, 2014; Jung et al., 2018), with zero-cost abstractions essential for nanosecond-critical high-frequency trading environments. This section includes:

- **ML Framework Strategy** leveraging Candle (Hugging Face, 2024) for end-to-end Rust-native ML
- **High-Performance Services** with async Tokio runtime (Tokio Contributors, 2024)
- **Rust-Native Training & Inference Pipeline** — full ML lifecycle in Rust
- **Neuromorphic Module** — complete brain-region implementations in pure Rust
- **Trading Strategies** — nine regime-aware strategies with gating and affinity
- **Deployment Architecture** (Docker Compose + Kubernetes)

## 1 Architectural Overview

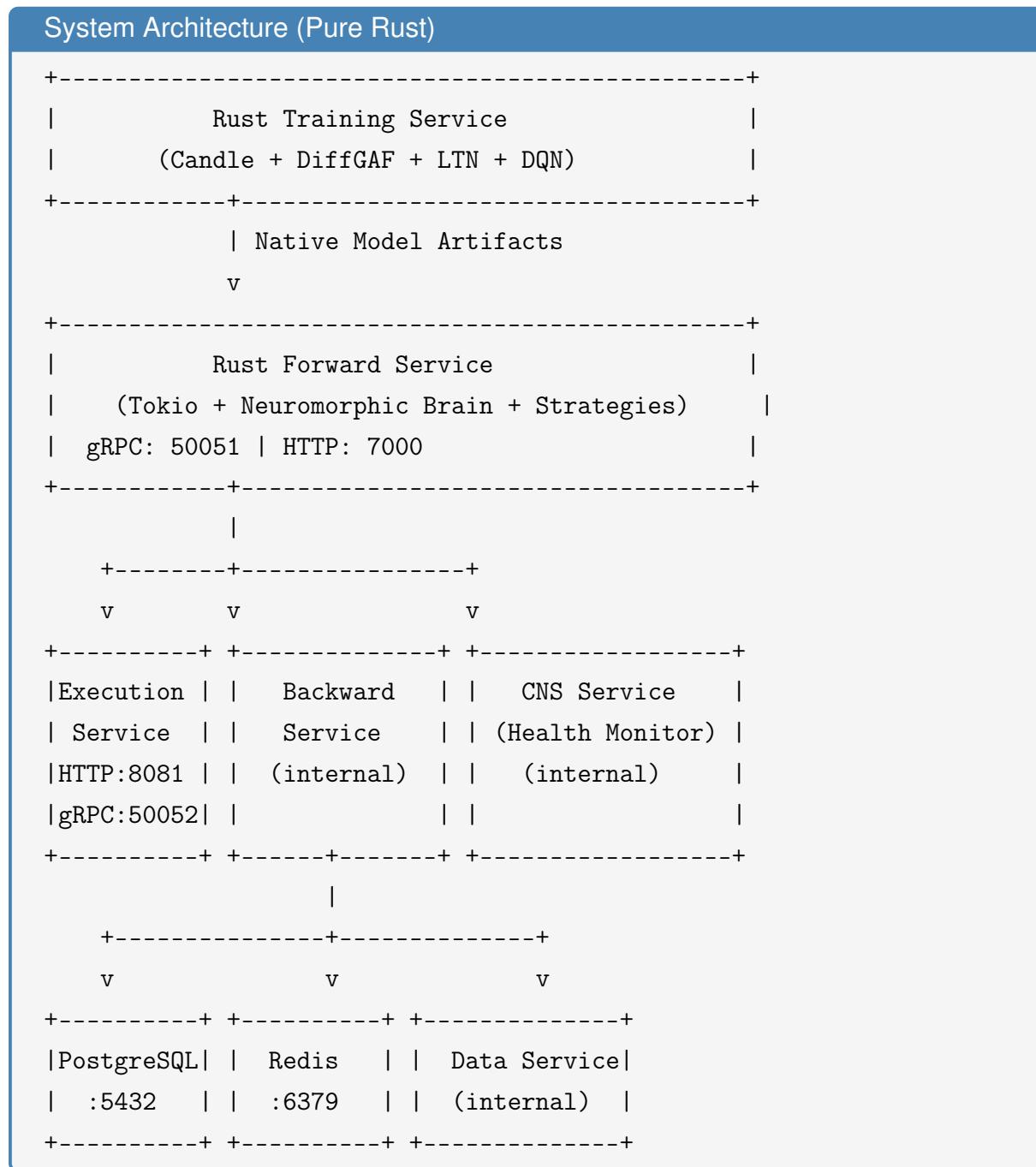
### 1.1 The Rust-Only Philosophy

The exclusive use of Rust across the entire stack—training, inference, and production services—is justified by the language’s formal safety guarantees (Matsakis and Klock, 2014; Jung et al., 2018) and the performance requirements of high-frequency trading systems:

1. **Performance:** Zero-cost abstractions, no GC pauses—critical for sub-microsecond latency
2. **Safety:** Memory safety without runtime overhead, preventing undefined behavior
3. **Concurrency:** Fearless async/await with Tokio (Tokio Contributors, 2024) for handling high-frequency websocket feeds
4. **Ecosystem:** End-to-end ML training and inference via Candle (Hugging Face, 2024)

5. **Unified Stack:** Single language for the entire pipeline eliminates cross-language serialization overhead, deployment complexity, and FFI boundary risks

## 1.2 Component Diagram



## 2 Machine Learning Framework Strategy

## 2.1 Framework Stack

The following Rust-native frameworks comprise the JANUS ML stack (Hugging Face, 2024):

Framework	Pros	Cons	Use Case
Candle (Hugging Face, 2024)	Pure Rust, HuggingFace integration, minimal deps, autograd	Younger ecosystem	Primary training & inference (DiffGAF, ViViT, LTN, DQN)
ndarray	Zero-dependency numerical arrays, mature ecosystem	No autograd	Traditional fuzzy logic, GAF encoding, feature engineering
Polars (Polars Contributors, 2024)	High-speed DataFrames	N/A	Data manipulation

## 2.2 Rust-Native ML Architecture

The system operates as a **fully Rust-native** ML pipeline with no external language dependencies:

- End-to-end training and inference in Rust via Candle with autograd support
- Custom differentiable kernels for DiffGAF and LTN operations
- Double DQN with online/target networks for reinforcement learning
- Native model serialization using `safetensors` format
- GPU acceleration via `wgpu` (Candle) and CUDA (optional)
- Zero cross-language overhead—no FFI bridges, no serialization boundaries
- Training infrastructure: AdamW/SGD optimizers, warmup+cosine LR scheduling, prioritized replay buffers

## 3 Forward Service: Rust Implementation

### 3.1 Performance Requirements

FPGA acceleration using AMD Alveo U55C cards (AMD, 2023; Vemeko, 2023; Marino et al., 2023) is planned as future work. Current targets:

- Latency: p99 < 10ms (target: < 1 $\mu$ s with FPGA)
- Throughput: 10,000 req/s
- Memory: < 2GB RSS

### 3.2 Core Data Structures

The system maintains several key data structures for real-time processing:

#### Market State Representation:

$$\mathcal{S}_t = (\tau_t, \mathbf{f}_t, \mathcal{O}_t, \mathbf{c}_t) \quad (53)$$

where:

- $\tau_t \in \mathbb{Z}^+$  is the timestamp
- $\mathbf{f}_t \in \mathbb{R}^d$  is the feature vector
- $\mathcal{O}_t = (\mathcal{B}_t, \mathcal{A}_t)$  is the order book with bids  $\mathcal{B}_t$  and asks  $\mathcal{A}_t$
- $\mathbf{c}_t$  contains contextual metadata (volatility, spreads, volume)

#### Order Book Structure:

$$\mathcal{B}_t = \{(p_i, q_i) : p_i \in \mathbb{R}^+, q_i \in \mathbb{R}^+\}_{i=1}^{N_{\text{bid}}} \quad (54)$$

$$\mathcal{A}_t = \{(p_j, q_j) : p_j \in \mathbb{R}^+, q_j \in \mathbb{R}^+\}_{j=1}^{N_{\text{ask}}} \quad (55)$$

### 3.3 GAF Transformation Algorithm

The GAF transformation converts time series to 2D images via the following algorithm (Wang and Oates, 2015). For training data generation, JANUS employs a Rust-native GPU-accelerated limit order book simulator, inspired by JAX-LOB (Fu, Pakkanen, and Cont, 2024), that enables parallel simulation of thousands of order books, solving the data scarcity problem inherent in traditional trading systems.

**Computational Complexity:**  $\mathcal{O}(W^2)$  for matrix construction, where  $W$  is the window size.

**Algorithm 1** GAF Computation

---

```

1: Input: Time series  $X = \{x_1, \dots, x_W\}$ , window size  $W$ 
2: Output: Gramian matrix  $\mathbf{G} \in \mathbb{R}^{W \times W}$ 
3:
4:  $\tilde{X} \leftarrow \text{Normalize}(X)$  to  $[-1, 1]$ 
5:  $\phi_i \leftarrow \arccos(\tilde{x}_i)$  for  $i = 1, \dots, W$ 
6: for  $i = 1$  to  $W$  do
7:   for  $j = 1$  to  $W$  do
8:      $\mathbf{G}_{ij} \leftarrow \cos(\phi_i + \phi_j)$ 
9:   end for
10: end for
11: return  $\mathbf{G}$  reshaped to  $[1, W, W]$  tensor

```

---

### 3.4 LTN Constraint Evaluation

Each constraint is represented as a weighted predicate function:

**Constraint Structure:**

$$\mathcal{C}_k = (P_k, w_k) \quad (56)$$

where  $P_k : \mathcal{S} \rightarrow [0, 1]$  is a predicate and  $w_k \in \mathbb{R}^+$  is the weight.

**Evaluation Function:**

$$\text{Eval}(\mathcal{C}_k, \mathcal{S}_t) = w_k \cdot P_k(\mathcal{S}_t) \quad (57)$$

**T-norm Operations (already defined in Part 2):**

$$a \wedge_{\mathcal{L}} b = \max(0, a + b - 1) \quad (\text{Conjunction}) \quad (58)$$

$$a \Rightarrow_{\mathcal{L}} b = \min(1, 1 - a + b) \quad (\text{Implication}) \quad (59)$$

**Total Constraint Satisfaction:**

$$\mathcal{L}_{\text{constraint}} = 1 - \frac{1}{K} \sum_{k=1}^K \text{Eval}(\mathcal{C}_k, \mathcal{S}_t) \quad (60)$$

### 3.5 Async Service Architecture

The service follows an event-driven architecture with the following characteristics:

**Request Processing Pipeline:**

1. **Initialization:** Load native Rust model  $\mathcal{M}_{\text{ViViT}}$  and LTN engine  $\mathcal{E}_{\text{LTN}}$
2. **Connection Handling:** Bind gRPC listener on port 50051 and HTTP on port 7000
3. **Concurrent Processing:** For each incoming request:

- Spawn asynchronous task with model clone
- Process request independently (non-blocking)
- Return prediction and constraint satisfaction scores

### Concurrency Model:

$$\text{Throughput} = \frac{N_{\text{workers}} \times 1000}{T_{\text{avg}}} \quad (61)$$

where  $N_{\text{workers}}$  is the thread pool size and  $T_{\text{avg}}$  is average processing time in ms.

**Performance Characteristics:** Non-blocking I/O via `async/await`, zero-copy model sharing across tasks, and bounded memory through connection limiting.

## 4 Backward Service: Batch Processing

### 4.1 Prioritized Experience Replay

The replay buffer maintains experiences with importance-based sampling.

#### Buffer State:

$$\mathcal{B} = \{(e_i, p_i)\}_{i=1}^N \quad (62)$$

where  $e_i$  is an experience and  $p_i \in \mathbb{R}^+$  is its priority.

#### Hyperparameters:

- $\alpha \in [0, 1]$ : Priority exponent (0 = uniform, 1 = full prioritization)
- $\beta \in [0, 1]$ : Importance sampling correction
- $C$ : Buffer capacity

---

#### Algorithm 2 Prioritized Experience Sampling

- 1: **Input:** Buffer  $\mathcal{B}$ , batch size  $B$
  - 2: **Output:** Sampled batch  $\{e_{i_1}, \dots, e_{i_B}\}$
  - 3:
  - 4: Compute probabilities:  $P(i) = \frac{p_i^\alpha}{\sum_j p_j^\alpha}$
  - 5: **for**  $k = 1$  to  $B$  **do**
  - 6:     Sample index  $i_k \sim \text{Categorical}(P)$
  - 7:     Add  $e_{i_k}$  to batch
  - 8: **end for**
  - 9: **return** batch
- 

#### Importance Weights:

$$w_i = \left( \frac{1}{N \cdot P(i)} \right)^\beta \quad (63)$$

These weights correct for the non-uniform sampling distribution.

## 4.2 Schema Consolidation Algorithm

Schemas are formed by clustering experience embeddings and storing centroids.

---

### Algorithm 3 Schema Update

---

- 1: **Input:** Experiences  $\mathcal{E} = \{e_1, \dots, e_N\}$ , number of clusters  $K$
  - 2: **Output:** Updated schema database
  - 3:
  - 4: Extract embeddings:  $\mathbf{h}_i = \text{Embed}(e_i)$  for  $i = 1, \dots, N$
  - 5: Cluster:  $\mathcal{C} \leftarrow \text{K-means}(\{\mathbf{h}_i\}, K)$
  - 6: **for**  $k = 1$  to  $K$  **do**
  - 7:     Compute centroid:  $\mathbf{z}_k = \frac{1}{|C_k|} \sum_{i \in C_k} \mathbf{h}_i$
  - 8:     Compute statistics:
  - 9:          $n_k = |C_k|$
  - 10:          $\bar{r}_k = \frac{1}{|C_k|} \sum_{i \in C_k} r_i$  (average reward)
  - 11:         **Upsert** schema  $k$  with vector  $\mathbf{z}_k$  and metadata  $(n_k, \bar{r}_k)$
  - 12: **end for**
- 

**Note:** In production, the primary schema classification uses deterministic feature-range matching for interpretability. The K-means clustering algorithm above is used for schema *formation* from accumulated experiences during deep consolidation phases.

**Schema Metadata:** Each schema  $k$  stores:

- Centroid vector  $\mathbf{z}_k \in \mathbb{R}^d$
- Member count  $n_k$
- Average reward  $\bar{r}_k$
- Volatility  $\sigma_k$  (standard deviation of returns)

**K-means Objective:**

$$\min_{\mathcal{C}} \sum_{k=1}^K \sum_{i \in C_k} \|\mathbf{h}_i - \mathbf{z}_k\|^2 \quad (64)$$

## 5 Execution Service

The Execution Service is a dedicated microservice responsible for multi-exchange order routing, providing a clean separation between decision-making (Forward Service) and order management.

### 5.1 Multi-Exchange Support

The service supports multiple cryptocurrency exchanges through a unified adapter interface (exchanges crate), with exchange-specific adapters handling authentication, rate limiting, and order format translation:

- **Kraken:** Full REST and WebSocket adapter with order normalization (primary exchange)
- **Bybit:** Dedicated client crate (`bybit-client`) with unified order types
- **Coinbase:** REST adapter with authentication and rate limiting
- **OKX:** REST adapter with order format translation
- **Binance:** Legacy connector maintained for backward compatibility
- **Kucoin:** Legacy connector maintained for backward compatibility

Each adapter implements a common `ExchangeAdapter` trait, enabling transparent exchange selection at runtime. The `normalizer` module translates exchange-specific order responses into canonical JANUS types.

## 5.2 Service Interface

- HTTP API on port 8081 for order management
- gRPC on port 50052 for low-latency inter-service communication
- Standardized order lifecycle (place, modify, cancel, query)
- Circuit breaker per exchange with configurable failure thresholds and recovery timeouts
- Rate limiter with token bucket and sliding window algorithms

## 5.3 Execution Algorithms

The system provides production execution algorithms:

- **TWAP:** Time-Weighted Average Price execution with configurable slice intervals
- **VWAP:** Volume-Weighted Average Price execution with volume profile estimation
- **Iceberg:** Hidden large orders by exposing small visible tip orders to minimize market impact
- **Execution Analytics:** Per-venue latency tracking, fill rate monitoring, and slippage analysis

## 6 CNS Service: System Health Monitoring

The Central Nervous System (CNS) Service provides system-wide health monitoring and operational safety, analogous to the autonomic nervous system's regulation of vital functions.

### 6.1 Preflight Validation

Before entering live or paper trading, the CNS Service executes a five-phase preflight sequence, each with configurable criticality levels (Critical, Required, Optional):

1. **Infrastructure Phase:** Database connectivity, Redis availability, Qdrant health, shared memory paths
2. **Sensory Phase:** Exchange API authentication, WebSocket feed health, market data freshness
3. **Regulatory Phase:** Kill switch state verification, compliance rule loading, wash sale detector initialization
4. **Strategy Phase:** Model file availability, version consistency, strategy affinity configuration
5. **Executive Phase:** Memory and CPU resource adequacy, neuromorphic brain region initialization

Critical failures abort the boot sequence; required failures block trading but allow monitoring; optional failures are logged but permit full operation. The `PreFlightRunner` supports both sequential and parallel-within-phase execution modes, producing a comprehensive `BootReport` with Discord-formatted notifications.

### 6.2 Runtime Monitoring

During operation, the CNS Service provides:

- **Watchdog Monitoring:** Heartbeat-based component liveness detection with configurable degraded/dead thresholds, per-component criticality levels (Critical, Important, NonEssential), and automatic kill switch triggering when critical components die
- **Boot Reports:** Comprehensive system state summary with pass/fail/skip counts per phase

- **Prometheus Metrics:** 15+ custom metrics exposed for Grafana dashboards (6+ pre-built dashboards)
- **Alert Integration:** Slack, PagerDuty, and generic webhook notifications with severity-based routing
- **Distributed Tracing:** Jaeger integration for cross-service latency analysis
- **Circuit Breakers:** Per-component circuit breakers with Closed → Open → HalfOpen state machine, configurable failure windows, and recovery timeouts
- **Reflex Actions:** Automated responses including component restart, throttling, graceful shutdown, and safe command execution with allowlist validation
- **Neuromorphic Brain Coordinator:** Topological initialization ordering of all 10 brain regions based on dependency graphs, per-region health scoring, and global brain activation/deactivation

## 7 Trading Strategies

The strategies crate implements nine regime-aware trading strategies, each designed for specific market conditions as identified by the ensemble regime detector:

### 7.1 Trend-Following Strategies

- **EMA Flip:** 8/21 EMA crossover with ATR-based stops, trading pullbacks to the fast EMA in the trend direction
- **EMA Ribbon Scalper:** 8/13/21 EMA ribbon with volume confirmation for higher-quality pullback entries
- **Trend Pullback:** Fibonacci retracement entries within established trends, confirmed by RSI divergence and candlestick patterns (pin bars, engulfing)
- **Momentum Surge:** Detects sudden price surges with volume spikes, entering on the first pullback within the surge
- **Multi-Timeframe Trend:** EMA 50/200 crossover with ADX strength and higher-timeframe alignment

## 7.2 Mean-Reversion Strategies

- **Mean Reversion:** Bollinger Bands with RSI confirmation and ATR-based stops
- **Bollinger Squeeze Breakout:** Detects low-volatility squeeze periods and generates breakout signals when price escapes the bands
- **VWAP Scalper:** Mean reversion scalping around the Volume-Weighted Average Price with standard deviation bands
- **Opening Range Breakout:** Trades breakouts above or below the first N candles of a session with volume confirmation

## 7.3 Strategy Gating and Affinity

The StrategyGate module controls which strategies execute based on:

- **Regime Compatibility:** Each asset can define preferred strategies per regime (e.g., Trending → EMA Flip, MeanReverting → Bollinger Squeeze)
- **Affinity Scoring:** Strategy-asset affinity tracker with performance-weighted scoring
- **Allowlists/Denylists:** Per-asset strategy filtering via TOML configuration
- **Untested Strategy Policy:** Configurable flag to allow or block strategies without historical performance data

## 8 Neuromorphic Module: Brain-Region Implementations

The `neuromorphic` crate is the largest module in the JANUS codebase, implementing all ten brain regions as production Rust code. Each region is a self-contained submodule with its own configuration, state management, and comprehensive test suite.

## Neuromorphic Crate Structure

```

neuromorphic/
  +-+ visual_cortex/      # GAF, ViViT, preprocessing, UMAP viz
  +-+ cortex/             # Schemas, knowledge base, planning
  |   +-+ memory/         # 7 regime schemas + Markov transitions
  +-+ hippocampus/        # Episodic buffer, SWR, consolidation
  |   +-+ swr/             # Ripple detection, compressed replay
  +-+ thalamus/            # Attention, fusion, gating, routing
  |   +-+ fusion/          # Orderbook, price, volume, sentiment
  +-+ hypothalamus/        # Position sizing, homeostasis, energy
  |   +-+ position_sizing/ # Drawdown scaling, Kelly criterion
  +-+ basal_ganglia/       # Direct/Indirect pathways, OpAL
  |   +-+ praxeological/  # Go/No-Go signals, confidence
  +-+ amygdala/            # Threat detection, VPIN, kill switch
  |   +-+ vpin/            # Calculator, flash crash, toxicity
  +-+ cerebellum/           # Forward models, error correction
  |   +-+ error_correction/ # PID controller, feedback loops
  |   +-+ forward_models/  # Smith predictor, fill prob
  +-+ prefrontal/           # LTN, fuzzy logic, conscience, goals
  |   +-+ ltn/              # Hedges: very, somewhat, extremely
  +-+ integration/          # Coordinator, message bus, bridges
  |   +-+ engine/            # Cognitive core, orchestrator
  +-+ distributed/           # Multi-node coordination

```

## 8.1 Key Implementation Details

**Basal Ganglia — Opponent Actor Learning (OpAL):** The praxeological module implements the full Go/No-Go architecture with dopamine-modulated action selection. The GoSignal evaluates action value against adaptive thresholds with dopamine sensitivity, urgency boosting, and facilitation bias. The NoGoSignal evaluates 12 inhibition reasons (risk threshold, position limit, loss limit, high volatility, low liquidity, cooling off, correlation risk, drawdown protection, time restriction, external halt, learned pattern, and custom) with learned inhibition patterns and adaptive thresholds. The actor-critic framework includes Generalized Advantage Estimation (GAE) and TD( $\lambda$ ) learning for stable policy updates, with winner-take-all selection and habit caching for frequently-encountered states.

**Cerebellum — Forward Models and Error Correction:** The almgren\_chriss module implements optimal execution trajectories with permanent/temporary impact coefficients. The pid\_controller provides a full PID implementation with anti-windup,

dead band, derivative filtering, cascaded PID, and Ziegler-Nichols auto-tuning. The `forward_models` submodule includes adverse selection detection, Smith predictor for latency compensation, order latency estimation, and fill probability prediction.

**Amygdala — VPIN and Kill Switch:** The `vpin` module implements volume-synchronized probability of informed trading with bulk volume classification, rolling bucket computation, and configurable high/critical thresholds. The `kill_switch` provides a four-scope design (per-strategy, per-instrument, per-service, global) with emergency actions (cancel all orders, close all positions, disable trading, send alerts). The fear network integrates reinforcement learning (FNI-RL) to adapt threat responses based on historical outcomes.

**Cortex — Schema Formation:** The `schemas` module implements all seven regime schemas (Bull, Bear, Ranging, Crisis, Recovery, Bubble, Deflation) with 10 market features (trailing return, realised volatility, average correlation, max drawdown, vol rate of change, momentum signal, mean reversion signal, credit spread, yield curve slope, relative volume), weighted feature-range matching, a full Markov transition matrix with stationary distribution computation, and EMA-smoothed confidence tracking. The cortex also implements a hierarchical RL manager with feudal goal-setting and subgoal generation for multi-level strategic planning.

**Prefrontal — Fuzzy Logic with Linguistic Hedges:** The `ltn` submodule implements the complete Łukasiewicz fuzzy logic system with five linguistic hedges: very ( $x^2$ , concentration), somewhat ( $\sqrt{x}$ , dilation), slightly ( $\sqrt{x} - x$ ), extremely ( $x^3$ ), and `more_or_less` ( $\sqrt{x}$ , synonym for dilation). A full expression evaluator supports nested hedge application.

**Thalamus — Multimodal Fusion:** Four specialized fusion engines process distinct data streams: `orderbook_fusion` (multi-venue book consolidation with weighted mid-price, imbalance detection, and EMA smoothing), `price_fusion` (multi-timeframe price action), `volume_fusion` (volume profile analysis), and `sentiment_fusion` (news and social sentiment signals). The `gating` submodule implements Wilson-Cowan oscillatory dynamics (H. R. Wilson and Cowan, 1972; Y. Zhang et al., 2024) with Hilbert-transform-based amplitude/phase estimation, bifurcation analysis, and neuromodulator-driven regime switching, alongside sensory gates with relevance scoring and threshold-based filtering. The `sources` submodule integrates Chronos time series forecasting (Ansari et al., 2024b; Ansari et al., 2024a) via ONNX inference with quantile tokenization and confidence intervals, BERT/FinBERT sentiment embeddings (Hugging Face, 2024) running natively via Candle with Qdrant-backed storage, plus news feeds, weather data, and celestial/space weather for commodity correlation analysis.

**Hippocampus — Sharp-Wave Ripple Simulation:** The `swr` submodule implements 11 ripple types (large profit, large loss, novel pattern, market anomaly,

strategy breakthrough, risk event, regime change, correlation breakdown, volatility spike, liquidity event, periodic) with priority-weighted detection, inspired by biological SWR replay (M. A. Wilson and Bruce L. McNaughton, 1994; Buzsáki, 2015). The compressed\_replay module provides compressed experience sequences, and consolidation\_sync coordinates the transfer from hippocampal to neocortical storage, implementing CLS theory (Kumaran, Hassabis, and James L. McClelland, 2016). The hippocampus also houses a feudal RL worker agent with a skill library and tactical policy for executing subgoals issued by the cortex's strategic planner. A regime-conditional DDPM diffusion model (Coletta et al., 2024) generates synthetic market data for training data augmentation, with configurable noise schedules (linear, cosine, quadratic), Min-SNR loss weighting, and automated quality assessment comparing synthetic vs. real feature distributions.

**Distributed Training Infrastructure:** The distributed submodule provides multi-GPU and multi-node training coordination with AllReduce, Parameter Server, and Ring AllReduce gradient synchronization strategies. Distributed data loading supports multiple sharding strategies (Contiguous, RoundRobin, Random, Stratified). The infrastructure includes NCCL GPU-to-GPU communication, gRPC-based inter-node coordination, and distributed checkpointing with cloud storage backends.

#### GPU Compute Infrastructure:

**GPU Compute Infrastructure:** The gpu submodule provides custom wgpu compute kernels for hardware-accelerated numerical operations: MatMul, Softmax, LayerNorm, Attention, GELU, Reduce, Embedding Lookup, and Pairwise Distance. The module includes a device manager, buffer pool, kernel registry, and a Candle tensor bridge for CPU $\leftrightarrow$ GPU data transfer. All GPU functionality is gated behind a gpu feature flag, with graceful CPU fallback when unavailable.

**Prefrontal — Quantum-Inspired Portfolio Optimization:** The planning submodule includes quantum-inspired portfolio optimizers (Mugel, Lizaso, and Orus, 2022): a QAOA (Quantum Approximate Optimization Algorithm) simulator for combinatorial asset selection via QUBO formulation, a VQE (Variational Quantum Eigensolver) portfolio optimizer for continuous weight optimization using parameterized circuits, and simulated quantum annealing for escaping local minima in non-convex portfolio landscapes. These complement classical Mean-Variance, Risk Parity, and Black-Litterman optimizers.

**Cerebellum — LOB Simulator:** The lob\_simulator module provides a full limit order book simulator (Fu, Pakkanen, and Cont, 2024) with price-time priority matching, support for Limit, Market, IOC, FOK, Post-Only, Iceberg, Stop, and Stop-Limit order types, self-trade prevention, maker/taker fee computation, L2/L3 snapshots, VWAP and weighted mid-price calculations, Almgren-Chriss market impact estimation, order event history, and configurable tick/lot sizing. The simulator enables parallel training

data generation, solving the data scarcity problem for reinforcement learning.

## 8.2 Supporting Crates

The modular Rust workspace includes dedicated crates:

- **training:** End-to-end training loop with AdamW/SGD optimizers, learning rate schedulers (warmup+cosine, step, exponential), prioritized experience replay with SWR sampling, gradient clipping, checkpointing with metadata, early stopping, and pluggable callbacks
- **ml:** LSTM and MLP models via Candle, Double DQN with online/target networks and soft updates, feature engineering (price, volume, technical indicators, normalizers), and dataset management
- **logic:** Dual-mode LTN system—non-differentiable (ndarray) for inference-time constraint checking and differentiable (Candle) for gradient-based training—with three t-norm families ( $\bar{\text{Łukasiewicz}}$ , Product, Gödel), learnable/threshold/similarity predicates, and comprehensive rule composition (ForAll, Exists, Implies, Iff, AndN, OrN)
- **Itn:** Domain-specific neuro-symbolic engine with 10 market axioms encoded as logical rules (e.g., Trending + Positive Divergence  $\rightarrow$  Long, Low Confidence  $\rightarrow$  Neutral), integrating DSP features through fuzzy predicates with a hybrid supervised-semantic loss function
- **regime:** Ensemble regime detection combining HMM (Gaussian emissions, Baum-Welch, online parameter updates), indicator-based (ADX, Bollinger Bands, ATR, EMA, RSI), and ensemble fusion with agreement tracking and strategy routing
- **dsp:** Digital signal processing with FRAMA (Fractal Adaptive Moving Average) and Sevcik fractal dimension estimation producing an 8D feature vector (divergence, alpha, fractal dimension, Hurst exponent, regime, sign, deviation, confidence) at sub-microsecond latency per tick
- **lob:** Standalone limit order book crate with matching engine, fill probability models, L2 replay, latency simulation, and Almgren-Chriss market impact estimation
- **compliance:** Dedicated WashSaleDetector with full 30-day lookback/lookforward window, partial wash sale handling, cost basis tracking, and position-aware detection; plus ComplianceSheriff enforcing proprietary firm trading rules (daily loss limits, maximum loss thresholds, mandatory stop-losses)
- **optimizer:** Hyperparameter optimization with configurable samplers, constraints, backtesting integration, and result publishing

- **data-quality:** Market data validation and anomaly detection
- **gap-detection:** Multi-layer time series gap detection (sequence ID, heartbeat, statistical, volume-aware) with PostgreSQL persistence
- **indicators:** Technical indicator library (ADX, ATR, Bollinger Bands, EMA, RSI, MACD) with incremental O(1) per-tick computation
- **rate-limiter:** Token bucket and sliding window rate limiters with async circuit breaker, exchange-specific algorithms
- **questdb-writer:** High-performance time series ingestion to QuestDB via ILP, batched writes targeting >100K inserts/sec
- **memory:** Three-tier memory hierarchy with production Qdrant vector database client (circuit breaker, exponential backoff retries, TLS, mock fallback), predefined collections for market regimes, episodic memory, sentiment embeddings, and schema prototypes, plus prioritized experience replay buffer

## 9 Deployment Architecture

### 9.1 Service Orchestration

The system deploys as eight services plus supporting infrastructure:

#### Service Topology:

##### 1. Forward Service:

- Ports: gRPC 50051, HTTP 7000
- Dependencies: Native model artifacts, Redis, PostgreSQL
- Resource limits: 2GB memory, 2 CPU cores

##### 2. Backward Service:

- Internal service (no external ports)
- Dependencies: PostgreSQL, Redis
- Scheduling: Triggered during market close

##### 3. Execution Service:

- Ports: HTTP 8081, gRPC 50052
- Dependencies: Exchange API credentials, Redis (kill switch)
- Multi-exchange order routing

**4. Data Service:**

- Internal service for centralized market data management
- Dependencies: Exchange WebSocket feeds, QuestDB (time series)

**5. CNS Service:**

- Internal service for health monitoring and preflight validation
- Dependencies: All other services (monitoring target)

**6. API Service:**

- HTTP API gateway for external access
- Dependencies: Forward Service, Backward Service

**7. Registry Service:**

- Internal service discovery and registration
- Dependencies: Redis

**8. Optimizer Service:**

- Hyperparameter optimization with backtesting integration
- Metrics port: 9092

**Infrastructure:**

- **PostgreSQL** (port 5432): Primary relational storage for signals, portfolios, and performance
- **Redis** (port 6379): Operational state, kill switch coordination, hot-reloadable config
- **QuestDB** (ports 9000/9009): High-performance time series storage for market data
- **Qdrant**: Vector similarity search engine for schema pattern matching
- **Prometheus** (port 9090): Metrics collection with 1700+ lines of alert rules
- **Grafana** (port 3000): Visualization dashboards (6+ dashboards including strategy, regime, CNS, brain region monitors)
- **Alertmanager** (port 9093): Alert routing with Discord integration
- **Jaeger** (port 16686): Distributed tracing

- **Loki + Promtail**: Centralized log aggregation with label-based organization
- **Authelia**: OpenID Connect / SAML authentication gateway
- **Nginx**: Reverse proxy with SSL termination

### Service Communication:

Forward  $\xrightarrow[\text{gRPC}]{\text{signals}}$  Execution  $\xrightarrow[\text{HTTP}]{\text{orders}}$  Exchanges (65)

Forward  $\xrightarrow[\text{SQL}]{\text{experiences}}$  PostgreSQL  $\xrightarrow[\text{nightly}]{\text{batch}}$  Backward  $\xrightarrow[\text{SQL}]{\text{schemas}}$  PostgreSQL (66)

**Volume Management:** Model artifacts are stored on shared read-only volumes; PostgreSQL and QuestDB use persistent volumes with automated backups; Redis data is ephemeral with configurable persistence; experience buffers rotate daily.

## Part VI

# Limitations and Open Problems

## Abstract

No system of JANUS’s complexity can be presented without candid engagement with its limitations. This section addresses the challenges that peer reviewers will—and should—raise, and describes mitigation strategies where they exist.

## 1 The Brain Metaphor Critique

Every generation projects its latest technology onto the brain: clocks, telephone switchboards, computers, and now neural networks. Critics may argue that JANUS’s brain-region mapping is the latest instance of this pattern. We acknowledge the validity of this concern and offer two defenses:

1. **Functional, not literal:** Following Hassabis et al. (2017), JANUS uses neuroscience as *functional inspiration*, not biological simulation. The mappings are justified by computational principles (e.g., reinforcement learning for action selection, supervised error correction for timing) rather than cellular fidelity.
2. **Scientifically productive:** The Doya framework (Doya, 1999) has generated falsifiable predictions confirmed by neuroimaging. The super-learning hypothesis (Caligiore et al., 2019) demonstrates that multi-region integration produces capabilities exceeding single-region models. JANUS’s architecture is grounded in this validated framework.

## 2 Integration Complexity

Glasmachers (2017) demonstrated experimentally that end-to-end training can fail even for small multi-module systems due to non-trivial couplings between components. JANUS integrates ten brain-region modules, creating substantial coupling risk.

**Mitigations:** (i) Modular training: each brain region can be trained independently before integration. (ii) Service boundaries: the Forward and Backward services communicate via gRPC, creating natural decoupling points. (iii) The brain wiring pipeline (Section 5) defines explicit information flow, preventing uncontrolled coupling.

**Honest assessment:** Integration testing at scale remains an open challenge. Ablation studies (Part VII) are needed to quantify inter-region dependencies.

### 3 Non-Stationarity and Regime Shifts

Financial environments are fundamentally non-stationary. RL algorithms optimized for one regime may fail catastrophically in the next (Padakandla et al., 2020). JANUS’s ensemble regime detector is an important mitigation but is itself a heuristic—novel regimes not represented in training data may not be correctly classified.

**Mitigations:** (i) Three-timescale memory allows rapid adaptation at short timescales while preserving long-term knowledge (Masset et al., 2025). (ii) Brain-inspired replay (Ven, Siegelmann, and Tolias, 2020) helps mitigate catastrophic forgetting. (iii) The hypothalamic homeostatic regulator provides regime-independent risk controls.

**Honest assessment:** No existing approach fully solves the stability–plasticity dilemma in non-stationary environments. JANUS’s multi-timescale architecture is theoretically motivated but requires empirical validation across diverse market conditions.

### 4 Backtest Overfitting Risk

Bailey et al. (2015) established the Combinatorially Symmetric Cross-Validation (CSCV) framework showing that conventional backtesting dramatically overstates expected performance. López de Prado (2018b) warned that “ML algorithms will always find a pattern, even if there is none.” With ten brain regions and numerous hyperparameters, JANUS is particularly susceptible to overfitting.

**Mitigations:** (i) Walk-forward validation protocol described in Part VII. (ii) LTN constraints provide domain knowledge that regularizes against spurious patterns. (iii) The CNS service monitors for performance degradation relative to out-of-sample benchmarks.

### 5 Crowding Resistance Is Not Crowding Immunity

As discussed in Section IV, the literature supports heterogeneity as a mechanism for crowding *resistance*, not crowding *immunity*. Key limitations:

- Khandani and Lo (2011) showed that independently developed strategies converged on similar factors in 2007 despite nominal diversity
- Evolutionary dynamics can cause heterogeneous populations to converge over time if selection pressure favors similar strategies
- Tail events may produce correlated liquidation regardless of strategy diversity (Caccioli et al., 2014)

**Mitigations:** (i) Active monitoring of cross-instance correlation. (ii) Periodic re-randomization of heterogeneity parameters. (iii) Multi-agent LOB simulation for ongoing validation (Part VII).

## 6 LTN Scalability

Wan et al. (2024) found that LTN workloads exhibit a fundamental hardware utilization challenge: symbolic operations are memory-bounded while neural operations are compute-bounded, creating inefficiencies on standard hardware. As the axiom count grows, LTN inference latency may conflict with hot-path requirements.

**Mitigations:** JANUS implements a dual-mode LTN system: strict mode (full axiom evaluation) for the cold path and approximate mode (reduced axiom set with cached evaluations) for the hot path.

## 7 Adversarial Robustness

Goldblum et al. (2021) demonstrated that adversarial traders can fool automated ML systems by placing adversarial orders on public exchanges. JANUS's neural components—particularly the visual pattern recognition and sentiment analysis modules—are vulnerable to such attacks.

**Mitigations:** (i) The amygdala's multi-method anomaly detection provides some defense against adversarial inputs. (ii) The FNI-RL fear network can learn to recognize and avoid adversarial patterns. (iii) LTN constraints provide a logic-based floor that adversarial perturbations cannot violate.

**Honest assessment:** Adversarial robustness in financial ML is an open research problem. JANUS's defenses are heuristic, not provably robust.

## Part VII

# Validation Framework and Experimental Design

## Abstract

For academic peer review, empirical claims require a rigorous experimental protocol. This section describes the planned validation framework, including multi-agent simulation, walk-forward backtesting, ablation studies, and benchmark comparisons.

## 1 Multi-Agent LOB Simulation Protocol

The primary validation of JANUS's crowding-resistance claim requires multi-agent simulation using the Rust-native GPU-accelerated limit order book simulator (Fu, Pakkanen, and Cont, 2024).

### Experimental conditions:

- **Condition A (Homogeneous):** 50–200 identical JANUS instances with shared parameters trading simultaneously
- **Condition B (Heterogeneous):** 50–200 JANUS instances with engineered heterogeneity (Table 1)
- **Control:** Background liquidity provided by zero-intelligence agents (Farmer, Patelli, and Zovko, 2005)

### Metrics:

- Return kurtosis (tail risk under crowding)
- Net signed volume vs. price move (co-impact measurement)
- Preservation of the square-root impact law (Tóth et al., 2011)
- Cross-instance return correlation
- Effective market capacity (Sharpe ratio degradation as  $N$  increases)

## 2 Walk-Forward Backtesting

Following Bailey et al. (2015) and López de Prado (2018a), the backtesting protocol uses:

1. **Combinatorially Symmetric Cross-Validation (CSCV):** Partition historical data into  $S$  subsets, form all  $\binom{S}{S/2}$  train/test splits, and compute the probability of back-test overfitting
2. **Out-of-sample Sharpe ratio degradation:** Compare in-sample to out-of-sample Sharpe ratios across multiple walk-forward windows
3. **Regime-conditional performance:** Disaggregate performance by detected regime to identify regime-specific overfitting

## 3 Ablation Studies

Ablation studies isolate the contribution of each component:

- **Brain region ablation:** Remove each brain region individually and measure performance degradation, directly demonstrating the value of multi-region integration
- **Memory tier ablation:** Compare the three-tier CLS memory against single-tier experience replay
- **LTN ablation:** Remove symbolic constraints and measure constraint violation rates and risk-adjusted returns
- **Heterogeneity ablation:** Compare homogeneous vs. heterogeneous multi-agent performance (overlaps with LOB simulation)
- **Regime detection ablation:** Remove the ensemble regime detector and measure adaptation speed to market transitions

## 4 Benchmark Comparisons

- **Standard RL baselines:** PPO, SAC, and TD3 on the same market data and reward function
- **Rule-based systems:** Classic momentum, mean-reversion, and factor-based strategies (Quant 1.0/2.0 comparisons)

- **Single-region systems:** Pure basal ganglia RL without cerebellar timing, pure replay without CLS consolidation
- **Existing neuromorphic approaches:** SNN-based portfolio optimization (Mohan et al., 2025) where feasible

## Conclusion

Project JANUS represents a paradigm shift in quantitative trading: from opaque black boxes to transparent, brain-inspired systems that combine the best of deep learning and symbolic reasoning. Grounded in the Doya–Hassabis framework (Doya, 1999; Hassabis et al., 2017) and motivated by the growing threat of strategy crowding (Bucci, Mastromatteo, et al., 2020; Khandani and Lo, 2011), JANUS is—to our knowledge—the first trading system to map multiple brain regions to distinct functional subsystems within a unified architecture.

## Key Innovations

1. **Neuromorphic Architecture:** Ten functionally motivated brain regions with modular design, distributed training infrastructure, and Wilson-Cowan oscillatory dynamics (Doya, 1999; Caligiore et al., 2019)
2. **Crowding Resistance:** Engineered heterogeneity across deployments reducing co-impact costs and cross-instance correlation (Wagner, 2011; DeMiguel, Martin-Utrera, and Uppal, 2021)
3. **Neuro-Symbolic Fusion:** Dual-mode LTN system with domain-specific axioms and linguistic hedges bridging neural networks and logical constraints (Badreddine et al., 2022)
4. **Multi-Timescale Memory:** Three-tier hierarchy with sleep-phase consolidation mirrors hippocampal–neocortical transfer, implementing CLS theory (Kumaran, Hassabis, and James L. McClelland, 2016; Masset et al., 2025)
5. **OpAL Decision Engine:** First application of basal ganglia computational models to financial action selection (Collins and Michael J Frank, 2014; Jaskir and Michael J. Frank, 2023)
6. **Ensemble Regime Detection:** HMM, statistical, and technical methods fused for robust market state identification with strategy routing
7. **Production-Ready Safety:** Four-scope kill switch, multi-method anomaly detection, FNI-RL fear network, and CNS health monitoring with five-phase preflight
8. **Pure Rust Stack:** End-to-end training and inference in Rust (Matsakis and Klock, 2014) with high-performance, safe, and maintainable eight-service architecture
9. **Fractal Signal Processing:** Sub-microsecond DSP pipeline producing 8D feature vectors via FRAMA and Sevcik fractal dimension for regime-aware signal generation

10. **GPU Compute Infrastructure:** Custom wgpu kernels (MatMul, Softmax, LayerNorm, Attention, GELU) with Candle tensor bridge for hardware-accelerated inference

## Future Work

The following items remain as planned enhancements beyond the current implementation:

- Hardware acceleration using FPGAs (Marino et al., 2023; Vemeko, 2023; AMD, 2023) and neuromorphic chips for nanosecond-level latency in high-frequency trading applications
- Execution of the full validation framework described in Part VII
- Investigation of adversarial robustness guarantees beyond heuristic defenses

## Implementation Status

The following components have been implemented since initial specification:

Component	Description
Flow Toxicity	VPIN calculator with flash crash detection
Risk Control	Full PID controller with Ziegler-Nichols auto-tuning
Compliance	Wash sale detection, proprietary firm rule enforcement
Strategies	Nine regime-aware strategies with gating and affinity scoring
Safety	Four-scope kill switch, FNI-RL fear network
Training	Multi-GPU/multi-node with AllReduce, Parameter Server, Ring AllReduce
Feudal RL	Cortex manager and hippocampus worker agents
LTN	10 market axioms with hybrid supervised-semantic loss
Regime	Markov transition matrix with stationary distribution
DSP	8D feature vector pipeline with sub-microsecond latency
Vision	End-to-end DiffGAF → ViViT pipeline with dual-GAF
CLS Memory	Three-tier hierarchy with sleep-phase consolidation
Thalamus	Wilson-Cowan oscillation models (H. R. Wilson and Cowan, 1972; Y. Zhang et al., 2024) with Hilbert-transform estimation
Forecasting	Chronos (Ansari et al., 2024b; Ansari et al., 2024a) via ONNX inference
NLP	BERT/FinBERT sentiment via Candle (Hugging Face, 2024) with Qdrant storage
Visualization	Parametric UMAP (McInnes, Healy, and Melville, 2018b) with drift detection
Quantum	QAOA, VQE, simulated annealing (Mugel, Lizaso, and Orus, 2022)
Synthetic Data	Regime-conditional DDPM diffusion models (Coletta et al., 2024)
LOB Simulator	GPU-accelerated matching engine (Fu, Pakkanen, and Cont, 2024) with Almgren-Chriss impact
GPU Kernels	Custom wgpu (MatMul, Softmax, LayerNorm, Attention, GELU, Reduce)
Qdrant Client	Circuit breaker, exponential backoff, TLS, mock fallback

## Repository & Contact

**GitHub:** <https://github.com/nuniesmith/fks>

For implementation code, updates, and discussions, visit the repository.

*“The god of beginnings and transitions, looking simultaneously to the future and the past.”*

## References

- Alayrac, Jean-Baptiste, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. (2022). "Flamingo: A visual language model for few-shot learning". In: *Advances in Neural Information Processing Systems*. Vol. 35. Introduces gated cross-attention for multimodal fusion.
- Almgren, Robert and Neil Chriss (2001). "Optimal execution of portfolio transactions". In: *Journal of Risk* 3, pp. 5–40.
- AMD (2023). *AMD Alveo U55C data center accelerator card*. URL: <https://www.amd.com/en/products/accelerators/alveo/u55c.html>.
- Ansari, Abdul Fatir et al. (2024a). "Introducing Chronos-2: Large language model time series forecasting". In: *Amazon Science Blog*. URL: <https://www.amazon.science/blog/introducing-chronos-2>.
- Ansari, Abdul Fatir, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Syndar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, et al. (2024b). "Chronos: Learning the language of time series". In: *arXiv preprint arXiv:2403.07815*.
- Arnab, Anurag, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid (2021). "ViViT: A video vision transformer". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6836–6846.
- Badreddine, Samy, Artur d'Avila Garcez, Luciano Serafini, and Michael Spranger (2022). "Logic tensor networks". In: *Artificial Intelligence* 303, p. 103649. DOI: [10.1016/j.artint.2021.103649](https://doi.org/10.1016/j.artint.2021.103649).
- Bailey, David H., Jonathan M. Borwein, Marcos López de Prado, and Qiji Jim Zhu (2015). "The probability of backtest overfitting". In: *Journal of Computational Finance*. SSRN: 2326253.
- Baltas, Nick (2019). "The impact of crowding in alternative risk premia investing". In: *Financial Analysts Journal* 75.3, pp. 50–65. DOI: [10.1080/0015198X.2019.1600955](https://doi.org/10.1080/0015198X.2019.1600955).
- Bi, Yingjia and Vince D. Calhoun (2025). "The financial connectome: A brain-inspired framework for modeling latent market dynamics". In: *arXiv preprint arXiv:2508.02012*.
- Bouchaud, Jean-Philippe, Julius Bonart, Jonathan Donier, and Martin Gould (2018). *Trades, Quotes and Prices: Financial Markets Under the Microscope*. Cambridge University Press.
- Brown, Gregory, Philip Howard, and Christian Lundblad (2022). "Crowded trades and tail risk". In: *Review of Financial Studies* 35, pp. 3231–3271. DOI: [10.1093/rfs/hhab107](https://doi.org/10.1093/rfs/hhab107).

- Bucci, Frédéric, Michael Benzaquen, Fabrizio Lillo, and Jean-Philippe Bouchaud (2019). “Crossover from linear to square-root market impact”. In: *Physical Review Letters* 122.10. arXiv: 1811.05230, p. 108302.
- Bucci, Frédéric, Iacopo Mastromatteo, Zoltán Eisler, Fabrizio Lillo, Jean-Philippe Bouchaud, and Charles-Albert Lehalle (2020). “Co-impact: Crowding effects in institutional trading activity”. In: *Quantitative Finance* 20.2, pp. 193–205. DOI: [10.1080/14697688.2019.1660398](https://doi.org/10.1080/14697688.2019.1660398).
- Buzsáki, György (1989). “Two-stage model of memory trace formation: A role for “noisy” brain states”. In: *Neuroscience* 31.3, pp. 551–570. DOI: [10.1016/0306-4522\(89\)90423-5](https://doi.org/10.1016/0306-4522(89)90423-5).
- (2015). “Hippocampal sharp wave-ripple: A cognitive biomarker for episodic memory and planning”. In: *Hippocampus* 25.10, pp. 1073–1188. DOI: [10.1002/hipo.22488](https://doi.org/10.1002/hipo.22488).
- Caccioli, Fabio, Munik Shrestha, Christopher Moore, and J. Doyne Farmer (2014). “Stability analysis of financial contagion due to overlapping portfolios”. In: *Journal of Banking & Finance* 46, pp. 233–245.
- Caligiore, Daniele et al. (2017). “Consensus paper: Towards a systems-level view of cerebellar function: The interplay between cerebellum, basal ganglia, and cortex”. In: *The Cerebellum* 16, pp. 203–229.
- Caligiore, Daniele, Michael A. Arbib, R. Chris Miall, and Gianluca Baldassarre (2019). “The super-learning hypothesis: Integrating learning processes across cortex, cerebellum and basal ganglia”. In: *Neuroscience & Biobehavioral Reviews* 100, pp. 19–34.
- Chen, Jun-Hao and Yun-Cheng Tsai (2020). “Encoding candlesticks as images for pattern classification using convolutional neural networks”. In: *Financial Innovation* 6.1, p. 26. DOI: [10.1186/s40854-020-00187-0](https://doi.org/10.1186/s40854-020-00187-0).
- Coletta, Andrea et al. (2024). “Conditional generative models for simulation of EMG during naturalistic movements”. In: *arXiv preprint*. Generative diffusion models applied to sequential financial data simulation.
- Collins, Anne GE and Michael J Frank (2014). “Opponent actor learning (OpAL): Modeling interactive effects of striatal dopamine on reinforcement learning and choice incentive”. In: *Psychological Review* 121.3, p. 337. DOI: [10.1037/a0037015](https://doi.org/10.1037/a0037015).
- Daw, Nathaniel D, John P O’Doherty, Peter Dayan, Ben Seymour, and Raymond J Dolan (2006). “Cortical substrates for exploratory decisions in humans”. In: *Nature* 441.7095, pp. 876–879.
- DeMiguel, Victor, Alberto Martin-Utrera, and Raman Uppal (2021). “What alleviates crowding in factor investing?” In: *CEPR Discussion Paper* 16527. SSRN: 3928838.
- Dosovitskiy, Alexey, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold,

- Sylvain Gelly, et al. (2020). “An image is worth 16x16 words: Transformers for image recognition at scale”. In: *arXiv preprint arXiv:2010.11929*.
- Doya, Kenji (1999). “What are the computations of the cerebellum, the basal ganglia and the cerebral cortex?” In: *Neural Networks* 12.7–8, pp. 961–974. DOI: [10.1016/S0893-6080\(99\)00046-5](https://doi.org/10.1016/S0893-6080(99)00046-5).
- (2002). “Metalearning and neuromodulation”. In: *Neural Networks* 15.4–6, pp. 495–506. DOI: [10.1016/S0893-6080\(02\)00044-8](https://doi.org/10.1016/S0893-6080(02)00044-8).
- Easley, David, Marcos Lopez De Prado, and Maureen O’Hara (2012). “Flow toxicity and liquidity in a high-frequency world”. In: *The Review of Financial Studies* 25.5, pp. 1457–1493.
- Easley, David, Marcos M Lopez de Prado, and Maureen O’Hara (2011). “The microstructure of the “flash crash”: flow toxicity, liquidity crashes, and the probability of informed trading”. In: *The Journal of Portfolio Management* 37.2, pp. 118–128.
- Evans, Jonathan St B. T. and Keith E. Stanovich (2013). “Dual-process theories of higher cognition: Advancing the debate”. In: *Perspectives on Psychological Science* 8.3, pp. 223–241.
- Evans, Jonathan St BT (2008). “Dual-processing accounts of reasoning, judgment, and social cognition”. In: *Annual Review of Psychology* 59, pp. 255–278.
- Ezinwoke, B. and O. Rhodes (2025). “Predicting price movements in high-frequency financial data with spiking neural networks”. In: *arXiv preprint arXiv:2512.05868*.
- Fama, Eugene F and Kenneth R French (1993). “Common risk factors in the returns on stocks and bonds”. In: *Journal of Financial Economics* 33.1, pp. 3–56.
- Farmer, J. Doyne, Paolo Patelli, and Ilija I. Zovko (2005). “The predictive power of zero intelligence in financial markets”. In: *Proceedings of the National Academy of Sciences* 102.6, pp. 2254–2259.
- Foster, David J, Richard GM Morris, and Peter Dayan (2013). “Mechanisms of hierarchical reinforcement learning in corticostriatal circuits”. In: *Nature Neuroscience* 16, pp. 383–391.
- Frank, Michael J, Bryan Loughry, and Randall C O'Reilly (2006). “Making working memory work: a computational model of learning in the prefrontal cortex and basal ganglia”. In: *Neural Computation* 18.2, pp. 283–328.
- Fu, Sascha, Mikko S Pakkanen, and Rama Cont (2024). “JAX-LOB: A GPU-accelerated limit order book simulator to unlock large scale reinforcement learning”. In: *arXiv preprint arXiv:2408.08806*.
- Garcez, Artur d'Avila and Luís C Lamb (2023). “Neurosymbolic AI: The 3rd wave”. In: *Artificial Intelligence Review*. DOI: [10.1007/s10462-023-10448-w](https://doi.org/10.1007/s10462-023-10448-w).
- George, Dileep et al. (2025). “A detailed theory of thalamic and cortical microcircuits for predictive visual inference”. In: *Science Advances*. DOI: [10.1126/sciadv.adr6698](https://doi.org/10.1126/sciadv.adr6698).
- Glasmachers, Tobias (2017). “Limits of end-to-end learning”. In: *PMLR*. Vol. 77.

- Goldblum, Micah et al. (2021). "Adversarial attacks on machine learning systems for high-frequency trading". In: *ACM International Conference on AI in Finance (ICAIIF)*. arXiv: 2002.09565.
- Halassa, Michael M and Sabine Kastner (2017). "Thalamic functions in distributed cognitive control". In: *Nature Neuroscience* 20.12, pp. 1669–1679.
- Hassabis, Demis, Dharshan Kumaran, Christopher Summerfield, and Matthew Botvinick (2017). "Neuroscience-inspired artificial intelligence". In: *Neuron* 95.2, pp. 245–258. DOI: [10.1016/j.neuron.2017.06.011](https://doi.org/10.1016/j.neuron.2017.06.011).
- Hugging Face (2024). *Candle: Minimalist ML framework for Rust*. URL: <https://github.com/huggingface/candle>.
- Internal Revenue Service (2024). *Wash sale rule basics for active traders*. URL: <https://www.irs.gov/taxtopics/tc409>.
- Jaskir, Adam and Michael J. Frank (2023). "On the normative advantages of dopamine and striatal opponency for learning and choice". In: *eLife* 12, e85107.
- Joo, Hannah R. and Loren M. Frank (2023). "Selection of experience for memory by hippocampal sharp wave ripples". In: *Science* 380. Sharp wave ripple selection mechanisms in hippocampus, pp. 1171–1175.
- Jung, Ralf, Jacques-Henri Jourdan, Robbert Krebbers, and Derek Dreyer (2018). "RustBelt: Securing the foundations of the Rust programming language". In: *Proceedings of the ACM on Programming Languages* 2.POPL, p. 66. DOI: [10.1145/3158154](https://doi.org/10.1145/3158154).
- Kahneman, Daniel (2011). *Thinking, fast and slow*. Farrar, Straus and Giroux.
- Kahneman, Daniel and Amos Tversky (1979). "Prospect theory: An analysis of decision under risk". In: *Econometrica* 47.2, pp. 263–291. DOI: [10.2307/1914185](https://doi.org/10.2307/1914185).
- Khandani, Amir E. and Andrew W. Lo (2011). "What happened to the quants in August 2007?" In: *Journal of Financial Markets* 14.1, pp. 1–46. DOI: [10.1016/j.finmar.2010.07.005](https://doi.org/10.1016/j.finmar.2010.07.005).
- Kralingen, Marc van, Diego Garlaschelli, Karolina Scholtus, and Iman van Lelyveld (2021). "Crowded trades, market clustering, and price instability". In: *Entropy* 23.3, p. 336. DOI: [10.3390/e23030336](https://doi.org/10.3390/e23030336).
- Kumaran, Dharshan, Demis Hassabis, and James L. McClelland (2016). "What learning systems do intelligent agents need? Complementary learning systems theory updated". In: *Trends in Cognitive Sciences* 20.7, pp. 512–534. DOI: [10.1016/j.tics.2016.05.004](https://doi.org/10.1016/j.tics.2016.05.004).
- LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton (2015). "Deep learning". In: *Nature* 521.7553, pp. 436–444.
- LeDoux, Joseph E (2000). "Emotion circuits in the brain". In: *Annual Review of Neuroscience* 23, pp. 155–184. DOI: [10.1146/annurev.neuro.23.1.155](https://doi.org/10.1146/annurev.neuro.23.1.155).

- Lo, Andrew W. (2004). "The adaptive markets hypothesis". In: *Journal of Portfolio Management* 30.5, pp. 15–29. DOI: [10.3905/jpm.2004.442611](https://doi.org/10.3905/jpm.2004.442611).
- (2017). *Adaptive Markets: Financial Evolution at the Speed of Thought*. Princeton University Press.
- López de Prado, Marcos (2018a). *Advances in Financial Machine Learning*. Wiley.
- (2018b). "The 10 reasons most machine learning funds fail". In: *Journal of Portfolio Management* 44.6, pp. 120–133.
- Lou, Dong and Christopher Polk (2022). "Comomentum: Inferring arbitrage activity from return correlations". In: *Review of Financial Studies* 35.7, pp. 3272–3302. DOI: [10.1093/rfs/hhab117](https://doi.org/10.1093/rfs/hhab117).
- Marino, Kevin et al. (2023). "ME-ViT: A single-load memory-efficient FPGA accelerator for vision transformers". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*.
- Markwick, David (2023). *Solving the Almgren-Chriss model*. URL: <https://www.actuaries.digital/2023/01/19/solving-the-almgren-chriss-model/>.
- Marra, Giuseppe, Sebastijan Dumančić, Robin Manhaeve, and Luc De Raedt (2024). "From statistical relational to neurosymbolic artificial intelligence: A survey". In: *Artificial Intelligence* 328, p. 104062. DOI: [10.1016/j.artint.2023.104062](https://doi.org/10.1016/j.artint.2023.104062).
- Masset, Paul et al. (2025). "Multi-timescale reinforcement learning in the brain". In: *Nature* 642, pp. 682–690. DOI: [10.1038/s41586-025-08929-9](https://doi.org/10.1038/s41586-025-08929-9).
- Matsakis, Nicholas D. and II Klock Felix S. (2014). "The Rust language". In: *ACM SIGAda Ada Letters* 34.3, pp. 103–104. DOI: [10.1145/2692956.2663188](https://doi.org/10.1145/2692956.2663188).
- Mattar, Marcelo G. and Nathaniel D. Daw (2018). "Prioritized memory access explains planning and hippocampal replay". In: *Nature Neuroscience* 21, pp. 1609–1617.
- McClelland, James L, Bruce L McNaughton, and Randall C O'Reilly (1995). "Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory". In: *Psychological Review* 102.3, p. 419. DOI: [10.1037/0033-295X.102.3.419](https://doi.org/10.1037/0033-295X.102.3.419).
- McInnes, Leland, John Healy, and James Melville (2018a). *UMAP: Uniform manifold approximation and projection (Python package)*. Includes AlignedUMAP for temporal manifold alignment. URL: <https://umap-learn.readthedocs.io/>.
- (2018b). "UMAP: Uniform manifold approximation and projection for dimension reduction". In: *arXiv preprint arXiv:1802.03426*.
- Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, et al. (2015). "Human-level control through deep reinforcement learning". In: *Nature* 518.7540, pp. 529–533. DOI: [10.1038/nature14236](https://doi.org/10.1038/nature14236).
- Mohan, A. et al. (2025). "Spiking neural network for cross-market portfolio optimization". In: *arXiv preprint arXiv:2510.15921*.

- Monfils, Marie-H, Kiriana K Cowansage, Eric Klann, and Joseph E LeDoux (2009). “Extinction-reconsolidation boundaries: key to persistent attenuation of fear memories”. In: *Science* 324.5929, pp. 951–955.
- Mugel, Samuel, Enrique Lizaso, and Roman Orus (2022). “Dynamic portfolio optimization with real datasets using quantum processors and quantum-inspired tensor networks”. In: *Physical Review Research* 4, p. 013006. DOI: [10.1103/PhysRevResearch.4.013006](https://doi.org/10.1103/PhysRevResearch.4.013006).
- Padakandla, Sindhu et al. (2020). “Reinforcement learning algorithm for non-stationary environments”. In: *Applied Intelligence* 50. arXiv: 1905.03970, pp. 3590–3606.
- Polars Contributors (2024). *Polars: Lightning-fast DataFrame library*. URL: <https://www.pola.rs/>.
- Qdrant (2024). *Qdrant: Vector similarity search engine*. URL: <https://qdrant.tech/>.
- Schaul, Tom, John Quan, Ioannis Antonoglou, and David Silver (2015). “Prioritized experience replay”. In: *arXiv preprint arXiv:1511.05952*.
- Schultz, Wolfram, Peter Dayan, and P. Read Montague (1997). “A neural substrate of prediction and reward”. In: *Science* 275.5306, pp. 1593–1599. DOI: [10.1126/science.275.5306.1593](https://doi.org/10.1126/science.275.5306.1593).
- Serafini, Luciano and Artur S. d’Avila Garcez (2016). “Logic tensor networks: Deep learning and logical reasoning from data and knowledge”. In: *AI\*IA 2016*, pp. 334–348.
- Sokolov, Alexander A., R. Chris Miall, and Richard B. Ivry (2017). “The cerebellum: Adaptive prediction for movement and cognition”. In: *Trends in Cognitive Sciences* 21.5, pp. 313–332.
- Stein, Jeremy C. (2009). “Presidential address: Sophisticated investors and market efficiency”. In: *Journal of Finance* 64.4, pp. 1517–1548. DOI: [10.1111/j.1540-6261.2009.01472.x](https://doi.org/10.1111/j.1540-6261.2009.01472.x).
- Sterling, Peter (2012). “Allostasis: a model of predictive regulation”. In: *Physiology & Behavior* 106.1, pp. 5–15.
- Stillman, Namid R. and Rory Baggott (2024). “Neuro-symbolic traders: Assessing the wisdom of AI crowds in markets”. In: *arXiv preprint arXiv:2410.14587*. DOI: [10.48550/arXiv.2410.14587](https://doi.org/10.48550/arXiv.2410.14587).
- Sutton, Richard S. and Andrew G. Barto (2018). *Reinforcement Learning: An Introduction*. 2nd. MIT Press.
- Tokio Contributors (2024). *Tokio: Asynchronous runtime for Rust*. URL: <https://tokio.rs/>.
- Tóth, Bence, Yves Lepiérière, Cyril Deremble, Joachim de Lataillade, Julien Kockelkoren, and Jean-Philippe Bouchaud (2011). “Anomalous price impact and the critical nature of liquidity in financial markets”. In: *Physical Review X* 1.2, p. 021006. DOI: [10.1103/PhysRevX.1.021006](https://doi.org/10.1103/PhysRevX.1.021006).

- Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin (2017). “Attention is all you need”. In: *Advances in Neural Information Processing Systems* 30.
- Vemeko (2023). *How to use FPGAs for HFT acceleration*. URL: <https://www.vemeko.com/fpga-hft>.
- Ven, Gido M. van de, Hava T. Siegelmann, and Andreas S. Tolias (2020). “Brain-inspired replay for continual learning with artificial neural networks”. In: *Nature Communications* 11, p. 4069.
- Volpati, Valerio, Michael Benzaquen, Zoltán Eisler, Iacopo Mastromatteo, Bence Tóth, and Jean-Philippe Bouchaud (2020). “Zooming in on equity factor crowding”. In: *arXiv preprint arXiv:2001.04185*.
- Wagner, Wolf (2011). “Systemic liquidation risk and the diversity–diversification trade-off”. In: *Journal of Finance* 66.4, pp. 1141–1175. DOI: [10.1111/j.1540-6261.2011.01666.x](https://doi.org/10.1111/j.1540-6261.2011.01666.x).
- Wan, Zishen et al. (2024). “Towards efficient neuro-symbolic AI: From workload characterization to hardware architecture”. In: *IEEE Transactions on Circuits and Systems for AI*. arXiv: 2409.13153.
- Wang, Zhiguang and Tim Oates (2015). “Imaging time-series to improve classification and imputation”. In: *arXiv preprint arXiv:1506.00327*. URL: <https://arxiv.org/abs/1506.00327>.
- Wilson, Hugh R and Jack D Cowan (1972). “Excitatory and inhibitory interactions in localized populations of model neurons”. In: *Biophysical Journal* 12.1, pp. 1–24.
- Wilson, Matthew A. and Bruce L. McNaughton (1994). “Reactivation of hippocampal ensemble memories during sleep”. In: *Science* 265.5172, pp. 676–679. DOI: [10.1126/science.8036517](https://doi.org/10.1126/science.8036517).
- Wolpert, Daniel M., R. Chris Miall, and Mitsuo Kawato (1998). “Internal models in the cerebellum”. In: *Trends in Cognitive Sciences* 2.9, pp. 338–347.
- Yamakawa, Hiroshi (2021). “The whole brain architecture approach: Accelerating the development of artificial general intelligence by referring to the brain”. In: *Neural Networks* 144, pp. 478–495.
- Zhang, Yiming et al. (2024). “Bidirectionally regulating gamma oscillations in Wilson-Cowan model by self-feedback loops: A computational study”. In: *Neural Computation*. Computational study of Wilson-Cowan oscillatory dynamics with bidirectional gamma regulation.