LEARNING SYMBOLICALLY COMPOSABLE REPRESENTATIONS FOR FORMULAIC ALPHA FACTOR MINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Formulaic alpha mining seeks explicit scoring expressions that rank stocks by future returns. Its effectiveness depends on both the search algorithm and the input variables from which formulas are constructed. We investigate whether learning suitable representations before symbolic search can improve alpha discovery. We propose a two-stage framework that combines a feature-axis encoder (FAE) with genetic programming (GP). The FAE uses feature-axis convolutions and coarse-grained return classification to learn low-dimensional representations.GP then performs fine-grained search for alpha formulas over the learned features, using RankIC to evaluate candidate formulas. Experiments on CSI500 and CSI1000 show that the proposed framework improves out-of-sample predictive performance compared with symbolic search over raw indicators. Controlled comparative experiments and ablation studies further support the contribution of the learned representations beyond dimensionality reduction and validate the proposed encoder design. The resulting formulas expose how learned features are combined, providing inspectable scoring rules at the representation level. These findings highlight the value of learning suitable search variables for formulaic alpha discovery.

1 INTRODUCTION

Alpha mining aims to discover signals in financial data that are informative about future asset returns. These signals, commonly called alpha factors, score assets for stock ranking, portfolio construction, and trading decisions (Zhang et al., 2025). Extracting useful signals from price movements, trading volume, and technical indicators is challenging because financial data are noisy, predictive relationships change over time, and relevant information may emerge only through combinations of variables. Automated alpha mining seeks predictive relationships beyond manually designed factors (Yu et al., 2023; Shi et al., 2025), with existing approaches broadly categorized as non-formulaic and formulaic methods.

Non-formulaic methods map financial observations to return estimates or ranking scores using statistical or machine learning models (Vishwakarma & Bhosale, 2025; Soni et al., 2022). Neural approaches can capture nonlinear relationships and learn representations for stock prediction (Li et al., 2024b; Chen & Wang, 2025). However, their computations are distributed across many parameters and hidden features, making the resulting signals difficult to express as concise rules, inspect for variable interactions, or reuse as standalone factors. Strong predictive performance therefore does not necessarily explain how a financial signal is constructed.

Formulaic methods represent alpha factors as mathematical expressions whose inputs and operations can be inspected, modified, and reused (Shi et al., 2025). Existing approaches include evolutionary search, reinforcement learning, and structure-aware expression generation (Koza, 1992; Yu et al., 2023; Shi et al., 2025; Chen et al., 2026). Genetic programming (GP) evolves expression trees through selection, crossover, and mutation, with applications to financial prediction (Ding et al., 2020; Ari & Alagoz, 2021; Li et al., 2024a). Its performance depends on the variables supplied as tree terminals, which form the building blocks of candidate formulas. Large raw-indicator sets may contain weakly informative or redundant variables and dependencies that are difficult to exploit through simple operators. As the terminal set and expression depth grow, the number of possible expressions increases rapidly, consuming the search budget with unproductive combinations. Im-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

proving search alone thus leaves an important question open: are the input variables suitable for discovering compact, predictive formulas?

To address this challenge, we propose a two-stage framework that learns a compact terminal set to make useful combinations more accessible to GP. Our *Feature-Axis Encoder (FAE)* transforms raw indicator vectors into low-dimensional representations using feature-axis pointwise convolutions and hierarchical pooling, trained through coarse-grained return classification. GP then combines the learned features into explicit alpha expressions. We use *symbolic composability* to describe a representation's empirical usefulness for constructing predictive formulas with low symbolic complexity. Classification accuracy is not our primary objective; the ultimate goal is to discover alpha factor expressions that deliver stable investment returns.

Experiments on CSI500 and CSI1000 demonstrate the effectiveness of the proposed framework. Compared with raw features, conventional dimensionality reduction, and alternative representations, our approach offers advantages when predictive performance, portfolio outcomes, and symbolic expression complexity are considered together. These results support the encoder's ability to learn representations suited to formulaic alpha mining and demonstrate the benefits of learning input features for downstream GP performance.

Our key contributions are summarized as follows:

1. **Representation learning for formula discovery.** We study GP terminal variables as learnable representations and assess their value through out-of-sample predictive performance and formula complexity under common search conditions.
2. **Feature-Axis Encoder.** We introduce FAE, which learns low-dimensional variables from stock-level indicator vectors through feature-axis convolutions, hierarchical pooling, and coarse-grained return classification.
3. **Latent-feature-driven alpha formula generator.** We integrate standard GP with the learned variables to perform fine-grained, RankIC-guided formula search, producing explicit rules for selecting and combining latent features within the two-stage framework.

2 Related Work

Non-formulaic alpha mining. Non-formulaic approaches map financial observations to return estimates or ranking scores without searching for symbolic expressions. Representative predictors include XGBoost (Chen & Guestrin, 2016) and D-Va, which uses a diffusion variational autoencoder for multi-step stock prediction (Koa et al., 2023). Recent methods include MASTER for market-guided feature selection and temporal and cross-stock modeling (Li et al., 2024b), ADB-TRM for mitigating temporal and relational biases (Chen et al., 2024), and DHMoE for diffusion-based hierarchical expert modeling (Chen & Wang, 2025). Although these methods capture complex financial relationships, their predictions are not directly expressed as compact symbolic rules, and their learned features may not suit symbolic composition. Our framework instead learns representations as GP terminals, connecting supervised feature learning with explicit formula discovery.

Formulaic alpha mining. Formulaic approaches construct explicit expressions over financial variables. GP evolves expression trees through selection, crossover, and mutation (Koza, 1992), with financial applications exploring hyperparameter optimization (Ari & Alagoz, 2023), multi-objective trading (Long et al., 2023), and richer alpha programs (Cui et al., 2021). Reinforcement-learning methods include AlphaGen and SFAG for synergistic factor discovery (Yu et al., 2023; Shin et al., 2024), and QFR for variance-bounded policy gradients and reward shaping (Zhao et al., 2025). RiskMiner uses risk-seeking tree search (Ren et al., 2024), while AlphaForge combines factor mining with dynamic weighting (Shi et al., 2025). AlphaSAGE guides GFlowNet exploration through formula structures (Chen et al., 2026), and AlphaAgentEvo learns iterative evolution through tool interaction (Tang et al., 2026). LLM-based mining includes Alpha-GPT (Wang et al., 2025), with surveys and AlphaBench organizing and evaluating this direction (Zhang et al., 2025; Luo et al., 2026). These methods emphasize formula construction and exploration; our work instead focuses on GP input variables. By learning representations of the search features in advance, we aim to help GP discover alpha formulas that generate sustained returns more efficiently.

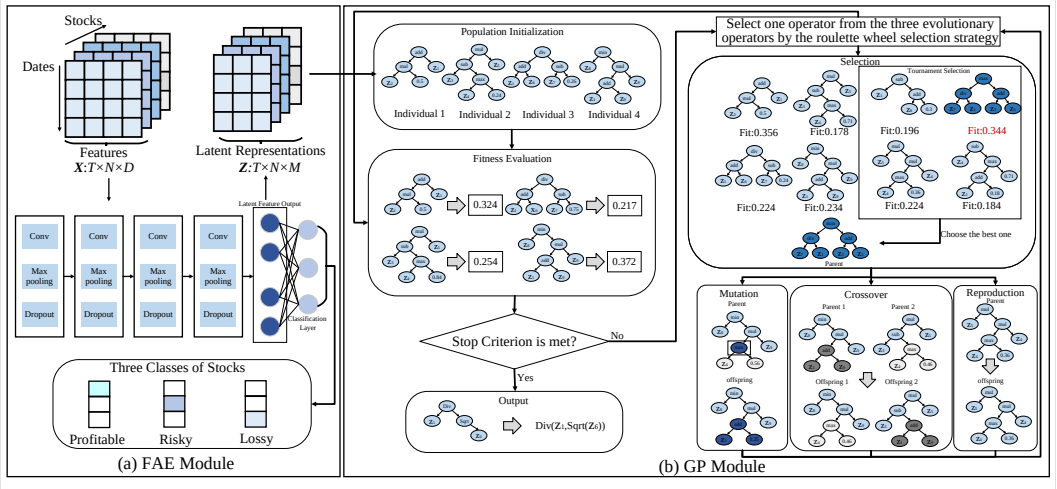

Figure 1: Overview of the proposed framework. (a) the FAE module and (b) the GP module.

3 METHODOLOGY

As illustrated in Figure 1, our framework consists of two modules: a Feature-Axis Encoder (FAE) that learns low-dimensional representations through coarse-grained return classification, and a GP module that constructs alpha formulas from the learned features. The FAE module learns from raw features to obtain compact, low-dimensional representations that aim to suppress noise; the GP module searches for combinations of these representations that predict cross-sectional return rankings.

3.1 FEATURE-AXIS ENCODER

GP-based alpha mining typically constructs formulas from a predefined set of financial indicators. The quality of these variables affects which predictive relationships can be expressed and discovered. A large input set introduces many possible combinations, including those involving redundant or noisy indicators. FAE addresses this difficulty by transforming the original indicators into a smaller set of return-related features before formula search.

Let $X_t \in \mathbb{R}^{N_t \times D}$ contain the indicators of the N_t eligible stocks at decision time t. Applying FAE to each stock's indicator vector produces

$$Z_t = E_\theta(X_t) \in \mathbb{R}^{N_t \times M}, \qquad M < D, \tag{1}$$

where θ denotes the encoder parameters and each row $z_{t,i}$ contains the M learned features of stock i. The encoder applies nonlinear transformations and feature aggregation along the indicator axis. This differs from a temporal convolutional encoder, which aggregates observations across consecutive dates to extract sequential patterns. Temporal prediction features need not be organized for subsequent symbolic combination. Here, the input indicators already summarize historical market behavior, and FAE focuses on transforming the information available within each stock's indicator profile. Its output provides a compact set of numerical variables for GP. Detailed architectural specifications are given in the appendix.

To make the representation relevant to alpha discovery, we train FAE using a coarse-grained classification task. Direct return regression requires fitting numerical return magnitudes and, under squared-error supervision, assigns greater weight to large residuals. In contrast, relative return groups emphasize broad predictive distinctions without requiring every fluctuation in return magnitude to be reproduced. This classification supervision is designed to substantially suppress noise associated with task-irrelevant return fluctuations while retaining information useful for distinguishing stocks with different return prospects.

Specifically, within each training date, stocks are sorted by their subsequent returns and divided into three approximately equal groups, corresponding to Profitable, Risky and Lossy relative re-

turns. This construction produces approximately balanced classes within each date and defines the labels relative to the contemporaneous stock universe. Let $c_{t,i} \in \{1, 2, 3\}$ be the resulting label. A classification head H_ϕ maps the latent features to class probabilities,

$$p_{t,i} = \mathrm{softmax}(H_\phi(z_{t,i})), \qquad \mathcal{L}_{\mathrm{cls}} = -\frac{1}{|\mathcal{D}_{\mathrm{enc}}|} \sum_{(t,i)\in\mathcal{D}_{\mathrm{enc}}} \log p_{t,i,c_{t,i}}, \tag{2}$$

where $\mathcal{D}_{\mathrm{enc}}$ denotes the representation-training observations. Minimizing this loss encourages the encoder to preserve distinctions associated with subsequent return groups within the restricted latent dimension.

The representation used for formula mining is taken before the classification head. Thus, GP receives the continuous latent features rather than the three predicted classes or their probabilities. Coarse labels guide the learning of these features, while continuous coordinates retain the capacity to express differences among stocks assigned to the same group. Compared with directly searching over all raw indicators, the resulting terminal set has fewer variables and is shaped by return-related supervision. GP can then combine these variables to discover finer-grained ranking relationships.

3.2 GP-Based Alpha Formula Generation

Given the learned features Z_t, we use GP to search for an explicit alpha expression (Koza, 1992). GP maintains a population of candidate formulas represented as expression trees. The leaves are latent variables $z^{(1)}, \ldots, z^{(M)}$ or numerical constants, and the internal nodes are mathematical operators. Each tree therefore specifies a computable relationship among the learned features. The operator vocabulary determines the permitted forms of combination, while FAE supplies the variables from which these combinations are built.

A candidate formula g assigns stock i the score $g(z_{t,i})$. We evaluate its predictive quality using RankIC, so that evolutionary search is directed toward stock ordering rather than exact return magnitudes. Over the fitness-evaluation dates $\mathcal{T}_{\mathrm{fit}}$, candidate fitness is

$$\begin{aligned} \mathrm{RankIC}(g) &= \frac{1}{|\mathcal{T}_{\mathrm{fit}}|} \sum_{t\in\mathcal{T}_{\mathrm{fit}}} \mathrm{Corr}_{\mathrm{Spearman}}\left(\{g(z_{t,i})\}_{i=1}^{N_t}, \{r_{t,i}\}_{i=1}^{N_t}\right), \\ \mathcal{F}(g) &= \mathrm{RankIC}(g) - \lambda C(g), \end{aligned} \tag{3}$$

where $r_{t,i}$ is the subsequent return and $C(g)$ counts all leaf and operator nodes in the tree. The coefficient λ controls the complexity penalty. This objective favors expressions that provide useful ranking information without unnecessary growth in formula size. It complements the encoder's three-class objective by evaluating finer distinctions among stocks using their continuous returns.

The search begins with a population of randomly generated valid expressions. After fitness evaluation, parent expressions are selected to produce offspring through crossover, mutation, and reproduction. Crossover exchanges subexpressions between parents, mutation modifies an existing expression, and reproduction retains selected candidates. The offspring are evaluated and the population is updated, repeating this process until the stopping criterion is met. The search retains the best expression encountered according to the fitness objective. The operator set and detailed search settings are provided in the appendix.

Denoting the selected expression by $g^\star$, the resulting alpha score is

$$\alpha_{t,i} = g^\star(z_{t,i}) = g^\star(E_\theta(x_{t,i})). \tag{4}$$

The score is continuous and supports cross-sectional stock ranking. The framework therefore connects coarse-grained supervised feature learning with fine-grained symbolic alpha discovery: FAE constructs the representation, and GP exposes how its coordinates are combined. The displayed formula describes this combination over learned features, with the encoder forming the preceding part of the predictor.

4 Experiments

We evaluate the proposed framework from four perspectives: performance against existing alpha-mining methods, the usefulness of different input representations, the complexity of discovered formulas, and the contributions of the two modules.

4.1 EXPERIMENTAL SETUP

Datasets. The experiments use daily China A-share data and evaluate performance on the CSI500 and CSI1000 stock universes. Each stock-date observation contains 52 indicators, including opening and closing prices, volume-weighted average price, trading volume, volatility, rates of change, and derived statistics. The documented training interval is January 1, 2016–December 31, 2020, and the test interval is January 1, 2021–December 31, 2023. Models are trained on the broader stock universe and evaluated on the two target universes. Input indicators use cross-sectional z-score normalization. The main FAE configuration produces $M = 10$ latent features. These reproducibility limitations are discussed in the appendix.

Backtesting Settings. At each evaluation date, stocks are ranked by alpha scores. The factor direction is predetermined using only information available before the evaluation period and is held fixed throughout the evaluation. The long-only portfolio equally weights stocks in the top decile, and its daily excess return is measured against the equal-weight return of the evaluated stock pool. The long–short portfolio assigns total weights of 0.5 and -0.5 to the top and bottom deciles, respectively, with equal weights within each decile. Portfolio weights are updated at each evaluation date. Returns are compounded and annualized using 252 trading days.

Evaluation Metrics. We assess ranking quality using RankIC. For test dates $\mathcal{T}_{\text{test}}$, alpha scores $\alpha_{t,i}$, subsequent returns $r_{t,i}$, and eligible stocks $\mathcal{U}_t$,

$$\text{RankIC} = \frac{1}{|\mathcal{T}_{\text{test}}|} \sum_{t \in \mathcal{T}_{\text{test}}} \text{Corr}_{\text{Spearman}} \left(\{\alpha_{t,i}\}_{i \in \mathcal{U}_t}, \{r_{t,i}\}_{i \in \mathcal{U}_t} \right). \tag{5}$$

Portfolio performance is measured by annualized long excess return (ALER). With daily long-portfolio and benchmark returns $R_{p,t}$ and $R_{b,t}$, respectively, the relative excess growth and its annualization are

$$1 + e_t = \frac{1 + R_{p,t}}{1 + R_{b,t}}, \qquad \text{ALER} = \left[\prod_{t=1}^{T} (1 + e_t) \right]^{252/T} - 1. \tag{6}$$

Here T is the number of evaluated trading days. ALER measures annualized relative wealth growth, rather than the difference between two annualized returns. Higher RankIC and ALER indicate better reported performance. For formula complexity, we count all operator and terminal occurrences in the original GP tree.

Comparison Methods. The external baselines are AlphaForge (Shi et al., 2025), QFR (Zhao et al., 2025), RiskMiner (Ren et al., 2024), SFAG (Shin et al., 2024), and D-Va (Koa et al., 2023). The first four construct or combine formulaic factors, whereas D-Va is a neural forecasting method. Representation comparisons supply GP with raw indicators, PCA components, temporal-CNN features, or FAE features. Raw GP uses 52 inputs; the other representations use ten coordinates. Component comparisons additionally include a direct encoder predictor, FAE followed by XGBoost, and raw-input XGBoost (Chen & Guestrin, 2016).

Implementation Details. The FAE-GP method is implemented by PyTorch. In FAE module, we adopt the Adam optimizer with batch size 512 and set the number of the output latent features (namely *M*) as 10. The normalization method utilizes the cs-zscore to normalize the stock data. In the GP module, the population size (namely *NP*) is set as 3000, the tournament size (namely *K*) is set as 50, and the coefficient λ is set as 0.001. As for the parameters of the comparison methods, they are all set as the recommended ones in the associated papers. At last, all algorithms are executed on a server with NVIDIA RTX 3090 GPU for independent 10 runs, so as to get fair comparisons.

4.2 EXPERIMENTAL RESULTS AND ANALYSIS

Predictive and portfolio performance. Table 1 compares FAE + GP with five existing methods on CSI500 and CSI1000. FAE + GP achieves the highest mean RankIC and ALER on both datasets, indicating improvements in both cross-sectional ranking and portfolio performance. Figure 2 complements these aggregate results with long excess and long–short NAV trajectories. FAE + GP

Table 1: External comparison on CSI500 and CSI1000. Entries are mean ± standard deviation across ten independent runs. Higher is better for both metrics.

	CSI500		CSI1000	
Method	RankIC	ALER	RankIC	ALER
FAE + GP	**0.0530** ± 0.0030	**0.1648** ± 0.0056	**0.0646** ± 0.0017	**0.1578** ± 0.0030
AlphaForge	0.0433 ± 0.0033	0.0892 ± 0.0029	0.0523 ± 0.0029	0.0632 ± 0.0024
QFR	0.0380 ± 0.0067	0.0380 ± 0.0056	0.0443 ± 0.0027	0.0618 ± 0.0030
RiskMiner	0.0214 ± 0.0044	0.0712 ± 0.0031	0.0337 ± 0.0033	0.0530 ± 0.0030
SFAG	0.0188 ± 0.0042	0.0110 ± 0.0006	0.0346 ± 0.0015	0.0826 ± 0.0030
D-Va	0.0341 ± 0.0032	0.0622 ± 0.0031	0.0435 ± 0.0033	0.0646 ± 0.0027

Table 2: GP performance with different input representations.

		CSI500		CSI1000	
Representation	Dim.	RankIC	ALER	RankIC	ALER
Raw indicators	52	0.0428	0.1081	0.0512	0.0772
PCA	10	0.0528	0.1276	0.0643	0.1282
Temporal CNN	10	0.0410	0.0381	0.0558	0.0648
FAE	10	**0.0530**	**0.1648**	**0.0646**	**0.1578**

reaches the highest ending NAV in all four panels over the displayed periods, although its relative advantage varies over time.

Returns across score groups. Figures 3 and 4 present annualized returns for ten portfolios formed by ranking stocks from highest to lowest alpha scores on CSI500 and CSI1000, respectively. For FAE + GP, returns generally decrease across the groups, with high-score portfolios outperforming low-score portfolios on both universes. Although adjacent groups do not always follow a strictly monotonic ordering, the overall pattern indicates that the discovered alpha expressions distinguish stocks with different return prospects. These results complement the aggregate RankIC and ALER comparisons by showing how the predicted rankings translate into portfolio-level return differences.

Effect of the input representation. Table 2 evaluates the features supplied to GP. FAE improves both RankIC and ALER over raw indicators and temporal-CNN features on both datasets, supporting the usefulness of learning the input variables before symbolic search. PCA provides a strong ranking baseline, with RankIC close to that of FAE. However, FAE achieves higher ALER, with relative gains of 29.2% on CSI500 and 23.1% on CSI1000 over PCA. This difference highlights the importance of evaluating learned representations through portfolio performance as well as ranking correlation. The aggregate metrics alone do not identify the portfolio effects responsible for this difference. These comparisons evaluate alternative representations; they do not isolate the effects of the convolution axis or classification supervision.

Complexity of the discovered expressions. Table 4 presents example formulas obtained from the different representations. The FAE-based expression contains four GP-tree nodes, compared

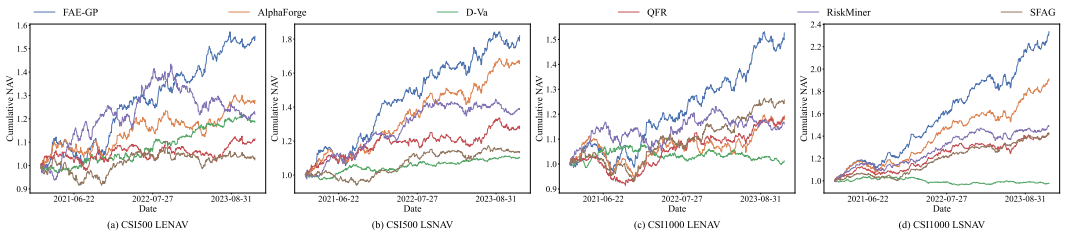

Figure 2: Cumulative NAV (LENAV and LSNAV) curves of the FAE-GP method and the five state-of-the-art comparison methods on the two datasets.

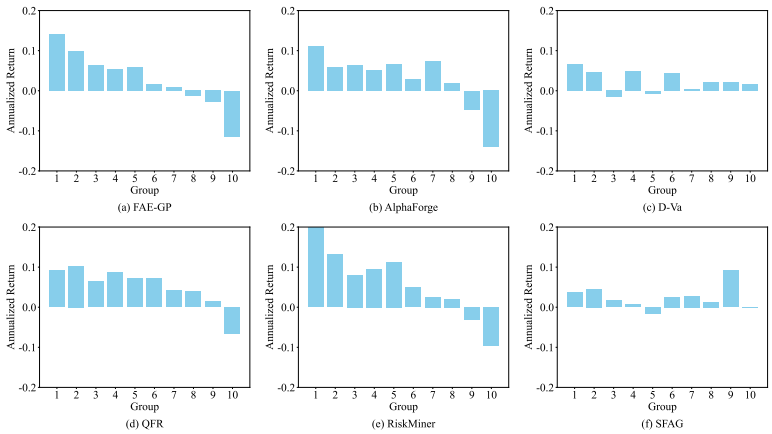

Figure 3: Annualized returns by group predicted by the optimal alpha factors found by the FAE-GP method and the five comparison methods on CSI500.

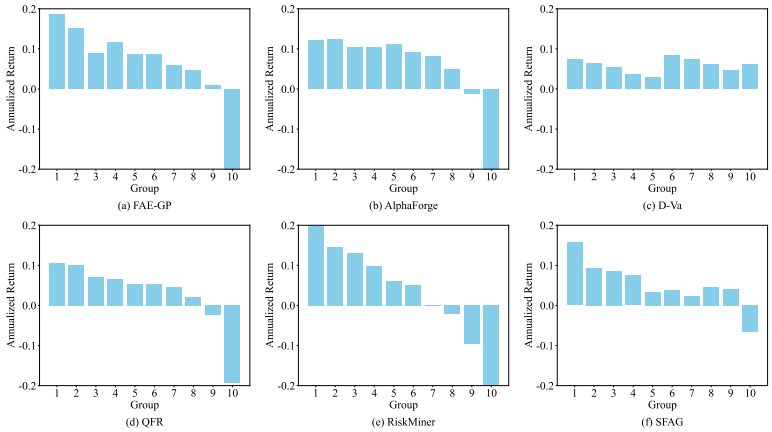

Figure 4: Annualized returns by group predicted by the optimal alpha factors found by the FAE-GP method and the five comparison methods on CSI1000.

with seven for raw indicators, six for PCA, and ten for temporal-CNN features. It combines two learned coordinates through square root and division, illustrating that the learned representation can support a compact symbolic expression. Each displayed formula is shared across the two evaluation universes, so a single node count is reported per method. Counts refer to the original trees before algebraic simplification; these examples do not establish lower average complexity across runs.

Computational cost. FAE + GP achieves a favorable balance between efficiency and investment performance. Its total runtime on CSI500 is 89.7 minutes, substantially lower than FAE + XG-

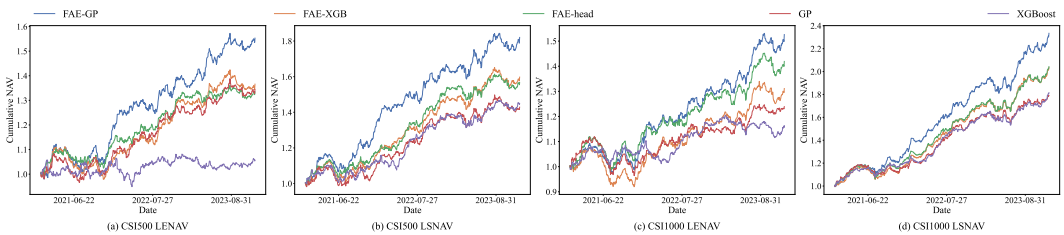

Figure 5: Cumulative NAV (LENAV and LSNAV) curves of instantiated FAE-GP method and their variants on the two datasets.

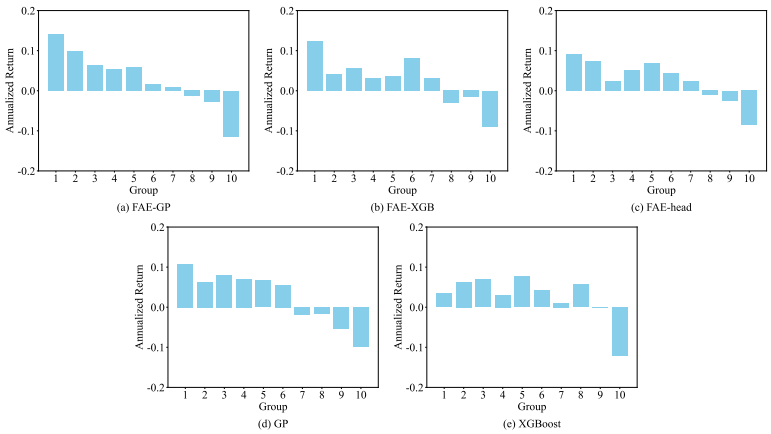

Figure 6: Annualized returns by group predicted by the optimal alpha factors found by the FAE-GP method and their variants on CSI500.

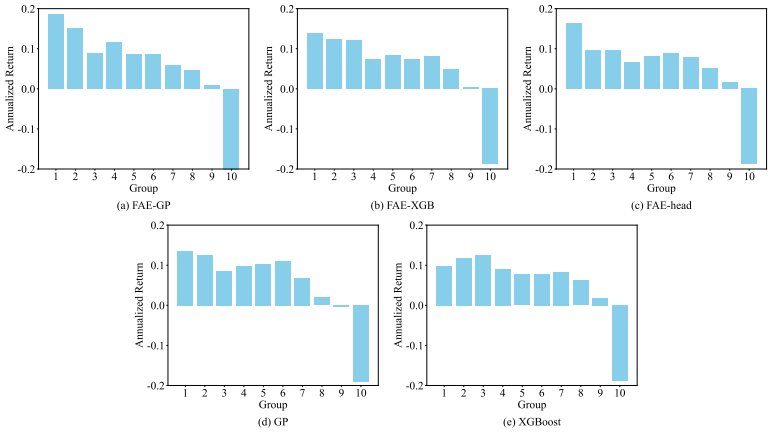

Figure 7: Annualized returns by group predicted by the optimal alpha factors found by the FAE-GP method and their variants on CSI1000.

Boost, the standalone encoder predictor, raw GP, AlphaForge, and D-Va. Although QFR, RiskMiner, SFAG, and raw-input XGBoost are faster, their predictive and portfolio performance is clearly weaker. FAE + GP achieves higher RankIC and ALER and more consistent grouped-return patterns, indicating that its additional computational cost relative to the fastest baselines is accompanied by stronger and more stable factor performance. The full runtime breakdown is provided in the appendix.

4.3 Ablation Studies

We examine the contributions of representation learning and symbolic prediction through component comparisons, complemented by latent-dimension sensitivity. The alternative input representations are compared in Section 4.2.

Contribution of learned features. Replacing FAE features with raw indicators gives the Raw GP baseline in Table 3. FAE + GP increases mean RankIC from 0.0428 to 0.0530 on CSI500 and from 0.0512 to 0.0646 on CSI1000, with corresponding improvements in ALER. Learned features also benefit XGBoost: FAE + XGBoost exceeds raw-input XGBoost on both metrics. The representation is therefore useful beyond the GP predictor in these comparisons.

Table 3: Component comparisons on CSI500 and CSI1000. Entries are mean ± standard deviation across ten independent runs. Higher is better for both metrics.

	CSI500		CSI1000	
Method	RankIC	ALER	RankIC	ALER
FAE + GP	$\mathbf{0.0530} \pm 0.0030$	$\mathbf{0.1648} \pm 0.0056$	$\mathbf{0.0646} \pm 0.0017$	$\mathbf{0.1578} \pm 0.0030$
FAE + XGBoost	0.0456 ± 0.0024	0.1141 ± 0.0029	0.0592 ± 0.0029	0.0985 ± 0.0028
Encoder + head	0.0444 ± 0.0062	0.1053 ± 0.0030	0.0584 ± 0.0015	0.1290 ± 0.0005
Raw GP	0.0428 ± 0.0027	0.1081 ± 0.0021	0.0512 ± 0.0032	0.0772 ± 0.0028
XGBoost	0.0328 ± 0.0032	0.0193 ± 0.0025	0.0491 ± 0.0029	0.0540 ± 0.0024

Table 4: Example alpha expressions obtained using different representations. Nodes are counted in the original GP trees before algebraic simplification.

Method	Expression	Nodes
Raw GP	$\max\{x_7, x_{46} + x_7\} + x_{19}$	7
PCA + GP	$\max\{\sin(z_3), z_6\} - z_0$	6
Temporal CNN + GP	$-z_0 - \sqrt{z_6} - \sqrt{z_5 z_1}$	10
FAE + GP	$z_1/\sqrt{z_6}$	4

Contribution of symbolic prediction. The Encoder + head variant uses a direct neural predictor, while FAE + XGBoost replaces GP with a tree ensemble. FAE + GP obtains higher mean RankIC and ALER than both alternatives on both universes. These comparisons support the use of symbolic composition after representation learning. They compare complete predictive pipelines and do not establish a causal effect independent of training and model-selection choices. The reported unadjusted rank-sum tests for these component comparisons are provided in the appendix.

Figure 5 shows the corresponding portfolio trajectories for the component variants. These curves complement the aggregate comparison by displaying how the performance differences accumulate over time.

Grouped-return analysis of the components. Figures 6 and 7 compare the annualized returns of score-ranked portfolios across the component variants. FAE + GP exhibits a more consistent decline from high-score to low-score groups than raw GP and XGBoost, whose returns fluctuate more across intermediate groups. The direct encoder head and FAE + XGBoost also separate high- and low-score stocks, while FAE + GP retains a clear return gradient on both universes. Together with Table 3, these patterns support the usefulness of learning the input representation and subsequently combining its coordinates through GP.

Latent-dimension sensitivity. The supplementary experiments in Figures S4 and S5 vary M from 8 to 30 in steps of two. RankIC changes within a comparatively narrow range, whereas ALER varies more noticeably, with different maximizing dimensions across the two stock universes. Considering RankIC and ALER jointly across CSI500 and CSI1000, together with representation compactness, we select $M = 10$ as the default configuration because it offers a favorable overall trade-off.

5 Conclusion

Whether the features supplied to GP are learned beforehand can influence its evolutionary search and the quality of the discovered formulas. We propose a two-stage framework in which a Feature-Axis Encoder learns compact representations of raw financial indicators, providing GP with a reduced set of variables designed to support more efficient formula search. Experiments show that the learned features improve the predictive and portfolio performance of GP relative to raw inputs, while the displayed formulas illustrate their potential to support compact symbolic expressions. Our work contributes to representation learning for GP-based alpha mining by demonstrating the value of learning the variables over which evolutionary search operates.

AI USE STATEMENT

In this work, we did not use generative AI tools for any tasks. We have not used generative AI tools for any tasks with required disclosure, and all other required disclosure categories are not applicable to this work. Additionally, we did not use generative AI tools for any tasks with recommended disclosure. Because no generative AI tools were used, there is no AI-assisted work to review or elaborate on. All research ideas, literature surveys, methodology, code, experiments, data analysis, results, figures, and manuscript writing and revisions were produced and verified by the authors without generative AI assistance. We take responsibility for the final content of this work, including all text, claims, and artifacts.

See the ICLR 2027 AI Policy for Authors for more details.

ETHICS STATEMENT

The authors have considered the ethical implications of this work. This study does not involve human subjects, animal experimentation, or the collection, processing, or release of personally identifiable or sensitive data. Any datasets used are publicly available or were obtained and used in accordance with their respective licenses and terms of use, and no IRB or ethics approval was required. The authors declare no competing interests or conflicts of interest; funding sources, if any, are disclosed in the acknowledgments. The proposed methods and applications are intended for legitimate research purposes, and we do not foresee direct harmful, discriminatory, or privacy-invasive applications. We have followed research integrity practices, including accurate documentation, transparent methodology, and proper citation of prior work. Potential limitations and biases are discussed in the main text. We take responsibility for the ethical conduct and final content of this work.

REPRODUCIBILITY STATEMENT

We have taken several steps to ensure that the results in this paper are reproducible. The full specification of our method, including the model architecture, training objective, and algorithm, is provided in Section 3 and Appendix. For theoretical results, all assumptions and complete proofs are included in Appendix. For the experiments, we describe the datasets, preprocessing steps, data splits, evaluation metrics, and hyperparameter configurations in Section 4 and Appendix. Computational resources and runtime are documented in Appendix. We believe these materials provide sufficient information to reproduce the main results.

REFERENCES

Davut Ari and Baris Baykant Alagoz. Modeling daily financial market data by using tree-based genetic programming. In *ICIT*, pp. 382–386, 2021.

Davut Ari and Baris Baykant Alagoz. Dehypgpols: a genetic programming with evolutionary hyperparameter optimization and its application for stock market trend prediction: D. ari, bb alagoz. *Soft Comput.*, 27(5):2553–2574, 2023.

Binqi Chen, Hongjun Ding, Ning Shen, Taian Guo, Jinsheng Huang, Luchen Liu, and Ming Zhang. AlphaSAGE: Structure-aware alpha mining via GFlowNets for robust exploration. In *ICLR*, 2026.

Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *KDD*, pp. 785–794, 2016.

Weijun Chen and Yanze Wang. DHMoE: Diffusion generated hierarchical multi-granular expertise for stock prediction. In *AAAI*, volume 39, pp. 11490–11499, 2025.

Weijun Chen, Shun Li, Xipu Yu, Heyuan Wang, Wei Chen, and Tengjiao Wang. Automatic de-biased temporal-relational modeling for stock investment recommendation. In *IJCAI*, pp. 1999–2008, 2024.

Can Cui, Wei Wang, Meihui Zhang, Gang Chen, Zhaojing Luo, and Beng Chin Ooi. Alphaevolve: A learning framework to discover novel alphas in quantitative investment. In *SIGMOD*, pp. 2208–2216, 2021.

Shusheng Ding, Tianxiang Cui, Xihan Xiong, and Ruibin Bai. Forecasting stock market return with nonlinearity: a genetic programming approach. *J. Ambient Intell. Humaniz. Comput.*, 2020.

Kelvin J. L. Koa, Yunshan Ma, Ritchie Ng, and Tat-Seng Chua. Diffusion variational autoencoder for tackling stochasticity in multi-step regression stock price prediction. In *CIKM*, pp. 1087–1096, 2023.

John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

Qi Li, Norshaliza Kamaruddin, Siti Sophiayati Yuhaniz, and Hamdan Amer Ali Al-Jaifi. Forecasting stock prices changes using long-short term memory neural network with symbolic genetic programming. *Sci. Rep.*, 14(1):422, 2024a.

Tong Li, Zhaoyang Liu, Yanyan Shen, Xue Wang, Haokun Chen, and Sen Huang. MASTER: Market-guided stock transformer for stock price forecasting. In *AAAI*, volume 38, pp. 162–170, 2024b.

Xinpeng Long, Michael Kampouridis, and Panagiotis Kanellopoulos. Multi-objective optimisation and genetic programming for trading by combining directional changes and technical indicators. In *CEC*, pp. 1–8, 2023.

Haochen Luo, HoTin Ko, Jiandong Chen, David Sun, Yuan Zhang, and Chen Liu. AlphaBench: Benchmarking large language models in formulaic alpha factor mining. In *ICLR*, 2026.

Tao Ren, Ruihan Zhou, Jinyang Jiang, Jiafeng Liang, Qinghao Wang, and Yijie Peng. RiskMiner: Discovering formulaic alphas via risk seeking monte carlo tree search. In *ICAIF*, pp. 752–760, 2024.

Hao Shi, Weili Song, Xinting Zhang, Jiahe Shi, Cuicui Luo, Xiang Ao, Hamid Arian, and Luis Angel Seco. AlphaForge: A framework to mine and dynamically combine formulaic alpha factors. In *AAAI*, volume 39, pp. 12524–12532, 2025.

Hong-Gi Shin, Sukhyun Jeong, Eui-Yeon Kim, Sungho Hong, Young-Jin Cho, and Yong-Hoon Choi. Synergistic formulaic alpha generation for quantitative trading based on reinforcement learning. In *ICOIN*, pp. 42–46, 2024.

Payal Soni, Yogya Tewari, and Deepa Krishnan. Machine learning approaches in stock price prediction: a systematic review. In *J. Phys.: Conf. Ser.*, volume 2161, pp. 012065, 2022.

Ziyi Tang, Xuexiong Yin, Weixing Chen, Zechuan Chen, Yongsen Zheng, Wenxuan Ye, Keze Wang, and Liang Lin. AlphaAgentEvo: Evolution-oriented alpha mining via self-evolving agentic reinforcement learning. In *ICLR*, 2026.

Vijay Kumar Vishwakarma and Narayan P Bhosale. A survey of recent machine learning techniques for stock prediction methodologies. *Neural Comput. Appl.*, 37(4):1951–1972, 2025.

Saizhuo Wang, Hang Yuan, Leon Zhou, Lionel Ni, Heung Yeung Shum, and Jian Guo. Alpha-gpt: Human-ai interactive alpha mining for quantitative investment. In *EMNLP: System Demonstrations*, pp. 196–206, 2025.

Shuo Yu, Hongyan Xue, Xiang Ao, Feiyang Pan, Jia He, Dandan Tu, and Qing He. Generating synergistic formulaic alpha collections via reinforcement learning. In *KDD*, pp. 5476–5486, 2023.

Junjie Zhang, Shuoling Liu, Tongzhe Zhang, and Yuchen Shi. A survey on large language model-based alpha mining. *Front. Inf. Technol. Electron. Eng.*, 26(10):1809–1821, 2025.

Junjie Zhao, Chengxi Zhang, Min Qin, and Peng Yang. QuantFactor REINFORCE: Mining steady formulaic alpha factors with variance-bounded REINFORCE. *IEEE Trans. Signal Process.*, 73: 2448–2463, 2025.

A Feature-Axis Encoder Architecture

Let $X \in \mathbb{R}^{T\times N\times D}$ denote the input financial observations, where T, N, and D are the numbers of trading days, stocks, and raw indicators, respectively. The Feature-Axis Encoder (FAE) processes the indicator vector of each stock on each trading day and maps it to a low-dimensional representation, yielding $Z \in \mathbb{R}^{T\times N\times m}$. The main configuration uses $m = 10$.

As shown in Figure S1, FAE contains four pointwise Conv1D blocks with channel transitions $1 \to 64$, $64 \to 128$, $128 \to 128$, and $128 \to 256$. Each block applies convolution, batch normalization, ReLU activation, max pooling, and dropout. Global average pooling then produces a 256-dimensional vector, and a further pointwise projection maps this vector to m latent coordinates. A classification head trains the encoder to distinguish three coarse-grained return groups. After training, the head is discarded, the encoder is frozen, and the latent coordinates become terminal variables for GP.

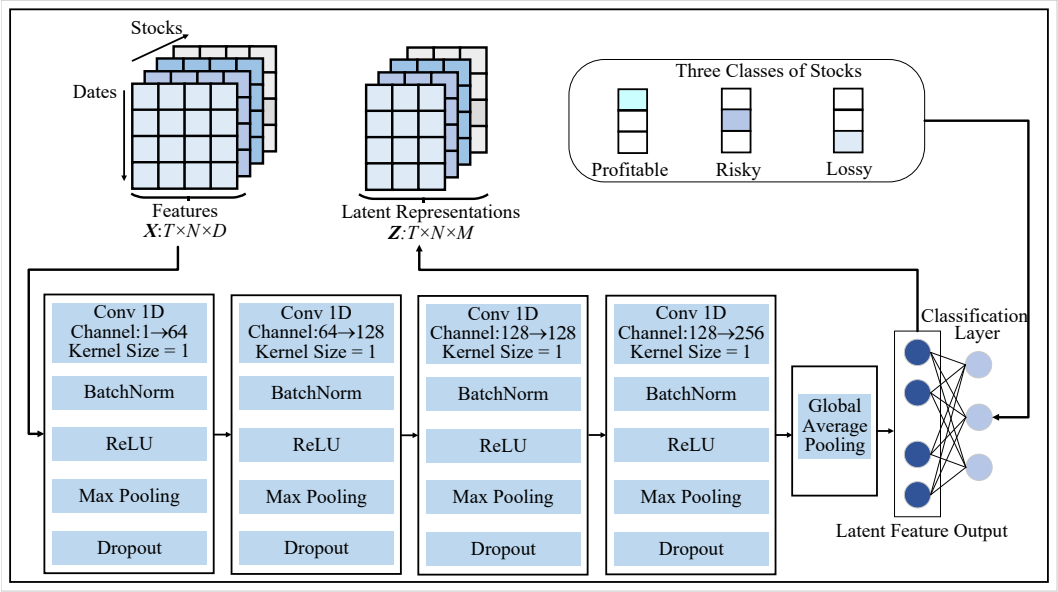

Figure S1: FAE architecture. Four pointwise convolution blocks and pooling produce a compact representation, which is projected to the latent coordinates used by GP. The classification head is used only during representation learning.

Unlike a temporal CNN, FAE processes indicator positions within a stock-date observation rather than sliding kernels over consecutive dates. Historical information can enter through the indicator definitions. Pointwise convolutions transform channels at each position, while pooling aggregates responses across positions. Because local pooling depends on the ordering of indicators, this architecture is not permutation invariant. The resulting GP expression exposes composition over learned variables; the neural encoder remains part of the complete predictor.

B Supplementary Statistical Analysis

Tables S1 and S2 report the means, standard deviations, medians, and pairwise test results from ten independent runs. The external comparison includes AlphaForge, D-Va, QFR, RiskMiner, and SFAG. The component comparison includes FAE + XGBoost, the direct encoder head, raw-input GP, and raw-input XGBoost. These names correspond to FAE-GP, FAE-XGBoost, FAE, GP, and XGBoost in the original experimental record, with FAE-GP denoting the full FAE + GP framework.

The reported two-sided Wilcoxon rank-sum p-values are below 0.05 for every listed comparison, with higher mean and median performance for FAE + GP on both datasets and metrics. These are unadjusted pairwise results; they should not be interpreted as family-wise error-controlled findings. The reported average ranks also place FAE + GP first in each comparison group. However, average

ranks alone do not establish statistically significant pairwise differences, and no such inference is made from the rank values. These tests do not cover the additional PCA or temporal-CNN representation controls.

Table S1: External comparison across ten independent runs.

Dataset / Metric	Statistic	FAE + GP	AlphaForge	D-Va	QFR	RiskMiner	SFAG
CSI500 RankIC	Mean	0.053020	0.043300	0.034100	0.038000	0.021400	0.018800
	Std	0.002953	0.003305	0.003155	0.006678	0.004364	0.004195
	Median	0.052650	0.043600	0.034500	0.037800	0.021350	0.017050
	p-value	—	2.46×10^{-4}	1.83×10^{-4}	2.46×10^{-4}	1.83×10^{-4}	1.83×10^{-4}
CSI500 ALER	Mean	0.164800	0.089200	0.062200	0.038010	0.071200	0.011000
	Std	0.005587	0.002858	0.003055	0.005625	0.003078	0.000606
	Median	0.167200	0.089250	0.062300	0.038500	0.071050	0.011100
	p-value	—	1.82×10^{-4}	1.82×10^{-4}	1.82×10^{-4}	1.82×10^{-4}	1.80×10^{-4}
CSI1000 RankIC	Mean	0.064600	0.052300	0.043500	0.044300	0.033700	0.034600
	Std	0.001678	0.002877	0.003337	0.002677	0.003349	0.001478
	Median	0.063950	0.052350	0.043500	0.044200	0.033500	0.034950
	p-value	—	1.83×10^{-4}	1.83×10^{-4}	1.83×10^{-4}	1.83×10^{-4}	1.83×10^{-4}
CSI1000 ALER	Mean	0.157810	0.063200	0.064620	0.061800	0.053000	0.082600
	Std	0.002954	0.002378	0.002726	0.003036	0.002997	0.002980
	Median	0.157450	0.063600	0.065050	0.061850	0.053050	0.083150
	p-value	—	1.82×10^{-4}	1.82×10^{-4}	1.82×10^{-4}	1.82×10^{-4}	1.82×10^{-4}
Average rank		1.00	2.50	3.75	4.00	5.00	4.75

Mean, standard deviation (Std), and median summarize ten runs. Reported p-values compare FAE + GP with each alternative using a two-sided Wilcoxon rank-sum test; no multiplicity adjustment is reported. Average ranks summarize the reported Friedman ranking and are descriptive; smaller values are better.

Table S2: Component comparison across ten independent runs. FAE + XGB denotes FAE + XGBoost; Encoder + head denotes the direct neural predictor.

Dataset / Metric	Statistic	FAE + GP	FAE + XGB	Encoder + head	Raw GP	XGBoost
CSI500 RankIC	Mean	0.053020	0.045630	0.044400	0.042800	0.032810
	Std	0.002953	0.002443	0.006196	0.002680	0.003199
	Median	0.052650	0.045650	0.042200	0.043200	0.032850
	p-value	—	4.40×10^{-4}	9.11×10^{-3}	1.83×10^{-4}	1.83×10^{-4}
CSI500 ALER	Mean	0.164800	0.114100	0.105300	0.108100	0.019330
	Std	0.005587	0.002937	0.002985	0.002072	0.002536
	Median	0.167200	0.114150	0.105100	0.108050	0.019150
	p-value	—	1.82×10^{-4}	1.82×10^{-4}	1.82×10^{-4}	1.82×10^{-4}
CSI1000 RankIC	Mean	0.064600	0.059200	0.058400	0.051200	0.049100
	Std	0.001678	0.002851	0.001534	0.003222	0.002928
	Median	0.063950	0.059200	0.058700	0.051750	0.049750
	p-value	—	1.83×10^{-4}	1.83×10^{-4}	1.83×10^{-4}	1.83×10^{-4}
CSI1000 ALER	Mean	0.157810	0.098480	0.129000	0.077200	0.054040
	Std	0.002954	0.002795	0.000459	0.002848	0.002422
	Median	0.157450	0.098300	0.129000	0.077350	0.053450
	p-value	—	1.82×10^{-4}	1.80×10^{-4}	1.82×10^{-4}	1.82×10^{-4}
Average rank		1.00	2.25	3.00	3.75	5.00

Mean, standard deviation (Std), and median summarize ten runs. Reported p-values compare FAE + GP with each alternative using a two-sided Wilcoxon rank-sum test; no multiplicity adjustment is reported. Average ranks summarize the reported Friedman ranking and are descriptive; smaller values are better.

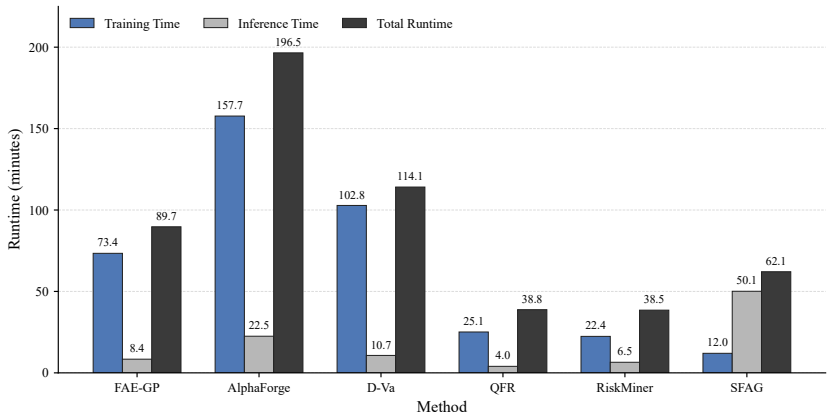

Figure S2: Execution time including the training time, the inference time and the backtesting time of FAE-GP and the five comparison methods on CSI500.

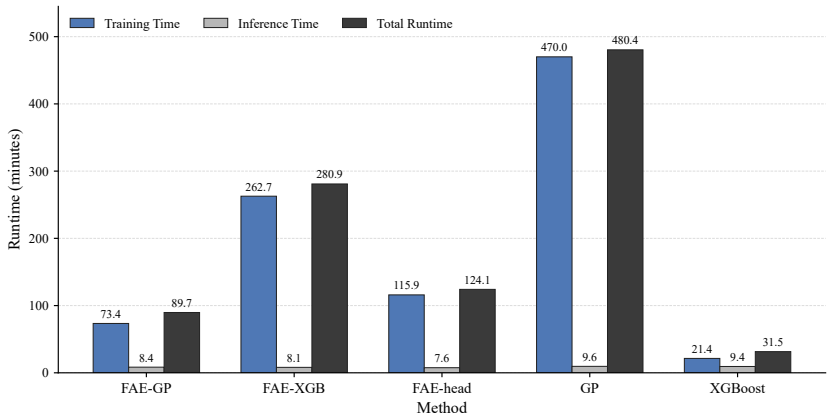

Figure S3: Execution time including the training time, the inference time and the backtesting time of FAE-GP and the four compared variants on CSI500.

C Runtime Analysis

This section investigates the computational cost of FAE-GP by comparing its runtime with those of the comparison methods. Specifically, we record the training time, the inference time, and the backtesting time for all methods. Then, the total time of each method by summing all the three kinds of time is also reported. Here, it should be mentioned that since we train all methods on all stocks and only use CSI500 and CSI1000 for test, the training time and the inference time, which occupy the most part of the execution time for each method, are the same for the two datasets. Therefore, we only report the computational time of all methods on one dataset, namely CSI500.

Figs. S2 shows the runtime of FAE-GP and the five comparison methods (namely AlphaForge, D-Va, QFR, RiskMiner, and SFAG) on CSI500, while Figs. S3 displays its runtime and those of the four developed variants (namely FAE-XGBoost, FAE, GP, and XGBoost) on CSI500.

As shown in Fig. S2, except for SFAG, the training time occupies the most part of the execution time for all the rest methods, while the inference time occupies the second most part of their execution time. Particularly, the training time of FAE-GP is much smaller than AlphaForge and D-Va, but larger than QFR, RiskMiner, and SFAG. In terms of the inference time, FAE-GP takes much less time than AlphaForge, D-Va, and SFAG, but more time than QFR and RiskMiner. Concerning the backtesting time, there are not so big difference among the six methods, but FAE-GP takes slightly smaller time than the five comparison methods.

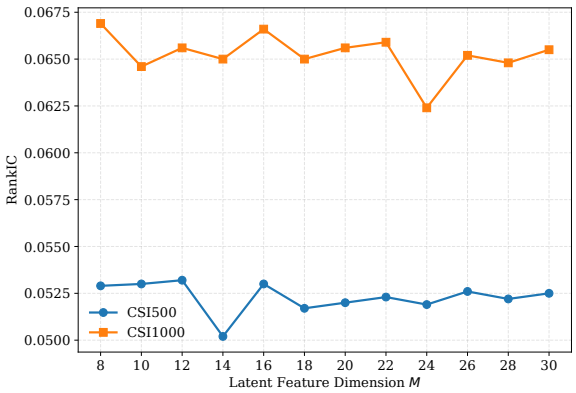

Figure S4: Change curves of FAE-GP with the increase of the number of the latent features M on CSI500 and CSI1000 regarding RankIC.

As a whole, even though FAE-GP takes more time than some comparison methods, its performance is significantly better than all the five comparison methods regarding RankIC and ALER.

As shown in Fig. S3, regarding the training time, FAE-GP takes much less time than FAE-XGBoost, FAE, and GP, but more time than XGBoost. As a whole, FAE-GP takes much less time than FAE-XGBoost, FAE, and GP, but more time than XGBoost. However, its performance is significantly better than all the four comparison variants in view of both RankIC and ALER. These actually demonstrate the effectiveness of hybridizing FAE and GP for alpha factor mining and analyzing the financial time-series data.

D Sensitivity Analysis of FAE-GP to the Number of the Latent Features

This section investigates the influence of the number M of the latent features extracted by FAE on the performance of FAE-GP because such a number has significant influence on the alpha factor mining by the GP module. For this purpose, we offer different settings for M ranging from 8 to 30 with step size 2.

Fig. S4 presents the change curves of FAE-GP with the increase of the number of the latent features M on CSI500 and CSI1000 regarding RankIC, while Fig. S5 shows the change curves of FAE-GP with the increase of the number of the latent features M on CSI500 and CSI1000 regarding ALER.

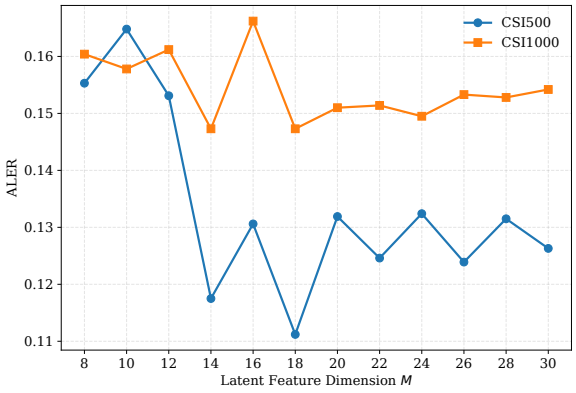

Figure S5: Change curves of FAE-GP with the increase of the number of the latent features M on CSI500 and CSI1000 regarding ALER.

From Fig. S4, in terms of RankIC, FAE-GP is not so sensitive to M since the gap of RankIC obtained by FAE-GP with different settings of M is not so large.

From Fig. S5, with respect to ALER, we find that the performance of FAE-GP differs obviously with different settings of M on the two datasets.

As a whole, it is observed that $M = 10$ helps FAE-GP gain the best overall performance on the two datasets. Therefore, $M = 10$ is set as the default setting for FAE-GP in this paper.

E GP FUNCTION SET

Table S3 lists the function set used to construct alpha expressions. Binary operators combine two subexpressions, whereas unary operators transform a single subexpression. The terminal set consists of the learned feature coordinates and numerical constants. These operators allow GP to express arithmetic combinations, nonlinear transformations, and comparisons of the learned features. The table describes their mathematical roles; domain-sensitive operations use the numerical safeguards of the GP implementation.

Table S3: Function set for GP-based alpha formula generation.

No.	Function	Definition
1	$\mathrm{add}(a, b)$	Sum of two inputs.
2	$\mathrm{sub}(a, b)$	Difference between two inputs.
3	$\mathrm{mul}(a, b)$	Product of two inputs.
4	$\mathrm{div}(a, b)$	Quotient of two inputs.
5	$\mathrm{abs}(a)$	Absolute value of an input.
6	$\mathrm{sqrt}(a)$	Square root of an input.
7	$\log(a)$	Natural logarithm of an input.
8	$\mathrm{inv}(a)$	Reciprocal of an input.
9	$\mathrm{neg}(a)$	Negation of an input.
10	$\max(a, b)$	Maximum of two inputs.
11	$\min(a, b)$	Minimum of two inputs.
12	$\sin(a)$	Sine of an input.
13	$\cos(a)$	Cosine of an input.
14	$\tan(a)$	Tangent of an input.

F GP SEARCH ALGORITHM

Algorithm 1 summarizes the GP search over the learned features. Candidate expressions are evaluated using the RankIC-based fitness with a complexity penalty in Equation 3. Tournament selection chooses parents, and crossover, mutation, or reproduction generates the next population. The best expression is tracked across the initial population and all subsequent generations. Here, G denotes the number of offspring generations after initialization, and K is the tournament size.

Algorithm 1: GP-based alpha formula generation

Input: Learned features $\mathcal{Z} = \{Z_t\}$; corresponding returns $\mathcal{Y} = \{\boldsymbol{r}_t\}$; function set F; population size NP; maximum offspring generations G; tournament size K; complexity coefficient λ.

Output: Best alpha expression g_{best}.

$\mathcal{P}_0 \leftarrow$ initialize NP valid expression trees using learned feature coordinates, numerical constants, and F;
Evaluate $\mathcal{F}(\tau)$ for each $\tau \in \mathcal{P}_0$ using Equation 3;
$g_{\text{best}} \leftarrow \arg\max_{\tau \in \mathcal{P}_0} \mathcal{F}(\tau)$;
$g \leftarrow 0$;
while $g < G$ **do**
    $\mathcal{Q}_g \leftarrow$ an empty offspring population;
    while $|\mathcal{Q}_g| < NP$ **do**
        Select crossover, mutation, or reproduction according to the configured probabilities;
        Select the required parent(s) from $\mathcal{P}_g$ using tournaments of size K;
        Generate one offspring expression τ using the selected operator;
        Append τ to $\mathcal{Q}_g$;
    Evaluate $\mathcal{F}(\tau)$ for each $\tau \in \mathcal{Q}_g$ using Equation 3;
    $\tau^\star \leftarrow \arg\max_{\tau \in \mathcal{Q}_g} \mathcal{F}(\tau)$;
    if $\mathcal{F}(\tau^\star) > \mathcal{F}(g_{\text{best}})$ **then**
        $g_{\text{best}} \leftarrow \tau^\star$;
    $\mathcal{P}_{g+1} \leftarrow \mathcal{Q}_g$;
    $g \leftarrow g + 1$;
return g_{best};