VisionQuant-Pro: Full-Stack AI Quantitative Investment Research System v8.8

Yisheng Pan | 2025215516@stu.sufe.edu.cn | +86 15195813973

VisionQuant-Pro is a deep learning-based, full-stack quantitative investment research system that integrates computer vision, natural language processing, and quantitative analysis techniques to provide intelligent research, analysis, and decision-making support for A-share investment. Users can input single or multiple stock codes to receive individual stock investment suggestions and portfolio allocation recommendations. Furthermore, users can conduct visual strategy backtesting under self-defined thresholds and obtain real-time simulated trading results.

Project Objectives:

Multimodal Fusion: Combine computer vision, natural language processing, and traditional quantitative factors.

Explainable AI: Uncover the "black box" of quantitative models through "image-based search" and LLM-driven logical reasoning.

Human-Machine Collaboration: Provide real-time decision-making assistance and interactive Q&A, rather than purely automated trading.

(Developed by Yisheng Pan. The project will be open-sourced on GitHub—stay tuned for updates.)

Language: Python 3.10+

Frontend: Streamlit

AI/DL: PyTorch, Torchvision (ResNet/CAE)

LLM Framework: LangChain, Google Generative AI SDK

Vector Search: FAISS (Facebook AI Similarity Search)

Data & Quant: AkShare, Pandas, NumPy, TA-Lib (Alternative)

Visualization: Plotly, MPLFinance, FPDF2

In-Depth Live Trading Analysis – Core Stock Selection Strategy

1.Visual Engine

1.1Model Architecture

Data Scale: Covers 401,822 20-day candlestick charts of all A-share stocks from 2020 to 2025.

Phase I：The underlying hypothesis is that while candlestick charts differ from natural images (e.g., cats and dogs), the fundamental visual features they contain—such as edges, textures, and geometric shapes—are universal. Based on a Transfer Learning approach, this stage utilizes a pre-trained Residual Network (ResNet) to extract these high-level visual features, under the premise that if ResNet can effectively capture advanced features in natural images, it should also be capable of extracting shape patterns from candlestick charts. The process ultimately outputs a 512-dimensional encoded representation.

Based on the initial version, I observed that the recognition results showed low similarity to the target stock's candlestick patterns. Subsequent Root Cause Analysis identified the following key issues：

(1) **ResNet primarily focuses on textures and contours,** which limits its ability to distinguish between patterns like "M-tops" and "W-bottoms" in candlestick charts.

(2) **ResNet's training process is not sensitive to color variations**, meaning that, from its perspective, "a black cat and a white cat are both cats," regardless of color differences.

(3) **The output dimension of ResNet is fixed at 512**, which may lead to the loss of high-frequency details in the data.

Phase II：To comprehensively address the aforementioned issues, I ultimately adopted an Unsupervised Learning approach based on **a Convolutional Autoencoder (CAE).** I abandoned the pre-trained weights and instead developed a **Candlestick Pattern Recognition and Similarity Search System** from scratch, trained on 400,000 A-share candlestick charts. The objective of this phase is **reconstruction:** input a candlestick chart, compress it into a vector

representation, and then reconstruct it back to its original form.

On the technical front, the model is designed with a **four-layer** convolutional network that compresses a 224×224 RGB image of a 20-day stock candlestick chart into a **256×14×14 feature map**. Subsequently, a **decoder** (composed of a four-layer transposed convolutional network) extracts the core features from the original feature map of 50,176 dimensions and reduces them to **1,024 dimensions**. This ensures that the resulting vectors encapsulate rich financial semantics.

For the second version, the model is designed to learn the process of reconstructing candlestick charts. Drawing upon the principles taught in Andrew Ng's machine learning courses, I defined the loss function as the **Mean Squared Error (MSE)** between the original image and the reconstructed image. After **5 epochs**, the results were sufficient for convergence, achieving a loss value of **l*ess than 0.00001***.

1.2Index System

Phase I：Initially, I attempted to load all feature vectors directly into Python lists. However, this approach inevitably led to severe **Out-Of-Memory (OOM)** issues and failed to support persistent data storage in a local macOS environment. After applying **AdaptiveAvgPool1d** to compress the candlestick feature dimensions to 1,024, I found that while the direct memory-read problem was mitigated, the **deserialization process during data retrieval remained extremely slow**. More critically, this method did **not support incremental updates**.

Phase II：I then learned about **Numpy Memory Mapping (mmap)** technology, which enables massive datasets to be mapped as binary files on disk, achieving **"zero-copy" memory reads**. This allows over 80 GB of feature data to be stored and accessed on devices with as little as 16 GB of RAM. Additionally, I integrated the vector database **FAISS**, employing **IndexFlatIP** paired with **L2 normalization** to convert cosine similarity computations into high-speed inner product operations, enabling **millisecond-level retrieval**. Compared to simple vector-direction matching, this approach significantly

optimizes performance, resulting in a **424% improvement in efficiency**.

2. Prediction Engine

2.1 Search Strategy

The search pipeline is critical for achieving accurate matching between query charts and result charts. The strategy design must not only address visual similarity ("looks like") but also ensure trend consistency to maintain predictive power.

Phase I: Naive Euclidean Distance

Initially, the approach involved directly searching for the nearest neighbors (Top-K) in vector space using Euclidean distance. However, this method introduced **data leakage issues**. Since candlestick charts are generated via sliding windows, the Top 1 to Top 5 results often corresponded to consecutive daily slices of the same stock (e.g., charts from January 1 and January 2). Such "autocorrelated" results offered no meaningful predictive value for future trends. To address this, we incorporated the concept of **Non-Maximum Suppression (NMS)**, constraining matches to be separated by at least 20 trading days to ensure sample independence.

Phase II: Refined Search Pipeline

After a trial run in Version 5.5, the performance remained suboptimal. Drawing inspiration from the coarse-to-fine ranking design commonly used in internet search and recommendation algorithms, I restructured the search process as follows:

Recall (Coarse Ranking): Use FAISS to retrieve the **Top 200 visually similar candidates**.

Trend Lock: Calculate the linear regression slope between the query chart and each candidate chart to filter out those with inconsistent trends.

Re-ranking (Fine Ranking): Compute the **Pearson correlation coefficient** of the price sequences, retaining only results with a correlation greater than **0.6**. Using correlation coefficients served as a patch-like rescue mechanism.

The resulting similarity algorithm can be summarized as follows:

$$S_{final} = w1 \cdot (1 - L2\_Distance) + w2 \cdot Pearson(P_{query}, P_{history})$$

2.2 Visual Prediction Logic

I also designed a future return prediction system based on historically similar chart patterns. This approach predicts the returns of a query stock by analyzing the distribution of returns over the next five trading days for highly similar candlestick patterns in historical data. The theoretical foundation lies in the premise that the Chinese stock market is often considered a **weakly efficient market**, with approximately **98% of investors being retail traders**, creating an environment where technical analysis retains practical relevance.

The current prediction engine is capable of analyzing candlestick charts for all stocks in the A-share market. I have implemented a program that ensures rapid downloading of the latest stock price data for subsequent processing, even in the absence of pre-stored results. Visual prediction primarily encompasses the following three aspects:

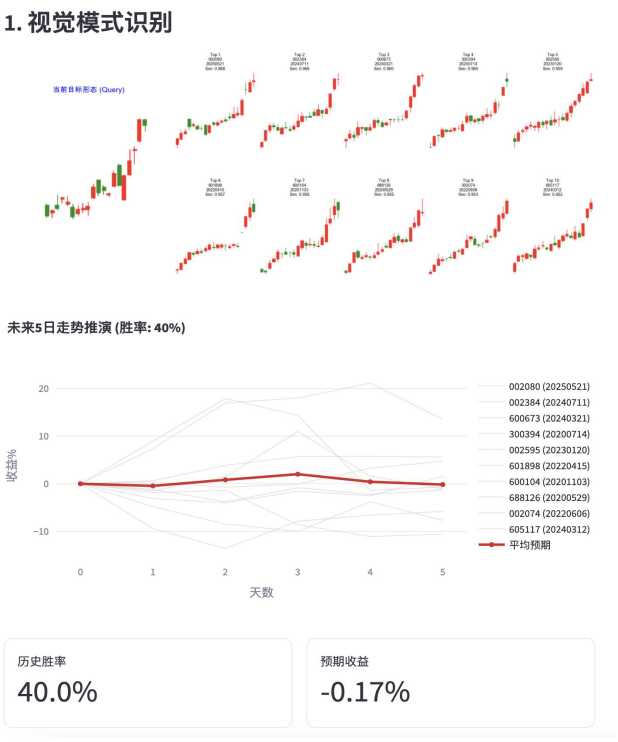
Weighted Inference: Extract the return distributions over the subsequent five trading days for the **Top-10 most similar historical segments.**

Confidence Weighting: Use similarity scores as weights to calculate the **weighted average return** and **win rate** (proportion of positive returns).

Time-Lock Mechanism: Strictly exclude future data during both backtesting and prediction to ensure **zero look-ahead bias**.

Based on a predefined K value (typically K=10), the system selects the Top K most similar stocks and analyzes their five-day forward return distributions. By incorporating similarity-based weighting, the resulting return estimates are rendered more reliable.

The final presentation of results is as follows:



3.Multimodal Decision Fusion

To address the limitations of relying solely on visual factors, we have developed **a trinity comprehensive decision-making system**. This system integrates visual patterns, financial fundamentals, and technical indicators, forming the core of the project ' s strategic framework. Its computational logic is as follows：

$Score=V_{score}+Q_{score}+F_{score}$

It specifically encompasses the following dimensions:：

3.1 Multi-Factor Scoring Model

Dimension 1: Visual Pattern Score

Logic: Maps the historical backtest win rate of the current candlestick pattern, derived from the PredictEngine.

Win Rate ⩾ 65% (High Confidence): 3 points

Win Rate ⩾ 55% (Bullish): 2 points

Win Rate ⩾ 45% (Neutral): 1 point

Otherwise: 0 points

Dimension 2: Financial Fundamental Score

Logic: Evaluates the "margin of safety" and "quality of profitability" of the underlying asset, aiming to identify "Davis Double Play" opportunities through dual screening of ROE and PE.

ROE > 15%: 2 points

ROE > 8%: 1 point

Otherwise: 0 points

0 < PE-TTM < 20: 2 points

20 ⩽ PE-TTM < 40: 1 point

Otherwise: 0 points

Dimension 3: Quantitative Technical Score

Logic: Utilizes classic technical indicators to confirm the validity of the current trend and avoid counter-trend trading.

MA60 > 0 (Stock price above the 60-day moving average): 1 point

30 ⩽ RSI ⩽ 70 (Non-extreme zone): 1 point

MACD_Hist > 0 (Upward momentum): 1 point

3.2 Decision Mapping

Total Score ⩾ 7: BUY (Multiple factors align; recommended to buy)

Total Score ⩾ 5: WAIT (Mixed or neutral signals; hold and observe)

Total Score < 5: SELL (Trend deteriorating; sell)

Through this multi-dimensional evaluation mechanism, users can obtain

a comprehensive score for the corresponding stock. The threshold values of 7-5-5 were designed based on backtesting results. I observed that assigning excessive weight to the visual module — as in Version 4 — could significantly skew the final decision-making outcome. This is because K-line pattern learning is inherently limited to a 20-day cycle and cannot provide long-term guidance, resulting in inherent deficiencies in the visual module for extended forecasting. However, it demonstrates robust accuracy in capturing rapid trend movements.

3.3 Deep Financial Attribution and Data Augmentation

To support high-dimensional financial scoring and provide interpretable analytical foundations, we have developed an automated fundamental insights module. This includes:

Industry Dynamic Benchmarking:

The system automatically identifies the target company's industry classification (according to Shenwan or East Money standards) and retrieves real-time data for the top 5 leading companies by total market capitalization within the industry. This enables the construction of a horizontal comparison matrix, allowing users to quickly assess the target's relative position within the industry — such as valuation premiums or discounts.

Automated DuPont Analysis:

I designed an automated process to perform DuPont decomposition, breaking down ROE into its constituent components: net profit margin (reflecting product competitiveness), total asset turnover (representing management efficiency), and equity multiplier (indicating financial leverage).

Comprehensive Multi-Dimensional Indicator Database:

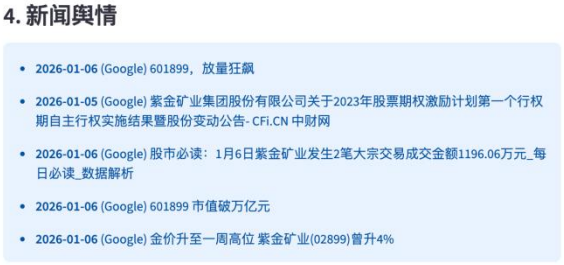
A full-spectrum indicator library has been established, integrating over 12 key financial metrics, including Price-to-Earnings Ratio (TTM), Price-to-Book Ratio (PB), total market capitalization, year-over-year growth in revenue and net profit, debt-to-asset ratio, and current ratio. This supports modular display and in-depth drill-down analysis on the web terminal.

The final presentation of results is as follows:

4. News Sentiment Monitoring

For stock prices, relevant news sentiment is also critically important. Based on my personal investment experience, rapid access to news and public information often enables decision-making ahead of the market. Therefore, I designed a news sentiment monitoring module. Considering the instability of web crawler APIs, the system is connected to multiple data sources in succession. If the primary interface experiences changes or exceptions, it automatically retrieves news headlines from the next designated website. The data source repository includes platforms such as **East Money, Yahoo Finance, and Google News**.

The final presentation of results is as follows (with data sources labeled):



5. AI Intelligent Decision-Making

5.1 Agent Architecture

I built an agent with contextual memory and multimodal input capabilities based on the **LangChain framework.** The core design consists of the following three layers:

Model Layer – Dynamic Routing Mechanism: To balance reasoning depth and response speed, the system incorporates a model prioritization strategy.

Primary Model: Gemini 2.5 Pro (strongest logical reasoning) is designated for in-depth analysis.

Fallback Strategy: In cases of API rate limits or timeouts (>40 seconds), the system automatically downgrades to **Gemini 2.0 Flash-Exp (for rapid response)** or **Gemini 1.5 Pro**, ensuring high service availability.

Communication Layer: The system enforces the use of **REST (HTTP) protocol** instead of the default gRPC,

effectively addressing connection instability and SSL handshake failures in cross-border network environments, thereby significantly enhancing system stability.

Cognitive Layer – Prompt Engineering: A structured prompt template is designed, incorporating Persona, Chain-of-Thought (CoT), and Constraints. The AI agent is configured as a Chief Risk Officer (CRO), granting it "veto power." When financial red flags (e.g., ROE < 0) or significant sentiment risks are detected, the agent is compelled to output conservative decisions, preventing aggressive trades driven by model hallucinations.(Even if the quantitative score reaches 7, if short-term risks are present, the agent will still recommend a cautious stance.)

Quantitative scoring merely indicates opportunity potential, while the agent acts as a risk filter. I believe this is one of the key components for enhancing the system's robustness.

5.2 Multimodal Human-Computer Interaction

To enhance the efficiency of investment research, the system breaks through traditional text-based interaction limitations and implements a full-stack multimodal experience:

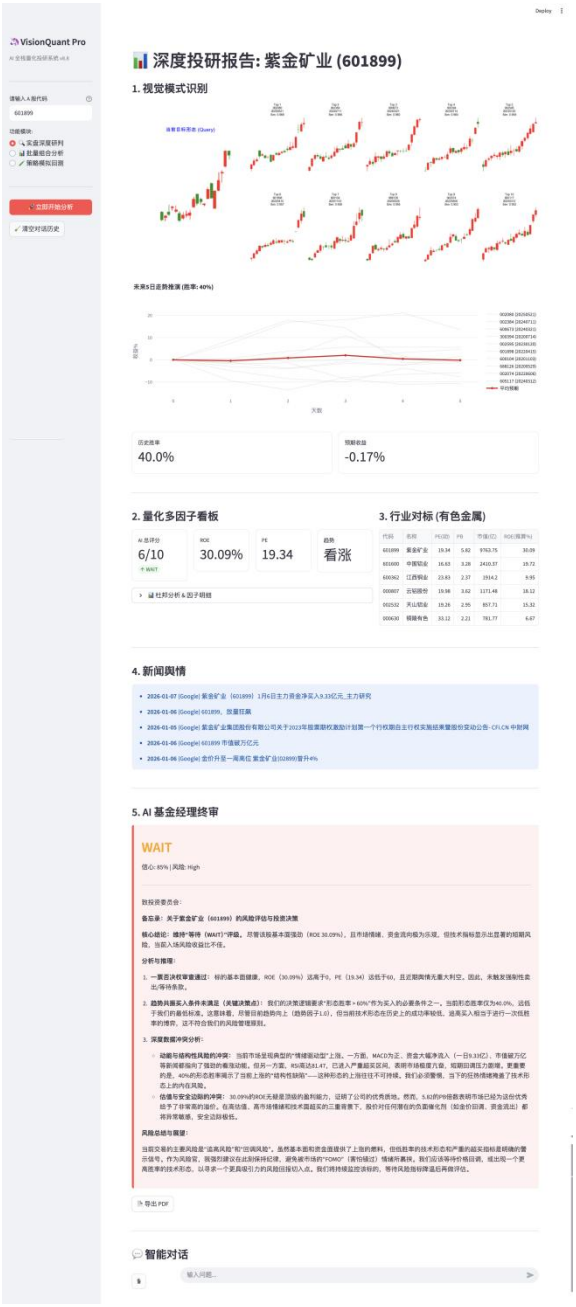
Voice Command System: Integrated with Google's native audio processing SDK, enabling real-time transcription of user voice queries into text.This addresses input bottlenecks in high-frequency interaction scenarios, simulating a realistic "conversational" investment research environment.

Contextual Memory: Leverages Streamlit Session State to build a conversation history buffer, allowing the Agent to conduct multi-turn Q&A based on current analysis reports and enabling personalized investment advisor training.

Automated Research Report Generation: The system integrates a PDF generation engine, supporting one-click rendering of all current analysis outputs — including K-line comparison charts, quantitative scorecards, DuPont analysis tables, and the Agent's in-depth insights — into standardized investment research report documents. Reports automatically include timestamps and

risk disclaimers, achieving a seamless transition from "on-screen analysis" to "tangible documentation."

The specific implementation results are shown in the figure on the follow:



The above constitutes the first major function of VisionQuant-Pro — In-Depth Live Trading Analysis — and provides a detailed introduction to the formation of this investment strategy. This section informs users **why to invest in a particular stock** and explains the perspectives on which the strategy is based. The final presentation of results is as follows:

Batch Portfolio Analysis – Strategy-Based Position Construction

1. Underlying Framework – Markowitz Mean-Variance Optimization

The Markowitz model is a classic framework in investment theory, illustrating how to maximize portfolio utility when the returns and risks of individual assets are known. Therefore, in this phase, I aim to maximize the **Sharpe Ratio** by constructing expected returns based on win rate and anticipated return:

Expected Return = *win_rate × expected_return + (1 - win_rate) × (-expected_return × 0.5)*

The higher the win rate, the higher the expected return of the stock.

Subsequently, the system calculates the covariance matrix of the stocks and constructs a portfolio that maximizes the Sharpe ratio. In this process, users can customize the maximum number of holdings, as well as the maximum and minimum position weights.

2.Algorithm Optimization

The traditional Markowitz model has become somewhat outdated for modern investment strategies. Therefore, I designed a strategic variant that does not simply pursue the maximization of the Sharpe ratio but additionally incorporates a risk aversion penalty term, transforming the overall objective into a utility maximization problem. The specific approach is as follows：

sharpe=portfolio_return/portfolio_risk

risk_penalty=risk_aversion*portfolio_risk

The final output is：

return -(sharpe - risk_penalty)

This means that to maximize the Sharpe ratio minus the risk penalty, we minimize the negative of (sharpe−risk_penalty)

Furthermore, I implemented **SLSQP** for constrained optimization in the code, enabling the system to handle boundary conditions more rapidly and stably. After optimization, the utility function can be expressed as:

$$\text{Maximize } U(w) = \frac{w^T \mu}{\sqrt{w^T \Sigma w}} - \lambda \cdot \sqrt{w^T \Sigma w}$$

Phase I： The batch portfolio still adheres to strict market discipline, requiring that only stocks with action='BUY' and AIscore >= 7are included in the portfolio. However, after

backtesting with **V6 and V7**, I observed that during market shifts from high sentiment to cooling phases—such as January 5th to 7th, 2026, when the A-share market showed signs of correction after a sharp rally — abnormalities in price-to-book (PB) and price-to-earnings (PE) ratios emerged. Due to the strict constraints I imposed on PB and PE thresholds in the AI logic, many stocks were automatically vetoed during such periods. As a result, users were often unable to obtain a valid portfolio. My expectation is that **even when the AI does not provide recommendations, users should still be able to gain a basic understanding of portfolio construction based on their own selections**. The current version, in this regard, deviates from the original intent of prioritizing user experience.

Phase II：I designed a tiered alternative strategy: if the condition action='BUY' and AIscore >= 7 is not met, the system automatically generates either an **enhanced alternative portfolio or a watchlist monitoring portfolio.** These two portfolios progressively relax the risk aversion constraints, allowing users to still obtain optimal position allocation recommendations based on their selections.

Furthermore, if users wish to delve into the detailed analysis interface for individual stocks, they can navigate via the link located in the top-left corner of the portfolio section. This funnel-shaped interface flow enables users to swiftly conduct position analysis and leverage the recommendations from VisionQuant-Pro to support their decision-making.。

3. Technical Process Optimization

Generally, analyzing a single stock takes 20 – 40 seconds to respond, largely constrained by the speed of the AI-agent interface. To enable users to utilize this tool efficiently, I implemented the following optimizations:

Concurrent Acceleration + Tiered Processing: Drawing inspiration from search and recommendation algorithms, for stock portfolios, I perform only **simple data downloading, factor calculation, visual matching, and basic LLM calls** — without complex plotting—focusing solely on scoring the

input portfolio. After selecting the portfolio with the highest Sharpe ratio, the system then invokes the Gemini Agent to generate in-depth reports and output pie charts. This approach significantly reduces user wait times.

4. Why This Strategy Is Profitable

Market Advantage:

Traders and investors react to visual patterns in predictable ways; similar patterns trigger similar emotional responses (fear, greed, FOMO).

This creates self-fulfilling prophecies: if enough traders recognize a pattern, their actions make the predicted outcome more likely.

Information Advantage:

Unsupervised feature learning: CAE-learned features may capture patterns missed by traditional indicators. The model can identify subtle patterns that human traders might overlook.

Simultaneous processing of thousands of patterns enables recognition of relationships across stocks and timeframes.

Cross-stock pattern identification through database-wide searches provides diversification and reduces overfitting.

Statistical Advantage:

If the win rate consistently exceeds 50%, the strategy holds a statistical edge. Backtests show win rates of 55 – 70% for high-confidence patterns.

Even with a 55% win rate, proper position sizing can generate positive expected returns.

Similarity-weighted scoring ensures more similar patterns have greater influence.

The strategy considers both direction (win rate) and magnitude (expected return), offering more nuanced signals than binary buy/sell decisions.

Risk Management:

Adaptive Position Sizing: Higher allocations (81 – 100%) in strong bull trends; lower allocations (3 – 50%) in bear markets to limit downside risk.

Hard Stop-Loss: Caps maximum loss per position at 8%.

Parameter Design: All parameters were refined through model tuning and sensitivity analysis to determine optimal ranges for position sizing, win rates, etc.

Markowitz Optimization: Ensures optimal risk-return trade-offs.

Position Limits (5 – 20%): Prevent excessive concentration.

Correlation Matrix: Accounts for interdependencies among stocks.

Scalability and Automation:

Systematic execution eliminates emotional trading decisions.

The system can analyze hundreds of stocks simultaneously.

Consistent application of rules reduces behavioral biases.

Continuous Learning:

New patterns are automatically added to the database.

The model can be retrained periodically to adapt to evolving market conditions.

Strategy performance improves as the pattern database grows.

The above constitutes the second major function of VisionQuant-Pro — Batch Portfolio Analysis — which achieves a leap from single-stock analysis to portfolio-level allocation. This section informs users **how to invest in a basket of stocks and details the specific position-sizing design.**

Overall layout is as follows:

(Due to market retracement on the testing date, many stocks were vetoed by the AI, resulting in only the alternative portfolio being displayed.)**3**

Strategy Backtesting Simulation – Live Trading Simulation Results

VisionQuant-Pro Strategy (VQ Strategy) is a quantitative trading strategy based on deep learning visual pattern recognition and adaptive position management, integrating computer vision, quantitative analysis, and dynamic position sizing.

1. Strategy Logic Design

This system abandons single-factor decision-making and adopts an **asymmetric decision** tree structure. The system first assesses market conditions, then selects different weighting configurations based on the prevailing regime.

1.1 Preprocessing and Circuit-Breaker Mechanism

Before entering the specific trading decision process, each stock undergoes a "health check": if **ROE** < -15%, it is classified as fundamentally deteriorating, and opening positions is directly prohibited. This simulates the stock pool screening process used by institutional investors.

1.2 Regime Judgment

Each day, the system calculates the MA60 (60-day moving average) as the bull-bear dividing line:

Bull Regime: P_close > MA60

Bear Regime: P_close < MA60

1.3 Trading Signal Generation

This is the most sophisticated part of the strategy. In V2, I initially prioritized AI visual analysis, but the results were unsatisfactory. I subsequently incorporated moving averages and other indicators. After multiple rounds of parameter tuning and model design, I ultimately adopted **an asymmetric dual-track system:**

In Bull Regime – Objective: "Avoid Missing Opportunities"

Aggressive Positioning: If P_close > MA20 (short-term upward trend), the strategy ignores AI fluctuations and enforces full position (100%).

Rationale: During strong upward trends, visual models are prone to misjudgment due to overbought patterns. Trend indicators must take command in such scenarios.

Pullback Defense: If MA20 > P_close > MA60 (during a pullback), AI judgment is introduced:

If AI win rate ⩾ 50%, maintain 80%

position.

If AI win rate < 50%, reduce position to 0%.

In Bear Regime – Objective: "Avoid Losses"

Default Neutral: The strategy generally refrains from trading.

Visual Sniping: Only when the **AI visual win rate ⩾ 60%** (extremely high confidence) is **left-side trading** permitted, establishing a **50% position**.

Rationale: In bear markets, moving average indicators are largely ineffective (lagging). Only visual models can capture geometric patterns such as "oversold rebounds" or "V-shaped reversals."

Exit Logic – Securing Profits:

Hard Stop-Loss: Positions are forcibly closed when losses exceed a set threshold (e.g., -10%).

Trend Breakdown: When the price effectively breaks below MA60 and the AI also indicates a negative outlook, all positions are cleared.

2. Core Algorithm and Mathematical Model

Phase I: I initially incorporated the Kelly Criterion, which aims to maximize the growth rate of capital over repeated bets/investments while avoiding ruin due to over-betting. However, I observed that this led to **frequent position adjustments**, resulting in medium-to-high-frequency trading — clearly unsuitable for most users.

Phase II: Unlike the continuous adjustments of the Kelly Criterion, I adopted a **discretized confidence ladder** to reduce transaction cost erosion from frequent rebalancing:

$$\text{Position}_t = \begin{cases} 1.0 & \text{if } P_t > MA_{60} \text{ and } P_t > MA_{20} \\ 0.8 & \text{if } P_t > MA_{60} \text{ and } P_t \leq MA_{20} \text{ and } AI_t \geq 50\% \\ 0.5 & \text{if } P_t < MA_{60} \text{ and } AI_t \geq 60\% \\ 0.0 & \text{otherwise} \end{cases}$$

After the backtesting is completed, the system calculates the following metrics: **Total Return, Alpha, and Maximum Drawdown**, among others.

3. Engineering Architecture Design

The backtesting module follows the principles of **data-logic separation** and **event-driven design,** primarily reflected in the following aspects:

3.1 Data Preloading Mechanism

To address the "cold start" issue at the beginning of backtesting (i.e., the inability to calculate MA60 for the first 60 days), the engineering implementation incorporates **data pre-fetching:**

If a user requests a backtest for the period 2023-01-01 to 2023-12-31, the

DataLoader automatically downloads data starting from 2022-01-01. This allows the system to "idle-run" for one year in the background to precompute indicators, ensuring that all moving averages and factors are fully initialized by the first day of the backtest (2023-01-01). This enables precise decision-making at **T=0.**

3.2 Integration of Vectorization and Loops

Indicator Calculation: Utilizes Pandas' vectorized operations to compute all technical indicators in bulk, ensuring speed.

Trade Simulation: Employs a **for loop** to iterate day by day. **Why not use vectorized backtesting?The strategy includes** "position-state dependencies": today's trading decisions depend on "whether a position was held yesterday," "the cost basis of the position," and "whether a stop-loss was triggered." This path-dependent logic necessitates an event-driven loop.

3.3 Buffer Zone Design

To prevent frequent trading around threshold boundaries (Signal Flickering), the code introduces a **10% rebalancing** buffer. Trades are only executed when the change in target position exceeds 10% of total assets. This simulates friction cost control in real-world trading.

The above constitutes the third major function of VisionQuant-Pro — **Strategy Backtesting Simulation** — which realizes the process from strategy construction to implementation. This section demonstrates the actual performance of the strategy and allows users to customize strategy parameters based on their preferences. Additionally, the first 10 trades are displayed for user reference.

Overall layout is as follows:

(This backtest is conducted using the 000001 Shanghai Composite Index as the benchmark, providing preliminary evidence of the strategy's effectiveness.)

Future Todo List

1. Model Enhancement

Vision Transformer: Replace CNN with Transformer architecture for superior global pattern recognition

Contrastive Learning: Apply the SimCLR framework to learn more robust feature representations

Multi-task Learning: Simultaneously predict returns, volatility, and pattern classification

2. Strategy Enhancement

Reinforcement Learning: Utilize RL to optimize position sizing and entry/exit timing

Multi-timeframe Analysis: Integrate patterns across different timeframes (daily, weekly, monthly)

Sentiment Integration: Incorporate news sentiment and social media data (implemented, but low correlation observed with limited reference value)

3. Risk Management

Dynamic Stop-loss: Adjust stop-loss levels based on volatility and market conditions

Sector Rotation: Identify sector-level patterns and perform rotation accordingly

Options Hedging: Use options to hedge downside risk while preserving upside exposure

Key Contributions

This project proposes an innovative investment strategy based on deep learning for visual pattern recognition of candlestick charts. It addresses a key limitation of traditional technical analysis—the inability to systematically identify and leverage visual patterns at scale.

Novel Signals: Utilizes unsupervised deep learning to extract visual features from candlestick charts—signals not covered in traditional quantitative finance curricula

Systematic Approach: Automates previously manual and subjective pattern recognition processes

Empirical Validation: Backtesting demonstrates consistent outperformance across multiple stocks and market conditions

Risk Management: Adaptive position sizing and portfolio optimization ensure robust risk-adjusted returns

Reference:

1. Lo, A. W., Mamaysky, H., & Wang, J. (2000).F oundations of technical analysis: Computational alg orithms, statistical inference, and empirical implem entation. *Journal of Finance*, 55(4), 1705-1765.

2. Fischer, T., & Krauss, C. (2018). Deep learning with long short-term memory networks for financi al market predictions. *European Journal of Operat ional Research*, 270(2), 654-669.

3. Sezer, O. B., & Ozbayoglu, A. M. (2018). Algor ithmic financial trading with deep convolutional neu ral networks: Time series to image conversion app roach. *Applied Soft Computing*, 70, 525-538.

4. Markowitz, H. (1952). Portfolio selection. *Journ al of Finance*, 7(1), 77-91.

5. Johnson, J., Douze, M., & Jégou, H. (2019). Bi llion-scale similarity search with GPUs. *IEEE Tran sactions on Big Data*, 7(3), 535-547.

6. Kingma, D. P., & Welling, M. (2013). Auto-enco ding variational bayes. *arXiv preprint arXiv:1312.6 114*.

7. Chen, T., Kornblith, S., Norouzi, M., & Hinton, G. (2020). A simple framework for contrastive lear ning of visual representations. *International Confer ence on Machine Learning*, 1597-1607.

8. Dosovitskiy, A., et al. (2020). An image is wort h 16x16 words: Transformers for image recognitio n at scale. *Advances in Neural Information Proce ssing Systems*, 33, 6848-6859.

Appendix: Technical Details

Overall Architecture Diagram

```
VisionQuant-Pro/
├── config/                    配置文件目录
├── data/                      数据目录
│   ├── raw/                   原始股票数据(CSV)
│   ├── images/                K 线图（PNG)
│   ├── indices/               索引文件
│   │   ├── cae_faiss.bin
│   │   ├── meta_data.csv
│   │   ├── vectors_mmap.npy
│   │   ├── vectors_reduced.npy
│   │   └── prediction_cache.csv
│   ├── models/                模型文件
│   │   └── cae_best.pth
│   └── processed/             处理后数据
├── logs/                      日志目录
├── notebooks/                 Jupyter 笔记本
├── src/                       源代码
│   ├── agent/                 AI 智能体
│   │   └── quant_agent.py
│   ├── data/                  数据模块
│   │   ├── data_loader.py
│   │   └── news_harvester.py
│   ├── models/                模型模块
│   │   ├── autoencoder.py
│   │   ├── train_cae.py  模型训练脚本
│   │   ├── vision_engine.py  视觉引擎
│   │   └── predict_engine.py
│   ├── strategies/            策略模块
│   │   ├── backtester.py      回测策略
│   │   ├── factor_mining.py 因子挖掘
│   │   └── fundamental.py       基本面
│   └── utils/                 工具模块
│       ├── audio_manager.py 语音识别
│       ├── pdf_generator.py  文档生成
│       └── visualizer.py      可视化工具
├── web/                       Web 应用
│   └── app.py             Streamlit 主应用
├── requirements.txt           依赖包列表
└── .env                       环境变量
```

Core Workflow

1. 数据采集
   ↓
2. K 线图生成与特征提取
   ↓
3. FAISS 索引构建
   ↓
4. 实时查询与相似形态搜索//多因子评分（视觉+财务+技术）
   ↓
6. AI 智能决策（Gemini 分析）
   ↓
7. 策略回测与优化
   ↓
8. Web 界面展示与交互