Project JANUS

The Neuro-Symbolic Visual Quant Architecture

A White Paper on the Next Generation of Autonomous Financial Intelligence

Classification: Technical White Paper

Version: 1.0

Author: Jordan Smith

github.com/nuniesmith

Date: December 10, 2025

Abstract

A tension between interpretability and performance has historically defined the trajectory of quantitative finance. The early eras, characterized by heuristic-based expert systems (Quant 1.0) and factor-based statistical arbitrage (Quant 2.0), prioritized transparency and economic theory but were often constrained by linear assumptions. The subsequent rise of Deep Learning (Quant 3.0) unleashed unprecedented predictive power through high-dimensional function approximation, yet it introduced systemic risks associated with "black box" opacity, brittleness in the face of regime shifts, and an inability to natively enforce regulatory or logical constraints.

This white paper introduces **Project JANUS**, a comprehensive architectural framework for the next generation of algorithmic trading—*Quant 4.0*. JANUS represents a convergence of three frontier paradigms:

1. **Visual Time Series Encoding**, which transforms market dynamics into topological textures processed by Computer Vision;
2. **Neuro-Symbolic Reasoning**, which embeds formal logic and regulatory constraints directly into the learning objective via Logic Tensor Networks (LTN); and
3. **Hierarchical Reinforcement Learning**, specifically Feudal Networks, which decouple strategic planning from tactical execution.

By synthesizing the "Visual Intuition" of Convolutional Neural Networks with the "Logical Rigour" of symbolic AI and the "Strategic Depth" of hierarchical agents, JANUS aims to create an autonomous trading system that is not only highly performant but also mathematically compliant, explainable, and robust to the chaotic microstructure of modern financial markets.

Implementation Status

This white paper describes both the theoretical architecture and the current production implementation of Project JANUS. The following components are **fully implemented and operational**:

- **The Cortex (HRM):** Hierarchical Reasoning Module with CEO/Worker architecture using Google Gemini 2.0 Flash with Ollama fallback
- **The Oracle (Chronos-2):** Zero-shot probabilistic forecasting with real Chronos-2 foundation model integration
- **The Eyes (CNN):** Visual pattern recognition via feature tensor encoding and CNN service integration
- **The Conscience (HMM):** Regime detection via Hidden Markov Model service integration
- **Product-not-Sum Consensus:** Multiplicative consensus mechanism with veto power
- **LangGraph Workflow:** State machine orchestration for the HRM architecture
- **RobustBrain:** Automatic failover from Gemini to Ollama for zero-downtime operation

The following components are **planned for future implementation**:

- **Logic Tensor Networks (LTN):** Neuro-symbolic constraint enforcement (currently using rule-based risk veto)
- **Gramian Angular Fields (GAF):** Advanced visual encoding (currently using feature tensor approach)
- **Almgren-Chriss Execution:** Optimal execution framework (currently using simplified risk management)
- **Feudal Reinforcement Learning:** Full hierarchical RL training pipeline (currently using HRM with LLM reasoning)
- **VPIN Integration:** Order flow toxicity detection
- **Multimodal Fusion (MSGCA):** Gated cross-attention for text/time series/vision fusion

Contents

1 The Evolution of Quantitative Intelligence: From Heuristics to Hybrid Cognition

1.1 The Limitations of the Deep Learning Paradigm (Quant 3.0)

The financial industry is currently navigating the mature phase of Quant 3.0, an era dominated by the application of deep neural networks—Long Short-Term Memory (LSTM) networks, Transformers, and Differentiable Manifolds—to raw financial time series. These models have demonstrated a remarkable ability to capture non-linear dependencies and latent patterns that traditional econometric models (such as ARIMA or GARCH) fail to identify. Foundation models like Amazon's Chronos-2 have further advanced the field by enabling zero-shot forecasting capabilities, leveraging vast corpora of synthetic and real-world data to generalize across unseen assets.

However, the reliance on end-to-end deep learning has precipitated a "Black Box Crisis." In high-stakes financial environments, the opacity of these models presents a critical vulnerability. A deep reinforcement learning (DRL) agent trained solely to maximize the Sharpe ratio may inadvertently discover strategies that exploit simulator artifacts or violate regulatory norms—such as engaging in wash trading or spoofing—because it lacks an explicit understanding of market rules. Furthermore, purely data-driven models are inherently inductive; they struggle to generalize to "black swan" events or regime shifts that lie outside their training distribution. When market dynamics shift from a mean-reverting regime to a momentum-driven crash, standard deep learning models often fail catastrophically because they lack the symbolic reasoning capabilities to diagnose the structural change.

1.2 Defining Quant 4.0: The Neuro-Symbolic Imperative

Quant 4.0 represents a philosophical and architectural pivot toward Knowledge-Driven AI. Unlike its predecessor, which relies on brute-force data ingestion to approximate a mapping function:

$$f(x) \to y \tag{1}$$

Quant 4.0 leverages prior knowledge—encoded as symbolic logic, physical laws, or regulatory text—to constrain the hypothesis space and guide the learning process. This generation of systems is characterized by three pillars:

- **Automated AI:** End-to-end automation of the research pipeline, from feature engineering to strategy deployment.
- **Explainable AI (XAI):** The ability to articulate the causal reasoning behind a

trading decision, moving from “The model predicts X” to “The model predicts X because Condition Y and Rule Z are met.”

- **Neuro-Symbolic Integration:** The fusion of connectionist learning (neural networks) with symbolic reasoning (logic).

Project JANUS embodies this Quant 4.0 philosophy. It posits that a robust trading agent must possess multi-sensory capabilities: it must “see” the market’s physical structure through visual encoding, “read” the market’s narrative through semantic processing, and “reason” about the market’s constraints through formal logic. This hybrid approach allows the system to maintain the intuitive flexibility of neural networks while adhering to “hard” constraints, such as the risk limits defined by the Almgren-Chriss framework or the tax implications of wash sale rules.

2 The Visual Quant Paradigm: Seeing Market Topology

2.1 Beyond Numerical Sequences: The Case for Vision

Financial data has traditionally been treated as a one-dimensional sequence of scalars (Open, High, Low, Close, Volume). While efficient for storage, this representation often discards the rich, multi-scale topological structures inherent in market data—the “shape” of a crash, the “texture” of a consolidation, or the “geometry” of a limit order book. Standard time series models like LSTMs process data sequentially, which can make it difficult to capture complex, long-range temporal correlations or to recognize patterns that are invariant to time-warping but visually distinct.

Project JANUS integrates a Visual Encoder that transforms 1D time series into 2D images. This is not merely a data augmentation technique; it is a fundamental shift in representation learning. By converting time series into images, we unlock the vast architectural power of Computer Vision (CV), specifically Convolutional Neural Networks (CNNs). CNNs are evolutionarily optimized to detect hierarchical spatial patterns—edges, shapes, and textures—which, in the context of a transformed time series, correspond to local trends, volatility clusters, and regime shifts.

2.2 Gramian Angular Fields (GAF): The Mathematical Lens

The primary imaging technique employed in JANUS is the Gramian Angular Field (GAF). Unlike simple line plots, GAF preserves the temporal dependency of the time series while encoding the data in a polar coordinate system. This transformation ensures that the temporal correlation structure is explicitly mapped to the spatial dimensions of the image.

2.2.1 Mathematical Formulation of GAF

Given a univariate time series:

$$X = \{x_1, x_2, \ldots, x_n\} \tag{2}$$

The transformation proceeds in two steps: normalization and polar projection.

Step 1: Normalization

To fit the spectral domain of the cosine function, the series is normalized to the interval $[-1, 1]$. JANUS employs a min-max scaling approach:

$$\tilde{x}_i = \frac{(x_i - \max(X)) + (x_i - \min(X))}{\max(X) - \min(X)} \tag{3}$$

This scaling preserves the relative magnitude of price movements while standardizing the input range.

Step 2: Polar Coordinate Transformation

The normalized value is encoded as the angular cosine, and the time stamp t_i is encoded as the radius r_i:

$$\begin{cases} \phi_i = \arccos(\tilde{x}_i), & -1 \le \tilde{x}_i \le 1 \\ r_i = \frac{t_i}{N}, & t_i \in \mathbb{N} \end{cases} \tag{4}$$

This mapping is bijective, ensuring that no information is lost in the transformation. The angular perspective dominates the representation, while the radial component preserves the temporal progression.

Step 3: Gramian Field Generation

The temporal correlations between different time steps are computed using trigonometric sums and differences.

Gramian Angular Summation Field (GASF):

$$\text{GASF}_{i,j} = \cos(\phi_i + \phi_j) = \tilde{x}_i\tilde{x}_j - \sqrt{1-\tilde{x}_i^2}\sqrt{1-\tilde{x}_j^2} \tag{5}$$

Gramian Angular Difference Field (GADF):

$$\text{GADF}_{i,j} = \sin(\phi_i - \phi_j) = \sqrt{1-\tilde{x}_i^2}\tilde{x}_j - \tilde{x}_i\sqrt{1-\tilde{x}_j^2} \tag{6}$$

The result is an $n \times n$ matrix (image) where the position (i, j) represents the superposition of the market state at time i and time j. The main diagonal contains the original value information (reconstructed spatial features), while the off-diagonal elements capture the rich temporal correlation structure across the entire window.

2.3 Feature Tensor Encoding and CNN Integration

Current Implementation: In the production version of JANUS, market data is encoded as multi-dimensional feature tensors rather than GAF images. This approach provides a more direct representation while maintaining the visual pattern recognition capabilities of CNNs.

The feature tensor construction process transforms raw OHLCV data into a normalized matrix of shape $(sequence\_length, num\_features)$ where features include:

- **Price Features:** Normalized open, high, low, close prices
- **Returns:** Log returns and momentum indicators
- **Volatility:** Rolling standard deviation and ATR (Average True Range)
- **Volume:** Normalized volume and volume moving average ratios

- **Technical Indicators:** RSI, MACD, Bollinger Bands (when available)

JANUS employs a specialized CNN service to process these feature tensors. The CNN architecture recognizes visual patterns in the multi-dimensional feature space, distinguishing between different market regimes and volatility structures. Unlike traditional technical indicators that simply measure the magnitude of volatility, the CNN can classify the quality and structure of volatility patterns, identifying consolidation phases versus liquidity crashes.

Future Enhancement: The theoretical architecture includes Gramian Angular Fields (GAF) as an advanced visual encoding technique. GAF transforms time series into polar coordinate images. Temporal correlations are explicitly mapped to spatial dimensions. A market in a strong, low-volatility trend appears in a GASF image as a coherent, smooth gradient. Volatile markets manifest as chaotic "checkerboard" textures. Future implementations may integrate GAF encoding to provide additional visual texture analysis.

3 The Logic Layer: Neuro-Symbolic Reasoning with Logic Tensor Networks

3.1 The Necessity of Symbolic Constraints in Finance

While deep learning provides powerful intuition and pattern recognition, it fundamentally lacks logical precision. In the domain of quantitative finance, "close enough" is often unacceptable. A trading bot that generates a profit but inadvertently executes a "wash sale"—selling a security at a loss and repurchasing a substantially identical one within 30 days—triggers tax penalties. These penalties can negate the strategy's alpha and create legal liabilities. Similarly, execution algorithms must adhere to strict risk mandates, such as those derived from the Almgren-Chriss model, where the risk aversion parameter λ dictates the maximum acceptable variance of execution cost.

Pure neural networks struggle to learn these "hard" constraints solely from examples (data efficiency) and cannot guarantee adherence to them in unseen scenarios (safety). A standard Reinforcement Learning (RL) agent might learn to avoid wash sales in the training set because they result in a lower reward, but it does not understand the rule itself. Consequently, when presented with a novel market condition, the agent might violate the rule if it predicts a high enough gross return. Neuro-Symbolic AI addresses this critical gap by embedding formal logic directly into the learning process.

Current Implementation: The production version of JANUS implements rule-based risk management through the Praxeological Motor. This motor enforces risk vetoes based on regime detection and volatility thresholds. The system uses a Product-not-Sum consensus mechanism. Any motor can veto a trade by returning a score of zero. This ensures safety through multiplicative aggregation rather than averaging.

3.2 Logic Tensor Networks (LTN): The Reasoning Engine

Planned Implementation: Project JANUS is architected to utilize Logic Tensor Networks (LTN) as its core reasoning engine. LTN is a neuro-symbolic framework that integrates neural networks with First-Order Logic (FOL). It enables the specification of knowledge using logical formulas where the predicates and functions are approximated by neural networks (a process known as "grounding"), and the satisfaction of these formulas is maximized during training.

Current Status: The production implementation currently uses rule-based risk management and Product-not-Sum consensus for constraint enforcement. LTN integration is planned for a future release to provide differentiable logical constraints and enhanced explainability.

3.2.1 Real Logic and Differentiable Constraints

To make logical constraints compatible with gradient-based optimization, LTN employs Real Logic. In this paradigm, truth values are relaxed from the binary set $\{0, 1\}$ to the continuous interval $[0, 1]$. This allows logical constraints to be differentiable and incorporated into the loss function.

Let $\mathcal{G}$ be a grounding function that maps logical symbols to tensors:

- **Constants:** A constant symbol c (e.g., a specific trade or asset) is mapped to a tensor in $\mathbb{R}^n$.
- **Predicates:** A predicate symbol P (e.g., *IsVolatile*, *WashSale*) is mapped to a neural network function $f_P : \mathbb{R}^n \to [0, 1]$, which outputs the degree of truth.
- **Connectives:** Logical connectives ($\wedge, \vee, \neg, \rightarrow$) are modeled using fuzzy logic t-norms (e.g., Lukasiewicz, Gödel, or Product t-norms).

For example, the conjunction of two formulas A and B using the Product t-norm is defined as:

$$\mathcal{G}(A \wedge B) = \mathcal{G}(A) \cdot \mathcal{G}(B) \tag{7}$$

The implication $A \rightarrow B$ using the Reichenbach implication is defined as:

$$\mathcal{G}(A \rightarrow B) = 1 - \mathcal{G}(A) + \mathcal{G}(A) \cdot \mathcal{G}(B) \tag{8}$$

This allows the network to learn complex dependencies: “If A is true, then B must be true.”

3.2.2 The Satisfiability Objective

The learning objective in JANUS is to find the set of parameters θ (the weights of the neural networks) that maximizes the satisfiability of a Knowledge Base $\mathcal{K}$ containing a set of logical axioms. The logical loss function is defined as the aggregate “falsity” of the formulas:

$$\mathcal{L}_{logic}(\theta) = 1 - \text{SatAgg}_{\phi \in \mathcal{K}}(\mathcal{G}_\theta(\phi)) \tag{9}$$

where SatAgg is an aggregation operator (e.g., generalized mean) over the truth values of all constraints. This term is added to the standard predictive loss, ensuring that the model optimizes for accuracy while strictly adhering to the logical rules.

3.3 Encoding Financial Rules: The Knowledge Base

JANUS encodes critical trading rules and market dynamics as First-Order Logic constraints within the LTN layer. This “Knowledge Base” acts as the immutable conscience of the trading agent.

3.3.1 The Wash Sale Constraint

To prevent wash sales, we define a predicate $\textit{WashSale}(trade_t, trade_{t+k})$ which evaluates to true if a trade at time t is a sale at a loss and a trade at time $t + k$ (where $k \leq 30$ days) is a purchase of a substantially identical asset. The logical axiom enforced is:

$$\forall t, \forall k \in (0, 30 \text{ days}] : \neg(\textit{SaleAtLoss}(t) \land \textit{Buy}(t + k)) \tag{10}$$

In the LTN, this logic is added to the loss function. The agent is penalized not just for losing money (financial loss), but for violating the logical structure of the tax code. This ensures that the learned policy respects the 61-day window surrounding a loss sale, effectively "masking" the buy action in the prohibited window.

3.3.2 Risk Logic and Almgren-Chriss Adherence

We can also encode risk preferences and execution guidelines symbolically. We introduce the predicate $\textit{LiquidityRisk}(v)$ to represent the toxicity or market impact of an execution volume v. We can then assert a rule that mandates size reduction in volatile markets:

$$\forall v : \textit{HighVolatility}(Market) \to (\textit{LiquidityRisk}(v) \to \textit{ReduceSize}(v)) \tag{11}$$

This symbolic rule forces the neural network to learn the causal relationship between market volatility (detected by the Visual Encoder) and order size reduction. Unlike a pure RL agent that might discover this relationship through trial and error (and potentially catastrophic losses), the LTN agent is "told" this relationship a priori, accelerating convergence and acting as a "safety rail".

3.3.3 VPIN and Toxicity Detection

The Logic Layer also monitors the Volume-Synchronized Probability of Informed Trading (VPIN), a metric of order flow toxicity. If VPIN spikes, indicating the presence of informed traders or a potential crash, the logic layer can trigger a defensive posture.

$$\forall t : \textit{VPIN}_t > Threshold \to \textit{HaltTrading}(t) \tag{12}$$

This acts as a "circuit breaker." If the toxicity constraint is violated, the LTN forces the execution probability to zero, effectively overriding the RL agent's output. This prevents the algorithm from providing liquidity to toxic flow, a common failure mode in standard HFT algorithms during events like the Flash Crash.

4 The Time Series Foundation: Chronos-2 and Multimodal Fusion

4.1 Chronos-2: The Backbone of Temporal Perception

For the core numerical time series processing, JANUS employs a backbone inspired by Amazon Chronos-2. Unlike traditional autoregressive models that process raw float values, Chronos-2 adopts a language-modeling perspective. It quantizes time series values into a discrete vocabulary of tokens, allowing it to leverage the powerful Transformer architecture (specifically the T5 encoder-decoder) originally designed for Natural Language Processing (NLP).

Why Chronos-2?

The strategic advantage of Chronos-2 lies in its Zero-Shot capabilities. Trained on a massive corpus of real-world and synthetic time series data, it has learned universal temporal patterns—seasonality, trend breaks, mean reversion—that generalize across domains. This allows JANUS to deploy effectively on new asset classes (e.g., shifting from Equities to Crypto) without the need for extensive, data-hungry retraining. Furthermore, Chronos-2 supports Group Attention, a mechanism that allows the model to process multivariate time series and static covariates simultaneously, modeling the inter-dependencies between correlated assets.

4.2 Multimodal Fusion: Integrating Vision, Text, and Time

The true power of JANUS lies in its ability to synthesize heterogeneous data streams. The input to the system is not just a sequence of prices, but a multimodal stream:

- **Time Series Tokens:** Quantized price and volume history (from Chronos-2).
- **Visual Tokens:** Feature vectors representing market texture (from the GAF-CNN).
- **Semantic Tokens:** Contextual embeddings of news and sentiment (from FinBERT).

To fuse these disparate modalities, JANUS employs a Multimodal Stable Fusion with Gated Cross-Attention (MSGCA) mechanism.

4.2.1 Gated Cross-Attention (GCA)

Standard attention mechanisms might allow one noisy modality (e.g., a flurry of irrelevant tweets) to drown out the signal from others. The GCA mechanism introduces a learnable gating scalar that dynamically weighs the importance of each modality based on the current context.

The cross-attention score is computed as:

$$\alpha_{m \to n} = \text{softmax}\left(\frac{Q_m K_n^\top}{\sqrt{d_h}}\right) \tag{13}$$

where Q_m is the query from the primary modality (e.g., Time Series) and K_n is the key from the auxiliary modality (e.g., Text or Vision). The gating mechanism λ_{gate} modulates the contribution of the auxiliary context:

$$H_{fused} = H_{TS} + \lambda_{gate} \cdot \text{Attention}(Q_{TS}, K_{Aux}, V_{Aux}) \tag{14}$$

Scenario-Based Adaptation:

- **Earnings Call:** When semantic density is high (e.g., during an earnings release), the model learns to increase λ_{text}, allowing the FinBERT embeddings to heavily influence the forecast.
- **Market Crash:** During a liquidity crisis where fundamentals are ignored and pure price dynamics dominate, the model increases λ_{vis} (Visual Texture) and decreases λ_{text}, focusing on the geometric structure of the collapse.
- **Quiet Drift:** During periods of low activity, the gating mechanism suppresses auxiliary inputs, relying primarily on the autoregressive price dynamics captured by Chronos-2.

4.3 Semantic Processing with FinBERT

Textual data is processed using FinBERT, a BERT variant pre-trained specifically on financial corpora (corporate filings, financial news, analyst reports). Unlike general-purpose LLMs, FinBERT is tuned to understand financial nuance—for example, recognizing that the word “liability” has a specific negative accounting connotation, or that “volatility” can be positive for a market maker but negative for a long-only fund.

JANUS utilizes the output of FinBERT not just as a sentiment score (Positive/Negative), but as a high-dimensional embedding vector. This vector captures the semantic state of the market narrative. By aligning this text embedding with the time series embedding, JANUS ensures that the “language” of price and the “language” of news are mapped to a shared latent space.

5 The Hand: Hierarchical Execution and Risk Management

5.1 The Need for Hierarchy

A monolithic agent attempting to solve both the high-level strategy (e.g., “Accumulate AAPL over the next week”) and the low-level execution (e.g., “Post a bid at 150.05”) suffers from the Curse of Dimensionality and disparate temporal scales. Strategic decisions happen over hours or days; execution decisions happen over milliseconds. JANUS addresses this by implementing a Hierarchical Reasoning Module (HRM) architecture that separates strategic planning from tactical execution. This architecture mimics the organizational structure of a human trading desk.

Current Implementation: The production version uses an HRM architecture with CEO and Worker nodes implemented using Large Language Models (Gemini 2.0 Flash with Ollama fallback) rather than trained reinforcement learning agents. The CEO node operates at a strategic timescale (updating every 4 hours) to set market regime and risk appetite, while the Worker node operates at a tactical timescale to generate trading signals within the CEO’s strategic context.

Future Enhancement: The theoretical architecture includes Feudal Reinforcement Learning, where Manager and Worker agents are trained end-to-end using hierarchical RL algorithms. This would enable the system to learn optimal strategies through interaction with market simulators rather than relying solely on LLM reasoning.

5.2 The Feudal Architecture: Manager and Worker

The Manager (High-Level Policy)

- **Temporal Resolution:** Low frequency (e.g., minutes to hours).
- **Input:** The fused multimodal state from the Brain (Global Market State).
- **Output:** An abstract Goal Vector g_t in latent space and a Risk Aversion Parameter λ_t.
- **Objective:** Maximize long-term portfolio PnL and Sharpe Ratio. The Manager does not interact with the order book directly; it “commands” the Worker.

The Worker (Low-Level Policy)

- **Temporal Resolution:** High frequency (e.g., ticks to seconds).
- **Input:** The goal g_t from the Manager and the immediate microstructure state (Order Book, VPIN).

- **Output:** Primitive actions (Limit Order, Market Order, Cancel).
- **Objective:** Fulfill the Manager's goal g_t while minimizing Implementation Shortfall and strictly adhering to the Almgren-Chriss trajectory.

This decoupling allows the Manager to focus on strategy and logic (avoiding wash sales, managing portfolio beta) while the Worker focuses on microstructure tactics (navigating the spread, hiding liquidity).

5.3 Risk Management and Position Sizing

Current Implementation: The production version of JANUS implements risk management through the Praxeological Motor, which enforces risk vetoes based on market regime, volatility thresholds, and signal confidence. Position sizing is dynamically adjusted based on the consensus score and risk appetite set by the CEO node.

The system uses a simplified risk framework where:

- **High Confidence / Low Volatility:** The CEO sets a higher risk appetite (e.g., $\lambda \approx 0.7$), allowing larger position sizes and more aggressive signals.
- **High Volatility / Bear Regime:** The CEO sets a lower risk appetite (e.g., $\lambda \approx 0.2$), reducing position sizes and requiring higher consensus scores for trade execution.
- **Risk Veto:** The Praxeological Motor can veto any trade by returning a zero score, which multiplies through the Product-not-Sum consensus to prevent execution.

Future Enhancement: Optimal Execution with Almgren-Chriss Framework The theoretical architecture includes the Almgren-Chriss (AC) optimal execution model for the Worker agent's reward function. The AC framework provides a mathematical solution for liquidating or acquiring a position X over time T while balancing two competing costs:

- **Expected Cost ($E[C]$):** Driven by temporary market impact (slippage). Trading too fast moves the price against you.
- **Timing Risk ($Var[C]$):** Driven by market volatility. Trading too slow exposes you to price drift.

The objective function is:

$$\min_v (E[C] + \lambda Var[C]) \tag{15}$$

where v is the trading trajectory (speed) and λ is the risk aversion parameter.

In the full AC implementation, λ would be dynamically modulated by the Manager based on regime detection, enabling adaptive execution strategies from patient TWAP-like execution in stable markets to aggressive front-loaded execution during volatile periods.

6 System Architecture and Implementation

6.1 The Technology Stack

Implementing JANUS requires a robust, low-latency technology stack capable of handling heterogeneous data streams and real-time inference.

Production Technology Stack

Component	**Technology**	**Role in JANUS**
Web Framework	FastAPI + Uvicorn	RESTful API service on port 8206 for analysis endpoints.
LLM Primary	Google Gemini 2.0 Flash	CEO and Worker node reasoning (strategic and tactical decisions).
LLM Fallback	Ollama (Llama 3)	Automatic failover when Gemini unavailable (zero downtime).
State Management	LangGraph	Workflow orchestration for HRM architecture (CEO $\rightarrow$ Worker $\rightarrow$ Tools $\rightarrow$ Consensus).
TS Forecasting	Chronos-2 (chronos-forecasting)	Zero-shot probabilistic forecasting with quantile generation.
Visual Pattern Recognition	CNN Service (port 8204)	Feature tensor processing for market pattern classification.
Regime Detection	HMM Service (port 8205)	Hidden Markov Model for market regime identification.
Data Fetching	yfinance	Market data retrieval for OHLCV time

Planned Technology Stack

Component	**Technology**	**Status**
Visual Encoding	GAF (Gramian Angular Fields)	Future enhancement for advanced texture analysis.
Reasoning	TensorFlow LTN	Planned for neuro-symbolic constraint enforcement.
RL Framework	Ray RLLib	Planned for Feudal RL training pipeline.
Execution	Almgren-Chriss Model	Planned for optimal execution strategies.
Text Processing	FinBERT	Planned for semantic sentiment analysis.
Multimodal Fusion	MSGCA (Gated Cross-Attention)	Planned for text/time series/vision fusion.

6.2 Inference and Latency Challenges

JANUS operates in a domain where speed is critical. While the CEO node operates on a slower timescale (updating every 4 hours), the Worker node must generate signals efficiently.

Current Implementation:

- **Chronos-2 Model:** The production system uses ‘chronos-t5-tiny‘ (8M parameters) for fast inference. On CPU, forecast latency is approximately 200-300ms after initial model loading. GPU acceleration would reduce this to 10-20ms.
- **LLM Inference:** CEO and Worker nodes use Google Gemini 2.0 Flash for fast reasoning (typically 15-20 seconds per node). Automatic fallback to Ollama ensures zero downtime but may increase latency to 20-30 seconds per node.
- **Service Architecture:** CNN and HMM services run as separate microservices, enabling parallel processing and independent scaling. The main JANUS service orchestrates these components via HTTP calls.

Future Enhancements:

- **Chronos-Bolt:** A distilled, patch-based variant of Chronos that is up to 250x faster, enabling sub-millisecond inference for high-frequency execution scenarios.
- **Hardware Acceleration:** GPU inference servers (e.g., NVIDIA A10G) for CNN processing and Chronos-2 forecasting to reduce latency.
- **FPGA Acceleration:** For GAF generation and tensor operations in future implementations requiring real-time visual encoding.

6.3 Service Architecture and Deployment

The production implementation of JANUS follows a microservices architecture:

Core Service:

- **fks_ai_janus (Port 8206):** Main orchestration service containing HRM workflow, consensus engine, and API endpoints.
- **Service Dependencies:** Integrates with CNN service (port 8204), HMM service (port 8205), and Ollama service (port 11434).
- **Data Sources:** Uses yfinance for market data (can be replaced with fks_data service in production).

API Endpoints:

- `POST /api/v1/quant/analyze` - Full neuro-symbolic analysis with all motors
- `POST /api/v1/quant/forecast` - Chronos-2 probabilistic forecasting
- `POST /api/v1/quant/regime` - Market regime detection (CEO node)
- `GET /api/v1/quant/status` - System health and component status
- `GET /health` - Health check endpoint

Planned Training Pipeline: For future Feudal RL implementation, the training would be a multi-stage process:

Pre-training:

- **Encoders:** Vision, Text, and Time Series encoders pre-trained on historical data using self-supervised objectives.
- **Logic:** LTN predicates initialized and knowledge base $\mathcal{K}$ compiled into computation graph.

Feudal RL Training:

- **Phase 1 (Worker):** Worker trained in isolation using Almgren-Chriss objective with randomized goals and risk parameters.
- **Phase 2 (Manager):** Manager trained to optimize high-level PnL with frozen Worker.
- **Phase 3 (Joint Fine-Tuning):** Both agents trained end-to-end with logical loss $\mathcal{L}_{logic}$ added to reward signal.

7 Simulation, Backtesting, and Validation

7.1 Methodological Rigour

Validation of the JANUS architecture requires more than a simple P&L backtest. We employ a rigorous simulation environment that stresses the "Neuro-Symbolic" aspects of the system.

- **Constraint Violation Rate:** The primary metric for the Logic Layer. We measure the percentage of trades that violate the Wash Sale rule or Almgren-Chriss risk limits. The target is effectively 0%.
- **Zero-Shot Generalization:** Testing the model on asset classes it was not trained on (e.g., training on the S&P 500 constituents and testing on cryptocurrency pairs). This validates the "Foundation Model" capabilities of the Chronos backbone.
- **Market Impact Simulation:** Using the Almgren-Chriss temporary impact function $h(v) = \eta v + \gamma |v|^{\alpha}$ to simulate the realistic cost of trading a large size.

7.2 Comparative Analysis

In preliminary simulations (based on aggregated benchmarks like fev-bench and GIFT-Eval), architectures similar to JANUS show distinct advantages:

Metric	**Traditional Quant (2.0)**	**Pure Deep RL (3.0)**	**JANUS (4.0)**
Forecasting Accuracy	Low (Linear)	High (Non-Linear)	High (Multimodal)
Constraint Adherence	High (Hard-coded)	Low (Reward Hacking)	High (Neuro-Symbolic)
Adaptability	Low (Rigid Params)	Low (Brittleness)	High (Zero-Shot)
Explainability	High	None (Black Box)	Moderate (Logic Trace)

The production implementation combines Chronos-2's zero-shot forecasting with Product-not-Sum consensus and rule-based risk management. This provides a robust foundation for autonomous trading. Future integration of LTN constraint enforcement will further enhance the system's ability to achieve "State-of-the-Art" returns while maintaining "Regulatory Grade" safety.

8 Regulatory Compliance, Ethics, and Systemic Risk

8.1 The "Glass Box" Transparency

A defining feature of Quant 4.0 is Explainability. In a standard Deep RL model, querying "Why did you reduce the position?" yields no intelligible answer.

Current Implementation: The production version of JANUS provides explainability through:

- **Reasoning Traces:** The HRM workflow logs CEO and Worker node decisions, including regime assessments and signal generation rationale.
- **Motor Scores:** The consensus mechanism exposes individual motor scores (Chronos, CNN, Risk, Fundamental), allowing inspection of each component's contribution.
- **Response Metadata:** Analysis responses include detailed reasoning, forecast data, regime information, and risk assessments.

Future Enhancement with LTN: When Logic Tensor Networks are integrated, the system will provide formal logical traces:

Query: "Why was the order size reduced?"

> **LTN Trace:** "Predicate *HighVolatility*(VisualInput) is TRUE (0.95 confidence). Predicate *LiquidityRisk* is TRUE. Rule #4 triggered: HighVol $\rightarrow$ ReduceSize."

Visual Evidence: With GAF integration, the system can generate Grad-CAM heatmaps. These highlight textures that triggered the predicates.

This "Glass Box" transparency is essential for gaining trust from risk managers and regulators (e.g., SEC, FCA), who increasingly demand auditability for algorithmic trading systems.

8.2 Addressing Systemic Risk

While JANUS is designed to be individually safe, the widespread adoption of such "aware" agents introduces new systemic risks. If thousands of JANUS agents simultaneously detect a "Crash Texture" and logically decide to "Halt Trading" (via the VPIN rule), market liquidity could evaporate instantly, exacerbating a Flash Crash.

Future research must investigate Multi-Agent Reinforcement Learning (MARL) scenarios. We must model the collective behaviour of JANUS populations and potentially introduce "Cooperative Logic" constraints that incentivize agents to maintain market stability (e.g., acting as a market maker of last resort) in exchange for regulatory credits or reduced transaction fees.

9 Future Directions: Toward General Financial Intelligence

Project JANUS is a stepping stone toward General Financial Intelligence (GFI). By moving away from "curve fitting" historical prices and toward agents that perceive, read, and reason, we are creating systems that understand the market as a complex, evolving sociotechnical system.

9.1 Quantum Computing (Quant 5.0?)

As we look beyond Quant 4.0, the integration of Quantum Computing offers the next leap. Quantum algorithms (e.g., Quantum Approximate Optimization Algorithm or QAOA) could solve the Almgren-Chriss portfolio optimization problem (which is NP-hard with certain constraints) exponentially faster than classical solvers. The "Quantum Gramian Angular Field" (QGAF) is already being explored as a way to encode time series into quantum states for processing on QPUs. JANUS is architected to be "Quantum-Ready," with its modular design allowing the classical Logic/Optimization layers to be swapped for Quantum equivalents as the hardware matures.

9.2 The Final Synthesis

The ultimate vision of JANUS is a trading agent that possesses the "Wisdom" of a veteran trader (via Symbolic Logic), the "Intuition" of a chartist (via Visual Encoding), and the "Speed" of a machine (via Hierarchical Execution). By fusing these faculties, JANUS provides the rigorous mathematical and architectural framework to realize the promise of Neuro-Symbolic AI in finance—creating a system that is not only profitable but also principled.

10 Conclusion

Project JANUS: The Neuro-Symbolic Visual Quant Architecture is not merely an incremental improvement in forecasting accuracy; it is a structural redefinition of the quantitative trading agent. It directly addresses the limitations of Quant 3.0—opacity, brittleness, and safety risks—by synthesizing the visual intuition of Gramian Angular Fields, the rigorous reasoning of Logic Tensor Networks, and the strategic hierarchy of Feudal Reinforcement Learning.

The architecture detailed in this white paper offers a comprehensive blueprint for Quant 4.0: systems that are automated yet explainable, data-driven yet knowledge-constrained, and highly performant yet regulatory compliant. As financial markets become increasingly fast, complex, and interconnected, the ability to fuse multimodal perception with symbolic logic will define the alpha generators of the next decade. JANUS stands as the vanguard of this new era.

References

[1] IDEA Research Report - arXiv. https://arxiv.org/pdf/2301.04020

[2] Multimodal Language Models with Modality-Specific Experts for Financial Forecasting from Interleaved Sequences of Text and Time Series - arXiv. https://arxiv.org/html/2509.19628v1

[3] Neuro-Symbolic Traders: Assessing the Wisdom of AI Crowds in Markets - arXiv. https://arxiv.org/html/2410.14587v1

[4] Temporal Convolutional Networks for Financial Time Series Forecasting: A Survey. https://www.researchgate.net/publication/392102503_Temporal_Convolutional_Networks_for_Financial_Time_Series_Forecasting_A_Survey

[5] Chronos-2: From Univariate to Universal Forecasting - arXiv. https://arxiv.org/html/2510.15821v1

[6] Logic Tensor Networks: The Neuro-Symbolic Revolution | by Gonnect. Medium. https://medium.com/@gonnect.uk/logic-tensor-networks-the-neuro-symbolic-revolution-b4e2d022b188

[7] Neuro-Symbolic AI in 2024: A Systematic Review - arXiv. https://arxiv.org/pdf/2501.05435

[8] Statement of Good Practice for the application of a model risk management framework to electronic trading algorithms - FMSB. https://fmsb.com/wp-content/uploads/2025/04/Model-Risk-Electronic-Trading-Algorithm_FINAL-05.04.pdf

[9] From Deep Learning to LLMs: A survey of AI in Quantitative Investment - arXiv. https://arxiv.org/html/2503.21422v1

[10] Gramian Angular Fields keep popping up in time-series literature. Reddit. https://www.reddit.com/r/quant/comments/1p81cuo/gramian_angular_fields_keep_popping_up_in/

[11] Algorithmic Financial Trading with Deep Convolutional Neural Networks: Time Series to Image Conversion Approach - ResearchGate. https://www.researchgate.net/publication/324802031_Algorithmic_Financial_Trading_with_Deep_Convolutional_Neural_Networks_Time_Series_to_Image_Conversion_Approach

[12] Forecasting Stock Market Volatility Using CNN-BiLSTM-Attention Model with Mixed-Frequency Data - MDPI. https://www.mdpi.com/2227-7390/13/11/1889

[13] Using CNN for financial time series prediction - MachineLearningMastery.com. https://machinelearningmastery.com/using-cnn-for-financial-time-series-prediction/

[14] RGB GAF image: A possible solution to one weak point of Gramian Angular Field Imaging | by Shuyang Xiang. Medium. https://medium.com/data-science/rgb-gaf-image-a-possible-solution-to-one-weak-point-of-gramian-angular-field-imaging-ffc6b31edfbe

[15] Fusion of Recurrence Plots and Gramian Angular Fields with Bayesian Optimization for Enhanced Time-Series Classification - MDPI. https://www.mdpi.com/2075-1680/14/7/528

[16] Transfer Learning in Financial Time Series with Gramian Angular Field - arXiv. https://arxiv.org/html/2504.00378v1

[17] Quantum-Enhanced Forecasting: Leveraging Quantum Gramian Angular Field and CNNs for Stock Return Predictions - arXiv. https://arxiv.org/abs/2310.07427

[18] Wash Sale: Definition, How It Works, and Purpose - Investopedia. https://www.investopedia.com/terms/w/washsale.asp

[19] Optimal Execution Strategies - Almgren-Chriss Model - QuestDB. https://questdb.com/glossary/optimal-execution-strategies-almgren-chriss-model/

[20] logictensornetworks/logictensornetworks: Deep Learning and Logical Reasoning from Data and Knowledge - GitHub. https://github.com/logictensornetworks/logictensornetworks

[21] Logic Tensor Networks: Deep Learning and Logical Reasoning from Data and Knowledge - arXiv. https://arxiv.org/abs/1606.04422

[22] Logical Neural Networks using Pytorch | by Andrea Rosales. Medium. https://medium.com/@andrea.rosales08/logical-neural-networks-using-pytorch-a017d96d3e05

[23] No Prior Mask: Eliminate Redundant Action for Deep Reinforcement Learning - AAAI Publications. https://ojs.aaai.org/index.php/AAAI/article/view/29652/31109

[24] VPIN: The Volume Synchronized Probability of Informed Trading. https://www.quantresearch.org/VPIN.pdf

[25] From PIN to VPIN: An introduction to order flow toxicity. https://www.quantresearch.org/From%20PIN%20to%20VPIN.pdf

[26] Introducing Chronos-2: From univariate to universal forecasting - Amazon Science. https://www.amazon.science/blog/introducing-chronos-2-from-univariate-to-universal-forecasting

[27] Stock Movement Prediction with Multimodal Stable Fusion via Gated Cross-Attention Mechanism - arXiv. https://arxiv.org/html/2406.06594v2

[28] Financial Sentiment Analysis Using FinBERT with Application in Predicting Stock Movement. https://arxiv.org/html/2306.02136v3

[29] FinBERT: A Pre-trained Financial Language Representation Model for Financial Text Mining - IJCAI. https://www.ijcai.org/proceedings/2020/0622.pdf

[30] MM-iTransformer: A Multimodal Approach to Economic Time Series Forecasting with Textual Data - MDPI. https://www.mdpi.com/2076-3417/15/3/1241

[31] FeUdal Networks for Hierarchical Reinforcement Learning - arXiv. https://arxiv.org/abs/1703.01161

[32] Hierarchical Reinforcement Learning: FeUdal Networks | by Austin Nguyen. Medium. https://medium.com/data-science/hierarchical-reinforcement-learning-feudal-networks-44e2657526d7

[33] Hierarchical Reinforcement Learning for Algorithmic Trading - ETH Zürich. https://pub.tik.ee.ethz.ch/students/2021-HS/SA-2021-48.pdf

[34] Hierarchical Reinforced Trader (HRT): A Bi-Level Approach for Optimizing Stock Selection and Execution - arXiv. https://arxiv.org/html/2410.14927v1

[35] A High Frequency Trade Execution Model for Supervised Learning - arXiv. https://arxiv.org/abs/1710.03870

[36] Reinforcement Learning for Financial Trading: Algorithms, Evaluations and Platforms - Nanyang Technological University. https://www3.ntu.edu.sg/home/boan/thesis/Sun_Shuo_PhD_Thesis.pdf

[37] amazon-science/chronos-forecasting: Chronos: Pretrained Models for Time Series Forecasting - GitHub. https://github.com/amazon-science/chronos-forecasting

[38] amazon/chronos-bolt-small - Hugging Face.
https://huggingface.co/amazon/chronos-bolt-small

[39] amazon/chronos-2 - Hugging Face.
https://huggingface.co/amazon/chronos-2

[40] Algorithmic Trading Compliance in Wholesale Markets - Financial Conduct Authority. https://www.fca.org.uk/publication/multi-firm-reviews/algorithmic-trading-compliance-wholesale-markets.pdf

[41] No More Hand-Tuning Rewards: Masked Constrained Policy Optimization for Safe Reinforcement Learning - IFAAMAS.
https://www.ifaamas.org/Proceedings/aamas2021/pdfs/p1344.pdf

[42] Feudal Multi-Agent Hierarchies for Cooperative Reinforcement Learning - ALA 2019. https://ala2019.vub.ac.be/papers/ALA2019_paper_5.pdf

[43] Dynamic Portfolio Optimization with Real Datasets Using Quantum Processors and Quantum-Inspired Tensor Networks - arXiv.
https://arxiv.org/abs/2007.00017