IDEA Research Report

Future Trend and Perspective

Quant 4.0: Engineering Quantitative Investment with Automated, Explainable and Knowledge-driven Artificial Intelligence

Jian Guo [1, 2]
IDEA Research &
The Hong Kong University of Science and Technology (Guangzhou)

Saizhuo Wang [1]
The Hong Kong University of Science and Technology &
IDEA Research

Lionel M. Ni
The Hong Kong University of Science and Technology (Guangzhou) &
The Hong Kong University of Science and Technology

Heung-Yeung Shum [2]
IDEA Research &
The Hong Kong University of Science and Technology

Keywords: *AGI, Artificial Intelligence, AutoML, Causality Engineering, Deep Learning, Feature Engineering, Investment Engineering, Knowledge Graph, Knowledge Reasoning , Knowledge Representation, Model Compression, NAS, Quant 4.0, Quantitative Investment, Risk Graph, XAI*

1. Equal Contribution
2. Corresponding Author

idea.edu.cn

Table of Contents

Quant 4.0: Engineering Quantitative Investment with Automated, Explainable and Knowledge-driven Artificial Intelligence

Jian Guo[a,c,1,*], Saizhuo Wang[a,b,1,2], Lionel M. Ni[b,c], Heung-Yeung Shum[a,b,*]

[a]*IDEA Research, International Digital Economy Academy, 5 Shihua Road, Futian District, Shenzhen, 518045, Guangdong, China*
[b]*The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong, 999077, China*
[c]*The Hong Kong University of Science and Technology (Guangzhou), 1st Duxue Road, Nansha District, Guangzhou, 518055, Guangdong, China*

Abstract

Quantitative investment ("quant") is an interdisciplinary field combining financial engineering, computer science, mathematics, statistics, etc. Quant has become one of the mainstream investment methodologies over the past decades, and has experienced three generations: Quant 1.0, trading by mathematical modeling to discover mis-priced assets in markets; Quant 2.0, shifting quant research pipeline from small "strategy workshops" to large "alpha factories"; Quant 3.0, applying deep learning techniques to discover complex nonlinear pricing rules. Despite its advantage in prediction, deep learning relies on extremely large data volume and labor-intensive tuning of "black-box" neural network models. To address these limitations, in this paper, we introduce Quant 4.0 and provide an engineering perspective for next-generation quant. Quant 4.0 has three key differentiating components. First, *Automated AI* changes quant pipeline from traditional hand-craft modeling to the state-of-the-art automated modeling, practicing the philosophy of "algorithm produces algorithm, model builds model, and eventually AI creates AI". Second, *Explainable AI* develops new techniques to better understand and interpret investment decisions made by machine learning black-boxes, and explains complicated and hidden risk exposures. Third, *Knowledge-driven AI* is a supplement to data-driven AI such as deep learning and it incorporates prior knowledge into modeling to improve investment decision, in particular for quantitative value investing. Moreover, we discuss how to build a system that practices the Quant 4.0 concept. Finally, we propose ten challenging research problems for quant technology, and discuss potential solutions, research directions, and future trends.

1. Introduction

Quantitative investment is an important part of wealth management (a.k.a. asset management) industry. This section contains introductory knowledge about quant, including market situation, classification and principles of strategy development, historical landmarks, and concepts of Quant1.0–Quant4.0.

1.1. Wealth Management and Quant

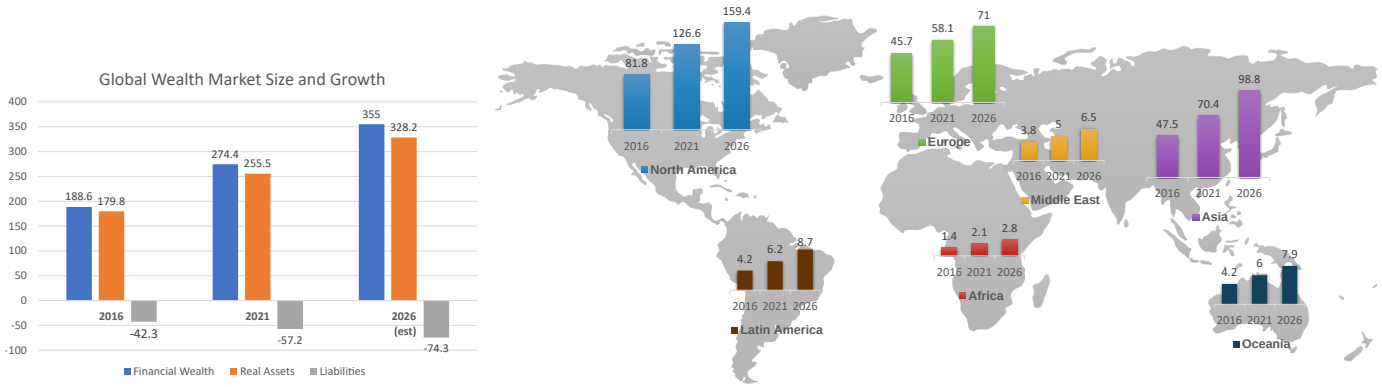
The wealth management industry is one of the largest sectors of the world's economy. According to a global wealth report from Boston Consulting Group (BCG) [1] and the illustration in Figure 1, the volume of global financial wealth has grown from 188.6 trillion USD in 2016 to 274.4 trillion USD in 2021, almost three times as the global nominal GDP in 2021. Moreover, the company predicts this number will increase to 355 trillion USD in 2026. It is not surprising that North America, Asia, and Europe are the three biggest regional markets of wealth management in the world, with approximately 46%, 26%, and 21% of the global market size in 2021, respectively. We also see the stable and sustainable growth of the wealth management market, both globally and regionally. Figure 2 shows an ecosystem of the wealth management industry, where investment funds as well as fund managers (a.k.a. investment managers) play core roles. They raise money from various capital providers, such as endowment foundations, fund of funds (FOF), family offices, billionaires, insurance companies, pension/sovereign funds and retail clients, and invest this money into financial markets to bet return and profit for their customers. Many types of investment instruments are liked by fund managers, such as stocks, exchange-traded funds (ETFs), bonds, futures, options, and foreign exchange [2]. Some investment funds even borrow money from depository institutions such as banks or peer-to-peer lending companies for investment and profit from the difference between investment return and loan interest. With the rapid development of digital economy, big data, and artificial intelligence, more and more new technologies are applied in the wealth management industry, leading to a branch of financial technology/engineering, called "investment engineering" [3]. Consequently, the pipeline of investment research, trading execution, and risk management is becoming a systematic, automated, and intelligent process, and this philosophy has been practiced in the recent evolution of quant.

As an important family of players in financial markets and the wealth management industry, contemporary quant applies

*Corresponding author.
Email addresses: guojian@idea.edu.cn (Jian Guo), swangeh@connect.ust.hk (Saizhuo Wang), ni@ust.hk (Lionel M. Ni), hshum@idea.edu.cn (Heung-Yeung Shum)

[1]Equal contribution.

[2]This work was done during the internship at IDEA Research.



(a) Global market sizes of wealth management.

(b) Regional market sizes of wealth management.

Figure 1: Global and regional market sizes of wealth management industry (unit: trillion USD). Panel (a) illustrates the volume of financial wealth, real assets and liabilities in 2016, 2021 and 2026 (estimated) in the world. Panel (b) shows the distribution of financial wealth in seven regional markets around the world in 2016, 2021 and 20226 (estimated). Data come from the report of BCG [1].

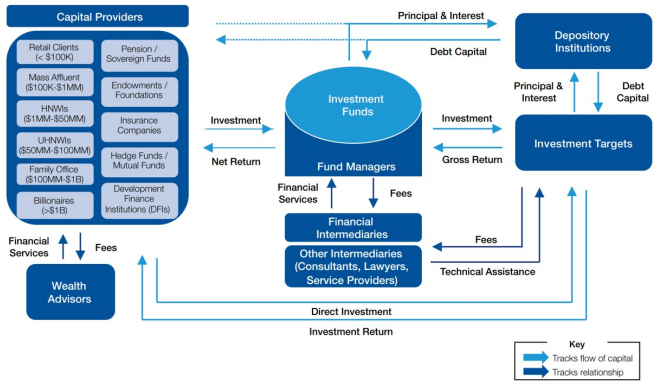


Figure 2: The asset management ecosystem [4]

rigorous mathematical and statistical modeling techniques, machine learning techniques, and algorithmic trading techniques to discover asset pricing abnormalities in financial markets and make money from the following arbitrage or investment opportunities. Compared with traditional fundamental and technical investment, quantitative investment has a number of advantages. Firstly, the performance of quant strategies can be examined and evaluated beforehand using back-test experiments based on historical data before the beginning of real trading. Secondly, quant trading has speed superiority in bidding orders with the best price. Thirdly, it eliminates the negative effect of human emotion in decision-making. Finally, quant research has significant advantages in data analysis with much deeper, broader and diversified coverage of information about financial markets and sectors. In the past 30 years, information infrastructure and computer technology are widely applied by financial exchange markets around the world. Nowadays, massive financial data are generated and millions of orders are executed every second, leading to the rapid growth of the quant industry. Taking the U.S. stock market as an example, over 60% of overall trading volumes comes from the orders placed by computer trading algorithms rather than human traders [5].

1.2. Quant Strategies

A quant strategy is a systematic function or trading methodology used for trading securities in financial markets based on predefined rules or trained models for making trading decisions. Strategies are usually the core intelligent property of a quantitative fund.

1.2.1. Components of Quant Strategy

A standard quant strategy contains a series of components, such as investment instrument, trading frequency, trading mode, strategy type and data type, and we introduce them one by one (see Figure 3).

- Investment instrument specifies which financial instruments are put in the universe by the strategy. Popular candidate instruments include stocks, ETFs, bonds, foreign exchanges, convertible bonds, and cryptocurrencies, as well as more complicated financial derivatives such as futures, options, swaps, and forwards [6]. An investment strategy could trade either a single type of instrument (e.g., a strategy for trading ETFs) or multiple types of instruments (e.g., an alpha hedging strategy that longs stocks and shorts index futures to eliminate market risks).

- Trading frequency specifies how to hold your asset in portfolio and how frequently to trade. Usually, high-frequency trading holds a position in several minutes or seconds, while low-frequency trading may hold an asset over several months or years. Comparing high-frequency trading and low-frequency trading, the dramatic discrepancy of holding periods result in very different consideration in strategy design. For example, asset capacity limitations and trading costs are big issues for high-frequency trading, while how to control the risk of drawdown [7] is what we should carefully think about for low-frequency trading.

- Model type characterizes how to formally model the trading problem. Examples include cross-sectional trading, time-series trading, and event-driven trading [7]. Cross-sectional trading is used commonly in stock selection, where all stocks

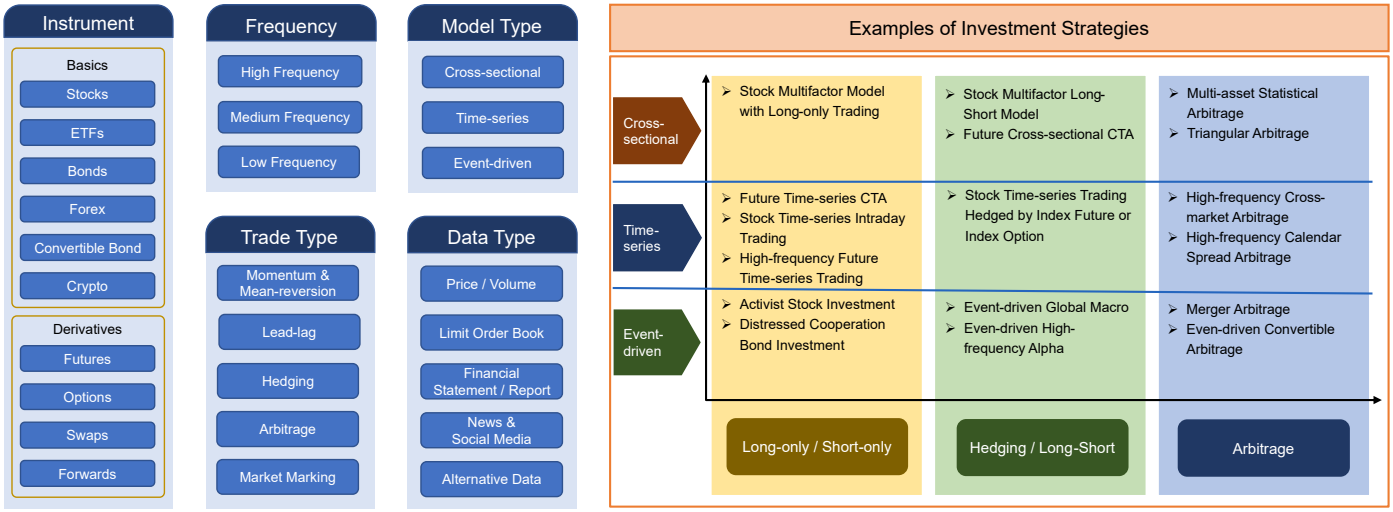


Figure 3: Classification of common strategies and investment instruments.

in a universe are ranked according to their scores of expected future returns predicted by a model, and portfolio managers could long stocks with the highest scores and short those with the lowest scores. Time series trading is relatively simple, where long/short trading operates only on a single instrument such as a certain stock or a certain future contract. Event-driven trading differs from time-series trading because the time intervals between events are not evenly distributed over time, while investment decisions and trading executions are triggered by the occurrence of events.

- Trade type is a series of thinking templates for us to design a strategy quickly. Examples include momentum trading [8], mean-reversion trading [9], arbitrage trading [10], hedging [11], market making [12], etc. By leveraging these strategy types, traders can explore profit chances from different aspects of financial markets. Specifically, momentum trading assumes the price trend is sustainable in the following time window and it follows this trend direction to trade. Mean-reversion trading, on the contrary, bets the price trend will move towards the opposite direction in recent future and buy opposite positions. Hedging is the purchase of one asset with the intention of reducing the risk of loss from another asset. Arbitrage is simultaneously longing and shorting the same asset in different markets or a pair of highly correlated assets in order to profit from the convergence of price discrepancy. Market making is a liquidity-providing trade that quotes both a buy and a sell price in a tradable asset held in inventory, hoping to make a profit on the bid–ask spread.

- Data type means what type of data is used in a strategy. Typical data types include quote data, limit order books [13] b, news data, financial statements, analysts' reports, and alternative data such as sentimental data, location data, satellite images, etc. A strategy researcher must consider what kind of data he has and what kind of data he needs in a strategy development process. For example, limit order book streams are usually used in building high-frequency trading strategies, while news data are used more commonly in event-driven strategies.

1.2.2. Examples of Popular Strategies

Figure 3 also list a number of popular strategies as examples. For example, stock hedging strategy based on multifactor model [14] is very popular in many main markets around the world. This strategy hedges market risk by longing the most favorable stocks and shorting the other end (in some markets prohibiting shorting, short the corresponding index future or index option instead). If we trade stocks with multifactor models in a long-only way without shorting and constrain the risk exposure between selected portfolios and certain stock indices, it is an enhanced indexing strategy, which is almost the most popular quantitative strategy in China's stock market if measured by assets under management (AUM).

1.3. Fundamental Principles of Asset Management

Similar to the situation that learning law of energy conservation could help avoid the trap of perpetual motion machine, it is beneficial to learn some fundamental principles of asset management so as to get rid of some common traps in strategy development.

1.3.1. Fundamental Law of Active Management

The first principle is the *fundamental law of active management* developed by Richard Grinold and Ronald Kahn [15]. This principle states that the performance of an active investment manager (or equivalently quant model) depends on the quality of investment skills and, consequently, the frequency of investment opportunities. This law can be expressed mathematically as follows:

$$IR = IC \times \sqrt{Breadth} \tag{1}$$

where IC is the information coefficient (correlation between the predicted return and true return in a future time window)

evaluating investment quality, *Breadth* means the number of independent investment decisions in a year, and *IR* is the ratio of portfolio returns above the returns of a benchmark to the volatility of returns, measuring the performance of asset management. Mathematically, the fundamental law of active management can be regarded as an application of the central limit theorem in mathematical statistics [16]. When applying this law in practice, we have to notice that *IC* and *Breadth* are usually not independent. For example, given a strategy, we may increase its *Breadth* by relaxing the threshold of trading signals, but in this way, *IC* may decrease because more false-positive noise is introduced to our decisions. Therefore, a good strategy should find an optimal trade-off between these two coupled variables. Figure 4a illustrates the distribution of various popular strategies on *IC* and *Breadth*, and their corresponding *IR* performance.

1.3.2. Impossible Trinity of Investment

The second principle is the impossible trinity of asset management. Specifically, any investment strategy can not meet the following three conditions simultaneously, i.e., high return, low risk (or equivalently high stability), and high capacity. Figure 4b illustrates the impossible trinity using a radar chart with three variables return, stability and capacity. For example, high-frequency market making and calendar arbitrage strategy could reach high return and stability (low portfolio volatility), but the capacity of its AUM is usually small, typically hard to exceed several billions of USD even in global trading. On the contrary, stock fundamental strategy has high capacity up to trillions of USD, but its return and stability are not as good as those of high-frequency trading.

1.4. History of Quantitative Investment

The origin of quant can trace back to over a century ago when French mathematician Louis Bachelier published his Ph.D. thesis "The Theory of Speculation" in 1900 [17] and he exhibited how to use probability law and mathematical tools to study the movement of stock prices. As a pioneer exploring the application of advanced mathematics in financial markets, Bachelier's work inspired academic research of quantitative finance despite the lack of industry application due to data scarcity at his age. Quantitative investment was first practiced by American mathematics professor Edward Thorp, who used probability theory and statistical analysis to win blackjack games, and his research was subsequently used to seek systematic and consistent returns in stock markets [18]. In this subsection, we introduce the history and landmarks in the development of quantitative finance through two routes: research landmarks in academia and evolution of quant in industry practice.

1.4.1. Q-Quant and P-Quant

People in academia and investment industry classify quantitative finance into two branches, which are usually referred to as "Q-quant" and "P-quant". These two branches are named after their differentiation in modeling based on risk-neural measure and probability measure, respectively. Generally speaking, Q-quant studies the problem of derivative pricing and *extrapolate the present*, using a model-driven research framework where data is usually used to adjust the parameters of models. On the other hand, P-quant studies quantitative risk and portfolio management to *model the future*, using a data-driven research framework where different models are built to improve the fitting of historical data. Usually, Q-quant research is conducted in sell-side institutes such as investment banks and security companies, while P-quant is popular in buy-side institutes such as mutual funds and hedge funds. Table 1 compares the characteristics of these two types of quant.

Table 1: Comparisons of P-quant and Q-quant [19].

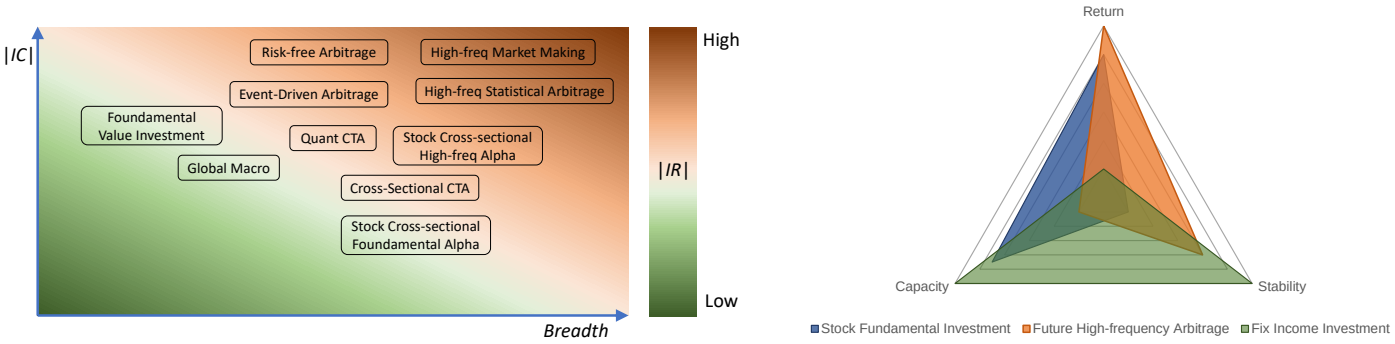
	Q-quant	**P-quant**
Goal	Extrapolate the present	Model the future
Scenario	Derivatives pricing	Portfolio management
Measure	Risk-neural measure	Probability measure
Modeling	Continuous stochastic process	Discrete time series
Example	Black-Scholes model	Multifactor model
Algorithm	Ito calculus, PDEs	Statistics, Machine Learning
Challenge	Calibration	Estimation/Prediction
Business	Sell-side	Buy-side

1.4.2. Landmarks in Q-Quant

In 1965, Paul Samuelson, American economist and the winner of 1970 Nobel Memorial Prize in Economic Sciences, introduced stochastic process and stochastic calculus tools in analyzing financial markets and modeling the stochastic movement of stock prices [20], and in 1965, he published a paper studying the lifetime portfolio selection problem using a stochastic programming method [21]. In the same year, another American economist Robert Merton published his work about lifetime portfolio selection as well. Different from Samuelson's work using discrete-time stochastic process, Merton's work modeled the random uncertainty of portfolio using continuous-time stochastic calculus [22]. Almost in the same year, economists Fischer Black and Myron Scholes demonstrated that the expected return and risk of assets under management could be removed by dynamically revising a portfolio, and thus inventing the risk-neutral strategy for derivative investment [23]. They applied the theory to real market trading and published it in 1973. The risk-neutral formula was later named in honor of them and called *Black-Scholes Model* [24], a partial differential equation (PDE) tool for pricing a financial market containing derivative investment instruments. Specifically, the Black–Scholes model establishes a partial differential equation governing the price evolution of a European option call or European option put, as follows:

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0 \tag{2}$$

where V is the price of the option as a function of stock price S and time t, r is the risk-free interest rate, and σ is the volatility of the stock. This PDE has a closed-form solution called Black-Scholes Formula. Since Robert Merton was the first to publish a paper expanding the mathematical understanding of the options pricing model, he was usually credited with the contribution of this theory as well. Merton and Scholes received the 1997



(a) Common investment strategies under the fundamental law of active management. The figure illustrates the relationship between the magnitude of *IR* with *breadth* and the magnitude of *IC* for different strategies.

(b) Illustration of the impossible trinity of return, capacity, and stability for active management using three typical strategies: stock fundamental investment, future high-frequency arbitrage and fixed income investment.

Figure 4: Illustration of the principles for active investment management with specific strategies.

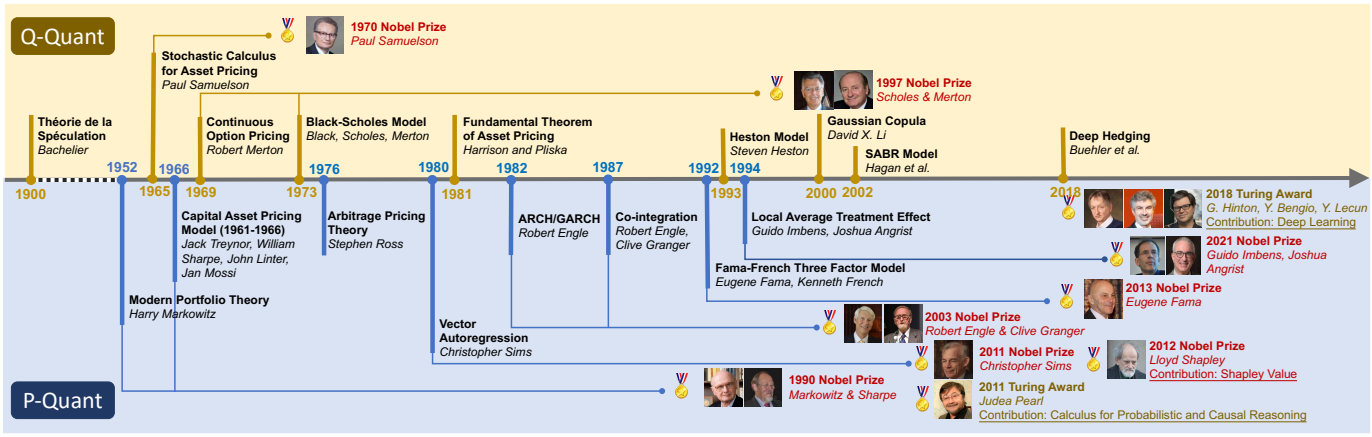


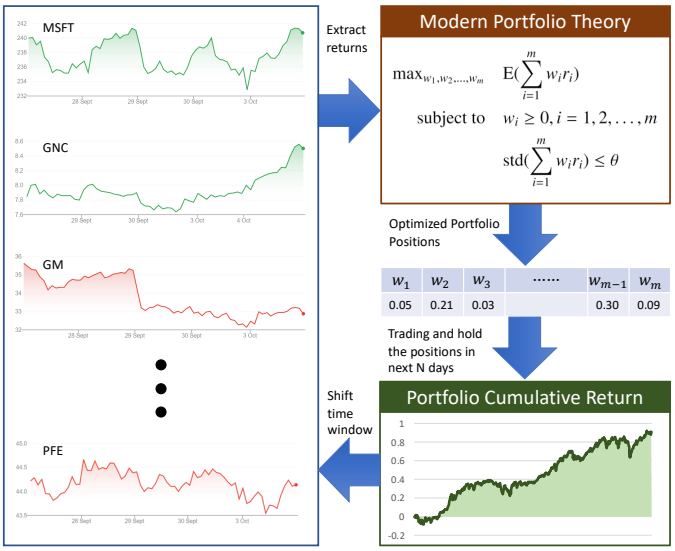
Figure 5: Main academic contributors and their works that deeply influence the development of quantitative investment. Photo credit: Wikipedia.

Nobel Memorial Prize in Economic Sciences for their discovery of the risk-neutral dynamic revision. The original Black-Scholes model was extended later for deterministically variable rates and volatilities, and was extended to characterize the price of European options on instruments paying dividends, as well as American options and binary options.

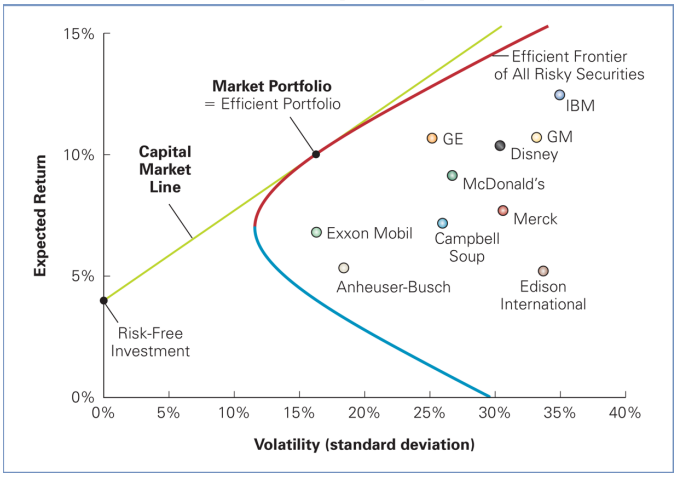
As a pioneering work in risk-neutral theory, Black-Scholes model has many limitations, one of which is the assumption that the underlying volatility is constant over the life of the derivative, and is unaffected by the changes in the price level of the underlying security. This assumption usually contradicts the phenomenon of the smile and skew shapes of implied volatility surfaces. A possible solution is to relax the constant volatility assumption. By characterizing the volatility of the underlying price using stochastic process, it is possible to model derivatives more accurately in practice, and this idea leads to a series of works about stochastic volatility, such as the Heston model [25] and the SABR model [26]. As a commonly used stochastic volatility model, the Heston model assumes the variation of the volatility process varies as a square root of the variance itself, and it exhibits a reversion trend towards the long-term mean of variance. Another popular stochastic volatility model is the SABR model, commonly used in interest rate derivative markets. This model uses stochastic differential equations to describe a single forward (such as a LIBOR forward rate, a forward swap rate, or a forward stock price) as well as its volatility, and has the ability to reproduce the effect of volatility smile. In recent years, deep learning and reinforcement learning techniques are applied to integrate with risk neural Q-quant modeling and a concept *learning to trade* was introduced by Hans Buehler [27], who proposed the deep hedging model, a framework for hedging a portfolio of derivatives in the presence of market frictions such as transaction costs, market impact, liquidity constraints or risk limits and for modeling the volatility stochastic process using deep reinforcement learning and market simulation. It does not use the Greeks anymore and naturally captures co-movements of relevant market parameters.

In addition to derivative pricing models, market efficiency theory and risk modeling theory are also very important in Q-quant, both in academia and industry. In 1980s, Harrison and Pliska established the fundamental theorem of asset pricing [28], which provides a series of necessary and sufficient conditions for an efficient market to be arbitrage free as well as complete. In 2000, David X. Li introduced the statistical model Gaussian copula [29] to evaluate the value-at-risk (VaR) of derivative pricing and portfolio optimization, especially the collateralized debt obligations (CDO). Gaussian copula quickly became a tool for financial institutions to correlate associations between mul-

tiple financial securities since it is relatively simple in modeling even for those assets too complex to price previously, such as mortgages.



(a) Illustration of Markowitz' portfolio optimization theory.



(b) Illustration of efficient frontier first formulated by Harry Markowitz. Figure cited from [30].

Figure 6: Portfolio optimization

1.4.3. Landmarks in P-Quant

Q-quant plays an extremely important role in quantitative finance. In this article, however, we stand on a buy-side point of view and focus on asset prediction and portfolio optimization problems, and thus all discussions about quantitative investment in the following content assume a P-quant statement unless otherwise specified. The origin of P-quant started from the establishment of modern portfolio theory introduced by Harry Markowitz. The theory was initialized in his Ph.D. thesis "Portfolio Selection" and later published in Journal of Finance in 1952 [31], and an extension published in his book *Portfolio Selection: Efficient Diversification of Investments* [32] in 1959. According to the old adage "Don't put all your eggs in one basket", Markowitz came up with the concept of efficient frontier of asset investment in financial market and formalized it mathematically as a quadratic optimization problem by maximizing the expected return of the portfolio given its risk (usually measured by the variance of the assets in a portfolio) at a certain level. Figure 6 illustrates the application of Markowitz's theory for allocating the best positions for assets in portfolio and illustrates the concept of efficient frontier.

Based on the modern portfolio theory, the Capital Asset Pricing Model (CAPM) was later introduced by Jack Treynor (1961, 1962) [33], William F. Sharpe (1964) [34], John Lintner (1965) [35] and Jan Mossin (1966) [36] independently. CAPM aims to describe the relationship between systematic risk from the market and expected return for assets.

$$E(R_p) - R_f = \alpha + \beta \cdot (E(R_m) - R_f) \tag{3}$$

where $E(R_p)$ is the expect return of portfolio, R_f is the risk-free return, $E(R_m)$ is the expected return of market. Specifically, CAPM decomposes asset return and risk into two separate parts, alpha and beta. Alpha measures the performance of a portfolio compared to a benchmark index (e.g., S&P500 index), while beta measures the variance of the portfolio in relation to a benchmark index, characterizing the risk from market volatility. One of the main contributors to CAPM, William Sharpe, shared the 1990 Nobel Prize with Harry Markowitz. A following important step in quantitative finance is the establishment of arbitrage pricing theory (APT) by MIT economist Stephen Ross in 1976 [37]. APT improved its predecessor CAPM by further introducing the multifactor model framework to build the relationship between asset price and various macroeconomic risk variables. Under the multifactor model framework, Nobel Prize-winning economist Eugene Fama proposed the famous Fama–French Three-Factor Model with his colleague Kenneth French at the University of Chicago in 1992 [14].

$$E(R_p) - R_f = \beta_0 + \beta_1 \cdot (E(R_m) - R_f) + \beta_2 \cdot SMB + \beta_3 \cdot HML \tag{4}$$

This model establishes the relationship between the expected portfolio return (up to subtracting a risk-free return) $E(r_p) - r_f$ with respect to three systematic risk factors: expected market return $E(R_m) - R_f$, size SMB (the spread between small capitalization stocks and large capitalization stocks), book-to-market values HML (the spread between high book-to-market companies and low book-to-market companies). The three-factor model was then extended to Fama and French Five Factor Model in 2015 [38], by adding two more factors: profitability (return spread of the most profitable firms minus the least profitable) and investment aggressiveness (the return spread of firms that invest conservatively minus aggressively).

Parallel with the progress of multifactor models, a number of significant research about time-series analysis appears in 1980s. In 1980, Nobel Prize winner Christopher Sims introduced the Vector Autoregression (VAR) model into economics and finance. As an extension of single sequence autoregressive (AR) model and autoregressive-moving-average (ARMA) model commonly used in time-series analysis, VAR characterizes the autoregressive properties over time across multiple

times series and it assumes constant variance of error terms in the regression formula. In 1982, Robert Engle introduced the Autoregressive Conditional Heteroskedasticity (ARCH) model and extend it to Generalized Autoregressive Conditional Heteroskedasticity (GARCH) model to characterize the pattern of financial volatility in the market by specifying stochastic variance in the model. In 1987, he introduced co-integration method with Clive Granger (inventor of Granger Causality for modeling lead-lag patterns among multiple time series) for testing the significance of mean-reversing patterns in financial time series, and co-integration test has been widely used in discovering promising asset pairs for statistical arbitrage strategies. Both Engle and Granger received the 2003 Nobel Prize for their contribution to time series analysis which has been widely applied in quantitative finance for market forecasting and investment research. In 2018, three pioneers in deep learning techniques, Yoshua Bengio, Geoffrey Hinton and Yann LeCun, are granted the Turing Award. Nowadays, deep learning has been widely used by academic researchers in finance and quant researchers in financial institutions to build complex nonlinear models in order to learn the relationship between financial signals and expected returns and to predict asset prices, and its powerful ability in fitting big data significantly improves the performance of market prediction and portfolio management.

Although accurately predicting the future trend of asset price is a very important task in P-Quant, how to explain the effect of model prediction and interpret how a model is really working seems more important for quant researchers since "know how" is more crucial than "know what" in risk management for portfolio managers. Causal effect analysis [39] and factor importance analysis are two core tasks in quant model interpretation. Clive Granger invented the Granger Causality Test in 1969 [40] for determining whether one time series is useful in forecasting another. The original Granger causality test does not account for latent confounding effects and does not capture instantaneous and non-linear causal relationships, though several extensions have been proposed to address these issues. Although there is an argument about whether Granger causality test can evaluate "real" causality in terms of statistics, this method has been widely applied in quant research such as searching and evaluating pairs of stocks with significant lead-lag effect and trading with corresponding strategies. In 1994, Guido Imbens and Joshua Angrist introduced the local average treatment effect (LATE) model to characterize the statistical causal effect in economics, finance and social sciences, and they shared the 2021 Nobel Prize in economics. Another important contributor in causal inference is the Turing Award winner Judea Pearl, who invented the causal diagram (Bayesian network) and it can be used to mine the causal effect among factors and returns in multifactor model. On the other hand, in the area of factor importance analysis, Shapley Value has become an important criterion for measuring the contribution of single feature in a complex nonlinear machine learning model. In fact, it is interesting that this criterion was invented originally to measure the contribution of individual player/agent in a cooperative gaming process when it was first proposed by Lloyd Shapely, a Nobel Prize winner and a pioneer in game theory research.

1.4.4. Development of Quant in Industry

The blooming era of quantitative investment funds started from 1990s, along with the emergence of the Internet and the development of electronic trading in exchanges. Here we briefly introduce the evolution of quant operating models and classify them into three generations, denoted as Quant 1.0–3.0, and summarize their characteristics in Figure 7.

- Quant 1.0 appeared in the early age of quantitative investment but it is still the most popular quant operating model in contemporary market. The features of Quant 1.0 includes: 1) Small but elite team, typically led by an experienced portfolio manager and composed of a few genius researchers and traders with strong mathematics, physics or computer science background; 2) applying or even inventing mathematical and statistical tools to analyze financial market analysis and discover mispriced assets for trading; 3) trading signals and trading strategies are usually simple, understandable and interpretable to reduce the risk of in-sample over-fitting in modeling. This operating model has high efficiency in quant trading but low robustness in management. Especially, the success of a Quant 1.0 team relies too much on particular genius researchers or traders, and such a team may decline or even bankrupt rapidly with the departure of genius. In addition, such a small "strategy workshop" limits the research efficiency on complex investment strategies such as quantitative stock alpha strategy which depends on diversified financial data types, extremely large data volume, and complex modeling techniques such as super large deep learning model.

- Quant 2.0 changes quant operating model from small *genius' workshop* to an industrialized and standardized *alpha factory*. In this model, hundreds or even thousands of investment researchers work on the same pipeline to mine effective alpha factors [41] out of the plethora of financial data, using standardized evaluation criteria, standardized back-test processes and standardized parameter configurations. These alpha mining researchers are rewarded by submitting qualified alpha factors which usually have high back-test returns, high Sharpe ratio, reasonable turnover rate and low correlation with existing factors in the alpha database. Traditionally, each alpha factor is a mathematical expression characterizing some pattern or profile of stocks, or some relationship between stocks, although more and more complicated machine learning factors are mined as well. Typical alpha factors include momentum factors, mean-reversion factors, event-driven factors, volume-price dispersion factors, growth factors, etc. Many alpha factors submitted by alpha researchers are combined into statistical models or machine learning models by portfolio managers to find the optimal asset positions after appropriate risk neutralization, expecting to obtain a stable and promising excess return in the market. However, large-scale team work results in huge costs for human resources, and the situation gets more and more serious with the team growing larger and larger. Specifically, we could expect the number of discovered effective alphas follows an approximately linear trend with the team size (actu-

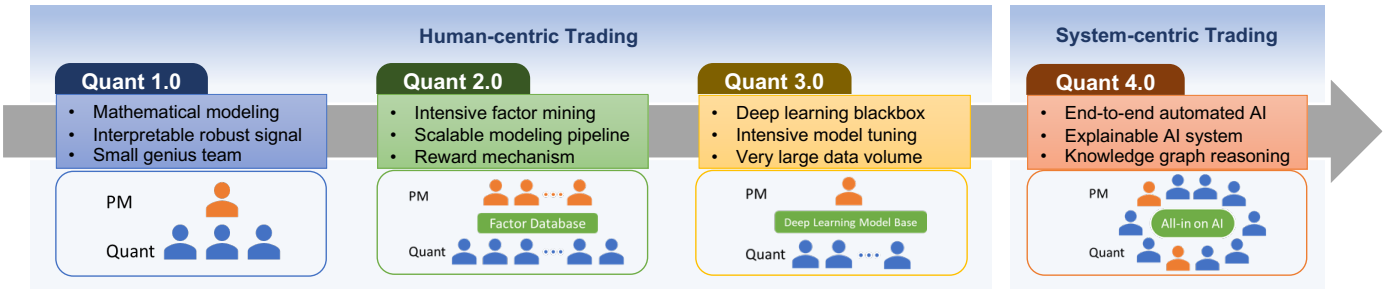


Figure 7: The development history of of quantitative investment in industry, from Quant 1.0 to Quant 4.0.

ally in practice, discovering new effective alphas is more and more difficult when the size of accumulated factors is already large), but the portfolio return grows significantly lower than the expand of alpha volume and team size, and this results in the profit margin getting smaller and smaller. This phenomenon is caused by a number of reasons such as the limitation of strategy market capacity, the growing difficulty in discovering new effective alphas, and even the limitation of human intelligence in searching all possibilities in strategy space.

- Quant 3.0 emerges with the rapid development of deep learning techniques which have exhibited success in many domain areas such as computer vision and natural language processing. Different from Quant 2.0 which puts more research efforts and human labor into mining sophisticated alpha factors, Quant 3.0 pays more attention to deep learning modeling. With relatively simpler factors, deep learning still has the potential to learn a prediction model performing as well as a Quant 2.0 model, by leveraging its powerful end-to-end learning ability and its flexible model fitting ability. In Quant 3.0, the cost of human labor of alpha mining is at least partially replaced by the cost of computing power, especially for the expensive GPU servers. But generally speaking, it is a more efficient way for quant research in the long run.

1.5. Quant 4.0: Why and What

1.5.1. Limitations of Quant 3.0

Although Quant 3.0 has demonstrated its success in some strategy scenarios such as high-frequency stock and future trading, it has three primary limitations.

1. Traditionally, building a "good" deep neural network is time-consuming and labor-intensive, because of the heavy work in network architecture design and model hyperparameter tuning, as well as the tedious work in model deployment and maintenance in trading ends.
2. It is a challenge to read understandable messages from a model encoded by deep learning black box, making it very unfriendly to investors and researchers who care much about the mechanism of financial markets and expect to know the source of profit and loss.
3. The good performance of deep learning relies heavily on extremely large volumes of data, and thus only high-frequency trading (or at least medium cross-sectional alpha trading with large breadth) belongs to the strategy pool that deep learning favorites. This phenomenon prevents deep learning techniques from application in low-frequency investment scenarios such as value investing, fundamental CTA and global macro.

New research and new techniques are needed to address these limitations, and this leads to our proposal for Quant 4.0 in this article.

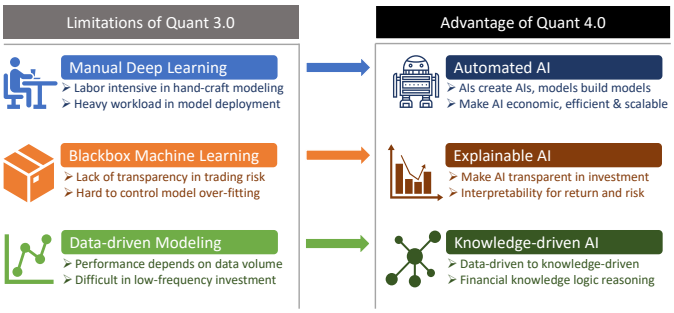


Figure 8: The three key components of Quant 4.0.

1.5.2. What is Quant 4.0?

We believe the limitations of Quant 3.0 are very likely to be solved or at least partially solved in the future with the quick development of the artificial intelligence (AI) technology frontier. Quant 4.0, the next-generation quant technology, is practicing the philosophy of "end-to-end going all-in on AI" and "AI creates AI" by incorporating the state-of-the-art automated AI, explainable AI and knowledge-driven AI and plotting a new picture for quant industry.

- Automated AI aims to build end-to-end automation for quant research and trading, in order to significantly reduce the cost of labor and time for quant research including data preprocessing, feature engineering, model construction and model deployment, and to dramatically improve R&D's efficiency and sustainability. In particular, we introduce state-of-the-art AutoML [42] techniques to automate every module in the whole strategy development pipeline. In this way, we propose to change traditional hand-craft modeling to an automated modeling workflow in an "algorithm produces algorithm, model builds model" manner, and eventually move towards a technical philosophy of "AI creates AI". Besides AI

automation, another important task is to make AI more transparent, which is essentially important for investment risk management.

- Explainable AI, usually abbreviated as XAI in machine learning area, attempts to open the black box encapsulating deep learning models. Pure black-box modeling is unsafe for quant research because people can not calibrate the risk accurately. It is difficult to know, for example, where returns come from and whether they rely on certain market styles, and what the reason for a specific drawdown is, under black-box modeling. More and more new techniques in the field of XAI could be applied in quant to enhance the transparency of machine learning modeling, and thus we recommend quant researchers to pay more attention to XAI. We have to notice that improving model explainability has costs. Figure 9 shows an impossible trinity of versatility, accuracy and explainability, and tells us that we have to sacrifice at least one apex in the triangle to obtain the benefit from the other two. For example, physical law $E = mc^2$ establishes an explainable and accurate relationship among energy, mass and speed of light, but this formula can be only applied in specific domains of physics and sacrifices versatility. Imagining that we provide more prior knowledge or domain experience in a model, it is equivalent to reducing the versatility to protect the performance of accuracy and explainability simultaneously.

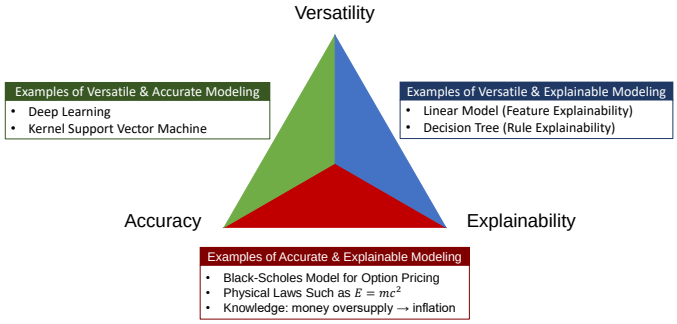


Figure 9: Impossible trinity of versatility, accuracy and explainability in modeling.

- Knowledge-driven AI differs from the data-driven AI which heavily depends on large volumes of data samples and thus is appropriate for investment strategies with large breadth such as high-frequency trading or stock cross-sectional trading. It is an important complement to data-driven AI techniques such as deep learning (illustrated in Figure 10 using Bayes' theorem). In this paper, we introduce knowledge graph which represents knowledge with a network structure composed of entities and relations, and stores knowledge with semantic triples. A knowledge graph of financial behaviors and events could be analyzed and inferred for investment decisions using symbolic reasoning and neural reasoning techniques. This implies potential applications to those investment scenarios with low trading frequency but intensive fundamental information in collection and analysis, including value investing and global macro investment.

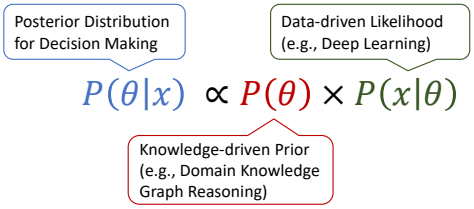


Figure 10: Complementary function of data-driven AI and knowledge-driven AI in decision making.

2. Automated AI for Quant 4.0

Automated AI for Quant 4.0 covers the automation of the full quant pipeline. In this section, we will first give an overview of the pipeline and then introduce how to upgrade it to an automated AI pipeline.

2.1. Automating Quant Research Pipeline

2.1.1. Traditional Quant Pipeline

Over decades of development, quant research has formed a standard workflow as shown in Figure 11 (blue part). This workflow consists of a number of modules, including data preprocessing, factor mining, modeling, portfolio optimization, order execution, and risk analysis.

- Data preprocessing is usually the first step in quant research. Original raw data may have many issues. Firstly, financial data usually have missing records, more or less. For example, in technical analysis, you may not receive price data at some time points due to packet loss during communication, or you may miss the price data on some trading days because of stock suspension. Similarly, in fundamental analysis, you may miss part of financial statement data since they are not reported on time. Although conventional statistical data imputation methods could be used to estimate and fill in missing records, we must avoid using future information in the imputation process. Secondly, financial data contain extreme values and outliers which may come from misrecording, data storage issues, data transfer issues, or extreme markets, and these outliers may lead to risky biases in investment decisions. Outliers could be eliminated by data winsorization methods [43] which limit extreme values in a certain percentile range, but we have to notice that some outliers are actually strong signals for quant trading rather than noise, and must differentiate the two during data preprocessing. Thirdly, many financial data, such as news event data, have low data coverage and irregular updating frequency. We must align these types of data with high coverage and regular frequency such as quotes data for the convenience of downstream factor mining and modeling tasks. Fourthly, different data features have quite different scales in value range and thus some "large" features may dominate "small" features in modeling. Therefore, data standardization methods are used to normalize the range of features. We have to take care of the way to standardize the data in order to reduce information loss.

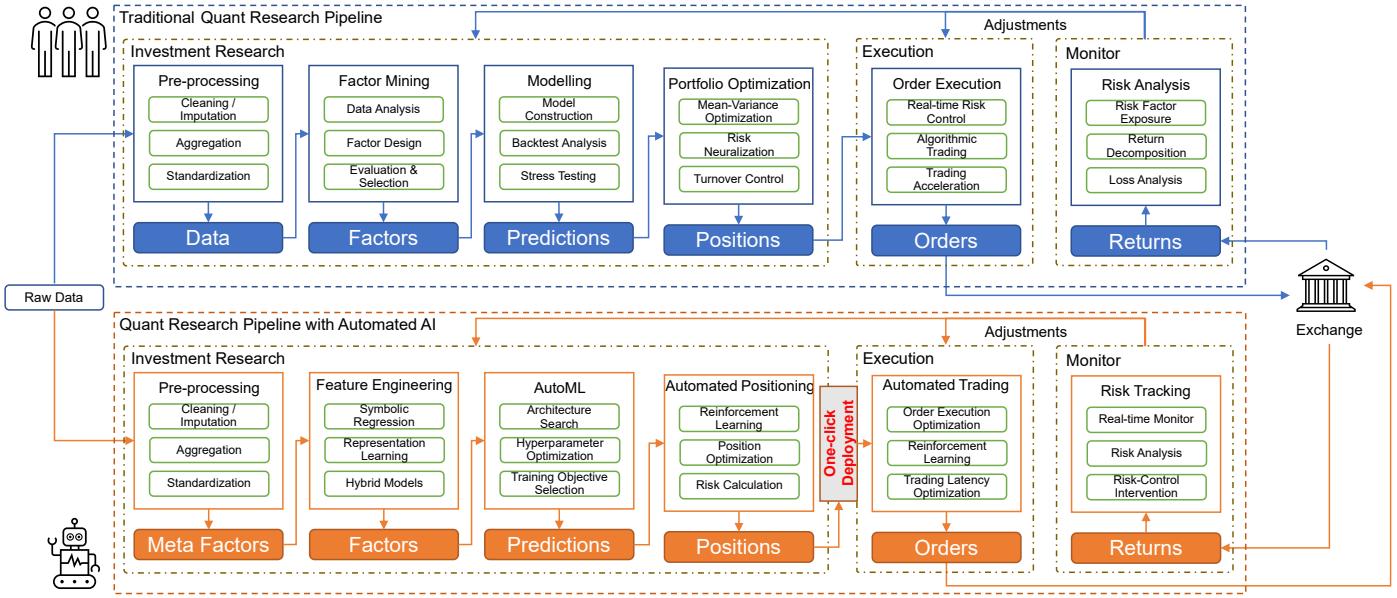


Figure 11: A prototypical workflow of quantitative investment with comparisons between the current quantitative investment system (manual, upper blue part) and AI investment engineering (automated, lower orange part).

- Factor mining is a task of feature engineering [44], which uses financial and economic domain knowledge to design, search, or extract factors (features for downstream modeling) from raw data. Usually, a larger factor value indicates a more significant trading signal. The motivation of factor mining is to find those signals from raw data for market prediction and improve the quality of downstream modeling tasks. Traditionally, financial factors could be represented as either algebraic formulas or rule-based expressions. Let's take a simple stock alpha factor as an example.

$$factor = -\text{ts\_corr}(\text{rank}(close), \text{rank}(volume), 50) \quad (5)$$

  where the ts_corr() function computes the correlation of daily close price and volume along time using the data from the previous 50 trading days, representing how similar the trend of *close* time series and *volume* time series are. The rank() function maps the values in a cross-section to their orders and normalizes them to the range of $[-1, +1]$ evenly according to their descending order, in order to remove the effect of extreme values. This factor prefers to select those stocks when their price and volume move in opposite directions, and the idea behind it is based on the assumption that a price trend can not sustain without the support of volume growth. Traditionally, factor mining is a labor-intensive job. Most quant researchers can only discover a limited number of "good" factors in a year. Different financial institutions have different definitions or criteria for a "good" factor, but most of them consider a few common aspects, such as return, Sharpe ratio, maximum drawdown, turnover rate, and correlation with other factors [41], and moreover, some institutions require the factors must be meaningful, understandable and explainable in economics.
- Modeling is a task to build statistical or machine learning models using factors and to predict market trends, asset price movements, best trading times, or most/least valuable assets. Usually, prediction models are evaluated through back-test experiments which simulate the prediction and trading process using historical data. Choices of models must consider a number of factors, such as prediction accuracy, model explainability, model robustness, and computational complexity, and find the best tradeoff according to the ultimate goal. In particular, we must notice that most statistical or machine learning models are not specifically developed for financial time series, and we have to adjust the application of these models in quant modeling. Firstly, financial time series prediction must avoid using future information, and thus we prefer forward-validation [45] (splitting the time series into training, validation, and test blocks over time) rather than cross-validation in model hyperparameter optimization. Secondly, financial time series are usually significantly nonstationary, far from the *independent and identically distributed* (i.i.d.) assumption required by many machine learning models. Therefore, data transformation is needed to make the data distribution closer to i.i.d. and if possible, look more like a normal distribution. Thirdly, market style moves over time and it results in the shift of financial time-series distribution. Therefore, periodic model retraining is necessary for keeping the model adapted to market style variation.
- Portfolio optimization aims to find the optimal asset allocation to expect high return and low risk simultaneously. While prediction models tell us what or when to buy/sell, portfolio optimization specifies how much to buy/sell. A typical portfolio optimizer attempts to solve a constrained convex quadratic programming problem which is extended from

Markowitz's efficient frontier theory.

$$\begin{aligned} \max_{w_t} \quad & w_t^T r_t \\ \text{subject to} \quad & w_t^T \Sigma w_t \le C_1 \\ & |w_t - w_{t-1}| \le C_2 \\ & 0 \le w_{i,t} \le C_3 \le 1, \text{for } i = 1, 2, \ldots, n \end{aligned}$$

where $r_t = (r_{1,t}, r_{2,t}, \ldots, r_{n,t})^T$ is the returns of n assets (e.g., stocks) at time t, and $w_t = (w_{1,t}, w_{2,t}, \ldots, w_{n,t})^T$ is the corresponding position weights (percentages of capital allocation). C_1, C_2, C_3 are positive constraint bounds. Σ is the volatility matrix of the n asset returns at time t. The target function tries to maximize the portfolio return and control the upper bound of risk and turnover rate (to reduce transaction cost). The key in this optimization problem is how to estimate the volatility matrix Σ whose estimation is usually unstable if historical data is not long enough, and in this case dimension reduction tricks such as regularization and factorization can be helpful to improve estimation robustness.

- Order execution is a task that buys or sells orders with optimal prices and minimal market impact. Usually buying (or selling) a big order at one time will push the price of the target asset in a harmful direction (market impact by this big order), and therefore increase the trading cost. A widely used solution is order splitting, which divides a big order into a number of small orders to reduce market impact. Algorithmic trading provides a series of mathematical tools for order splitting, from the simplest time-weighted average price (TWAP) and volume-weighted average price (VWAP) to the complicated reinforcement learning methods [46] in which optimal order flow is modeled as a (partially observable) Markov decision process.
- Risk analysis is an indispensable task for quant research and quant trading. We must discover and understand every possible risk exposure in order to better control unnecessary and harmful risks in quant research and trading [47]. In the monitor module, risks are measured in real-time and these messages and analysis are sent back to help quant researcher improve their strategies. The most popular risk model in stock trading is the BARRA model [48] which decomposes portfolio volatility into the exposures of a number of predefined risk factors, including style factors (size, growth, liquidity, etc.) and industry factors. However, the BARRA model could explain only about 30% of total volatility, leaving the risk hidden in the remaining 70% part still unknown.

2.1.2. Automated AI Quant Pipeline

The automated pipeline of Quant 4.0 is shown in Figure 11 (orange part), where modules in the pipeline are automated by applying state-of-the-art AI technology. In the following part of this section, we will concentrate on three core modules in the automated pipeline.

1. Automated factor mining (§2.2) applies automated feature engineering techniques to search and evaluate significant financial factors generated from meta factors. We will introduce popular search algorithms and demonstrate how to design the algorithmic workflow.
2. Automated modeling (§2.3) applies AutoML techniques to discover optimal deep learning models, automatically selecting the most appropriate models and the optimal model structures, and tuning the best hyperparameters;
3. One-click deployment (§2.4) builds an automated workflow to deploy trained large models on trading servers with limited computing power. It executes model compression, task scheduling, and model parallelization automatically, saving a lot of labor and time for tedious "dirty" work.

2.2. Automating Factor Mining

Feature engineering for quant refers to the process of extracting financial factors from original data, on which effective pattern recognition is difficult due to their intrinsic noisiness [49, 50, 51]. Traditionally, financial factors with significant "alpha" are explored and developed by quant researchers manually, they rely on professional domain expertise and comprehensive knowledge of financial markets. Although some financial institutions started using random search or generic programming algorithms, these techniques are mainly used as small-scale auxiliary tools to help improve the productivity of quant researchers. In Quant 4.0, We propose to automate the factor mining process by formulating feature engineering as a search problem and utilize corresponding algorithms to generate factors with satisfactory backtest performance at scale. In particular, according to their expression form, we classify factors as 1) symbolic [52] factors which are symbolic equations or symbolic rules, and 2) machine learning factors which are expressed by neural networks, and we will elaborate on the details in the following part of §2.2.

2.2.1. Symbolic Factors

Symbolic factor mining can be regarded as a special case of symbolic regression [56]. Traditional symbolic regression algorithms usually generate a large number of symbolic expressions from given operands and operators and select the symbolic expressions that maximize the predefined objective function. Figure 12 shows a framework for automated symbolic factor mining, which consists of four core parts: operand space, operator space, search algorithm, and evaluation criterion.

- Operand Space defines which meta factors could be used for factor mining. Meta factors are fundamental components for factor construction. Typical meta factors include basic price and volume information, sector categorizations, basic features extracted from limit/order books, common technical indices, basic statistics from financial analysts, important signals from financial reports, announcements and other research reports from public companies, sentiment signals from investor emotions [57, 58], etc.

- Operator space defines which operators could be used in the factor mining process. For example, in cross-sectional stock selection, the operators could be classified as main operators for constructing symbolic factors and post-processing

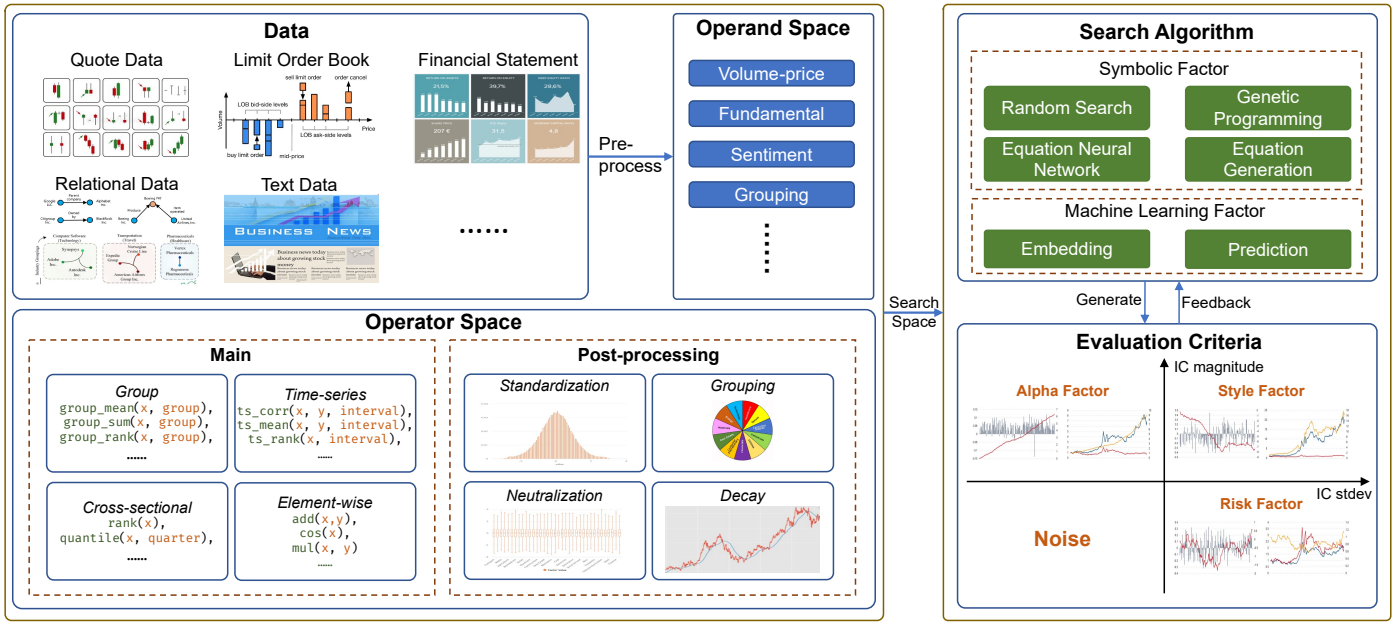


Figure 12: An example factor mining pipeline. The search space is defined by operators and meta-factors, where meta-factors are extracted from raw data in various forms. The search space is explored by search algorithms that are discrete or continuous. The evaluation module provides feedback to search algorithms based on certain criteria that serve as guidance for the next search iteration. Part of this figure is cited from [53, 54, 55].

operators for standardizing the factors for different trading environments. Main operators could be classified further as element-wise operators such as $\sqrt{()}$ and log(), time-series operators such as *ts_rank*() and *ts_mean*() which compute the rank order and mean along each stock respectively, cross-sectional operators such as *rank*() and *quantile*() which compute the rank and quantile along the cross-section at a specific trading time, and group operators such as *group_rank*() which compute rank order in each group (e.g., industry or sector) respectively. Post-processing operators are used to "fine-tune" the generated factors. Typical post-processing operators are standardization operators such as winsorization for outlier clipping [43] and normalization for unifying data range, neutralization operators for risk balancing, grouping operators for restricting the universe of stock selection, and decay operators for controlling turnover rate so as to reduce transaction cost.

- Search algorithms aim to search and find effective or qualified factors as efficiently as possible. A simple way to generate new factors is the Monte Carlo (MC) algorithm which randomly picks the elements in the operand and operator spaces and generates a symbolic expression tree recursively. Unfortunately, the search time may grow exponentially with the length and complexity of the generated formula, and push us to consider more efficient alternatives. The first option is Markov-chain Monte Carlo (MCMC) algorithm [59], which generates factors in sampling with importance way from a posterior distribution [60], and thus it is more efficient than MC. The second option is genetic programming [61], which is a special evolutionary algorithm for sampling and optimizing tree-type data. The third option is about gradient-based methods such as neural networks, which approximate the discrete symbolic formulas with continuous nonlinear functions and search along the gradient direction, significantly more efficient than random search.

- Evaluation criteria measure the quality of factors found by search algorithms. The performance of the generated factors is evaluated using backtest experiments. Typical evaluation criteria include information coefficient (IC), information ratio based on information coefficient (ICIR), as well as annualized return, maximum drawdown, Sharpe ratio, and turnover rate. In addition, it is very important to keep information diverse among factors by filtering out redundant factors which highly correlated with other factors.

Due to their importance in factor mining, we introduce about two types of search algorithms in detail.

- Genetic programming (GP) [64] (Figure 13), an extension of genetic algorithm [65], is a metaheuristic algorithm for searching tree-structured symbolic factor expressions. In an algorithmic loop, GP starts from a number of initial factors, and it uses an evolutionary mechanism to produce the next generation of factors, aiming to improve factor performance measured by the fitness function. There are two types of evolutionary mechanism in GP: *mutation* which replace a node randomly with another operand of an operator, and *crossover* where two trees swap their subtrees randomly. In each iteration, all factors are evaluated using IC or alternatives and only the best-performing factors are kept. This process is repeated until convergence.
- Neural symbolic regression utilizes gradient information to accelerate the search process. Neural networks are used to learn a continuous and nonlinear function to approximate those discrete symbolic expressions and use this function to

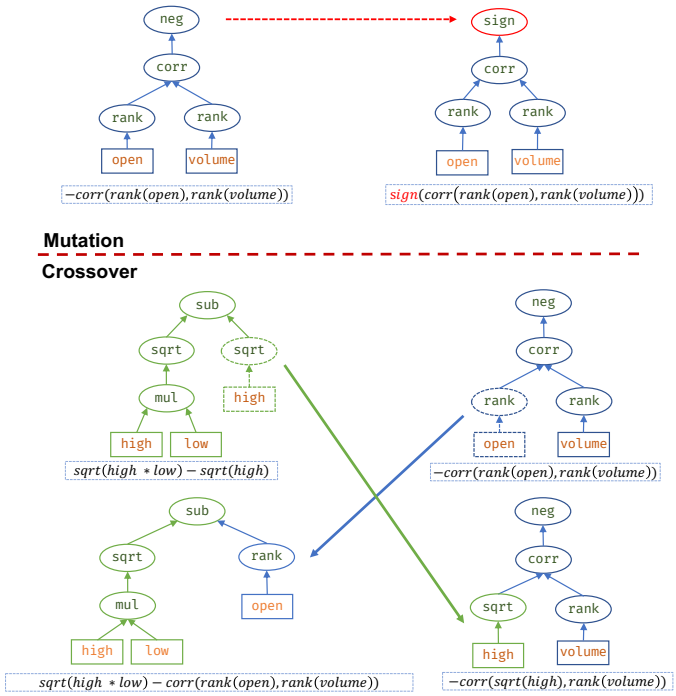
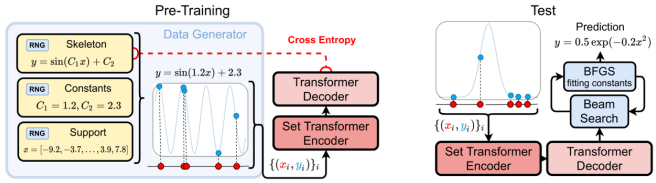
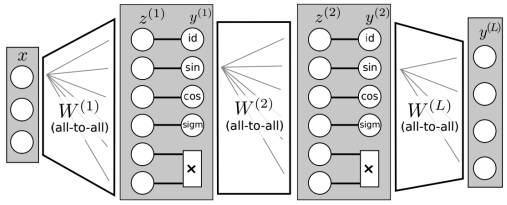


Figure 13: Illustrations of two evolutionary mechanism used in genetic programming.



(a) Neural symbolic regression in a sequence generation manner [62].



(b) Neural symbolic regression that directly uses the neural network as symbolic expressions [63].

Figure 14: Illustrations of neural symbolic regression algorithms.

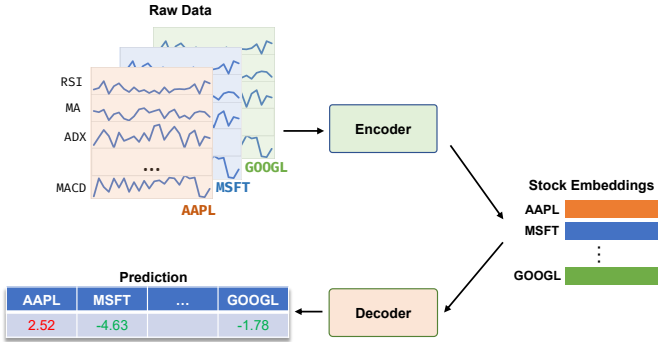


Figure 15: Illustration of the encoder-decoder architecture used in stock prediction. Both embeddings and predictions can be used as factors.

generate new formulas. We introduce two works about neural symbolic regression. The first paper [62] (Figure 14a) builds a transformer generative model from a number of existing symbolic expressions. In the training stage, a special transformer model (called set transformer [66]) encodes the formulas in the training set into embedded vectors, and they are delivered to a transformer decoder to update symbolic expressions in an autoregressive way using beam search, and this process is repeated until convergence. The generative model is trained by minimizing the cross-entropy loss between the generated expression and the original one. In the test stage, the trained model is used to generate new symbolic expressions. The second paper [63] (Figure 14b), designed a new neural network specifically for expressing symbolic formulas. In particular, the activation functions in this network are replaced with symbolic operators such as $sin(\cdot)$ and $\sqrt{(\cdot)}$. This special neural network has the flexibility to generate almost all formula expressions need to use in factor mining.

2.2.2. *Machine Learning Factors*

Symbolic factors have their advantages in simplicity and understandability, and are thus widely used in practice. However, their representation ability is limited by the richness of operands and operators. Machine learning factors, on the other hand, have more flexibility in representation to fit more complicated nonlinear relationships [67], and thus they have the chance to perform better in market prediction. In particular, mining machine learning factors [68, 69, 70, 71] is a process to fit neural networks, where gradients provide the optimal direction for fast search of solutions. As shown in Figure 15, most deep neural networks for stock prediction follows the encoder-decoder architecture [72], where the encoder maps meta factors to a latent vector representation and the decoder transforms this embedding to some outcome such as future return [73]. In fact, not only the final outcome, but also the embedding itself could be used as a (high-dimensional) machine learning factor [74], and further applied to various downstream tasks.

Machine learning factors have some limitations as well. Firstly, they are usually hard to interpret and understand because of the black-box nature of machine learning. Secondly, gradient search used by neural networks may be stuck at some local optima and result in model instability problems. Finally, neural networks may suffer more serious overfitting due to their flexibility, and this situation gets worse in quant because data are extremely noisy.

2.3. *Automated Modeling*

The automation of statistical machine learning such as SVM, decision tree and boosting has been extensively researched. A simple and direct automation method is the brute-force enumeration of all possible configurations, each including the choice of machine learning algorithms and the corresponding hyperparameters (Figure 17).

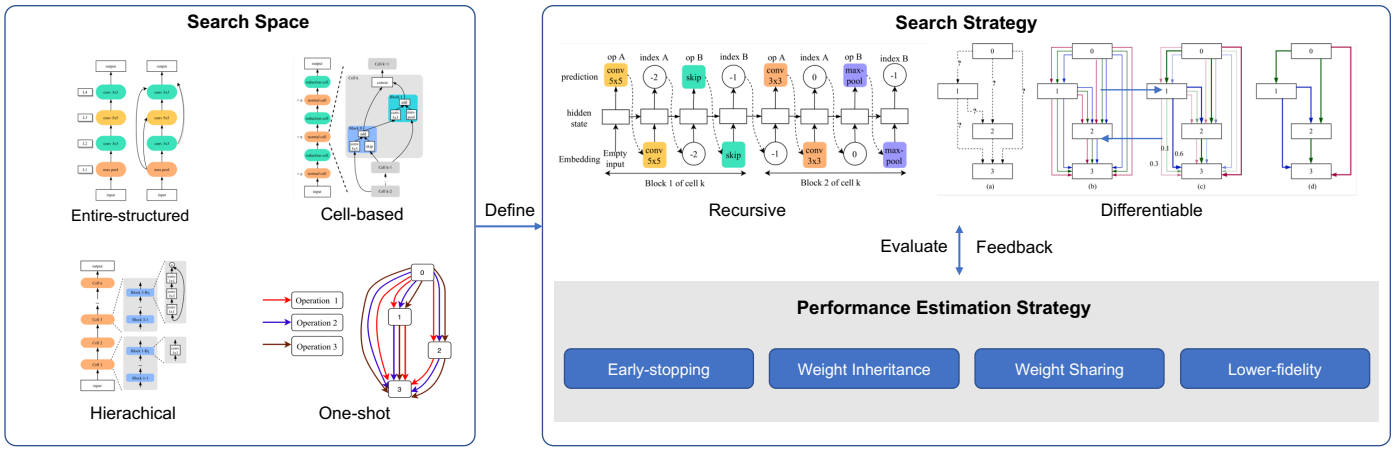


Figure 16: The automated modeling pipeline for architecture search. The structure of this figure is adapted from [75]. Illustrations of search spaces and search strategies are cited from [42, 76].

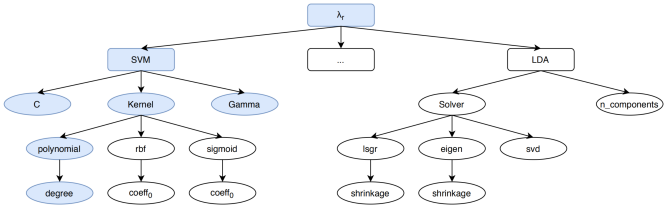


Figure 17: Illustration of early-stage automated machine learning based on brute-force search of algorithms and hyperparameters. Figure is cited from [77].

In this article, we focus on the state-of-the-art deep learning automation problem, which is more complex due to the end-to-end property and network architecture issue in modeling. The configuration of a deep learning model consists of three parts: architecture, hyperparameters, and objectives, and they jointly determine the final performance of the models. Traditionally, these configurations are tuned manually. In Quant 4.0, they are searched and optimized using various AutoML algorithms. A standard AutoML system needs to answer the following three questions: what to search (i.e., search space §2.3.1), how to search (i.e., search algorithm §2.3.2), and why to search (i.e., performance evaluation §2.3.3).

2.3.1. Search Space

Search space is designed for the three configuration settings that need to be optimized in an automatic way.

- Architecture configures a network structure. For example, the architecture of a multi-layer perceptron is specified by the number of hidden layers and the number of neurons at each layer. The architecture of a convolution neural network needs to consider more configurations such as the number of convolution kernels as well as their strides and receptive fields. The architecture of large-scale models such as Transformer is composed of a number of predefined blocks (e.g. self-attention blocks, residual blocks) linked together. As discussed above, architecture is complex and may have a hierarchical structure at different scales. Accordingly, the search space can be defined at various granularities, ranging from low-level operators such as convolutions and attentions to high-level modules such as LSTM cells. Early search algorithms run on the finest granularity and optimize the low-level structure of the neural network [78, 79]. Such a search process is flexible in network structure but inefficient in incorporating prior knowledge and abstraction. One solution is to assume a hierarchical structure in network architecture. Specifically, at a high level, the network is designed to be a graph of cells (a.k.a. blocks/motifs [42, 80]), each of which is a subnetwork. Many cells share the same internal structure at a low level in order to reduce the computational cost. Cell-based search algorithms [81, 82] need to find both high-level structures between cells as well as low-level structures within the cells.
- Hyperparameter controls the overall training process. For example, learning rates determines the step size moving towards a minimum of a loss function. A smaller learning rate is more accurate in solution but slower in convergence. The batch size determines the number of samples involved in a batch for gradient estimation, which also has an influence on training efficiency and stability. The search space for hyperparameters is simpler than that for architecture since most hyperparameters are continuous (e.g., learning rate) or approximately continuous values (e.g., batch size).
- Objective specifies the loss functions and labels used for training models. The loss function is the key component of machine learning models since it provides a goal towards which a model should be trained. Besides classic loss functions such as mean square loss and cross-entropy loss, new loss functions specifically designed for quant tasks can be also selected. Labels define the "ground-truth" target the model aims to fit. For example, either price raise/fall or future returns in different holding time windows can be considered in the search space.

2.3.2. Search Algorithm

Given the search space, we could use search algorithms to find the best model configuration. Table 2 lists various types of search algorithms and their corresponding tasks: network architecture search (NAS) [75], hyperparameter optimization (HPO) [83] and training objective selection (TOS).

Table 2: Search algorithms and their applicable search targets.

Algorithm \ Target	NAS	HPO	TOS	References
Grid/Random Search		✓	✓	[84, 85]
Evolutionary Algorithm	✓	✓	✓	[86, 87, 88, 89, 90]
Reinforcement Learning	✓	✓		[91, 81, 82, 92]
Bayesian Optimization	✓	✓		[93, 94, 95, 96]
Gradient-based Method	✓	✓		[97, 98, 99, 100, 101]

- Grid search is a brute-force algorithm that searches on a grid of configurations and evaluates all of them. It is a good choice when the search space is small due to ease of implementation and parallelization [84]. However, most NAS and HPO problems in deep learning have extremely large search spaces and grid search can not scale well for them. Moreover, grid search is used more popular in HPO and TOS than in NAS whose search space is difficult for grid layout except enumerating all possibilities.

- Random search generates a number of candidate configurations using some stochastic sampling mechanisms, such as Monte Carlo or MCMC. It is very straightforward to implement and parallelize (mainly for independent sampling mechanisms such as Monte Carlo sampling or importance sampling). Random search is very flexible and can be used for NAS, HPO, and TOS. Although random search is usually faster than grid search [84], it is still difficult to handle high-dimensional search space as the number of potential configurations grows exponentially with the number of hyperparameters.

- Evolutionary algorithm is an extension of random search. It utilizes evolution mechanisms to improve model configurations iteratively. It encodes the architecture of network networks as a population and performs the evolution steps on them to improve the model iteratively. Specifically, the models are first encoded according to their underlying computation graph. Then, a set of pre-defined evolutionary operators are applied to the encoded models, including architectural modifications such as inserting or deleting several operations and adding skipping connections, as well as hyperparameter-related operations such as learning rate adjustment. At each iteration, the best-performing models are selected via tournaments and combined via mutation and crossover operations to form the next generation. Evolutionary algorithms inherently support weight inheriting among generations, which helps accelerate the convergence in training and increase the searching efficiency.

- Reinforcement learning models the architecture search problem as a Markov decision process. In each step, an RNN controller chooses an action to sample a new architecture and the corresponding deep neural network model is trained. Then the performance of the model evaluated on the validation set is used as a reward which is forwarded to compute the policy gradient and update the RNN controller. This loop is iterated until convergence. The reinforcement learning framework is very universal for most optimization problems and it could be used for NAS, HPO, and TOS.

- Bayesian optimization explores the search space more efficiently by leveraging surrogate models to approximate the objective function that couldn't be expressed explicitly. Specifically, for a black-box objective function for HPO, Bayesian optimization initializes a prior distribution using a surrogate function such as Gaussian process or tree-structured Parzen estimator. Then it samples new data points from the prior distribution (with importance) and calculates their values using the underlying objective function. Given these new samples and prior, the posterior function can be calculated and is used as an updated surrogate function to replace the original prior function. This process is repeated until the optimal solution is found. Traditionally, Bayesian optimization is used for continuous search tasks such as HPO, but recent works have extended it to NAS tasks as well.

- Gradient-based methods is very efficient when the gradient of the objective function exists. However, for NAS, the search space is discrete and couldn't define a gradient directly. One solution is to "soften" the architecture and define an over-parameterized "super-architecture" which covers all possible candidates and is differentiable. A typical gradient-based NAS method is DARTS [97] which constructs an over-parameterized network where all types of candidate operations are present on the computation graph. The resulting value is the weighted sum of the results of all the operations, where the weights are the softmax values of a parameterized probability vector. Both model parameters and architectural parameters are trained via a bi-level optimization problem. In the inference process, the architecture and hyperparameters with the highest probability are selected. DARTS is substantially faster than random research and reinforcement learning in NAS and HPO tasks.

2.3.3. Accelerating Evaluation

The computational cost of automated model search comes from two parts: search algorithm and model evaluation, and the latter is usually the bottleneck of the computation because it is very time-consuming to train a deep neural network until convergence under a given configuration. Several methods are introduced in previous research to address this issue. Firstly, the training process of neural networks can be early-stopped before convergence to reduce the computational time for evaluation [81]. Secondly, the model can select fewer samples to accelerate the training process [94]. Thirdly, warm-start model training can be used to leverage the information from existing selected models [102] or inherit the information from an over-parameterized "parent" model [103, 97, 98] to accelerate the search loop.

2.4. Automated One-click Deployment

Model deployment is the task of transferring the developed model from offline research to online trading. It is not only simply transferring code and data, but also synchronizing data and factor dependency, adapting trading server and system, debugging model inference, testing computing latency, etc. In the following part, we focus on one important problem in model deployment: how to accelerate deep learning inference for high-frequency trading and algorithmic trading scenarios. We propose an automated one-click deployment solution utilizing techniques such as model compilation [104] and model compression [105, 106] to realize inference acceleration [104, 105, 106]. The former makes the inference faster without changing the model itself, and the latter seeks smaller and lighter alternative models to save inference time.

2.4.1. Acceleration by Model Compilation

At the development stage, deep learning models' functionality is the top priority for the underlying framework which implements the computations. Hence, at this stage, the framework strictly maps all the operations to the computation graph. However, such direct mapping introduces large room for optimization at the deployment stage where the computations are fixed. Therefore, the model's computation can be greatly simplified and adapted to hardware features without hurting its original semantics. Such optimization is one of the major topics in deep learning compilers [107, 108, 109], which can be categorized as *front-end optimizations* and *back-end optimizations*, which work on high-level and low-level intermediate representations (IRs) for deep learning models respectively. Following the summary in [104], we will briefly introduce relevant optimization techniques.

Front-end optimization, as illustrated in Figure 18a, focuses on simplifying the structure of the computation graphs. For example, algebraic simplification techniques such as constant folding [110] and strength reduction [111] convert expensive operations into cheaper ones via transformation or merging. Common subexpression extraction (CSE) [112] techniques identify repeated nodes in the computation graph and merge them into one node to avoid duplicate computations.

Back-end optimization, as illustrated in Figure 18b, is performed with an emphasis on the features of hardware architectures, such as locality and memory latency. For example, pre-fetching techniques [113] load data from main memory to GPU before they are needed, and speed up fetch operations. Loop-based optimizations [114] reorder, fuse and unroll operations inside loops to enhance locality among neighboring instructions. Memory latency hiding [115] techniques aim to increase instruction throughput so as to mitigate the problem of high latency in accessing memory. Parallelization techniques such as loop splitting [116], automatic vectorization [117] and loop skewing [118, 119], can also be applied to maximize the parallelism provided by modern processors.

2.4.2. Acceleration by Model Compression

Model compression aims to reduce model size for inference acceleration while minimizing drops in performance. In this way, the compressed model can be regarded as an approximation of the original model. Generally speaking, model compression can be performed at both micro- and macro-level, where the former focuses on the precision of individual model parameters and the latter focuses on simplifying the overall model structure.

At the micro-level, pruning and quantization techniques can be applied to reduce both the number of parameters and the bit size of the individual parameters. Model pruning [120], as shown in Figure 19a, removes unimportant connections and neurons in neural networks that have little influence on the activation of the neural. The identification of candidate parameters is usually based on their weights, where those with smaller weights are considered for pruning. Model quantization [121], as shown in Figure 19b, converts the parameters from floating-point numbers to low-bit representations. Specifically, a codebook is constructed to store the approximated values of the original parameters according to the distribution of all parameter values. The parameters are then quantized according to the codebook and thus the bit size for the parameters is reduced. Due to the inevitable performance drop induced by precision reduction, the compressed model is usually re-trained to approach its original performance.

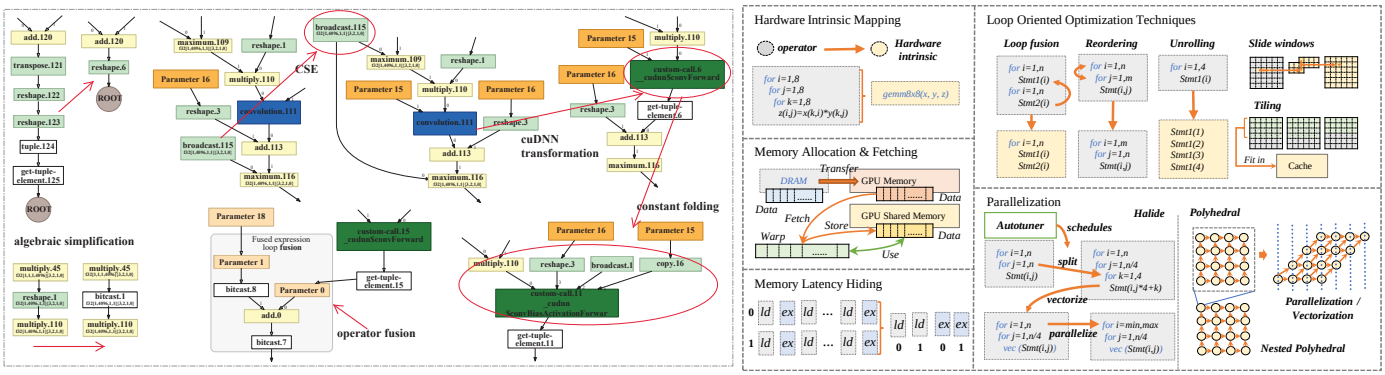
At the macro-level, the model can be significantly compressed into a smaller model with simpler architectures via knowledge distillation [122] and low-rank factorization [123, 124]. Knowledge distillation (Figure 19c) compresses a model by transferring useful knowledge from the original large model (called the teacher model) to a small and simple model (called the student model) with minimal knowledge loss. Low-rank factorization techniques (Figure 19d) assume the sparsity of model parameters and then split the parameter matrix of the original neural networks into products of low-rank matrices [123], and thus reduce the model complexity.

3. Explainable AI for Quant 4.0

XAI [126, 127], as an attractive research direction for decades, is critical to the trustworthiness and robustness of AI models. In the case of quant, improvement in the explainability of AI can make the decision process more transparent and easy to analyze, providing useful insights to researchers and investors and discovering potential risk exposures. In this section, we will discuss how to leverage XAI in Quant 4.0. §3.1 introduces common XAI techniques and §3.2 connects these techniques to real quant scenarios.

3.1. Overview of Explainable AI

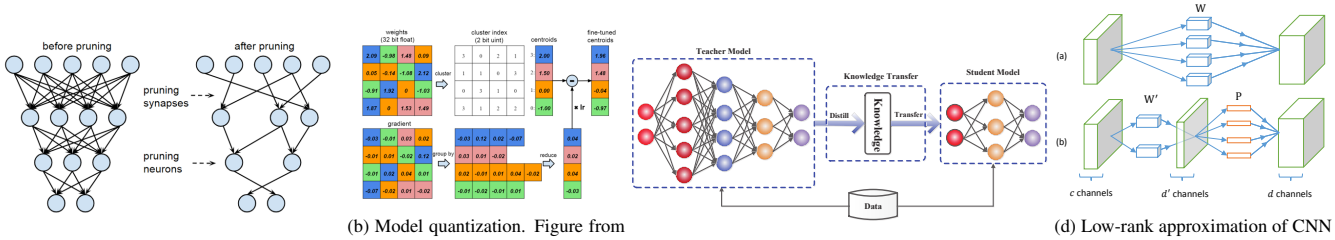
XAI is an emerging interdisciplinary research area covering machine/deep/reinforcement learning, statistics, game theory and visualization. Here we focus on two types of XAI: model-intrinsic explanation [128] and model-agnostic explanation [129].



(a) Front-end optimization

(b) Back-end optimization

Figure 18: Deep learning compiler optimization techniques. Figure from [104]



(a) Model pruning. Figure from [120]

(b) Model quantization. Figure from [121].

(c) Knowledge distillation. Figure from [125].

(d) Low-rank approximation of CNN models. Figure from [123].

Figure 19: Model compression techniques

3.1.1. Model-intrinsic Explanation in XAI

Risk control and management is the top priority of financial industry. When AI models are deployed in real-world applications, their decision process is usually required to be transparent by regulatory authorities for the safety of transactions. Moreover, model-intrinsic explainability is the requirement of many large financial institutions such as banks and insurance companies.

A machine learning model is intrinsically explainable if its internal structure or mechanisms can be easily explained. Some machine learning algorithms such as linear models and decision trees are inherently explainable, as many other algorithms such as deep neural networks and kernel learning methods (SVM, Gaussian process, etc.) are black boxes with poor explainabilities. Figure 20 illustrates many popular machine learning methods arranging along their general performance and explainability. We can see The increase in model-intrinsic explainability usually leads to a decrease in the model's prediction performance, and therefore the selection of machine learning algorithms is essentially a trade-off between explainability and performance. We briefly introduce a few typical machine learning methods in terms of explainability and predictive performances and discuss their applicable scenarios.

- Linear Models, such as linear regression, logistic regression, linear discriminant analysis, linear SVM and addition model, is a family of methods where features or transformation of a group of features are in an additive form and thus the performance of final prediction can be easily deposed to the effects from individual features or feature groups. Therefore, linear models are intrinsically understandable and explainable. For example, linear regression explicitly encodes the importance of each feature in their corresponding regression coefficients (assuming every feature is normalized to eliminate the effect from scales and units). Although linear models are easy to explain, they are suffering the poor prediction performance since they couldn't encode complicated nonlinear relationships between prediction outputs and features.
- Rule-based Learning is another type of easy-to-explain methods. Different from a linear model which fits a linear decision boundary, a rule-based learning method fits a stepwise decision boundary characterized by decision rules combining a number of logical expressions. Examples of rule-based learning include decision tree [133] and symbolic regression, as well as ensemble models such as random forest [134] and boosting [135, 136, 137]. Rule-based learning models are intrinsically explainable decision rules that are close to the logical thinking process of human beings. However, to better fit the training data and improve prediction performance, the decision rule is usually complicated, and it reduces the explainability and increases the risk of overfitting.
- Ensemble Learning combines multiple machine learning algorithms to achieve better decision performance than single models. Typical examples of ensemble learning include random forest and boosted trees that combine multiple tree models and make predictions based on the aggregation of individual decisions. Although there is controversy, in this article, we classify mixtures of experts (MoE) [138] as an ensemble method as well. MoE combines multiple expert networks in parallel in a layer and decides which expert (or experts) par-

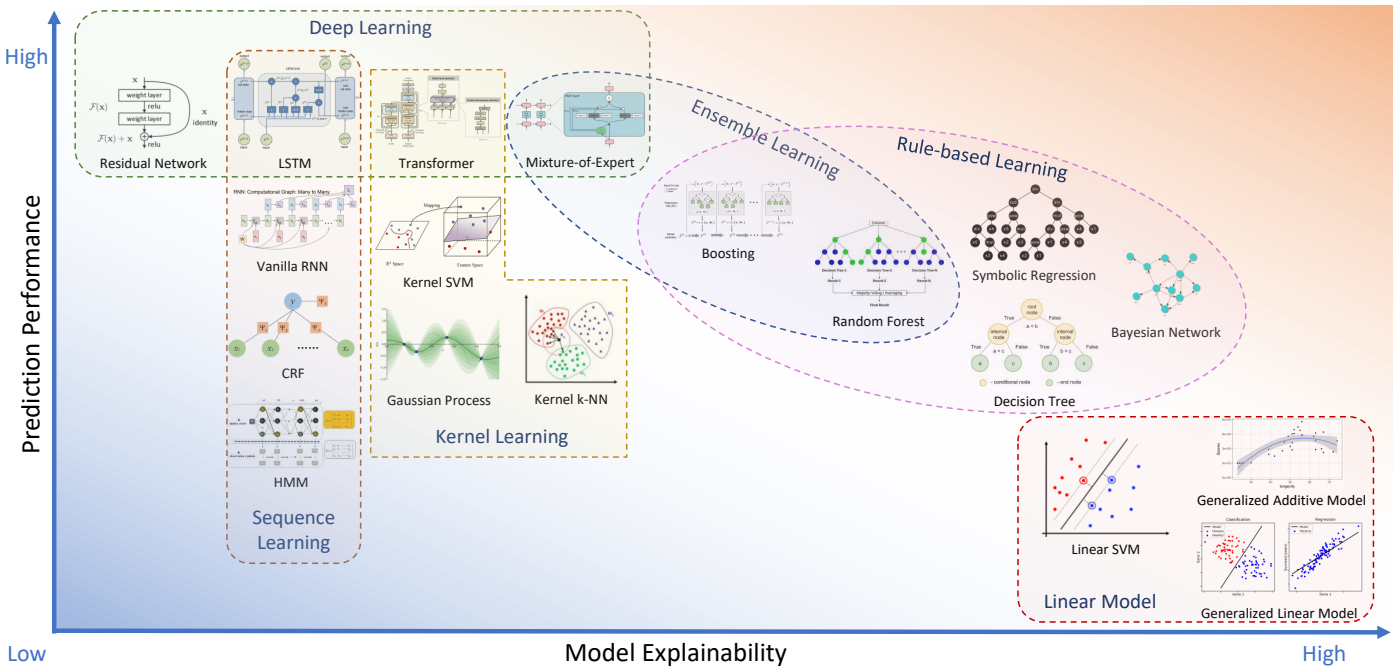


Figure 20: Comparison of popular machine learning algorithms according to prediction performance and model explainability. Part of this figure is cited from [130, 131, 132].

ticipates in the decision of a specific data point via a gating mechanism. Compared with other machine learning methods, ensemble learning provides high-level explainability by demonstrating the relative importance of single models or experts.

- Kernel Learning, also known as kernel method or kernel machine, is a family of nonparametric learning methods which make predictions by computing the similarity between samples. The similarity is characterized by a kernel function, which is a special inner product defined in a high-dimensional Hilbert space where original data samples are mapped [139]. For example, kernel SVM [140] transforms original positive/negative samples into another space where they can be easily separated using a linear decision boundary. In principle, kernel functions can be of arbitrary forms that satisfy the Mercer's condition [141], and they determine the nonlinear relationships between inputs and outputs. Moreover, the idea of kernel functions is extended to the self-attention mechanism [142] used in neural networks such as Transformer [143]. Traditionally, we think the kernel trick improves the performance of models but weakens their explainability. From another point of view, however, the definition of kernel itself encodes the prior insight of users and could help understand the model.
- Sequence Learning refers to a family of machine learning methods that work with sequential data such as time series or sentences. They are widely used in speech recognition, language understanding, DNA sequence analysis, and stock price prediction. Sequence learning methods characterize the underlying structure hidden in sequential data and discover implicit patterns. For example, hidden Markov model (HMM) [144] assumes that the underlying structure is a homogeneous Markov chain determined by a transition matrix (or transition kernel for continuous state space) and assumes the observed sequence is randomly generated from this chain through emission probabilities. The transition probabilities and emission probabilities are estimated during model training. Although HMM is generally a black-box model, its transition probability matrix provides some insight into the autoregressive structure in prediction. Conditional random field (CRF) [145] extends the first-order Markov assumption of HMM and characterizes longer-range time dependency using graphical model for probability modeling, and this extra flexibility usually brings better prediction performance for CRF. Recurrent neural networks (RNN) such as LSTM [146] and GRU [147] exhibit better performance in sequence prediction, but it is harder to explain their internal mechanism.
- Deep Learning usually has superior prediction performance [148] but its shortage in explainability is clear. Some special operators in deep neural networks such as convolution and attention provide partial and local explanations about their mechanisms. For example, the self-attention layer in a Transformer [143] characterizes the relative importance of each position in a sequence with respect to other positions.

The model-intrinsic explainability for machine learning is always a contradiction with its prediction capability. However, before the appearance of a brand-new machine learning model satisfying both high prediction accuracy and high explainability, we could rebuild and improve current machine learning methods. We could either start from an explainable model such as a linear or rule-based model and improve its prediction performance by incorporating more local nonlinear structures with

better predictive power. For example, starting from a decision tree model, we could replace the decision rule in each leaf node with a neural network [149], thus improving the model's flexibility. As another example, starting from a deep neural network, we could also improve its partial explainability by incorporating a special layer (e.g. self-attention layer [150]) identifying which important feature interacts more frequently.

3.1.2. Model-agnostic Explanation in XAI

To address the contradiction between performance and explainability, a suboptimal solution is to weaken the requirement and shift from model-intrinsic explanation to model-agnostic explanation. Based on the scope of explanations, model-agnostic XAI can be categorized as global methods (explanation applied to all samples) [151] and local methods (explanation applied to part of samples) [154].

Global methods explain the characteristics of features with respect to all samples in the dataset. These characteristics include the importance of features, the importance of feature sets, the interactive effect of features, and other high-order effects of features. There are various types of methods for estimating global feature importance.

- Feature marginalization methods estimate the importance of a specific feature or a specific feature set by marginalizing all other features in the model. Specifically, the importance of the first feature is calculated by integrating out all other features, and similarly, we calculate the second feature, the third feature, etc. For example, partial dependence plot (PDP) [135] computes the marginalized model function w.r.t. the features of interest, and it visualizes the corresponding feature importance. Accumulated local effect (ALE) plot [155] provides an unbiased estimate of marginal effects using conditional distributions that consider the correlation between features.
- Feature leave-one-out methods evaluate the importance of concerned features by comparing the difference in the model performance before and after leaving these features out of data. Specifically, the feature of a model can be left out by shuffling its values in samples [134, 156]. Leave-one-out methods can be extended to evaluate the interactive effects of features as well. For example, based on the partial dependence function, H-statistic [157] is proposed to test for the interaction between features. Other alternative techniques for evaluating and visualizing feature interactions have also been proposed in [158, 159]. Besides, we can also perform functional decomposition [158] on the original model to explore interactions between all possible sets of features.
- Feature surrogate methods interpret the model by learning a globally explainable surrogate model [160] that approximates the original model. The surrogate model is trained under the supervision of the original model using a dataset where the inputs remain the same while the outputs are produced by the original model.

Figure 21 summarizes popular global explanation methods, some of which have been introduced above. Moreover, these methods can be categorized as data-driven and model-driven, where the former methods treat the models as black boxes and query the model for explanations, and the latter methods treat models as white boxes and provide explanations using internal information such as gradients.

Local methods explain the feature importance at the sample level, i.e., how important a feature is for specific samples. Similar to the partial dependence plot, we can draw the individual conditional expectation (ICE) plot [161] for each sample that illustrates the effect of the concerned features when the values of other features are fixed at specific values. To interpret a black-box machine learning model at a specific sample, we can learn a surrogate model in the vicinity of a data sample to explain the original model locally. Typical examples include LIME [152] and Anchors [153]. In LIME (Figure 22), LASSO regression [162] is used as the surrogate model to fit the samples randomly generated by perturbing the original samples around the specified one. In this way, the selected samples by this local LASSO model contribute most to the fitness of the specified sample. Analogous to LIME, anchors are explanations on individual samples in the form of IF-THEN rules [153] that involve features. These anchors are generated by adding features to the rules iteratively using beam search. At each iteration, the candidate with the highest estimated precision is kept and used as the seed for the next iteration. Furthermore, we can also use feature importance to provide local explanations. For example, SHAP [163] proposes a unified framework for computing the importance of each feature in a sample using Shapley values [164]. Since the exact computation of SHAP values is computationally expensive, SHAP also proposes several approximation methods to accelerate the estimation. Gradient information can also be applied [165] in differentiable models such as deep neural networks to illustrate the importance of input features to the model's prediction. As shown in Figure 23, local explanations such as LIME and SHAP can also serve as global explanations using aggregations. Such global explanations can be attained by aggregating explanations for all samples to form explanations at the dataset level. For example, by computing the average importance of each feature across the whole dataset, we can identify the important features that make great contributions to model prediction for most samples in the dataset.

3.2. Explainable AI for Quant

In this part, we take stock alpha investment as an example. The input of deep learning models has three directions: stock, time, and factors (inner circle in Figure 24). Based on these directions, various tasks of interest can be formed to provide practical insights (outer circle in Figure 24). These tasks can be further instantiated as specific examples to provide explanations for realistic problems in investment.

3.2.1. Explanation on Stock

Explanations can be provided for individual stocks to illustrate their sensitivity to different factors at different times and their relationships with each other. Some tasks for explanations on stock are provided as follows:

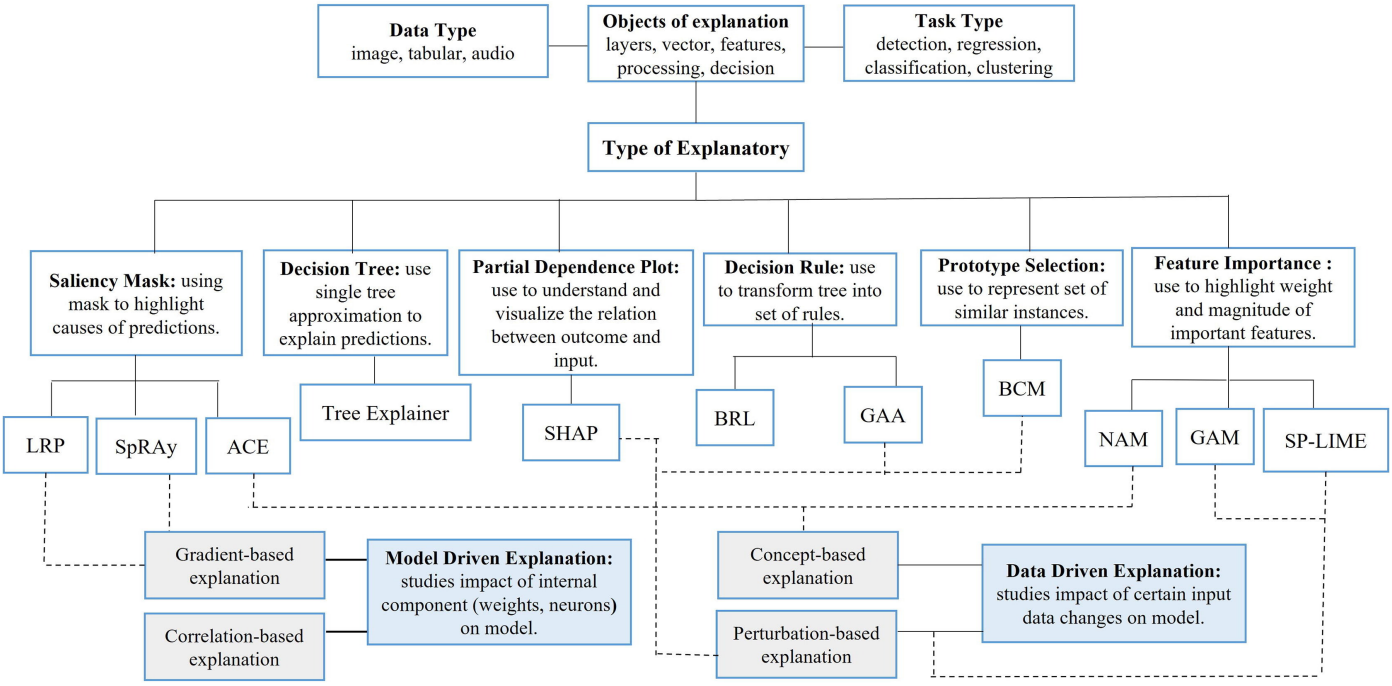


Figure 21: Taxonomy of global explanation methods. Figure cited from [151].

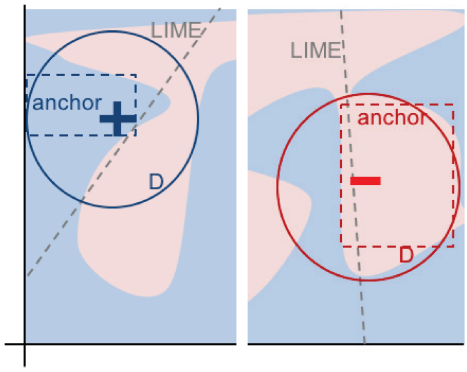


Figure 22: Illustration of LIME [152] and Anchors [153]. Figure cited from [153].

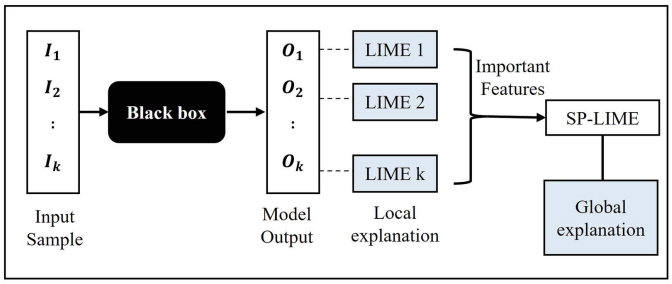
Stock similarity. Stocks are ubiquitously correlated with each other from many aspects (as shown in Figure 25a), and correlated stocks are expected to share common properties. In this sense, utilizing the relationships between financial instruments can bring advantages to our analysis and prediction over traditional methods that treat stocks individually. We can also better understand what the model has learned by analyzing the similarities between stock embeddings. However, the challenge lies in determining an appropriate similarity metric, which is expected to have enough flexibility and effectiveness. This problem is related to metric learning [168, 169] and graph structure learning [170]. A good similarity metric between stock embeddings is needed. Based on this metric, a graph structure can be constructed by computing an adjusted adjacency matrix based on pairwise similarity.

Lead-lag effect. In a lead-lag effect [171], the trend of a stock is followed by some other stocks with a lag in time. Following lead-lag effects, investors can observe the trends (i.e. price going up/down) of the leading stocks and take corresponding positions on the lagging stocks before the same trends duplicate. In this way, investors can profit by precisely identifying lead-lag effects on the market [172, 173]. However, it is not a trivial job to identify lead-lag effects, since duplicated trends appear frequently in financial markets but only few of them are actually caused by lead-lag effects. Strict identification of lead-lag effects needs to be conducted via causal inference that requires counterfactual explanations [174]: What will the trends of the lagging stocks be if the lead stock didn't go in this way? Nevertheless, counterfactual reasoning is usually infeasible in real-world financial markets.

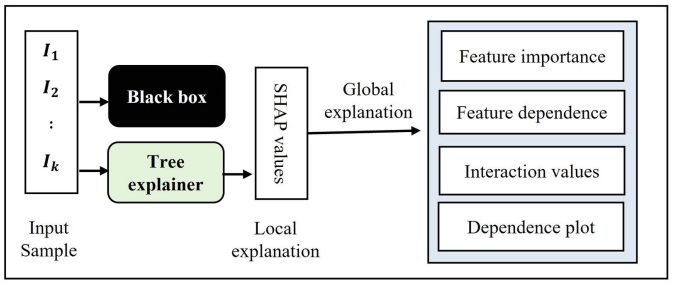
Sector trends. Sectors are stock categorizations defined according to certain criteria such as industry and market capitalization. Stocks in the same sector share certain common properties, and the trend of individual stocks can be influenced by their sectors. Therefore, it is important to identify sectors' contributions to individual stocks. To do this, we can treat stocks' membership to different sectors as categorical features and compute the importance of these factors via feature importance algorithms. Besides, investors can also gain insight into the sensitivity of sectors to different types of features by evaluating feature interactions between sector memberships and other ordinary features.

3.2.2. *Explanation on Time*

Explanation can be computed on individual time points to illustrate the situation of stocks and factors at that cross-section,



(a) SP-LIME (global explanation) from LIME (local explanation)



(b) Combining different local explanations and from SHAP values for Global explanation

Figure 23: Illustration of how local explanations can be aggregated to form global explanation. Figure cited from [151].

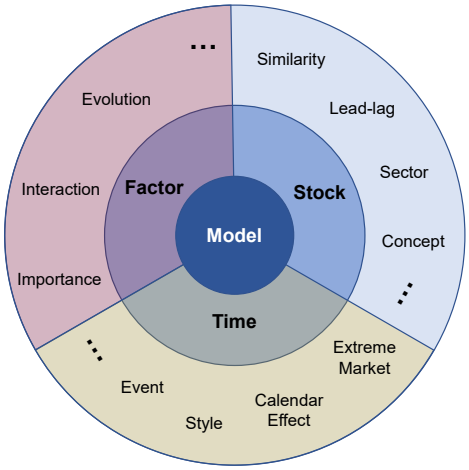


Figure 24: XAI from data dimensions.

and explanations across multiple cross-sections can be further combined to provide insights for market features in a time interval.

Extreme market. In stock markets, there are extreme conditions where nearly all stocks on the market experience severe price drops (Figure 26a). In extreme markets, it is hard for quant strategies to obtain excess return since the prices of all stocks drop together, and there is little room for arbitrage. Therefore, in extreme markets, it is important to identify the less affected stocks and trade them to earn excess returns. To achieve this, we can decompose stock returns from two aspects: those contributed by market trends and those contributed by stock-specific features The decomposition can be computed by categorizing factors into market factors and stock-specific factors. Then, the importance of these two types of factors can be computed via feature importance algorithms. And we need to select the stocks where the importance of stock-specific factors outweighs that of market factors.

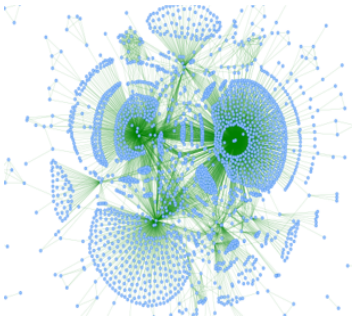
Calendar effect. Calendar effects [177] refer to market anomalies that are related to calendars, such as days in a week, months in a year, and event-related periods such as the U.S. presidential cycle. Calendar effects are caused by market participants' anticipations toward future trends and have a great influence on market trends. Thus, it is important in quant to identify calendar effects and make use of them to adjust investment strategies. Such identification can also be achieved via feature importance algorithms. By computing the importance of calendar factors, such as categorical features representing weekdays and days in the month, we can see whether model predictions heavily rely on these features. Stronger importance on calendar factors usually indicates potential calendar effects.

Style transition. Style factors are used in multiple factor models (as introduced in §1.4) such as BARRA [48] to describe the intrinsic features of stocks such as size, volatility, growth, etc. In such models, the returns of stocks are contributed by their exposures to these style factors, and the return contribution per unit exposure, also called factor return, differs across style factors. Moreover, the return of each factor also changes over time (Figure 26b) because of the transitions in the market's preference for different styles. If such transitions can be accurately recognized, investors can adjust their strategies accordingly to focus on stocks with large exposures to the dominating style factors. To detect style transitions, we can regard style exposures as factors and compute their contribution to stock returns using feature importance algorithms. We can then observe the distribution of factor contributions across time and detect shifts in this distribution as signals for style transitions.

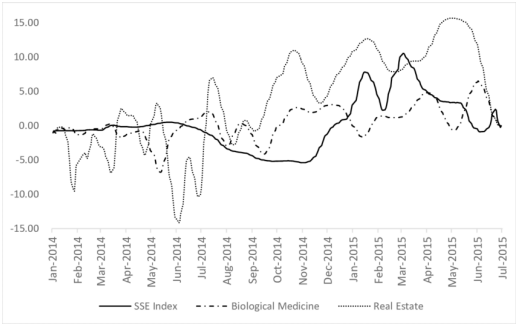
Event influence. Breaking events (Figure 26c) usually have a great influence on stock markets. Investors need to have a good understanding of the influences of breaking events to reduce the negative impacts or profit from the events. Usually, an event is associated with two pieces of information: the timestamp of its occurrence and the specific contents, which are usually represented in natural languages. Event contents can be encoded as specific factors using NLP techniques [176], and the effect of an event can be computed as the importance of the content factors concerning the market trends after its timestamp. Besides, we can also compute causal explanations to show the causal effect of the event.

3.2.3. *Explanation on Factors*

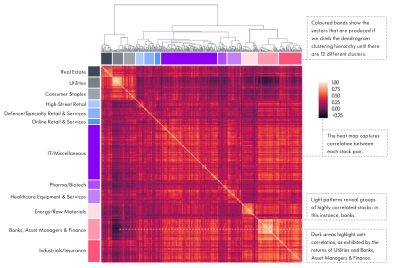
Explanations can be computed on each factor to illustrate the sensitivity of different stocks to it at different times. The explanations can be combined to show the interactive effects among factors for specific stocks.



(a) Stock similarity graph computed via return correlation.



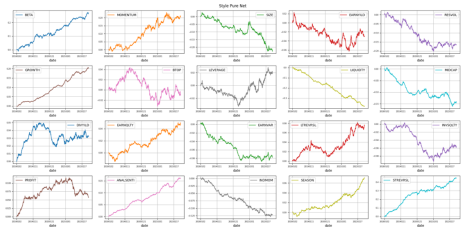
(b) Lead-lag effect in Chinese stock market. Figure from [166].



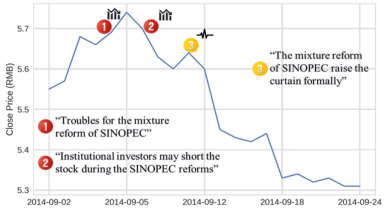
(c) Stock correlation matrix. Potential clustering can be visualized from this matrix. Figure from [167].

Figure 25: XAI in stock.

(a) Extreme market condition in 2020 of Dow Jones Industrial Average. Figure from [175].

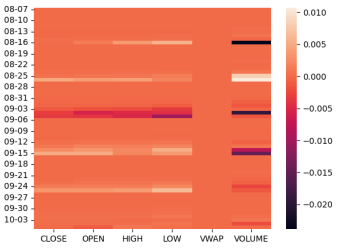


(b) Return curves of BARRA risk factors.

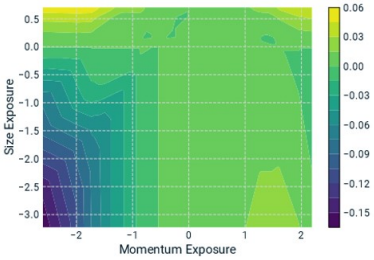


(c) Influence of breaking news across time. Figure from [176].

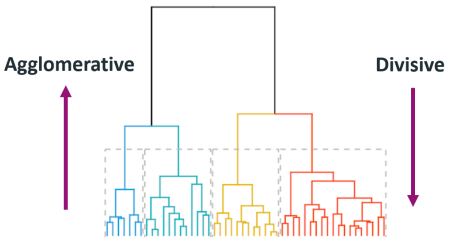
Figure 26: XAI in time.



(a) Feature importance for basic volume-price factors



(b) Interaction between size and momentum factors. Figure is cited from [178].



(c) Hierarchical clustering. Figure is cited from [179].

Figure 27: XAI in factors.

Factor type. Factors can be categorized from various aspects. For example, in terms of data sources, factors can be categorized as volume-price factors, sentiment factors, fundamental factors, etc. In terms of financial features, factors can be categorized as momentum factors, mean-reversion factors, lead-lag factors, etc. In terms of the time scales, factors can be categorized as tick-level factors, minute-level factors, day-level factors, etc. The contribution of different types of factors to portfolio returns can be computed by feature importance algorithms, and it provides investors with a better understanding of the investment strategy generated by AI, For example, in Figure 27a, the contribution to the model prediction of each factor at different positions in a time window is illustrated as a heatmap.

Factor interaction. Deep learning model is good at capturing the complicated associations between factors, and some weak factors can be combined to form strong factors. Such interactions reflect intriguing patterns among factors and provide new insights into finding new factors. Factor interactions can be revealed using feature crossing techniques. For example, in Figure 27b, the landscape of model prediction with respect to the value of two factors (exposure to size and momentum style factors) is illustrated in a contour map. And it can be seen from the map that model prediction decreases as both factor values drop.

Factor hierarchy. We can depict the semantic similarity among factors in a hierarchical way. Leveraging relevant techniques such as hierarchical clustering [180], a factor evolution graph demonstrates factor relations by arranging factors with higher similarities in lower-order neighborhoods (lower-level subtrees in the example of Figure 27c).

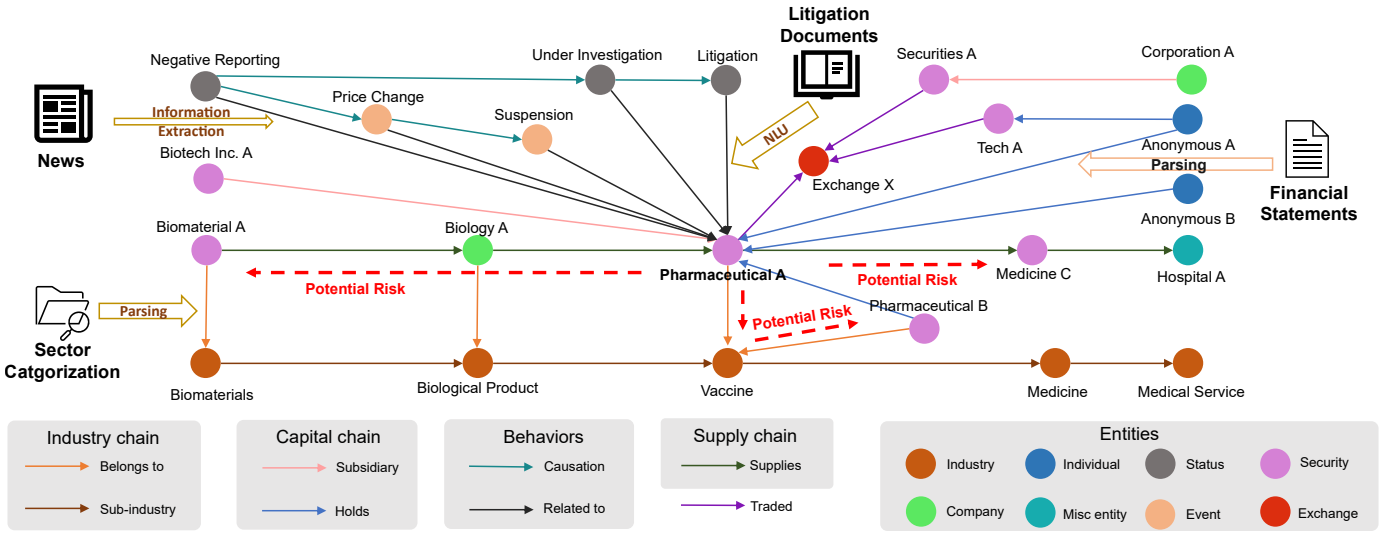


Figure 28: An example of financial knowledge graph that contains behavioral information. All the financial entities and events are fictitious and only for illustration purposes.

4. Knowledge-driven AI for Quant 4.0

As discussed in §1.5, knowledge-driven AI is an important complementary technology to data-driven AI, especially in low-frequency investment scenarios such as value investing and global macro investment. In this section we attempt to answer two technical questions: 1) how to build a practical knowledge-driven AI system, and 2) how to apply knowledge-driven AI to quant research.

The first question is about the components of a typical knowledge-driven AI system [181]. Generally speaking, such a system consists of two parts: a knowledge base which stores all knowledge we need in analysis, reasoning and decision, and a knowledge reasoning engine which analyzes and makes decisions based on the knowledge [182]. These two parts correspond to the problems of knowledge representation and knowledge reasoning [183][184], respectively, both are hot research directions in artificial intelligence. In practice, knowledge graph techniques [185] are widely used to build a knowledge base due to their simplicity and scalability. The second question considers how knowledge graph techniques can be applied to quant from the perspective of knowledge representation and reasoning. We introduce how financial knowledge is extracted from various sources and incorporated into a financial knowledge graph. We will also introduce some potential application scenarios where knowledge graph reasoning brings extra benefits for investment.

Figure 28 illustrates what a financial behavior knowledge graph looks like with an example extracted from a large knowledge graph. In this example, various entities including public companies, securities, sectors, individuals, events, as well as relationships such as supply chain, capital chain, and behavioral relations are characterized in the financial behavior knowledge graph. Information constituting the knowledge graph is collected from various data such as news reports, litigation documents, financial statements, research reports, and sectors, and extracted using NLP techniques such as natural language understanding and information extraction. Knowledge reasoning techniques can be applied to this graph for further analysis and decision. For example, negative new events or lawsuit outcomes may substantially affect the stock price of the company under study (*Pharmaceutical A*). Moreover, when the potential risk of this company is discovered from the knowledge graph, it may propagate to related entities such as important shareholders of this company and those companies on the same supply chain.

In this section, we introduce knowledge representation and reasoning techniques in §4.1 and §4.2, respectively, and introduce the application in quant in §4.3.

4.1. Knowledge Representation

The goal of knowledge representation is to encode human knowledge into machine-readable forms. It is the foundation of knowledge-driven AI and it determines the modeling method for downstream knowledge reasoning tasks.

4.1.1. Knowledge Base Techniques

In the beginning, we briefly review the history of knowledge base techniques (Figure 29) [187]. Early knowledge base concepts started from the semantic network which originates from ancient philosophers centuries ago [188] and it was first implemented by computer scientists in 1956 [189]. A semantic network is a special graph where the nodes represent concepts of interest and the edges represent semantic relationships between these concepts. These semantic graphs can be formalized equivalently as a set of semantic triples [190] (a base of the popular series of technical standards such as RDF [191]), and it is moreover applied widely in the later knowledge graph techniques. Construction of semantic networks uses many NLP techniques such as semantic parsing [192], which use entity recognition and relation extraction to parse text data and build the semantic network.

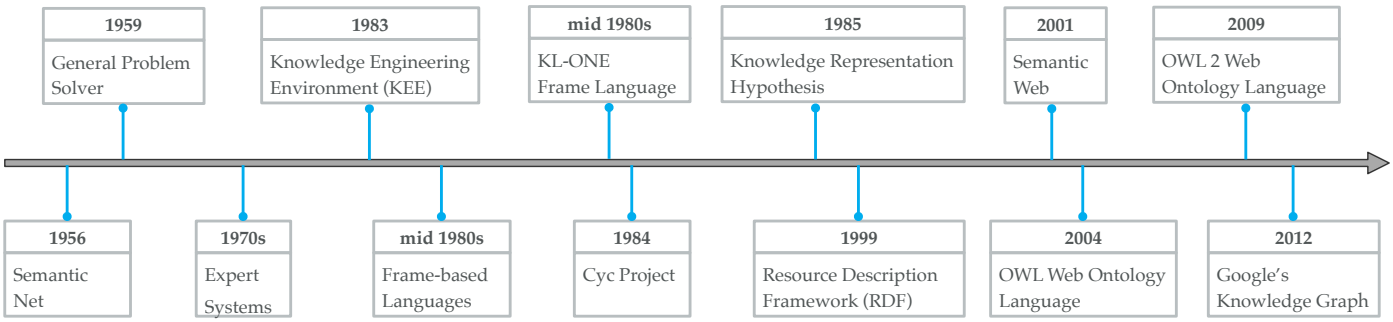


Figure 29: Knowledge base history. Figure cited from [186].

Later on, logic-based knowledge representation techniques such as propositional logic [193], descriptive logic [194] and first-order logic [195] were introduced, and they quickly became the mainstream techniques for decades. As an early logic-based knowledge system, *general problem solver* was proposed in 1959 [196]. It formalizes problems of interest as Horn clauses [197] and applies logic-based methods to solve them. The application of general problem solver is limited because most real-world problems are too complex, and this shortcoming leads to the development of expert systems [198] which restrict the knowledge to domain-specific tasks only. In addition to expert systems, frame-based languages [199] is an alternative technique for logic-based knowledge representation. Frame-based languages characterize real-world objects with concepts such as classes and inheritance relationships, similar to the abstraction in object-oriented programming. Successful examples of frame-based expert systems include the Knowledge Engineering Environment [200], KL-ONE framing language [201] and the Cyc system [202].

Traditional knowledge base techniques faced new challenges in the big data era, where data volume and data exchange exploded. The requirement of new protocols results in the emergence of various knowledge standards such as resource description framework (RDF) [203] and web ontology language (OWL) [204], and these new techniques are summarized and redefined as a new concept Semantic Web [205]. Semantic web techniques are further extended to satisfy large-scale applications in practice and it leads to the development of knowledge graph techniques which is the mainstream methods for building knowledge-based systems in data-driven scenarios.

4.1.2. Knowledge Graph Techniques

We introduce the structure of a knowledge graph and how to build it in practice.

Structure of Knowledge Graph. A knowledge graph typically consists of two parts [206]: ontology and instances.

- Ontology is the schema of a knowledge graph, which specifies the types and semantic meanings of the entities and relationships [207]. In practice, an ontology is usually encoded with formal languages such as OWL and RDF. For example, in the knowledge graph shown in Figure 28, the ontology specifies the eight types of financial entities (industry, individual, status, etc.) and the various types of relationships between them (supplies, shareholding, subsidiary, etc.). In addition to those regular elements, ontology can further define other attributes for entities and relationships such as timestamps, hyperlinks, etc.
- Instances are the main body of a knowledge graph. They can be expressed as semantic triples, which consist of subjects, predicates, and objects. For example, in Figure 28, the semantic triple (*Pharmaceutical A*, *supplies*, *Medicine C*) indicates the fact that *Pharmaceutical A* is a supplier of *Medicine C*. In a knowledge graph, the subjects and objects are entities and the predicates are relations. In practice, semantic triples are usually implemented in RDF and stored in graph databases.

Knowledge Acquisition. In practice, knowledge graphs are usually very large, with millions or even billions of entities. Therefore, knowledge acquisition techniques are required to automatically construct the knowledge graph. As summarized in [186], knowledge acquisition can be performed via knowledge extraction and knowledge graph completion.

- Knowledge Graph Completion aims to fill up the missing links in a knowledge graph based on existing information, and this problem has been extensively studied. For example, [208] proposes a rule learning model that extracts symbolic rules from knowledge bases with large-scale data. The rules can be further used to infer missing facts between entities. In [209], an RNN model is proposed to compose paths in the knowledge graph into embedding vectors. The composed embedding is then used to infer the missing relationship between the starting and ending entities of the path.
- Building from scratch extracts structural information from raw data. Most works on knowledge graph construction focus on the task of information extraction from text data [210, 211, 212]. This task consists of two major steps: entity recognition [213] and relationship extraction [214]). Entity recognition involves semantic role labeling [215] which identifies entity roles (e.g., subject, object, etc.) in semantic triples, and entity disambiguation [216] which aligns text tokens to entity names in the knowledge graph. Relationship extraction can be formalized as a prediction problem where the inputs are entity pairs of interest and the output is the relationships between entities. Various machine learning methods

such as convolutional neural networks [217], attention mechanism [218], and graph neural network [219] can be applied to this problem. In addition, some works [220, 221, 222] unify the entity recognition task and relationship extraction task in one end-to-end learning framework, where the inputs are token sequences (e.g., sentences) and the outputs are semantic triples.

4.2. Knowledge Reasoning

Knowledge reasoning refers to the process of analysis, inferences, proofs and decisions (e.g., inferring new facts, generating new conclusions, extracting new rules, etc.) based on existing knowledge and data. Reasoning can be conducted in various ways, including symbolic logic methods, neural methods, and neuro-symbolic methods.

4.2.1. Symbolic Reasoning

Symbolic reasoning can be conducted in either a deterministic way or a probabilistic way. Deterministic reasoning is performed by applying inference rules on given facts recursively until reaching the desired conclusion. Meanwhile, probabilistic methods are also applied in practice for more flexible reasoning.

Probabilistic symbolic reasoning methods model the distribution of the existence of fact triples given the knowledge on the graph. Therefore, logic rules are represented in a softer way, which allows more flexibility in the reasoning process. Various methods have been proposed for probabilistic reasoning. For example, probabilistic logic programming [223] models a logic program as a computation graph similar to Bayesian networks. Markov logic programming [224] involves a first-order knowledge base where each rule is assigned as scalar weight. In this way, the joint distribution of all the observed and hidden facts is modeled as the normalized exponential weighted sum of the grounds of each rule. There are also other probabilistic reasoning techniques such as stochastic logic programming [225] and TensorLog [226].

Symbolic reasoning has the advantage of transparency and explainability compared with neural methods. Specifically, it models knowledge in an explicit way by expressing facts and rules with logic chains or formulas. However, symbolic reasoning depends on curated knowledge bases that require enormous human effort in construction and maintenance. Moreover, it also suffers from high computation costs since most symbolic reasoning algorithms have to search in a high-dimensional search space. This fact usually limits their application in big data.

4.2.2. Neural Reasoning

Neural reasoning methods learn decision rules with deep neural networks and can represent non-linear associations. The information in entities and relations is embedded into neural networks. Neural methods utilize gradient descent search in the training process, and achieve better efficiency, especially in big data scenarios. Besides, neural methods also achieve better reasoning performance due to stronger expressiveness.

Neural methods train a deep learning model using knowledge graph structure described by semantic triples as well as the attributes of entities and relationships. During the inference process, the trained model predicts the fact of the input triple of interest. The following question is how to encode the input including both structures and attributes. Different embedding spaces can be used to represent entities and relationships. Many works represent entities as embedding vectors in Euclidean space but they differ in the modeling of relationships. [227] represents relationships as points in Euclidean space that translate subject embedding towards object embedding. [228] encodes relationships as projection matrices that map entity embeddings to low-dimensional space. [229] defines relationships as 3-D tensors that represent the bilinear similarities between entities from multiple dimensions. In addition to Euclidean space, other types of embedding space can also be used, including complex vector space [230, 231], Gaussian distribution [232, 233], and manifolds [234]. The possibility of semantic triples can be computed directly from the representations of the corresponding entities and relationships. For example, in [227], the possibility is computed as the distance between object embedding and subject embedding through relationship embedding. The computation of possibility can be extended to capture more complex interactions between entities and relationships using neural networks. For example, [235] concatenates subject embedding, object embedding and relationship embedding and feeds them to a convolutional neural network. Moreover, neural networks can also leverage the structural information of the knowledge graph. For example, in [236], a recurrent neural network encodes paths on the graph (similar to logic chains) involving multiple semantic triples. In [237], a graph neural network leverages the structural information of the knowledge graph and computes graph convolution to improve reasoning.

4.2.3. Neurosymbolic Reasoning

Symbolic reasoning and neural reasoning have different advantages, which can be combined in a neurosymbolic reasoning framework. Neurosymbolic reasoning can be conducted by either injecting logic structures into the embedding framework, or vice versa [238].

For the former idea, [239] incorporates conjunction rules into the computation of relation embedding in multi-hop paths. Following the framework of TransE [227], the relationships of semantic triples on a reasoning path are composited as one single embedding vector, which is used to translate subject embedding towards object embedding. [240] put constraints on the representation learning process to enhance the prediction confidence of the conclusions in entailments. Besides, other works also propose various ways to inject logical structures [241, 242] or ontological schemas [243, 244, 245] into the knowledge graph embedding framework.

For the latter idea, [246] proposes to infer the missing facts using neural networks and then reason over the queries with Markov logic networks. The whole model is optimized via the expectation-maximization algorithm [247], where the E-step corresponds to inferring hidden facts, and the M-step corresponds to maximizing the likelihood of the given facts. [248] uses knowledge graph embedding techniques to help shrink the size of candidate sets for fact inference in large-scale knowl-

edge bases. Then, inference via ground network sampling is performed on a Markov logic network to compute the final results. Moreover, there are also other works utilizing knowledge graph embedding for logic reasoning [249] and rule learning [250, 251].

4.3. Application in Quant

In this part, we will exhibit how knowledge-driven AI is applied in Quant 4.0 from two aspects: construction of financial behavioral knowledge graph and knowledge graph reasoning for quant.

4.3.1. Building a Financial Knowledge Graph

Ontology Design. The ontology of a financial behavioral knowledge graph should cover information from the following aspects: 1) the fundamental information of financial entities, 2) financial events happening between financial entities, and 3) the causal relationships between entities and events. Correspondingly, categories for entities include but are not limited to:

- Financial entities, including stocks, bonds, banks, public companies, important individuals, commodities, etc.
- Concepts, reflecting the fundamental information about financial entities, such as sectors, industries, exchanges, regions and countries, currencies, etc.
- Events, which are economic behaviors such as administrative punishment, illegal actions, litigation states, shareholding changes, personnel changes, etc.

Similarly, categories for relationships include but are not limited to:

- Relationship between entities, such as subsidiaries, belongs, shareholding, etc. These relationships are thus associated with timestamps indicating their beginning and ending times. For example, in Figure 28, the industrial chain and capital chain relationships describe the sector categorization and capital relationships between entities.
- Relationship between events, such as co-occurrence, lead-lag.
- Relationship between events and entities between events. For example, in Figure 28 the negative reporting leads to the price change and further investigation on "Pharmaceutical A", leading to a suspension in trading and litigation. Causal relations are usually inferred from existing knowledge and serve as important auxiliary information in downstream reasoning tasks. For example, in Figure 28, a "related to" relation connects the event of "negative reporting" and the corresponding financial entity "Pharmaceutical A" to indicate a negative reporting happening to the publicly traded company "Pharmaceutical A". These relations are usually associated with timestamps indicating the specific time point that the events happen.

Knowledge Acquisition. Knowledge that constitutes the financial behavioral knowledge graph can be acquired from various sources, and the most challenging part is to extract useful structural knowledge from unstructured data (text data as a representative example). Natural language expression has high flexibility and substantial personality, and thus it is a challenge for machines to accurately understand the information in documents and extract the most useful knowledge to build a knowledge graph. There is a lot of contradictory information in news and documents, leading to the difficulty in fact extraction. Therefore, probabilistic models and machine learning models play important roles in knowledge extraction with confidence evaluation. Moreover, the information is usually incomplete and extrapolation is needed to infer the knowledge that we are actually interested in. Graph completion is thus required to infer the missing knowledge from the given facts. Fourthly, different data sources share inconsistent update frequencies, which brings new challenges for an appropriate representation of the knowledge graph. Both snapshot-based representations and growing flat knowledge graphs can capture temporal information. A growing flat graph is friendly for temporal updates, but it is hard to perform temporal analysis directly on it. On the contrary, snapshot-based representations are naturally suitable for temporal analysis, while also bringing extra costs in storage and management.

4.3.2. Knowledge Reasoning for Quant

Given a knowledge graph, we can get meaningful representations of knowledge by reasoning on the graph. For quant, the knowledge representations can be incorporated into existing factors as external information and fed to deep learning models for better predictions. Figure 30 demonstrates a typical pipeline of knowledge reasoning in quant. Specifically, events and relations between stocks are represented as semantic triples, where the entities and relationships are embedded into vectors. Then, neural reasoning is performed on these semantic triples to compute the embedding of events, relations and the whole knowledge graph. After training on historical data, the knowledge representations are used in investment strategies to generate trading decisions.

There are also other works studying the application of knowledge reasoning for quant. [253] proposes incorporating relational and categorical knowledge for better event embeddings. Given a semantic triple representing an event, external information about the entities in the semantic triple is retrieved from a knowledge graph and involved in the computation of event embedding. [252] extracts events from news texts and uses entity linking techniques [254] to align the extracted information with the knowledge graph. Then event embeddings are generated using TransE. The embeddings are then combined with volume-price data in a temporal convolutional network [255] for stock prediction, as shown in Figure 31. [53] leverages fundamental information such as sector categorizations and supply chains to build a knowledge graph and uses temporal graph convolution to compute the embedding of each stock. The embeddings are then used to predict the stock returns, and the whole model is trained by maximizing the stock ranking loss.

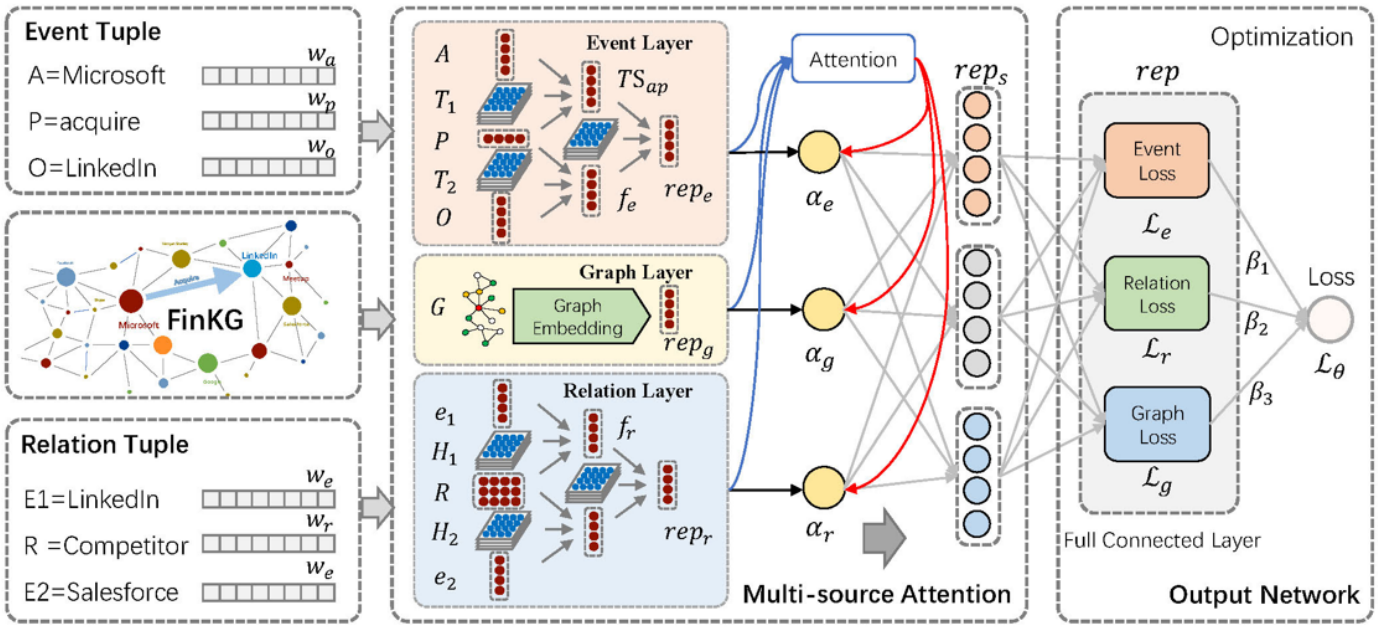


Figure 30: Knowledge graph reasoning for stock prediction. Figure is cited from [183].

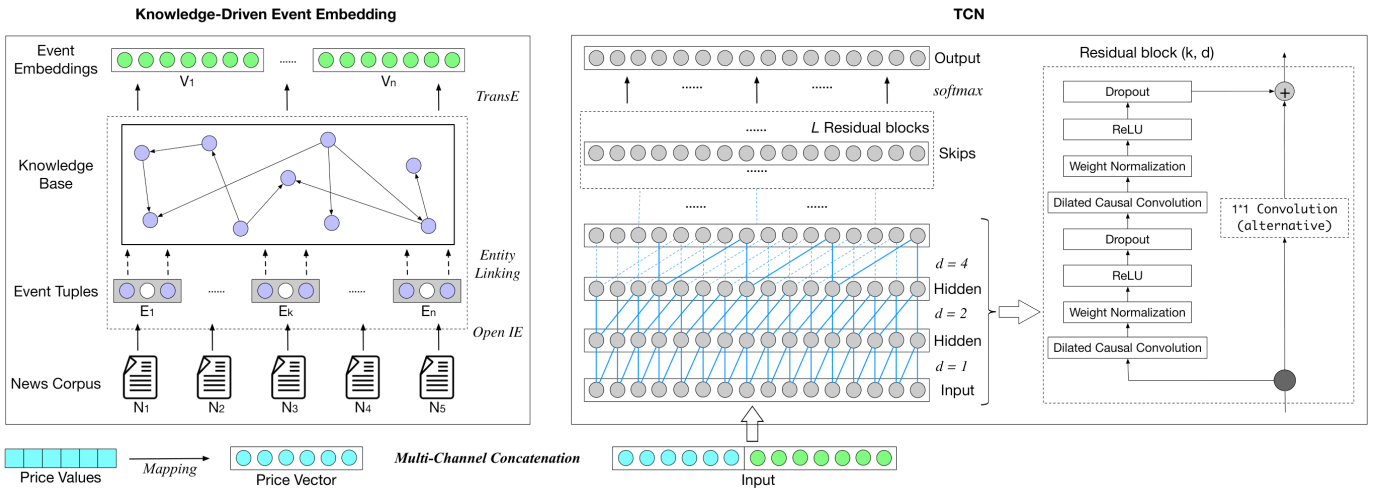


Figure 31: Knowledge graph event embedding combined with other factors in stock prediction. Figure cited from [252].

[256] uses node2vec [257] to generate stock embeddings based on a knowledge graph and uses these embeddings to compute the similarity between stocks. In this way, the top-K nearest neighbors are computed for each stock, and the factors from neighbors are used to enhance the original factors. Other works [258, 259] also uses knowledge graph to generate better stock embeddings or perform event-driven investment.

5. Building Quant 4.0: Engineering & Architecture

Sections 2, 3 and 4 introduce the three components of Quant 4.0 for the algorithmic perspective. In this section, we "retrospect" Quant 4.0 from a system point of view and study how to put all these components together in one system. Figure 32 illustrates the architecture of a proposed Quant 4.0 system framework, including the offline system for quant research and the online system for quant trading.

5.1. System for Offline Research

Quant 4.0 offline quant research system aims to improve the efficiency of quant research. It contains several layers (hardware layer, raw data layer, meta factor layer, factor layer, and model layer) and modules (high-performance computing clusters, data system, cache system, data preprocessing, automated factor mining, knowledge-based system, large-scale data analytics, AutoML, and risk simulation).

5.1.1. Hardware Platform Architecture

The underlying hardware platform for offline research is a high-performance computing cluster, consisting of many com-

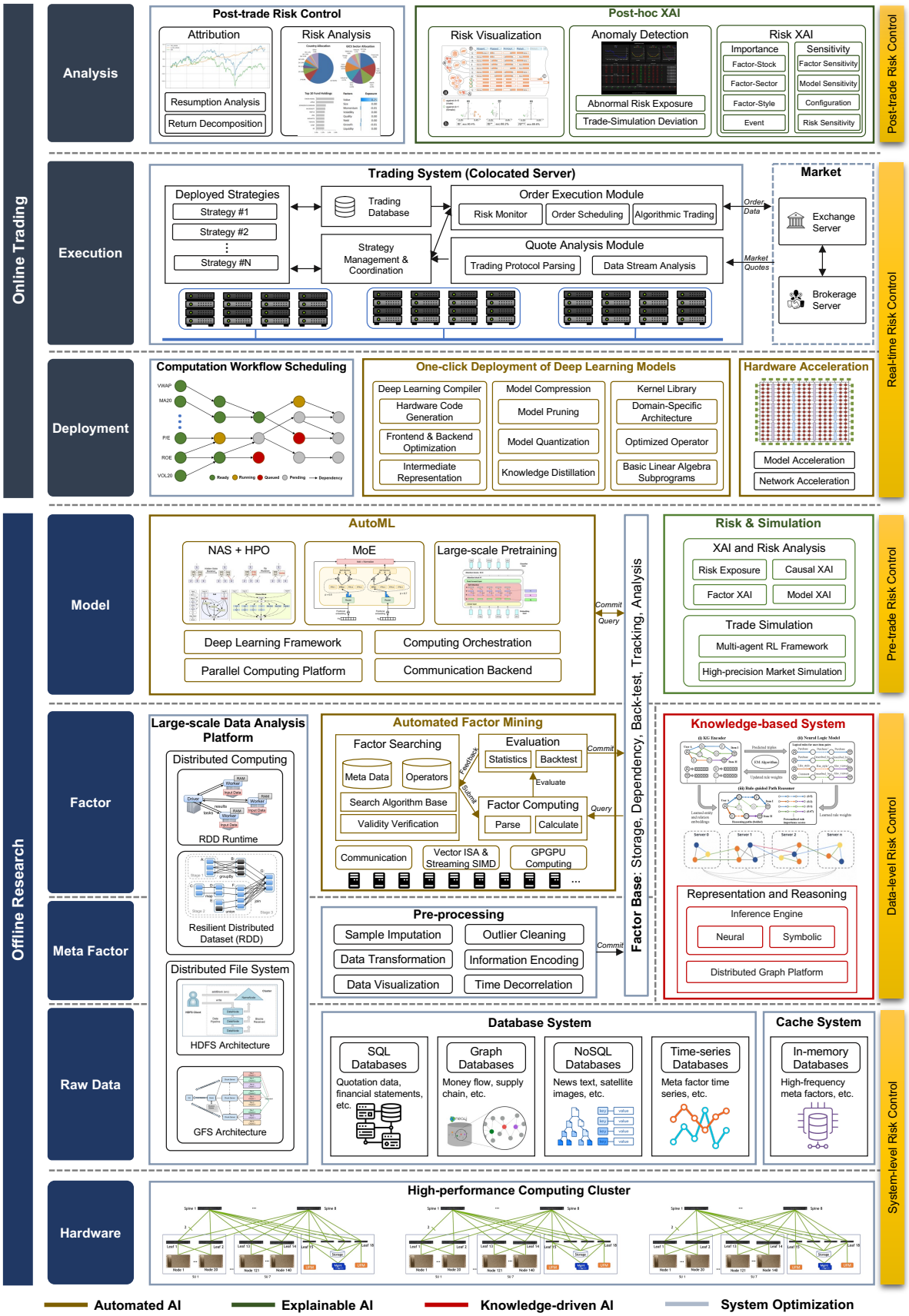


Figure 32: Architecture of an example Quant 4.0 engineering platform for investment research and trading. Part of this figure is cited from [260, 261, 262, 263, 264, 265, 89, 266].

puting nodes that are mounted with shared storage [267] and interconnected by high-bandwidth network [268]. The combination of multiple nodes aggregates the computing power distributed in various single nodes to support large-scale quant computing tasks. However, communication bottleneck usually serves as one of the constraints for scaling up computing power [269]. To address this problem, the network topology of the cluster adopts a hierarchical structure, where neighboring computing nodes are connected via lower-level switches to increase the overall throughput of the cluster.

5.1.2. Design of Data System

The data layer for Quant 4.0 system is to collect a tremendous amount of financial data and provide data management and query service for applications in upper layers. Since financial data are heterogeneous and multimodal, different types of database systems are involved in this layer to manage different types of data. Examples include SQL database [270], time-series database [271, 272], NoSQL database [273] and graph database [274].

- SQL database stores and manages data following the relational model proposed in [270], where data are stored in tables that represent relations among attributes. Most traditional financial data such as quote data and financial statements can be represented in this type and are suitable for SQL databases.
- Graph database [275] is designed to store and manage graph-structured data composed of nodes and edges. Such graph structures widely exist in financial data since financial entities are connected with each other through various relationships such as money flow and supply chain relations. Such links may indicate some latent patterns that are shared in neighborhoods. Graph database can be used to store and manage economic graphs, financial knowledge graphs and financial behavior graphs.
- NoSQL database [273] is used for storing non-tabular data such as key-value pair, document, and wide-column. It is appropriate to manage financial text data and image data, such as financial report text, news text, social media text, satellite and drone images etc.
- Time series database [272] is a special database type designed for quickly accessing, computing and managing time series data. Such kind of database engines is optimized to accelerate time series data processing, e.g., computing stock stream data collected in real-time, and computing time-series factors for high-frequency trading etc.

Moreover, the large volume of financial data requires a highly efficient distributed storage system to accelerate data access. To further improve the read/write speed for high-frequency tick data (limit order book etc), in-memory database [276] could be used as data cache to store the most frequently read data. It saves the data transfer time between hard disk and memory.

In addition to layer-specific components, the data layer, meta factor layer, and factor layer share a large-scale data processing platform which provides a complete solution and a convenient model for various data-driven tasks. The key components of this platform include a distributed file system that provides convenient and reliable data access on distributed storage systems, and a distributed computing engine that provides simple yet efficient programming interfaces for parallel computing on a large number of computing nodes. Specifically, distributed file systems such as HDFS [264] adopts the architecture consisting of name nodes and data nodes, where data replication is performed on multiple data nodes for fault tolerance. On the other hand, distributed computing engines such as MapReduce [277] and its open-source implement including Hadoop [278] and Spark [260, 261] provide programming models for parallel computing tasks. By abstracting parallel computing tasks into a set of primitive operations (e.g. map and reduce in MapReduce and transformation in Spark), the programming tool is ease to use by developers and is generic enough to handle many common parallel programming tasks.

5.1.3. Factor Mining System

Raw data have different formats, but factor mining requires unified input format. Therefore, corresponding to the workflow in Figure 12, the meta factor layer is involved to preprocess raw data with various modalities into meta factors with unified formats and appropriate values.

The factor layer builds automated factor mining engine and automated factor mining pipeline. The factor mining algorithms have been introduced in §2.2.1. Here we introduce how to implement factor mining at scale from a system point of view. In particular, we concern how to improve the system efficiency to discover more "good" factors per unit time.

- Factor mining system needs a parallelization architecture to improve computational efficiency.
- Syntactic validity of factors should be checked in real time during the factor generation process to reduce the CPU time wasted by invalid factors.
- Diversity of factors should be controlled in real time during factor generation process in order to reduce the CPU time consumption for redundant factors.

The whole system is backed by some key techniques in distributed execution and computing acceleration. Distributed execution tools such as message queues and distributed caches enable asynchronous parallel execution on multiple nodes, thus guaranteeing system scalability. Computing acceleration techniques such as vector and streaming SIMD instructions on CPU and massive parallel computing on general-purpose graphics processing unit (GPGPU) [279] significantly improve the computation efficiency for data frame operations and thus increasing the productivity of the whole system.

Meta factor layer, factor layer, and model layer are connected to the factor base, which is an integrated platform for the storage, computation, dependency management, backtest, tracking, and analysis of all factors (the output of models are also factors). Factors generated in these layers are committed to the factor base in various forms. Specifically, meta factors pre-processed from raw data are directly committed to the factor base in data frames, with appropriate descriptions about their data sources and pre-processing methods. Factors gener-

ated via automated factor mining are committed to the factor based in the form of symbolic expressions where the operands are other factors in the factor base. The model outputs generated in the model layer can also be regarded as speical machine learning factors, which can be committed to the factor base together with specifications of input factors, model architectures, hyperparameters and training descriptions, etc.

5.1.4. Knowledge-based System

In parallel with the factor mining system, a knowledge-based system is also involved in the overall architecture to provide knowledge-driven AI practice. As discussed in §4, the knowledge-based system consists of two modules: a knowledge base for knowledge representation and an inference engine for knowledge reasoning. Specifically, a distributed graph computing platform is used as the knowledge base to store large financial behavior graph and the inference engine is built upon for downstream financial knowledge reasoning and decision making.

Compared with traditional small knowledge bases, financial behavior knowledge graph in Quant 4.0 has the potential to grow up to a scale with billions of nodes and edges, and thus it requires a flexible and scalable architecture [280, 281] to store and manage large-scale dynamic graph data. Therefore, we need a distributed graph computing and management platform, which partitions the whole knowledge graph into a number of subgraphs distributed in different nodes of a computing cluster. Due to the sparse nature of knowledge graph, this partition is specifically arranged to maximize data locality, where graph nodes located in the same partition are more densely connected than those located in different partitions [282]. For the inference engine, both knowledge graph embedding and rule-based symbolic reasoning algorithms can be implemented on this platform.

5.1.5. Modeling System

The model layer is in charge of automatic generation of machine learning models and the corresponding risk evaluation and backtest simulation procedures before they are deployed into real-world environments. Therefore, this layer involves two major components: an AutoML module and a pre-trade risk analysis and simulation module.

The AutoML module implements automated model generation algorithms upon large-scale distributed deep learning systems. The bottom layer of its technology stack consists of parallel computing platforms such as CUDA [283], and communication backends such as message passing interface (MPI) [284] that provide standard interfaces for communications between computing nodes in a distributed system. In the second layer of the technology stack, deep learning frameworks such as PyTorch [285] provide basic interfaces for training and inference of deep neural networks via hardware-accelerated linear algebra operations and automatic differentiation engines. In addition, computing orchestration systems such as pathways [286] combine low-level communication primitives with deep learning framework functionalities to implement higher-level parallelisms such as model parallelism and pipeline parallelism, which is further wrapped as standard interfaces for upper-level programs. The top layer of the technology stack consists of implementations of model generation algorithms such as neural architecture search and hyperparameter optimization (NAS+HPO) [287], mixture of experts (MoE) [265] and large-scale pretraining [288, 289].

The risk and simulation module identifies and analyzes the potential risk exposures of the models before they are deployed to real-world trading environments. This module implements explainable AI techniques applied to analyzing factor, model, and causality to reveal nonlinear risk exposures that are more complex than ordinary risk exposures explained by the BARRA model. In addition, market simulator [290] is used to test the performance of trading strategies with higher precision than backtest, whose results may be biased by historical data. Specifically, the simulation environment system [290] can be built using multi-agent reinforcement learning [291, 292] where agents imitate the behavior of various market participants in real world [293].

5.2. System for Online Trading

The online trading system focuses on deploying investment strategies for real trading and executing post-trade analysis, and its major goal is to achieve low trading latency and high execution efficiency. The trading system consists of three modules: deployment, trading and analysis, and we introduce their functions in the following part.

5.2.1. Model Deployment

The deployment layer aims to implement the philosophy of technology "one-click deployment". It involves a computation scheduling module, an automated deployment module, and an optional hardware acceleration module. The computation scheduling module arranges a reasonable and efficient computation order for factors based on their intrinsic data dependencies which form a directed acyclic graph (DAG) over factors. In the computation scheduling system, each factor starts its computation if and only if all its preceding factors on the DAG have finished their computation. Computation scheduling is a tedious work and it should be automated by system due to the following reasons.

- We must synchronize offline factor dependencies with online factor dependencies and keep their consistency in real time.
- The number of factors may grow up quickly. Imagine what happens when you accumulate millions of factors? It is a nightmare for any quant researcher to deploy so many factors by hand.
- Adding, deleting and updating factors is the daily job of factor maintenance, which relies on correctly managing factor dependencies.

From the algorithmic point of view, the problem of computation scheduling can be regarded as topological sorting on the dependency DAG. In practice, the scheduler is designed to coordinate system components and schedule their executions according to the computation order on DAG. In common implementations [294] of such scheduling systems, asynchronous scheduling is

adopted to improve the overall execution efficiency. In this way, the pending steps can start their execution immediately after the previous steps finish.

The automated deployment module aims to deploy deep learning models trained from offline research to online trading. It implements the algorithmic techniques discussed in 2.4. In addition to deep learning compilers and model compression engines, this module also involves optimized hardware kernel libraries, which provide implementations of common data processing and modeling functions that are highly optimized based on hardware features. Popular examples of such libraries include cuDNN [295] and MKL [296]. Functions in kernel libraries typically include basic linear algebra subprograms (BLAS) [297] that are standard interfaces for scientific computing, and some optimized operators for deep learning such as 3 convolution. The implementations are specifically optimized based on domain-specific architectures [298] on devices (e.g. Tensor Core [299]) to maximize the potential of hardware in use.

The hardware acceleration module aims to improve the computation efficiency of data processing and model inference using special hardware technology such as field-programmable gate array (FPGA) acceleration. Strategy components such as network protocol stack or machine learning models implemented on FPGA with customized logic can bypass redundant logics that are inevitable on generic hardware, thus achieving lower latency and winning an advantage for traders over other market participants. However, to deploy strategies on specific hardware, it usually takes a huge amount of development effort to complete strategy migration with satisfactory speed. Therefore, high-level synthesis techniques [300, 301, 302] are developed to address this problem. They directly generates register transfer languages for high-level representations in which strategies are originally implemented.

5.2.2. Trading Execution

The execution layer converts trading decisions to actual orders that are executed in exchanges, and its goal is to reduce the trading latency as much as possible in order to capture the fleeting trading chances in the market. The latency can be decomposed into two parts: transmission latency is the delay of signal communication between trading servers and market servers (exchange server or brokerage server), and computation latency is the delay between quotes receiving and order sending. To reduce transmission latency, the trading system is usually deployed on servers that are colocated with market servers (such as racks and cabinets provided by brokers). To reduce computation latency, a trading system must be optimized in full strategy pipeline from data collection to order execution through various software and hardware acceleration techniques.

5.2.3. Trading Analysis

The analysis layer monitors the execution of investment strategies and performs analysis for further adjustments. It involves a post-trade risk control module responsible for ordinary performance monitor and a post-hoc XAI module for explaining strategy behavior from the perspective of AI. The post-trade risk control module performs return and risk attribution, aiming to analyze strategy's performance and revealing intrinsic risk structure hidden by machine learning blackboxes. Furthermore, the post-hoc XAI module provides in-depth analysis and explanation of investment strategies. It provides thorough risk analysis by analyzing all strategy components in terms of importance and sensitivity, and visalizes risk results.

Remarks:

Risk control is one of the core tasks of quantitative investment and is also the primary consideration in system design. In our proposed engineering framework, the idea of risk control runs throughout the whole architecture. Specifically, system-level risk control is reflected in hardware and raw data layers, where the reliability and stability of the underlying hardware and raw data are the top priority. Data-level risk control is reflected in meta factor and factor layers, where quality control and management of factors and knowledge are emphasized. Pre-trade risk control is reflected in the model layer, where model robustness and explainability are required. Real-time risk control is reflected in deployment and execution layers, where the system's trading behavior is confined within constrained areas to avoid unexpected situations. Post-trade risk control is reflected in the analysis layer, where detailed analysis is conducted to provide comprehensive and reasonable insights into strategy performance.

6. Discussion on 10 Challenges in Quant Technology

We have summarized 10 challenges in the development of next-generation quant technology. These challenges range from computing and data infrastructure, investment modeling, risk modeling, market simulation and cognitive AI technology, and they provide new research directions for researchers interested in AI technology and quant. We believe some problems might be solved in the next couple of years with the rapid development of AI technology, while others may remain challenging for a long time.

6.1. Exponentially Growing Demand of Computing Power

With the rapid development of GPU technology and parallel computing technology, AI models have been scaling up year by year. The number of parameters in deep learning models has grown exponentially from 94 million (ELMo in 2018 [304]) to over 500 billion (Megatron in 2022 [303]) (Figure 34). The rapid growth in model size not only improves models' performances, but also leads to a paradigm shift which is deeply affecting the technical roadmap of AI research. Accordingly, as an important domain going all-in on AI, Quant 4.0 heavily relies on ultra large-scale computing power and related engineering technology.

6.1.1. Quant 4.0 and Supercomputers

Investment Research of Quant 4.0 requires super-computing infrastructure as a fundamental support for large-scale factor mining, large-scale modeling, back-test and evaluation. One

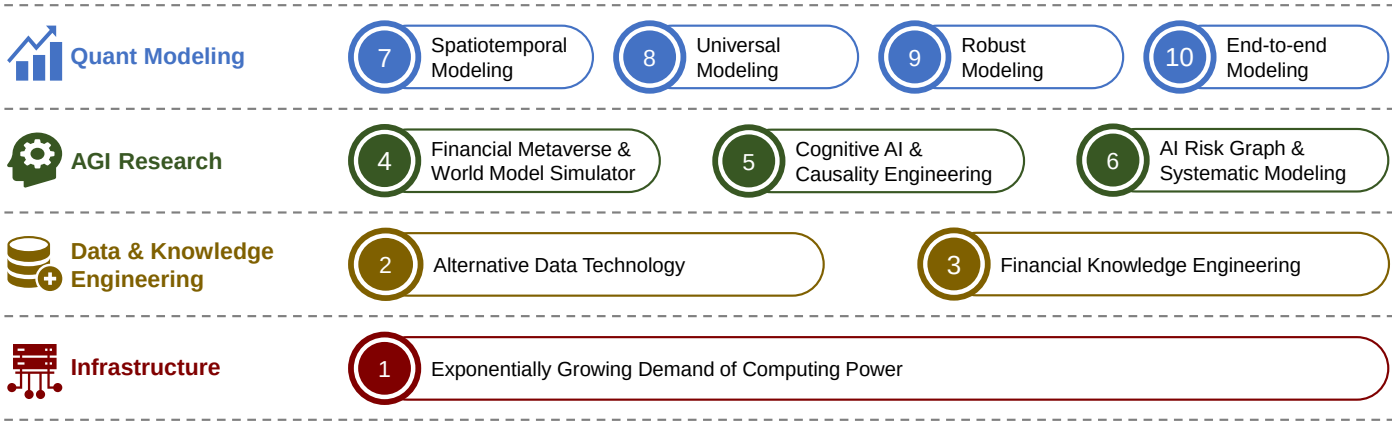


Figure 33: The 10 challenges in quantitative investment technology.

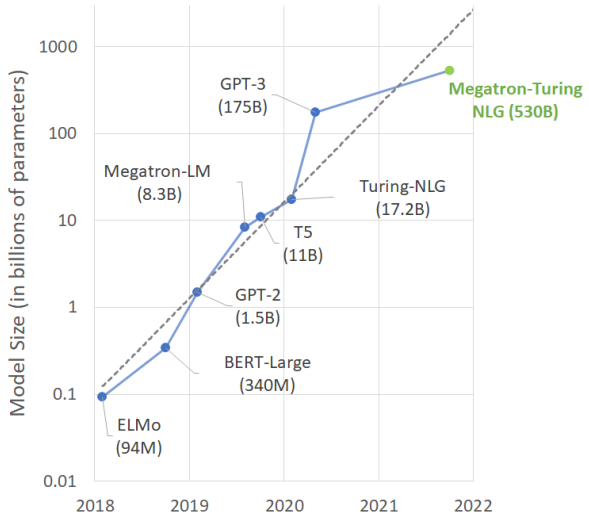


Figure 34: Growth of model sizes (measured by number of parameters) from 2018 to 2022. Figure is cited from [303].

important idea behind Quant 4.0 is to "replace human power with computing power" in quant research. We have to notice that Quant 4.0 has much more demand on computing power than what we could usually image ever before, because of the following reasons:

- Large-scale automated factor mining requires large distributed computing clusters with either homogeneous CPU parallelization or heterogeneous CPU-GPU parallelization. Even on a relatively simple homogeneously structured computing cluster, a high-frequency alpha factor mining task needs millions of CPU cores in order to achieve enough efficiency, performance and diversity in intensive factor search and evaluation. In particular, as more and more factors are discovered and accumulated, it will take longer and longer computing time (or equivalently computing power) to find a new qualified factor which not only performs well but also weakly correlated with existing factors discovered before. The same computation power problem happens similarly for heterogeneous clusters.

- Large-scale deep learning is necessary in Quant 4.0 research as some billion-parameter-level or even trillion-parameter-level deep learning models with large data volume do exhibit superior performance in many AI scenarios [305]. More and more large models are applied to quant tasks such as market prediction, portfolio position computing, risk forecasting and real trading by hedge funds and other institutional investors. However, the training process of large deep learning model requires extremely high computing power. For example, given a scenario of cross-sectional alpha model on 4000+ stocks and hundreds of daily factors of 10 years' history, a Transformer-type deep neural network with about ten billion parameters needs about 1-5 days to finish a single training/validation cycle on a cluster with 100 Nvidia A-100 GPUs, without counting the additional computational cost due to rolling training, model ensemble and autoML.

- Rolling training, i.e., iteratively retraining the same model with data sampled from a time window shifting towards the future, is a characteristic of quant modeling since the patterns of financial market and the patterns of investment instruments vary over time. In fact, financial time series are strongly non-stationary, and thus data features exhibit different distributions in different market styles and different time periods, and the performance of a model usually decays over time without retraining on recently incremented data. This phenomenon means a one-time model training on data like what people do in many other AI scenarios such as natural language process and image recognition doesn't work well in quant research. Therefore, rolling training is widely adopted in quant research for back-test simulation, paper trading and real trading. Imaging an experiment rolling training once a week, a ten years' back-test process retrain the model $10 \times 50 = 500$ times, significantly increasing the computational cost.

- Model ensemble [306] is a common way to improve performance and robustness of a quant strategy and reduce the risk of portfolio asset loss. Moreover, ensemble of diversified or uncorrelated models could usually improve strategy performance and asset return. However, the computational cost in-

creases linearly with the number of models combined in a strategy, and computer power becomes an important foundation for strategy stability and risk management.

- NAS and HPO are another two modules in quant pipeline heavily consuming computing power. Both of them are formalized as search-and-optimization problems in the network architecture space and the hyper-parameter space, respectively, and the computational cost is nonnegligible even if more and more fast algorithms are developed to improve the search efficiency.

- Feature selection (or feature importance computation) plays an important role in improving model performance and enhancing model explainability. Unfortunately, feature selection algorithms for deep learning are usually computationally intensive as well due to complicated feature interactive effects in the model, and they require sufficient computing power to identify significant factors within affordable time.

6.1.2. Solving Computing Power Dilemma

Learning large model is very expensive. Figure 35 compares a number of deep learning models on code generation tasks in three aspects: parameter size, sample (token) size and estimated cost. We can see that GPT-3 model has about 170 billion parameters and costs about 2 million USD to complete the computation, which is obviously too expensive to be afforded for most financial institutions. Given the expensive and limited computing power, what are the solutions to this dilemma? We attempt to provide a few suggestions and research directions.

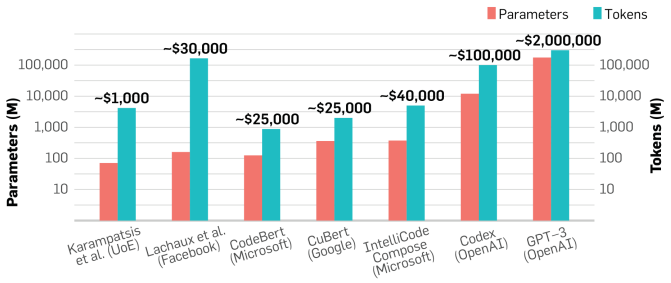


Figure 35: Comparison of large-scale deep learning pretraining models for code generation from model parameter size, data (token) size and estimated training cost. Figure is cited from [307].

- Research of faster algorithms is the most direct and most significant way to reduce computational cost. However, it is a difficult problem to reduce the computational complexities of deep neural network algorithms, NAS/HPO algorithms and feature selection algorithms without losing their performance and it needs the joint effort of researchers from multiple relevant areas including but not limited to applied mathematics, theoretical computer science and machine learning.

- Online learning [308] and incremental learning [309] are machine learning techniques where data are acquired sequentially along time and the model is updated once new data are received without retraining the model on all data, as opposed to batch learning techniques such as traditional deep learning which train a model using the entire training data set at once. The computation time can be saved by model updating rather than model retraining.

- Pretraining-finetuning paradigm could help save overall computation time as well. Specifically, in the pretraining stage, we build a general but computationally intensive deep learning model and in the finetuning stage, we adapt it to various domains, scenarios and tasks in trading. After the computationally expensive pretraining process has been finished, the following finetuning tasks cost significantly less computation power and can be repeatedly performed.

- Model diversification reduces model similarity and reduces information redundancy in model representation, thus decreasing the number of required models in the ensemble. New algorithms are required to improve the diversity of a new model compared with existing ones in model bases. In particular, we need new techniques to enforce the model diversity before or during the training process directly.

- Utilizing informative low-frequency data is a way to enhance computational efficiency without too much effort on engineering techniques or algorithmic tricks. Many types of low-frequency data such as various alternative data, fundamental data and event data are informative for investment decisions, complementing data with relatively high frequency including quote data and limit order book data. Due to the sparsity and small size nature of these low-frequency data, and moreover, easy to compute for modeling.

- Sharing computing power and large model is a potential business/operating model to avoid overlapping investment in computing and modeling infrastructure and to dilute the cost across the whole quant investment industry. Although cloud computing service is an option for computing power sharing, it is not designed specifically for quant investment. In particular, almost all architectures and software systems of cloud computing clusters are not optimized towards the computational demand of quant. Therefore, this industry needs its own professional quant computing sharing mechanism, platform and service. A similar sharing mechanism could be applied to large models as well. As a business model, financial institutes such as hedge funds and mutual funds could buy licenses for the usage of supercomputers and pretrained large models, just like buying financial data.

6.2. Alternative Data Technology

In principle, any data about economic activities have the potential to be used for investment and might be considered by quant researchers. Alternative data, a concept opposite to conventional financial data such as financial statements, stock quotes and limit order books, provides a much broader space for quant research. Although the concept of alternative data has appeared for a decade and is getting more and more popular in quant industry, it is still in the early stages of industry application because of a few reasons. Firstly, many alternative data

sets such as news and research reports have very narrow coverage and low breadth since a news event usually only relates to a limited number of stocks, and you couldn't expect a public company has news or significant events every day. Secondly, some alternative data such as satellite images are too expensive for small funds, and relevant satellite image detection and recognition techniques are not well-established due to limited labeled data and expensive labeling costs. Thirdly, processing and cleaning raw alternative data is a time-consuming "dirty work", and some informative alternative data can not be supplied legally and sustainably. In this subsection, we briefly introduce some examples of alternative data and the corresponding processing techniques, and discuss the difficulties in data acquisition and data aggregation.

6.2.1. Examples of Alternative Data

We list a few types of alternative data as examples and simply discuss the technical difficulties and opportunities for extracting informative signals from them. To avoid repetition, those alternative data already introduced in §4, such as supply chain data, are omitted here.

- Text data from news and social media are one of the most popular alternative data used by investment institutions. The key is to extract traders' sentiment signals correlated with future market trends. Natural language processing techniques such as opinion mining, sentiment modeling and trend tracking could be used to build useful alpha signals or factors.
- Satellite image data have been used by hedge funds and other institutional investors for many purposes. For example, investors could use satellite images to estimate how busy the parking lots of retailers are, providing investment signals for longing or shorting the retailers' stocks. Currently most satellite images are read and analyzed by human, but we believe computer vision techniques such as image recognition and object detection have huge application potential in the future when main technical issues (e.g., cheap labeled data) are solved.
- Merchandise sales data from e-commerce track sale records of consumer products companies from their online channels and estimate their actual income before an upcoming financial statement is reported. This kind of data is usually fragmented and so correct data aggregation is important. In addition, we have to follow relevant data protection laws to make sure the data legitimacy.
- Logistic and inventory data track transportation and logistic activities of a company to estimate their sales performance which might be helpful for predicting the corresponding stocks.
- Credit card or e-payment transaction data is another angle to learn the operation and sales performance of a company or an industry, providing a possible fine-grained analysis through the information of micro-economic activities. Data legitimacy and data sensitivity are the main concerns before using them.
- Geolocation data collect foot-traffic information by GPS or cellphone locators. For example, an investor could use foot-track data for pair-trading arbitrage of Adidas and Nike by analyzing their geolocation activities in different retail shops and predicting spreads between the stock prices of these two companies.

6.2.2. Problems in Data Acquisition

Traditionally, hedge funds and other institutional investors acquire data from third-party data providers or data vendors. With the rapid accumulation of internet data, more and more hedge funds started data collection from websites using web crawlers. However, collecting data by web crawling is suffering unprecedented restrictions because more and more websites are attempting to protect their data using protocol securities and anti-crawler techniques. In addition, many informative data sets such as transaction data are distributed in various banks, e-payment institutes and credit card companies, and few of them allow users to take data out of their own servers considering intellectual property protection, data security and privacy protection. Here we provide a few possible ideas to collect and utilize more protected data legally.

- Certificate techniques for data asset ownership are the foundation for data owners who are willing to share their data and obtain legal income from data users. Decentralized data management techniques such as blockchain techniques provide a potential solution for this mechanism design problem.
- Data encryption techniques [310], including data storage encryption, data transfer encryption and data computation encryption, provide technical solutions for protecting the interest of data owners and data users (e.g., investors), and potentially encourage the willing of data exchange and data sharing.
- Federated learning [311, 312] is a machine learning technique for modeling data sources distributed across multiple decentralized servers each only holding a part of the whole data samples and it realizes information fusion without exchanging those local data. Federal learning techniques have the potential to help investors improve their quant models without spending too much time dealing with data security and privacy issues with massive data owners.

6.2.3. Problems in Data Aggregation

Data aggregation is important for quant to find alpha signals across different types of data. However, data structures of alternative data are diverse and heterogeneous, resulting in difficulties in data aggregation, especially when integrating with traditional financial data.

- Heterogeneity in frequency is a characteristic of many alternative data types such as news data and geolocation data, since they are collected irregularly. Therefore, it's important to align time-series signals occurring at unaligned time points across different instruments and different data types. A series of problems naturally arise. For example, how to build prediction models on these irregular and sparse data samples? How to better estimate and impute missing information in alternative data? All these data processing problems need to be further explored. For example, it is worthwhile to consider

whether data embedding techniques learning "good" representations in lower-dimensional latent spaces help fuse heterogeneous data.

- Differentiating signals from noise is a difficult problem for the extraction of alpha signals from alternative data and should be carefully examined during data processing. Due to limited sample sizes and irregular sample timestamps, identifying true signals out of false positive patterns is more complicated than ever before. Therefore, it is demanding to develop new robustness techniques and significance test techniques to help evaluate the truth of signals and the significance of patterns.

6.3. Financial Knowledge Engineering

As we have introduced in §4, knowledge-driven AI will play an important role in future quantitative finance. Researching new knowledge representation methods, building complete and reliable knowledge bases and developing new knowledge reasoning and decision algorithms are crucial problems in knowledge engineering for financial investment.

6.3.1. Difficulties in Knowledge Engineering

Financial knowledge engineering is an engineering pipeline and research area for constructing effective AI knowledge systems covering knowledge acquisition, knowledge representation, knowledge management, knowledge reasoning, risk analysis and investment decision-making. Given the limitations of popular knowledge techniques, it is necessary for researchers to explore more efficient and more sophisticated methods to better represent and leverage different types of knowledge such as declarative knowledge [313], structured knowledge [314], procedural knowledge [315] etc. In particular, exploring and researching particular knowledge representation techniques for quant and other financial applications are valuable for contributing to the development of investment industry and for finding a broader investment field for quant. Before the wide application of knowledge engineering in quant, a number of technical difficulties need to be solved in the future.

- More sophisticated knowledge representation methods are extremely demanding for building an effective financial knowledge engineering in the future. In particular, we need better data structure and model structure to encode knowledge relevant to all types of economic and financial theories and practical activities.
- More advanced knowledge management system is the requirement of financial knowledge engineering. It requires more research on how to build an automatic and self-updating pipeline of knowledge acquisition, knowledge update, knowledge aggregation and knowledge correction, and on how to implement a reliable and scalable knowledge management system.
- Next-generation knowledge reasoning algorithms should be more explainable, providing more reliable analysis and prediction results. We encourage machine learning researchers to pay more attention to this area, which is part of next-generation AI core techniques.

6.3.2. Knowledge Engineering vs Large Model

Large-scale pretraining models have been widely used in many AI fields especially in natural language processing and computer vision and in multi-modal pattern recognition tasks. For example, large model GPT-3 [316] has about 175 billion parameters, and it could be imagined as a knowledge "crystal ball" for natural language generation and natural language reasoning. In this respect, large pretraining models play a similar role as knowledge engineering, and both of them could provide decision support and content generation function for downstream tasks. So whether large model is a good substitute for knowledge engineering? In our opinion, it is not true. Knowledge engineering has its particular advantage in explicit knowledge representation and knowledge logical reasoning, making the decision process transparent and understandable. Moreover, as the memory module in knowledge engineering, knowledge base has its advantage in flexibility, storing and managing all types of popular data structures including entities, facts and even rules. As a matter of fact, these two technology roadmaps are complementary to each other and have the potential to be integrated into a unified framework to improve the final performance of investment decisions.

6.4. Financial Metaverse & World Model Simulator

In quant research, it is very important to understand the underlying logic and micro-structure of financial markets. For example, we would like to know how the market will react to specific news about a financial statement fraud event, or how much a big order affects the asset price in the market. Unfortunately, empirical studies using historical data usually result in biased conclusions because they do not provide experimental access to all relevant information. In particular, even if certain extreme market events have never happened in history, it doesn't mean they will not happen in the future. What will happen if our trading strategy meets these assumed extreme events? Back-test experiments based on historical data can't answer this question, but we expect that financial metaverse can do it. Financial metaverse aims to build a simulated financial market parallel to real-world financial markets and use it as an experimental environment to simulate situations other than what have happened in real markets.

6.4.1. Financial Metaverse Market Simulator

Daniel Freidman, UCSC Economics Professor, expressed his view that simulation of markets provides a powerful tool to analyze not only individual participant behavior, but also overall market reactions that emerge from the interaction of individual agents [290]. Financial metaverse should be able to support various kinds of research experiments (about traders' behavior and market phenomenon) that are difficult to complete using historical data or trading experiments in real markets. Such a simulator could be used in a number of different scenarios. We list a number of examples as follows.

- Accurately estimating the market impact of a big order at certain market conditions and certain time point

- Analysis of trader behaviors reacting to a particular market event
- Understand the impact of a certain type of traders in markets
- High-precision simulation of transaction cost
- Running treatment-control random experiments to test causal effect to answer "what if" questions against particular historical dates

Besides the above applications, financial metaverse can also be applied to accurately justify causal effects of economic factors. This motivation is achieved by designing and conducting randomized experiments in this simulated market. Moreover, financial metaverse can also provide a fundamental experimental platform for causality engineering discussed in §6.5. Although financial metaverse is of great value for quant research and financial market research, there are many technical difficulties we have to face.

- There is no way to collect fine-grained data identifiable to individual traders, with which market simulation will become much easier.
- The data we have are lack information about the motivation and intent of every trader.
- A simulator can not enumerate and involve all possible factors affecting a market.
- Current computing power can not support high-precision simulation with a large number of trading agents in financial metaverse.

6.4.2. *World Model for Simulation*

A technical problem of financial metaverse is the complexity of the simulation environment from which agents in the same reinforcement learning system are hard to learn, and it makes traditional reinforcement learning algorithms hard to learn millions of weights of a large model. Usually a financial metaverse requires at least thousands of agents, each of which is modeled by a large neural network and plays a specific group of traders with some typical style, to participate in simulated market trading and it aims to recurrent the real-market by a reinforcement learning simulation as precise as possible. This makes regular computing power incapable of completing such accurate simulations. World model [317] provides a potential solution for this difficulty in financial metaverse. It is inspired by the function of human brain which develops a mental model of the world by learning an abstract representation of complex information flushing into the brain. In a world model, the dimension of outputs from the simulator (environment) is reduced using unsupervised learning such as variational autoencoder (VAE). We can use this abstraction to train small network controllers which let the training algorithm search on a small space for credit assignment task, and thus the computation can be accelerated. Figure 36 illustrates the structure of a world model for computer vision problems. The cognition abstraction idea in world model could be used in financial metaverse to solve the simulation computation problem for financial markets.

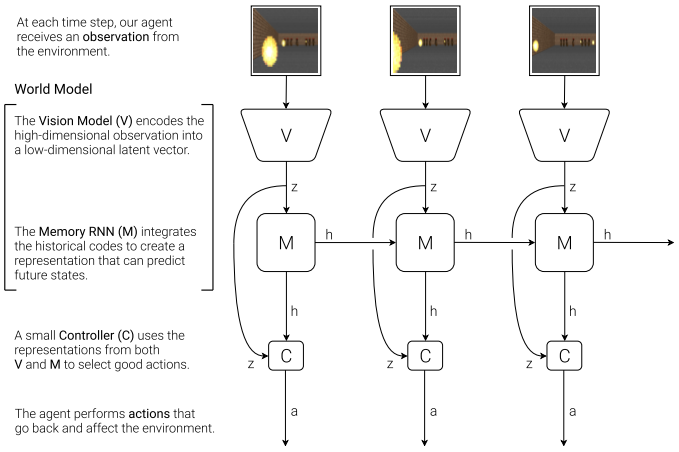


Figure 36: An example of world model architecture. Figure is cited from [317].

6.5. *Cognitive AI & Causality Engineering*

Cognitive AI has been regarded as the future direction of artificial general intelligence (AGI) [318] by many domain experts. In this subsection, we discuss its advantages and technical challenges in investment application, and we propose a new concept *causality engineering* as a potential solution for causal machine learning in AGI.

6.5.1. *Cognitive AI for Investment*

In 2011, Israeli-American psychologist and economist Daniel Kahneman published his book *Thinking, Fast and Slow* [319], introducing two modes of thought for the first time. Specifically, system 1 thinking, driven by instinct and experiences, is a near-instantaneous process and happens automatically, intuitively, and with little effort. On the other hand, system 2 thinking, driven by comprehensive logical reasoning, is slower, more conscious, more deliberative, and requires more effort. Contemporary mainstream AI technology such as deep learning and reinforcement learning is running on a fast track approaching system 1 thinking, and we refer to it as *perceptive AI*. On the contrary, cognitive AI techniques aim to simulate system 2 thinking of human, providing more sophisticated, more logical and more understandable AI solutions. Many pioneering researchers have proposed their opinions and/or solutions for cognitive AI. For example, Gary Marcus proposes "a hybrid, knowledge-driven, cognitive-model-based approach" towards robust artificial intelligence [320]. Yoshua Bengio identifies system 2 deep learning as being able to "understand, reason and generalize beyond training distributions" [321] and proposes corresponding techniques such as causal machine learning and generative flow networks [322] to achieve this goal. Figure 37 illustrates the concepts of system 1 and system 2, and explains their appropriate scenarios and tasks for investment decisions. Different from perceptive AI mainly applied in high-frequency and high-breadth trading tasks, cognitive AI gives quant opportunities to touch those high-capacity but low-frequency investment strategies, including value investing which buys/sells securities that appear underpriced/overpriced through multi-dimensional fundamental analysis and usually holds the

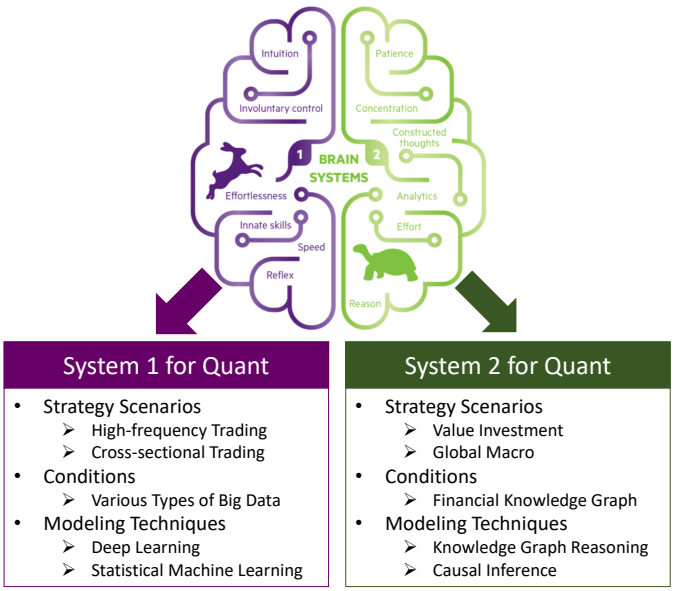


Figure 37: Comparison of System 1 and System 2 in AI technology for quant research.

position over months or even over years, as well as global macro investment which selects assets and establishes portfolio based on the interpretation and prediction of large-scale events related to national economies (interest rate trends, CPIs growth, GDPs growth, unemployment rates, policies, etc.), and international relations (inter-governmental relations, international trade, cross-border payments, etc.) to decide long/short positions in various equity, fixed income, currency, commodities, and futures markets.

6.5.2. *Causality Engineering*

Causal inference [323] and causal machine learning [324, 325] have been regarded as one potential technical route towards AGI [321]. For investment tasks, understanding the true causal relationships among numerous factors is extremely challenging, due to the difficulty in enumerating all possible confounding variables that affect the future trend in one single experiment and give us a misleading conclusion. Therefore, we propose a potential solution through causality engineering, a proposed concept which aims to build and maintain a large-scale causal diagram database storing and managing all known/inferred causal effect relationships and potential confounding variables [326] and causal variables. Different from regular alpha factor bases, a causal diagram database collects as many economic factors and computes their intrinsic conditional probability of causality between each other. Moreover, causality engineering will develop a series of algorithmic tools for different purposes, including testing the statistical significance of causal pairs, testing the effect of potential confounders, and searching significant causal pairs or causal clauses. Leveraging causality engineering, we expect that most main confounding factors disturbing the correct justification of true causal effect factors could be discovered in experiments and could be eliminated correctly using appropriate statistical methods.

6.6. *AI Risk Graph & Systematic Modeling*

The rapid growth of various types of financial big data brings us opportunities to model and analyze financial risk systematically, ranging from macroeconomy to micro-market. We propose the concept of *AI risk graph*, a special financial knowledge graph for recording, computing, analyzing and forecasting financial risks at different hierarchical dimensions, including country, district, industry, sector, etc.

6.6.1. *Risk Graph for Systematic Modeling*

An AI risk graph should be able to represent the causal dependency of different types of risk among public companies, private companies, banks, important individuals and many other economic entities, and represent risk transfer between various entities. Such a graph could be used to systematically model various types of risks at different levels by computing the conditional probability of risk for a specified object (public company, sector, industry, etc) at certain conditions (time, market environment, debt leverage, etc). Various statistical graphical models and machine learning algorithms could be applied to estimate risk conditional probability. Moreover, AI risk graph could help decompose observed risk value in a more scientific and more intuitive way in order to improve the interpretability of risk analysis and risk management.

6.6.2. *Complex Risk Measure for Investment*

Classic risk management techniques such as BARRA [48] risk factor analysis model are used in measuring the overall risk associated with securities relative to the market risks. Specifically, the model decompose the overall risk into a number of exposures from different risk factors with a linearly additive interpretation. However, the limitation of linear risk modeling is obvious, in particular when the prediction model is built with highly nonlinear machine learning sample fitting. How to characterize, measure, and evaluate nonlinear risk is an important research topic in quant. In particular in nonlinear modeling scenarios.

- How to define a reasonable and practically useful nonlinear risk measure?
- How to make sure the nonlinear risk by definition exists and is identifiable?
- How to tell the difference between risk and noise in an extreme market?
- What proportion of overall risk could be explained by linear and nonlinear risk?

6.7. *Spatiotemporal Modeling*

The data structure for stock modeling is typically a tensor with three orthogonal axes: time, stock, and factor. Traditionally, stock strategies are developed either along the time axis (called time-series modeling or temporal modeling) or along the stock axis (called cross-sectional modeling or spatial modeling). These two modeling types have significant differences in strategy development. Specifically, cross-sectional modeling only compares relative strengths of investment signals within

the same cross-section at some time point, and the signal strengths of the same stock at different time points are usually not comparable. Cross-sectional modeling has its advantage in neutralizing market risk automatically by longing top stocks and shorting bottom stocks ranked by cross-sectional trading signals. On the other hand, time-series modeling treats each stock individually, and the trading signals of different stocks at the same time point are not comparable.

6.7.1. Unifying Cross-section & Time-series

A technically difficult but practically feasible idea is to merge cross-sectional modeling and time-series modeling in a unified framework, in order to absorb the advantages from both sides. For this aim, we attempt to provide a few tips on how to build a unified model and what potential difficulties may exist.

- A unified model needs to update stock cross-sections in a very high frequency (in seconds, for example) in order to match the prediction paces of time-series modeling.
- A unified model should be very selective in longing and shorting stock positions at each trading point in order to reduce the turnover rate of strategy so as reduce transaction costs.
- The model should provide a hyperparameter for users to tune the balance between neutralization (alpha-oriented strategy) and absolute return (beta-oriented strategy), according to the design of the portfolio style and the demand of customers.

6.7.2. Spatiotemporal Graph for Quant

Although a unified model aims to combine the advantage of cross-sectional modeling and time-series modeling, the relationships (or interactions) among stocks are hard to be incorporated in it. Spatiotemporal graph modeling could help complement this part of the information. In particular, a spatiotemporal graph [327, 328] can be embedded as a vector of latent factors using graph convolution or graph attention representation learning techniques, the latent vector could be concatenated with the factor vector used in unified modeling to improve the prediction performance.

6.8. Universal Modeling

As data volume and computing power is growing rapidly, large-scale pretrained model has become one of the mainstream AI paradigms in practice. Successful examples such as BERT [329], GPT-3 [316], CLIP [330], Codex [331] and DALL-E [332] have demonstrated the effectiveness of large-scale pretrained models on a number of application domains such as machine translation, multi-modal understanding, creative content generation and so on. Many successful pretraining models have exhibited their universal power to help improve many downstream tasks with different tasks and data sources. We expect this phenomenon could be reproduced in quant applications.

6.8.1. Pretraining-Funetuning Paradigm

The pretraining-finetuning paradigm in AI has demonstrated its success in natural language processing [316] and computer vision [333, 334]. By pretraining a large model on as much data in a self-supervised manner, one can obtain a model extracting the common information across various tasks and use it to achieve superior performances on a series of downstream tasks, compared to task-specific training. We think the pretraining-finetuning paradigm may be transferred to quant scenarios due to a number of reasons.

- Many finance prediction tasks have sufficiently large volumes of data with diversified types and this helps train a large-scale neural network pretraining model during the pretraining stage.
- Many different quant prediction tasks may share the same pattern and information in modeling, especially for high-frequency trading where trading decisions mainly depends on market trends and traders' behaviors in the market. A general pretraining model could learn common patterns across different markets and even across different types of instruments, and these common patterns may help improve the downstream prediction tasks.
- Pretraining-finetuning improves the automation of the modeling process and saves computational cost since we only need to maintain and update a pretrained main model which is a large neural network and adapt it to solve different downstream tasks by finetuninng the main model.

6.8.2. Challenge in Quant Pretraining

The pretraining-finetuning paradigm has to face a number of difficulties if it is applied to quant.

- Investment data have extremely low signal-to-noise ratio, and make the pretraining process difficult to converge to a satisfying solution.
- The design of the quant pretraining process must be careful to avoid using future information. Therefore, the masks in self-supervised pretraining or labels in supervised pretraining can be allocated only on the right-hand side like what GPT-3 does.
- Pretraining should retain both flexibility and versatility. Specifically, we should figure out how to define appropriate labels or how to set appropriate masking structures.

6.9. Robust Modeling

Data noise is the biggest issue in quant modeling. It leads to three problems that negatively affect the robustness and correctness of our model.

1. The signal-to-noise ratio in financial data is extremely low. For example, empirically, the information coefficient of an effective stock cross-sectional alpha model for daily trading lies around the level of 0.1, indicating a signal-to-noise ratio at 10% level, and thus the low signal-to-noise ratio makes it more difficult for machine learning algorithms to tell true patterns out of false positive noise when the sample size is not sufficiently large.
2. It is difficult to correctly and robustly model the distribution of data with heavy noise. For example, the performance of many machine learning models is sensitive to model configurations as well as model hyperparameters when they are

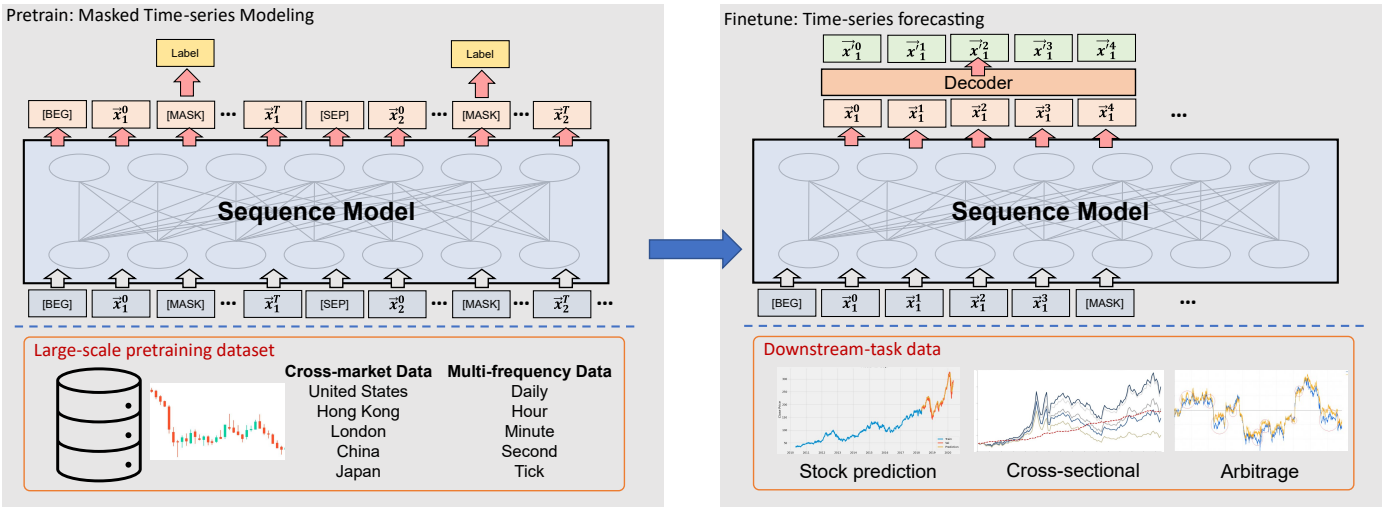


Figure 38: Pretrain-finetune paradigm. Figure is adapted from [329, 335].

trained on very noisy financial data. This makes models very sensitive to sample outliers, time-series change points and the option of parameters, and thus increases the risk of overfitting.

3. The noisy financial data do not follow an independent and identical distribution (i.i.d.) condition which is considered as the underlying assumption of most machine learning algorithms. Since market style is constantly changing as a consequence of an enormous number of factors, it is hard to find a model that persistently generates effective trading signals forever. Such noisy nature of financial data hinders the effectiveness of transferring existing algorithms directly, requiring both theoretical and practical innovations in the study of AI technology.

To address these problems, we suggest researchers considering the following directions in the future.

- Causal effect modeling [336] can be applied to explore the causal effect between factors and financial decisions. By removing the interference from confounding variables, we have the chance to approximate the causal effect by estimating an underlying stable relationship between input factors and output decisions, and reduce the uncertainty of the model. Moreover, causal effect learning techniques could be applied to machine learning modeling in order to improve their out-of-distribution generalization.
- Continual learning techniques [309] aim to cumulatively acquire new information from data without forgetting the knowledge obtained from previous tasks. They could be used to improve out-of-distribution generalization [337] and thus increase model robustness through iteratively retraining with accumulated data over time and frequently updating the prediction model.
- Model ensemble technique is useful in many scenarios to improve prediction stability by combining multiple single models, especially when single models have sufficiently differentiable outputs. Classic ensemble methods include bootstrap aggregating (bagging) [338], boosting [339, 340], stacking [341, 342], Bayesian model averaging [343], etc.
- Model diversification for ensemble expands the diversity of single models prepared to be combined, and the robustness of the ensemble model could benefit from diversification among single models. Various techniques could be applied to increase model diversity and discrepancy, such as model randomization (bootstrap, permutation, dropout [344], etc.), model neutralization, model decorrelation, mixture-of-experts [138], etc.
- Incorporating diversified data from different sources and different patterns helps improve model robustness as well. For example, models trained from fundamental data extracted from financial statements usually complement well with models trained from stock prices and volumes, and a combination of the two types of models may help improve stability.

6.10. *End-to-end Modeling*

As we have introduced in §2.1, the traditional quant research pipeline consists of a number of steps (as shown in the blue blocks in Figure 39), and each step has its own optimization direction. For example, the factor mining module searches "good" factors aiming to find effective alpha signals with significant single-factor IC or back-test profit-and-loss ratio. The modeling module trains machine learning models aiming to minimize some sort of loss functions that measure the difference between labels and prediction outcomes. The portfolio optimization module allocates assets with optimal positions aiming to maximize some sort of value-at-risk (VaR) target. And the trading execution module computes the optimal order size in real-time aiming to minimize the market impact and reduce transaction costs. We can see the optimization goals of these steps are somewhat different from each other. For example, a "good" factor with high IC doesn't necessarily contribute positively to the final model output in a complicated nonlinear relationship. Therefore, it is natural to consider if there exists an end-to-end model taking meta factors as input and trading

orders as output, and if its performance has an advantage compared with traditional separate modeling.

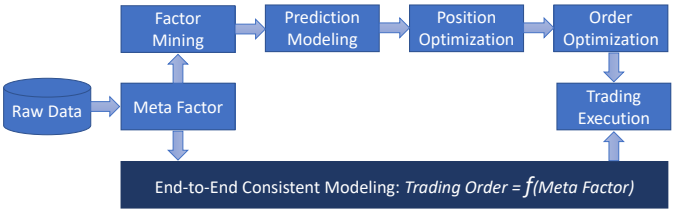


Figure 39: Comparison of traditional quant research pipeline and end-to-end consistent modeling.

6.10.1. End-to-end Consistent Optimization

It is a difficult task to build a consistently optimal model due to technical reasons. Firstly, end-to-end modeling is usually hard to be formulated as a typical supervised learning problem because there is no clear label definition for the whole pipeline. In particular, the steps of portfolio optimization and algorithmic trading are dynamic optimization problems relevant to Markov time dependency which are difficult to be incorporated into a supervised learning framework. Secondly, trading decisions in the order execution step are as frequent as a few seconds, while meta factors are usually updated every couple of minutes or days, and therefore it is difficult to construct "(x, y) pairs" of samples for machine learning models. Thirdly, the noisy financial data make it difficult to train machine learning models difficult, and in particular, it is easy to be stuck at some local optimum during the training process, making the model sensitive to disturbance, noise and outliers. Finally, the computational cost of an end-to-end model is extremely high, which redirects to the challenge of computing power discussed in §6.1.

The opportunity for solutions to consistent modeling relies on the development of new machine learning paradigms satisfying the following requirements.

1. The machine learning model should have a hierarchical structure supporting data and decisions at various time granularities including millisecond-level, second-level, minute-level, day-level, week-level, or even month-level.
2. The computational complexity should be controlled to finish the computation in an affordable time.
3. The optimization direction of the model should depend on predefined labels if available, and should depend on the effect of trading executions measured by typical evaluation criterion for quant.

Based upon the above analysis, we would recommend researchers interested in the problem to think from an existing baseline model under the reinforcement learning framework [73, 74], and pay more attention to fusing data, factors, decisions and executions in multiple time granularities.

6.10.2. Learning Unstructured Data

Another problem in end-to-end learning is how to model unstructured data automatically. Many raw financial data are unstructured in format, says, they are impossible or inappropriate to be formatted as matrices, tensors, or data frames. Examples include order volume distributions on various bid/ask prices from limit order book data, interactions or causal relationships from economic behavior and event data, and investor emotion from news text data. How to effectively train a deep learning quant model end-to-end with a mixture of structured and unstructured financial data is still an open problem to us. New representation learning techniques (especially new information extraction and embedding methods) for unstructured data are needed for future investment research.

7. Conclusion and Perspective

In this article, we proposed the concept of Quant 4.0 which describes what the next-generation quant looks like. We think AI technology will become the core of quant research in the future. In particular, we claim that Automated AI, explainable AI, and knowledge-driven AI are three key components in Quant 4.0, and we think the development of these research areas will not only drive the evolution of quant research but also promote the progress of next-generation AI technology. We emphasize the importance of engineering in Quant 4.0. All three key components can not be implemented at scale without excellent system architecture and powerful computation infrastructure. Furthermore, we summarize 10 main challenges in quant technology, including one challenge about infrastructure, two challenges about financial data, three challenges about AGI technology and quant application, and four challenges about AI quant modeling.

We must emphasize that Quant 4.0 is a dynamic concept and it will evolve and improve itself with the emergence of more and more new technology in the future. We encourage all researchers from related research areas to pay more attention to this interdisciplinary field which may become one of the demands driving the development of next-generation AI technology. To be honest, there is still a long way to go before achieving an ideal Quant 4.0 level since a lot of technical challenges in both artificial intelligence and quant engineering need to be solved. However, the rapid growth of AI technology in the past decade always moves beyond our expectations and brings us exciting progress, brand-new ideas, and more importantly, more confidence to explore this direction. Finally, we hope this perspective article could provide some insights to both academia and industry and encourage more researchers to study and contribute to this interdisciplinary field.

Acknowledgement

This work would not have been possible without the support of International Digital Economy Academy (IDEA). We would like to thank Prof. Qi Liu at Hong Kong University who provides helpful suggestions for the automated AI part of this article, thank Mr. Hang Yuan at IDEA Research for advice on the engineering part and thank Dr. Zhouchi Lin at IDEA Research for advice on the introduction part.

References

[1] A. Zakrzewski, B. Bacchetti, K. Burchardi, D. Frankle, H. Andrew, M. Kahlich, D. Kessler, S. Knobel, S. Kumar, H. Montgomery, E. Palmisani, O. Shipton, A. Soysal, J. Tan, T. Tang, Global Wealth 2022: Standing Still Is Not an Option, Tech. rep., Boston Consulting Group (Jun. 2022).
URL https://www.bcg.com/publications/2022/standing-still-not-an-option

[2] F. J. Fabozzi, The handbook of financial instruments, The Frank J. Fabozzi series., Wiley, Hoboken, N.J, 2002.

[3] P. Blakey, An introduction to investment engineering, IEEE Microwave Magazine 6 (2) (2005) 16–26, conference Name: IEEE Microwave Magazine. doi:10.1109/MMW.2005.1491247.

[4] From the Margins to the Mainstream: Assessment of the Impact Investment Sector and Opportunities to Engage Mainstream Investors (2013).
URL http://wef.ch/1WYVdeG

[5] E. Cheng, Just 10% of trading is regular stock picking, JPMorgan estimates, CNBC (Jun. 2017).
URL https://www.cnbc.com/2017/06/13/death-of-the-human-investor-just-10-percent-of-trading-is-regular-stock-picking-jpmorgan-estimates.html

[6] J. C. Hull, Options, futures, and other derivatives, 6th Edition, Pearson Prentice Hall, Upper Saddle River, NJ [u.a.], 2006.
URL http://gso.gbv.de/DB=2.1/CMD?ACT=SRCHA&SRT=YOP&IKT=1016&TRM=ppn+563580607&sourceid=fbw_bibsonomy

[7] X. Guo, T. L. Lai, H. Shek, S. Po-Shing Wong, Quantitative Trading: Algorithms, Analytics, Data, Models, Optimization, CRC Press LLC, Boca Raton, 2016. doi:10.1201/9781315371580.

[8] L. K. C. Chan, N. Jegadeesh, J. Lakonishok, Momentum Strategies, The Journal of Finance 51 (5) (1996) 1681–1713, publisher: [American Finance Association, Wiley]. doi:10.2307/2329534.
URL http://www.jstor.org/stable/2329534

[9] J. M. Poterba, L. H. Summers, Mean reversion in stock prices: Evidence and Implications, Journal of Financial Economics 22 (1) (1988) 27–59. doi:10.1016/0304-405X(88)90021-9.
URL https://www.sciencedirect.com/science/article/pii/0304405X88900219

[10] A. Pole, Statistical arbitrage : algorithmic trading insights and techniques, Wiley finance., John Wiley & Sons, Hoboken, N.J, 2007, publication Title: Statistical arbitrage : algorithmic trading insights and techniques.

[11] J. D. Koziol, Hedging : principles, practices, and strategies for the financial markets, Wiley, New York, 1990.

[12] A. J. Baird, Option market making: trading and risk analysis for the financial and commodity option markets, Wiley finance editions., Wiley, New York, 1993.

[13] M. D. Gould, M. A. Porter, S. Williams, M. McDonald, D. J. Fenn, S. D. Howison, Limit order books, Quantitative Finance 13 (11) (2013) 1709–1742, publisher: Routledge _eprint: https://doi.org/10.1080/14697688.2013.803148. doi:10.1080/14697688.2013.803148.
URL https://doi.org/10.1080/14697688.2013.803148

[14] E. F. Fama, K. R. French, The Cross-Section of Expected Stock Returns, The Journal of Finance 47 (2) (1992) 427–465, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1540-6261.1992.tb04398.x. doi:10.1111/j.1540-6261.1992.tb04398.x.
URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1540-6261.1992.tb04398.x

[15] R. C. Grinold, The fundamental law of active management, The Journal of Portfolio Management 15 (3) (1989) 30–37, publisher: Institutional Investor Journals Umbrella Section: Primary Article. doi:10.3905/jpm.1989.409211.
URL https://jpm.pm-research.com/content/15/3/30

[16] Central Limit Theorem, in: The Concise Encyclopedia of Statistics, Springer, New York, NY, 2008, pp. 66–68. doi:10.1007/978-0-387-32833-1_50.
URL https://doi.org/10.1007/978-0-387-32833-1_50

[17] L. Bachelier, Théorie de la Spéculation, Annales Scientifiques de L'Ecole Normale Supérieure 17 (1900) 21–88.

[18] E. Thorp, S. Kassouf, Beat the Market: A Scientific Stock Market System, Random House, 1967.

[19] A. Meucci, 'P' Versus 'Q': Differences and Commonalities between the Two Areas of Quantitative Finance (Jan. 2011).
URL https://papers.ssrn.com/abstract=1717163

[20] P. A. Samuelson, Proof That Properly Anticipated Prices Fluctuate Randomly, IMR 6 (2) (1965) 41, num Pages: 9 Place: Cambridge, United States Publisher: Massachusetts Institute of Technology, Cambridge, MA.
URL http://www.proquest.com/docview/214192447/citation/821F08A656E463EPQ/3

[21] P. A. Samuelson, Lifetime portfolio selection by dynamic stochastic programming, in: W. ZIEMBA, R. VICKSON (Eds.), Stochastic Optimization Models in Finance, Academic Press, 1975, pp. 517–524. doi:https://doi.org/10.1016/B978-0-12-780850-5.50044-7.
URL https://www.sciencedirect.com/science/article/pii/B9780127808505500447

[22] R. C. Merton, Lifetime Portfolio Selection under Uncertainty: The Continuous-Time Case, The Review of Economics and Statistics 51 (3) (1969) 247–257, publisher: The MIT Press. doi:10.2307/1926560.
URL http://www.jstor.org/stable/1926560

[23] N. Taleb, Dynamic hedging: managing vanilla and exotic options, Wiley series in financial engineering, Wiley, New York, 1997.

[24] F. Black, M. Scholes, The Pricing of Options and Corporate Liabilities, Journal of Political Economy 81 (3) (1973) 637–654, publisher: University of Chicago Press.
URL https://www.jstor.org/stable/1831029

[25] S. Heston, A closed-form solution for options with stochastic volatility with applications to bond and currency options, Review of Financial Studies 6 (1993) 327–343.

[26] P. S. Hagan, D. Kumar, A. Lesniewski, D. E. Woodward, Managing smile risk, Wilmott Magazine September (2002) 84–108.

[27] H. Buehler, L. Gonon, J. Teichmann, B. Wood, Deep hedging, Quantitative Finance 19 (8) (2019) 1271–1291, publisher: Routledge _eprint: https://doi.org/10.1080/14697688.2019.1571683. doi:10.1080/14697688.2019.1571683.
URL https://doi.org/10.1080/14697688.2019.1571683

[28] J. M. Harrison, S. R. Pliska, Martingales and stochastic integrals in the theory of continuous trading, Stochastic Processes and their Applications 11 (3) (1981) 215–260. doi:10.1016/0304-4149(81)90026-0.
URL https://www.sciencedirect.com/science/article/pii/0304414981900260

[29] D. X. Li, On Default Correlation: A Copula Function Approach, The Journal of Fixed Income 9 (4) (2000) 43–54, publisher: Institutional Investor Journals Umbrella Section: Primary Article. doi:10.3905/jfi.2000.319253.
URL https://jfi.pm-research.com/content/9/4/43

[30] J. Berk, P. DeMarzo, Corporate Finance, 3rd Edition, Pearson, Boston, 2013.

[31] H. Markowitz, Portfolio Selection, The Journal of Finance 7 (1) (1952) 77–91, publisher: [American Finance Association, Wiley]. doi:10.2307/2975974.
URL http://www.jstor.org/stable/2975974

[32] H. M. Markowitz, Portfolio Selection: Efficient Diversification of Investments, Yale University Press, 1959.
URL http://www.jstor.org/stable/j.ctt1bh4c8h

[33] J. L. Treynor, Market Value, Time, and Risk (Aug. 1961). doi:10.2139/ssrn.2600356.
URL https://papers.ssrn.com/abstract=2600356

[34] W. F. Sharpe, Capital Asset Prices: A Theory of Market Equilibrium Under Conditions of Risk*, The Journal of Finance 19 (3) (1964) 425–442, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1540-6261.1964.tb02865.x. doi:10.1111/j.1540-6261.1964.tb02865.x.
URL http://onlinelibrary.wiley.com/doi/abs/10.1111/j.1540-6261.1964.tb02865.x

[35] J. Lintner, The Valuation of Risk Assets and the Selection of Risky Investments in Stock Portfolios and Capital Budgets, The Review of Economics and Statistics 47 (1) (1965) 13–37, publisher: The MIT Press. doi:10.2307/1924119.
URL http://www.jstor.org/stable/1924119

[36] J. Mossin, Equilibrium in a Capital Asset Market, Econometrica 34 (4) (1966) 768–783, publisher: [Wiley, Econometric Society]. doi:10.2307/1910098.

URL http://www.jstor.org/stable/1910098
[37] S. A. Ross, The arbitrage theory of capital asset pricing, Journal of Economic Theory 13 (3) (1976) 341–360. doi:10.1016/0022-0531(76)90046-6.
URL https://www.sciencedirect.com/science/article/pii/0022053176900466
[38] E. F. Fama, K. R. French, A five-factor asset pricing model, Journal of Financial Economics 116 (1) (2015) 1–22. doi:https://doi.org/10.1016/j.jfineco.2014.10.010.
URL https://www.sciencedirect.com/science/article/pii/S0304405X14002323
[39] M. Eichler, Causal Inference in Time Series Analysis, in: Causality, John Wiley & Sons, Ltd, 2012, pp. 327–354, section: 22 _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781119945710.ch22. doi:10.1002/9781119945710.ch22.
URL http://onlinelibrary.wiley.com/doi/abs/10.1002/9781119945710.ch22
[40] C. W. J. Granger, Investigating Causal Relations by Econometric Models and Cross-spectral Methods, Econometrica 37 (3) (1969) 424–438, publisher: [Wiley, Econometric Society]. doi:10.2307/1912791.
URL http://www.jstor.org/stable/1912791
[41] I. Tulchinsky, Introduction to Alpha Design, in: Finding Alphas, John Wiley & Sons, Ltd, 2019, pp. 1–6, section: 1 _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781119571278.ch1. doi:https://doi.org/10.1002/9781119571278.ch1.
URL https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119571278.ch1
[42] X. He, K. Zhao, X. Chu, AutoML: A Survey of the State-of-the-Art, Knowledge-Based Systems 212 (2021) 106622, arXiv: 1908.00709. doi:10.1016/j.knosys.2020.106622.
URL http://arxiv.org/abs/1908.00709
[43] J. Adams, D. Hayunga, S. Mansi, D. Reeb, V. Verardi, Identifying and treating outliers in finance, Financial Management 48 (2) (2019) 345–384, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/fima.12269. doi:10.1111/fima.12269.
URL http://onlinelibrary.wiley.com/doi/abs/10.1111/fima.12269
[44] A. Zheng, A. Casari, Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists, 1st Edition, O'Reilly Media, Inc., 2018.
[45] M. Schnaubelt, A comparison of machine learning model validation schemes for non-stationary time series data, Working Paper 11/2019, FAU Discussion Papers in Economics (2019).
URL https://www.econstor.eu/handle/10419/209136
[46] Y. Nevmyvaka, Y. Feng, M. Kearns, Reinforcement learning for optimized trade execution, in: Proceedings of the 23rd international conference on Machine learning, ICML '06, Association for Computing Machinery, New York, NY, USA, 2006, pp. 673–680. doi:10.1145/1143844.1143929.
URL http://doi.org/10.1145/1143844.1143929
[47] T. Coleman, A Practical Guide to Risk Management (Jul. 2011).
URL https://papers.ssrn.com/abstract=2586032
[48] MSCI, Barra's risk models (1996).
URL https://www.msci.com/www/research-paper/barra-s-risk-models/014972229
[49] F. Black, Noise, The Journal of Finance 41 (3) (1986) 528–543, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1540-6261.1986.tb04513.x. doi:10.1111/j.1540-6261.1986.tb04513.x.
URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1540-6261.1986.tb04513.x
[50] Y. Aït-Sahalia, J. Yu, High frequency market microstructure noise estimates and liquidity measures, The Annals of Applied Statistics 3 (1) (2009) 422 – 457, publisher: Institute of Mathematical Statistics. doi:10.1214/08-AOAS200.
URL https://doi.org/10.1214/08-AOAS200
[51] C. Mancini, Measuring the relevance of the microstructure noise in financial data, Stochastic Processes and their Applications 123 (7) (2013) 2728–2751. doi:https://doi.org/10.1016/j.spa.2013.04.003.
URL https://www.sciencedirect.com/science/article/pii/S0304414913000951
[52] Z. Kakushadze, 101 Formulaic Alphas, arXiv:1601.00991 [q-fin]ArXiv: 1601.00991 (Mar. 2016).
URL http://arxiv.org/abs/1601.00991
[53] F. Feng, X. He, X. Wang, C. Luo, Y. Liu, T.-S. Chua, Temporal Relational Ranking for Stock Prediction, ACM Transactions on Information Systems 37 (2) (2019) 1–30, arXiv: 1809.09441. doi:10.1145/3309547.
URL http://arxiv.org/abs/1809.09441
[54] R. Sawhney, S. Agarwal, A. Wadhwa, R. R. Shah, Spatiotemporal Hypergraph Convolution Network for Stock Movement Forecasting, in: 2020 IEEE International Conference on Data Mining (ICDM), 2020, pp. 482–491, iSSN: 2374-8486. doi:10.1109/ICDM50108.2020.00057.
[55] Y. Wu, D. Magazzeni, M. Veloso, How Robust are Limit Order Book Representations under Data Perturbation?, in: ICML Workshop on Representation Learning for Finance and E-Commerce Applications, 2021.
[56] W. La Cava, P. Orzechowski, B. Burlacu, F. de Franca, M. Virgolin, Y. Jin, M. Kommenda, J. Moore, Contemporary Symbolic Regression Methods and their Relative Performance, in: J. Vanschoren, S. Yeung (Eds.), Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks, Vol. 1, 2021.
URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/file/c0c7c76d30bd3dcaefc96f40275bdc0a-Paper-round1.pdf
[57] A. Rashid, M. Fayyaz, M. Karim, Investor sentiment, momentum, and stock returns: an examination for direct and indirect effects, Economic Research-Ekonomska Istraživanja 32 (1) (2019) 2638–2656, publisher: Routledge _eprint: https://doi.org/10.1080/1331677X.2019.1650652. doi:10.1080/1331677X.2019.1650652.
URL https://doi.org/10.1080/1331677X.2019.1650652
[58] Z. Abdul Karim, F. S. R. Muhamad Fahmi, B. Abdul Karim, M. A. Shokr, Market sentiments and firm-level equity returns: panel evidence of Malaysia, Economic Research-Ekonomska Istraživanja 35 (1) (2022) 5253–5272, publisher: Routledge _eprint: https://doi.org/10.1080/1331677X.2021.2025126. doi:10.1080/1331677X.2021.2025126.
URL https://doi.org/10.1080/1331677X.2021.2025126
[59] C. Andrieu, N. de Freitas, A. Doucet, M. I. Jordan, An Introduction to MCMC for Machine Learning, Machine Learning 50 (1) (2003) 5–43. doi:10.1023/A:1020281327116.
URL https://doi.org/10.1023/A:1020281327116
[60] Y. Jin, W. Fu, J. Kang, J. Guo, J. Guo, Bayesian Symbolic Regression, arXiv:1910.08892 [stat] (Jan. 2020). doi:10.48550/arXiv.1910.08892.
URL http://arxiv.org/abs/1910.08892
[61] T. Chen, W. Chen, L. Du, An Empirical Study of Financial Factor Mining Based on Gene Expression Programming, in: 2021 4th International Conference on Advanced Electronic Materials, Computers and Software Engineering (AEMCSE), 2021, pp. 1113–1117. doi:10.1109/AEMCSE51986.2021.00228.
[62] L. Biggio, T. Bendinelli, A. Neitz, A. Lucchi, G. Parascandolo, Neural Symbolic Regression that scales, in: M. Meila, T. Zhang (Eds.), Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event, Vol. 139 of Proceedings of Machine Learning Research, PMLR, 2021, pp. 936–945.
URL http://proceedings.mlr.press/v139/biggio21a.html
[63] G. Martius, C. H. Lampert, Extrapolation and learning equations, in: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings, OpenReview.net, 2017.
URL https://openreview.net/forum?id=BkgRp0FYe
[64] M. T. Ahvanooey, Q. Li, M. Wu, S. Wang, A Survey of Genetic Programming and Its Applications, KSII Transactions on Internet and Information Systems (TIIS) 13 (4) (2019) 1765–1794, publisher: Korean Society for Internet Information. doi:10.3837/tiis.2019.04.002.
URL https://koreascience.kr/article/JAKO201919761177651.page
[65] S. Katoch, S. S. Chauhan, V. Kumar, A review on genetic algorithm: past, present, and future, Multimedia Tools and Applications 80 (5) (2021) 8091–8126. doi:10.1007/s11042-020-10139-6.
URL https://doi.org/10.1007/s11042-020-10139-6
[66] J. Lee, Y. Lee, J. Kim, A. Kosiorek, S. Choi, Y. W. Teh, Set Trans-

former: A Framework for Attention-based Permutation-Invariant Neural Networks, in: Proceedings of the 36th International Conference on Machine Learning, PMLR, 2019, pp. 3744–3753, iSSN: 2640-3498.
URL https://proceedings.mlr.press/v97/lee19d.html

[67] K. Hornik, M. B. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Networks 2 (5) (1989) 359–366.

[68] A. Thakkar, K. Chaudhari, A comprehensive survey on deep neural networks for stock market: The need, challenges, and future directions, Expert Systems with Applications 177 (2021) 114800. doi:10.1016/j.eswa.2021.114800.
URL https://www.sciencedirect.com/science/article/pii/S0957417421002414

[69] Z. Hu, Y. Zhao, M. Khushi, A Survey of Forex and Stock Price Prediction Using Deep Learning, Applied System Innovation 4 (1) (2021) 9. doi:10.3390/asi4010009.
URL https://www.mdpi.com/2571-5577/4/1/9

[70] W. Jiang, Applications of deep learning in stock market prediction: recent progress, Expert Systems with Applications 184 (2021) 115537, arXiv: 2003.01859. doi:10.1016/j.eswa.2021.115537.
URL http://arxiv.org/abs/2003.01859

[71] O. B. Sezer, M. U. Gudelek, A. M. Ozbayoglu, Financial time series forecasting with deep learning : A systematic literature review: 2005–2019, Applied Soft Computing 90 (2020) 106181. doi:10.1016/j.asoc.2020.106181.
URL https://www.sciencedirect.com/science/article/pii/S1568494620301216

[72] I. Sutskever, O. Vinyals, Q. V. Le, Sequence to Sequence Learning with Neural Networks, in: Advances in Neural Information Processing Systems, Vol. 27, Curran Associates, Inc., 2014.
URL https://proceedings.neurips.cc/paper/2014/hash/a14ac55a4f27472c5d894ec1c3c743d2-Abstract.html

[73] J. Wang, Y. Zhang, K. Tang, J. Wu, Z. Xiong, AlphaStock: A Buying-Winners-and-Selling-Losers Investment Strategy using Interpretable Deep Reinforcement Attention Networks, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, ACM, Anchorage AK USA, 2019, pp. 1900–1908. doi:10.1145/3292500.3330647.
URL https://dl.acm.org/doi/10.1145/3292500.3330647

[74] Z. Wang, B. Huang, S. Tu, K. Zhang, L. Xu, DeepTrader: A Deep Reinforcement Learning Approach for Risk-Return Balanced Portfolio Management with Market Conditions Embedding, Proceedings of the AAAI Conference on Artificial Intelligence 35 (1) (2021) 643–650, number: 1.
URL https://ojs.aaai.org/index.php/AAAI/article/view/16144

[75] T. Elsken, J. H. Metzen, F. Hutter, Efficient Multi-Objective Neural Architecture Search via Lamarckian Evolution, in: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019, OpenReview.net, 2019.
URL https://openreview.net/forum?id=ByME42AqK7

[76] M. Zhang, H. Li, S. Pan, X. Chang, C. Zhou, Z. Ge, S. Su, One-Shot Neural Architecture Search: Maximising Diversity to Overcome Catastrophic Forgetting, IEEE Transactions on Pattern Analysis and Machine Intelligence 43 (9) (2021) 2921–2935, conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence. doi:10.1109/TPAMI.2020.3035351.

[77] M.-A. Zöller, M. F. Huber, Benchmark and Survey of Automated Machine Learning Frameworks, Journal of Artificial Intelligence Research 70 (2021) 409–472. doi:10.1613/jair.1.11854.
URL http://doi.org/10.1613/jair.1.11854

[78] H. Mendoza, A. Klein, M. Feurer, J. T. Springenberg, F. Hutter, Towards Automatically-Tuned Neural Networks, in: Proceedings of the Workshop on Automatic Machine Learning, PMLR, 2016, pp. 58–65, iSSN: 1938-7228.
URL https://proceedings.mlr.press/v64/mendoza_towards_2016.html

[79] B. Baker, O. Gupta, R. Raskar, N. Naik, Accelerating Neural Architecture Search using Performance Prediction, in: 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Workshop Track Proceedings, OpenReview.net, 2018.
URL https://openreview.net/forum?id=HJqk3N1vG

[80] T. Elsken, J. H. Metzen, F. Hutter, Neural architecture search: a survey, The Journal of Machine Learning Research 20 (1) (2019) 1997–2017.

[81] B. Zoph, V. Vasudevan, J. Shlens, Q. V. Le, Learning Transferable Architectures for Scalable Image Recognition, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 8697–8710, iSSN: 2575-7075. doi:10.1109/CVPR.2018.00907.

[82] Z. Zhong, Z. Yang, B. Deng, J. Yan, W. Wu, J. Shao, C.-L. Liu, BlockQNN: Efficient Block-Wise Neural Network Architecture Generation, IEEE Trans. Pattern Anal. Mach. Intell. 43 (7) (2021) 2314–2328. doi:10.1109/TPAMI.2020.2969193.
URL https://doi.org/10.1109/TPAMI.2020.2969193

[83] M. Feurer, F. Hutter, Hyperparameter Optimization, in: F. Hutter, L. Kotthoff, J. Vanschoren (Eds.), Automated Machine Learning: Methods, Systems, Challenges, The Springer Series on Challenges in Machine Learning, Springer International Publishing, Cham, 2019, pp. 3–33. doi:10.1007/978-3-030-05318-5_1.
URL https://doi.org/10.1007/978-3-030-05318-5_1

[84] J. Bergstra, Y. Bengio, Random Search for Hyper-Parameter Optimization, Journal of Machine Learning Research 13 (10) (2012) 281–305.
URL http://jmlr.org/papers/v13/bergstra12a.html

[85] L. Li, A. Talwalkar, Random Search and Reproducibility for Neural Architecture Search, in: Proceedings of The 35th Uncertainty in Artificial Intelligence Conference, PMLR, 2020, pp. 367–377, iSSN: 2640-3498.
URL https://proceedings.mlr.press/v115/li20c.html

[86] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. V. Le, A. Kurakin, Large-Scale Evolution of Image Classifiers, in: Proceedings of the 34th International Conference on Machine Learning, PMLR, 2017, pp. 2902–2911, iSSN: 2640-3498.
URL https://proceedings.mlr.press/v70/real17a.html

[87] L. Xie, A. Yuille, Genetic CNN, in: 2017 IEEE International Conference on Computer Vision (ICCV), 2017, pp. 1388–1397, iSSN: 2380-7504. doi:10.1109/ICCV.2017.154.

[88] M. Suganuma, S. Shirakawa, T. Nagao, A Genetic Programming Approach to Designing Convolutional Neural Network Architectures, in: J. Lang (Ed.), Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden, ijcai.org, 2018, pp. 5369–5373. doi:10.24963/ijcai.2018/755.

[89] E. Real, A. Aggarwal, Y. Huang, Q. V. Le, Regularized Evolution for Image Classifier Architecture Search, Proceedings of the AAAI Conference on Artificial Intelligence 33 (01) (2019) 4780–4789, number: 01. doi:10.1609/aaai.v33i01.33014780.
URL https://ojs.aaai.org/index.php/AAAI/article/view/4405

[90] L. Tani, D. Rand, C. Veelken, M. Kadastik, Evolutionary algorithms for hyperparameter optimization in machine learning for application in high energy physics, The European Physical Journal C 81 (2) (2021) 170. doi:10.1140/epjc/s10052-021-08950-y.
URL https://doi.org/10.1140/epjc/s10052-021-08950-y

[91] B. Zoph, Q. V. Le, Neural architecture search with reinforcement learning, arXiv preprint arXiv:1611.01578 (2016).

[92] H. S. Jomaa, J. Grabocka, L. Schmidt-Thieme, Hyp-RL : Hyperparameter Optimization by Reinforcement Learning, arXiv:1906.11527 [cs, stat] (Jun. 2019). doi:10.48550/arXiv.1906.11527.
URL http://arxiv.org/abs/1906.11527

[93] S. Falkner, A. Klein, F. Hutter, BOHB: Robust and Efficient Hyperparameter Optimization at Scale, in: J. G. Dy, A. Krause (Eds.), Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018, Vol. 80 of Proceedings of Machine Learning Research, PMLR, 2018, pp. 1436–1445.
URL http://proceedings.mlr.press/v80/falkner18a.html

[94] A. Klein, S. Falkner, S. Bartels, P. Hennig, F. Hutter, Fast Bayesian Optimization of Machine Learning Hyperparameters on Large Datasets, in: Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, PMLR, 2017, pp. 528–536, iSSN: 2640-3498.
URL https://proceedings.mlr.press/v54/klein17a.html

[95] C. White, W. Neiswanger, Y. Savani, BANANAS: Bayesian Optimization with Neural Architectures for Neural Architecture Search,

in: Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021, AAAI Press, 2021, pp. 10293–10301.
URL https://ojs.aaai.org/index.php/AAAI/article/view/17233

[96] K. Kandasamy, W. Neiswanger, J. Schneider, B. Poczos, E. P. Xing, Neural Architecture Search with Bayesian Optimisation and Optimal Transport, in: Advances in Neural Information Processing Systems, Vol. 31, Curran Associates, Inc., 2018.
URL https://proceedings.neurips.cc/paper/2018/hash/f33ba15effa5c10e873bf3842afb46a6-Abstract.html

[97] H. Liu, K. Simonyan, Y. Yang, DARTS: Differentiable Architecture Search, in: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019, OpenReview.net, 2019.
URL https://openreview.net/forum?id=S1eYHoC5FX

[98] H. Cai, L. Zhu, S. Han, ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware, in: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019, OpenReview.net, 2019.
URL https://openreview.net/forum?id=HylVB3AqYm

[99] Y. Bengio, Gradient-Based Optimization of Hyperparameters, Neural Computation 12 (8) (2000) 1889–1900, conference Name: Neural Computation. doi:10.1162/089976600300015187.

[100] D. Maclaurin, D. Duvenaud, R. P. Adams, Gradient-based hyperparameter optimization through reversible learning, in: Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37, ICML'15, JMLR.org, Lille, France, 2015, pp. 2113–2122.

[101] L. Franceschi, M. Donini, P. Frasconi, M. Pontil, Forward and Reverse Gradient-Based Hyperparameter Optimization, in: Proceedings of the 34th International Conference on Machine Learning, PMLR, 2017, pp. 1165–1173, iSSN: 2640-3498.
URL https://proceedings.mlr.press/v70/franceschi17a.html

[102] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. L. Yuille, J. Huang, K. Murphy, Progressive Neural Architecture Search, in: V. Ferrari, M. Hebert, C. Sminchisescu, Y. Weiss (Eds.), Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part I, Vol. 11205 of Lecture Notes in Computer Science, Springer, 2018, pp. 19–35. doi:10.1007/978-3-030-01246-5_2.
URL https://doi.org/10.1007/978-3-030-01246-5_2

[103] T. Wei, C. Wang, Y. Rui, C. W. Chen, Network Morphism, in: Proceedings of The 33rd International Conference on Machine Learning, PMLR, 2016, pp. 564–572, iSSN: 1938-7228.
URL https://proceedings.mlr.press/v48/wei16.html

[104] M. Li, Y. Liu, X. Liu, Q. Sun, X. You, H. Yang, Z. Luan, L. Gan, G. Yang, D. Qian, The Deep Learning Compiler: A Comprehensive Survey, IEEE Transactions on Parallel and Distributed Systems 32 (3) (2021) 708–727, conference Name: IEEE Transactions on Parallel and Distributed Systems. doi:10.1109/TPDS.2020.3030548.

[105] Y. Cheng, D. Wang, P. Zhou, T. Zhang, Model Compression and Acceleration for Deep Neural Networks: The Principles, Progress, and Challenges, IEEE Signal Processing Magazine 35 (1) (2018) 126–136, conference Name: IEEE Signal Processing Magazine. doi:10.1109/MSP.2017.2765695.

[106] T. Choudhary, V. Mishra, A. Goswami, J. Sarangapani, A comprehensive survey on model compression and acceleration, Artificial Intelligence Review 53 (7) (2020) 5113–5155. doi:10.1007/s10462-020-09816-7.
URL https://doi.org/10.1007/s10462-020-09816-7

[107] T. Chen, T. Moreau, Z. Jiang, L. Zheng, E. Q. Yan, H. Shen, M. Cowan, L. Wang, Y. Hu, L. Ceze, C. Guestrin, A. Krishnamurthy, TVM: An Automated End-to-End Optimizing Compiler for Deep Learning, in: A. C. Arpaci-Dusseau, G. Voelker (Eds.), 13th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2018, Carlsbad, CA, USA, October 8-10, 2018, USENIX Association, 2018, pp. 578–594.
URL https://www.usenix.org/conference/osdi18/presentation/chen

[108] C. Lattner, M. Amini, U. Bondhugula, A. Cohen, A. Davis, J. Pienaar, R. Riddle, T. Shpeisman, N. Vasilache, O. Zinenko, MLIR: scaling compiler infrastructure for domain specific computation, in: Proceedings of the 2021 IEEE/ACM International Symposium on Code Generation and Optimization, CGO '21, IEEE Press, Virtual Event, Republic of Korea, 2021, pp. 2–14. doi:10.1109/CGO51591.2021.9370308.
URL http://doi.org/10.1109/CGO51591.2021.9370308

[109] S. Cyphers, A. K. Bansal, A. Bhiwandiwalla, J. Bobba, M. Brookhart, A. Chakraborty, W. Constable, C. Convey, L. Cook, O. Kanawi, R. Kimball, J. Knight, N. Korovaiko, V. Kumar, Y. Lao, C. R. Lishka, J. Menon, J. Myers, S. A. Narayana, A. Procter, T. J. Webb, Intel nGraph: An Intermediate Representation, Compiler, and Executor for Deep Learning, arXiv:1801.08058 [cs] (Jan. 2018). doi:10.48550/arXiv.1801.08058.
URL http://arxiv.org/abs/1801.08058

[110] A. V. Aho, R. Sethi, J. D. Ullman, Compilers, Principles, Techniques, and Tools, Addison-Wesley, 1986.

[111] K. D. Cooper, L. T. Simpson, C. A. Vick, Operator strength reduction, ACM Transactions on Programming Languages and Systems 23 (5) (2001) 603–625. doi:10.1145/504709.504710.
URL http://doi.org/10.1145/504709.504710

[112] G. A. Kildall, A unified approach to global program optimization, in: Proceedings of the 1st annual ACM SIGACT-SIGPLAN symposium on Principles of programming languages, POPL '73, Association for Computing Machinery, New York, NY, USA, 1973, pp. 194–206. doi:10.1145/512927.512945.
URL https://doi.org/10.1145/512927.512945

[113] S. P. Vanderwiel, D. J. Lilja, Data prefetch mechanisms, ACM Computing Surveys 32 (2) (2000) 174–199. doi:10.1145/358923.358939.
URL http://doi.org/10.1145/358923.358939

[114] V. Sarkar, R. Thekkath, A general framework for iteration-reordering loop transformations, ACM SIGPLAN Notices 27 (7) (1992) 175–187. doi:10.1145/143103.143132.
URL http://doi.org/10.1145/143103.143132

[115] V. Volkov, Understanding Latency Hiding on GPUs, Ph.D. thesis, UC Berkeley (2016).
URL https://escholarship.org/uc/item/1wb7f3h4

[116] R. Allen, Optimizing compilers for modern architectures: a dependence-based approach, Morgan Kaufmann, San Francisco, Calif, 2002.

[117] M. J. Wolfe, High performance compilers for parallel computing, Addison-Wesley, Redwood City, Calif, 1996.

[118] R. M. Karp, R. E. Miller, S. Winograd, The Organization of Computations for Uniform Recurrence Equations, Journal of the ACM 14 (3) (1967) 563–590, publisher: ACM.
URL http://dl.acm.org/citation.cfm?id=321418

[119] L. Lamport, The Parallel Execution of DO Loops, Communications of the ACM 17 (2) (1974) 83–93, publisher: ACM.
URL http://research.microsoft.com/en-us/um/people/lamport/pubs/do-loops.pdf

[120] S. Han, J. Pool, J. Tran, W. Dally, Learning both weights and connections for efficient neural network, Advances in neural information processing systems 28 (2015).

[121] S. Han, H. Mao, W. J. Dally, Deep Compression: Compressing Deep Neural Network with Pruning, Trained Quantization and Huffman Coding, in: Y. Bengio, Y. LeCun (Eds.), 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings, 2016.
URL http://arxiv.org/abs/1510.00149

[122] G. Hinton, O. Vinyals, J. Dean, Distilling the Knowledge in a Neural Network, arXiv:1503.02531 [cs, stat]ArXiv: 1503.02531 (Mar. 2015).
URL http://arxiv.org/abs/1503.02531

[123] X. Zhang, J. Zou, X. Ming, K. He, J. Sun, Efficient and accurate approximations of nonlinear convolutional networks, in: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 1984–1992, iSSN: 1063-6919. doi:10.1109/CVPR.2015.7298809.

[124] X. Yu, T. Liu, X. Wang, D. Tao, On Compressing Deep Models by Low Rank and Sparse Decomposition, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 67–76, iSSN: 1063-6919. doi:10.1109/CVPR.2017.15.

[125] J. Gou, B. Yu, S. J. Maybank, D. Tao, Knowledge Distillation: A Survey, International Journal of Computer Vision 129 (6) (2021) 1789–1819. doi:10.1007/s11263-021-01453-z.
URL https://doi.org/10.1007/s11263-021-01453-z

[126] V. Belle, I. Papantonis, Principles and Practice of Explainable Machine Learning, Frontiers in Big Data 4 (2021).
URL https://www.frontiersin.org/articles/10.3389/fdata.2021.688969

[127] W. J. Murdoch, C. Singh, K. Kumbier, R. Abbasi-Asl, B. Yu, Definitions, methods, and applications in interpretable machine learning, Proceedings of the National Academy of Sciences 116 (44) (2019) 22071–22080, publisher: Proceedings of the National Academy of Sciences. doi:10.1073/pnas.1900654116.
URL https://www.pnas.org/doi/10.1073/pnas.1900654116

[128] C. Rudin, Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead, Nature Machine Intelligence 1 (5) (2019) 206–215, number: 5 Publisher: Nature Publishing Group. doi:10.1038/s42256-019-0048-x.
URL https://www.nature.com/articles/s42256-019-0048-x

[129] M. T. Ribeiro, S. Singh, C. Guestrin, Model-Agnostic Interpretability of Machine Learning, CoRR abs/1606.05386, arXiv: 1606.05386 (2016).
URL http://arxiv.org/abs/1606.05386

[130] S. Towfighi, pySRURGS - a python package for symbolic regression by uniform random global search, Journal of Open Source Software 4 (2019) 1675. doi:10.21105/joss.01675.

[131] Y. Meshalkin, A. Shakirov, E. Popov, D. Koroteev, I. Gurbatova, Robust well-log based determination of rock thermal conductivity through machine learning, Geophysical Journal International 222 (May 2020). doi:10.1093/gji/ggaa209.

[132] M.-X. Wang, D. Huang, G. Wang, D.-Q. Li, SS-XGBoost: A Machine Learning Framework for Predicting Newmark Sliding Displacements of Slopes, Journal of Geotechnical and Geoenvironmental Engineering 146 (2020) 04020074. doi:10.1061/(ASCE)GT.1943-5606.0002297.

[133] J. R. Quinlan, Induction of decision trees, Machine Learning 1 (1) (1986) 81–106. doi:10.1007/BF00116251.
URL https://doi.org/10.1007/BF00116251

[134] L. Breiman, Random Forests, Machine Learning 45 (1) (2001) 5–32. doi:10.1023/A:1010933404324.
URL https://doi.org/10.1023/A:1010933404324

[135] J. H. Friedman, Greedy Function Approximation: A Gradient Boosting Machine, The Annals of Statistics 29 (5) (2001) 1189–1232, publisher: Institute of Mathematical Statistics.
URL http://www.jstor.org/stable/2699986

[136] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, T.-Y. Liu, LightGBM: A Highly Efficient Gradient Boosting Decision Tree, in: Advances in Neural Information Processing Systems, Vol. 30, Curran Associates, Inc., 2017.
URL https://papers.nips.cc/paper/2017/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html

[137] T. Chen, C. Guestrin, XGBoost: A Scalable Tree Boosting System, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, Association for Computing Machinery, New York, NY, USA, 2016, pp. 785–794. doi:10.1145/2939672.2939785.
URL http://doi.org/10.1145/2939672.2939785

[138] S. Masoudnia, R. Ebrahimpour, Mixture of experts: a literature survey, Artificial Intelligence Review 42 (2) (2014) 275–293. doi:10.1007/s10462-012-9338-y.
URL https://doi.org/10.1007/s10462-012-9338-y

[139] T. Hofmann, B. Schölkopf, A. J. Smola, Kernel methods in machine learning, The Annals of Statistics 36 (3) (2008) 1171 – 1220, publisher: Institute of Mathematical Statistics. doi:10.1214/009053607000000677.
URL https://doi.org/10.1214/009053607000000677

[140] C. Cortes, V. Vapnik, Support-vector networks, Machine Learning 20 (3) (1995) 273–297. doi:10.1007/BF00994018.
URL https://doi.org/10.1007/BF00994018

[141] V. N. Vapnik, Statistical learning theory, Adaptive and learning systems for signal processing, communications, and control., Wiley, New York, 1998.

[142] Y.-H. H. Tsai, S. Bai, M. Yamada, L.-P. Morency, R. Salakhutdinov, Transformer Dissection: An Unified Understanding for Transformer's Attention via the Lens of Kernel, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Association for Computational Linguistics, Hong Kong, China, 2019, pp. 4344–4353. doi:10.18653/v1/D19-1443.
URL https://aclanthology.org/D19-1443

[143] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, in: Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17, Curran Associates Inc., Red Hook, NY, USA, 2017, pp. 6000–6010.

[144] L. Rabiner, B. Juang, An introduction to hidden Markov models, IEEE ASSP Magazine 3 (1) (1986) 4–16, conference Name: IEEE ASSP Magazine. doi:10.1109/MASSP.1986.1165342.

[145] J. D. Lafferty, A. McCallum, F. C. N. Pereira, Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data, in: Proceedings of the Eighteenth International Conference on Machine Learning, ICML '01, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2001, pp. 282–289.

[146] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural computation 9 (8) (1997) 1735–1780.

[147] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, Doha, Qatar, 2014, pp. 1724–1734. doi:10.3115/v1/D14-1179.
URL https://aclanthology.org/D14-1179

[148] I. J. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, Cambridge, MA, USA, 2016.

[149] P. Kontschieder, M. Fiterau, A. Criminisi, S. R. Bulò, Deep Neural Decision Forests, in: 2015 IEEE International Conference on Computer Vision (ICCV), 2015, pp. 1467–1475, iSSN: 2380-7504. doi:10.1109/ICCV.2015.172.

[150] W. Song, C. Shi, Z. Xiao, Z. Duan, Y. Xu, M. Zhang, J. Tang, AutoInt: Automatic Feature Interaction Learning via Self-Attentive Neural Networks, in: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM '19, Association for Computing Machinery, New York, NY, USA, 2019, pp. 1161–1170. doi:10.1145/3357384.3357925.
URL http://doi.org/10.1145/3357384.3357925

[151] R. Saleem, B. Yuan, F. Kurugollu, A. Anjum, L. Liu, Explaining deep neural networks: A survey on the global interpretation methods, Neurocomputing 513 (2022) 165–180. doi:10.1016/j.neucom.2022.09.129.
URL https://www.sciencedirect.com/science/article/pii/S0925231222012218

[152] M. T. Ribeiro, S. Singh, C. Guestrin, "Why Should I Trust You?": Explaining the Predictions of Any Classifier, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, Association for Computing Machinery, New York, NY, USA, 2016, pp. 1135–1144. doi:10.1145/2939672.2939778.
URL http://doi.org/10.1145/2939672.2939778

[153] M. T. Ribeiro, S. Singh, C. Guestrin, Anchors: High-Precision Model-Agnostic Explanations, Proceedings of the AAAI Conference on Artificial Intelligence 32 (1), number: 1 (Apr. 2018). doi:10.1609/aaai.v32i1.11491.
URL https://ojs.aaai.org/index.php/AAAI/article/view/11491

[154] C. Molnar, Interpretable Machine Learning: A Guide for Making Black Box Models Explainable, 2nd Edition, 2022.
URL https://christophm.github.io/interpretable-ml-book

[155] D. W. Apley, J. Zhu, Visualizing the effects of predictor variables in black box supervised learning models, Journal of the Royal Statistical Society Series B 82 (4) (2020) 1059–1086, publisher: Royal Statistical Society.
URL https://econpapers.repec.org/article/blajorssb/v_3a82_3ay_3a2020_3ai_3a4_3ap_3a1059-1086.htm

[156] A. Fisher, C. Rudin, F. Dominici, All Models are Wrong, but Many are

Useful: Learning a Variable's Importance by Studying an Entire Class of Prediction Models Simultaneously, Journal of Machine Learning Research 20 (177) (2019) 1–81.
URL http://jmlr.org/papers/v20/18-760.html

[157] J. H. Friedman, B. E. Popescu, Predictive Learning via Rule Ensembles, The Annals of Applied Statistics 2 (3) (2008) 916–954, publisher: Institute of Mathematical Statistics.
URL http://www.jstor.org/stable/30245114

[158] G. Hooker, Discovering additive structure in black box functions, in: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '04, Association for Computing Machinery, New York, NY, USA, 2004, pp. 575–580. doi:10.1145/1014052.1014122.
URL http://doi.org/10.1145/1014052.1014122

[159] B. M. Greenwell, B. C. Boehmke, A. J. McCarthy, A Simple and Effective Model-Based Variable Importance Measure, arXiv:1805.04755 [cs, stat] (May 2018). doi:10.48550/arXiv.1805.04755.
URL http://arxiv.org/abs/1805.04755

[160] C. Schwartzenberg, T. M. v. Engers, Y. Li, The fidelity of global surrogates in interpretable Machine Learning, in: BNAIC/BeneLearn 2020, 2020.
URL https://bnaic.liacs.leidenuniv.nl/wordpress/wp-content/uploads/papers/BNAICBENELEARN_2020_Final_paper_59.pdf

[161] A. Goldstein, A. Kapelner, J. Bleich, E. Pitkin, Peeking Inside the Black Box: Visualizing Statistical Learning With Plots of Individual Conditional Expectation, Journal of Computational and Graphical Statistics 24 (1) (2015) 44–65, publisher: Taylor & Francis _eprint: https://doi.org/10.1080/10618600.2014.907095. doi:10.1080/10618600.2014.907095.
URL https://doi.org/10.1080/10618600.2014.907095

[162] R. Tibshirani, Regression Shrinkage and Selection via the Lasso, Journal of the Royal Statistical Society. Series B (Methodological) 58 (1) (1996) 267–288, publisher: [Royal Statistical Society, Wiley].
URL http://www.jstor.org/stable/2346178

[163] S. M. Lundberg, S.-I. Lee, A Unified Approach to Interpreting Model Predictions, in: Advances in Neural Information Processing Systems, Vol. 30, Curran Associates, Inc., 2017.
URL https://papers.nips.cc/paper/2017/hash/8a20a8621978632d76c43dfd28b67767-Abstract.html

[164] L. S. Shapley, 17. A Value for n-Person Games, in: H. W. Kuhn, A. W. Tucker (Eds.), Contributions to the Theory of Games (AM-28), Volume II, Princeton University Press, Princeton, 1953, pp. 307–318. doi:doi:10.1515/9781400881970-018.
URL https://doi.org/10.1515/9781400881970-018

[165] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, D. Batra, Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization, in: 2017 IEEE International Conference on Computer Vision (ICCV), 2017, pp. 618–626, iSSN: 2380-7504. doi:10.1109/ICCV.2017.74.

[166] K. Guo, Y. Sun, X. Qian, Can investor sentiment be used to predict the stock price? Dynamic analysis based on China stock market, Physica A: Statistical Mechanics and its Applications 469 (2017) 390–396. doi:https://doi.org/10.1016/j.physa.2016.11.114.
URL https://www.sciencedirect.com/science/article/pii/S0378437116309384

[167] Systematic Methods for Classifying Equities (Aug. 2018).
URL https://www.winton.com/research/systematic-methods-for-classifying-equities

[168] B. Kulis, Metric Learning: A Survey, Foundations and Trends® in Machine Learning 5 (4) (2013) 287–364, publisher: Now Publishers, Inc. doi:10.1561/2200000019.
URL https://www.nowpublishers.com/article/Details/MAL-019

[169] M. Kaya, H. S. Bilge, Deep Metric Learning: A Survey, Symmetry 11 (9) (2019) 1066, number: 9 Publisher: Multidisciplinary Digital Publishing Institute. doi:10.3390/sym11091066.
URL https://www.mdpi.com/2073-8994/11/9/1066

[170] Y. Zhu, W. Xu, J. Zhang, Q. Liu, S. Wu, L. Wang, Deep Graph Structure Learning for Robust Representations: A Survey, arXiv:2103.03036 [cs]ArXiv: 2103.03036 (Mar. 2021).
URL http://arxiv.org/abs/2103.03036

[171] K. Hou, Industry Information Diffusion and the Lead-Lag Effect in Stock Returns (Nov. 2003). doi:10.2139/ssrn.463005.
URL https://papers.ssrn.com/abstract=463005

[172] Y. Li, T. Wang, B. Sun, C. Liu, Detecting the lead–lag effect in stock markets: definition, patterns, and investment strategies, Financial Innovation 8 (1) (2022) 1–36, number: 1 Publisher: SpringerOpen. doi:10.1186/s40854-022-00356-3.
URL https://jfin-swufe.springeropen.com/articles/10.1186/s40854-022-00356-3

[173] N. Fan, Z.-P. Fan, Y. Li, M. Li, Does the lead-lag effect exist in stock markets?, Applied Economics Letters 29 (10) (2022) 895–900, publisher: Routledge _eprint: https://doi.org/10.1080/13504851.2021.1897068. doi:10.1080/13504851.2021.1897068.
URL https://doi.org/10.1080/13504851.2021.1897068

[174] L. Bottou, J. Peters, J. Quiñonero-Candela, D. X. Charles, D. M. Chickering, E. Portugaly, D. Ray, P. Simard, E. Snelson, Counterfactual Reasoning and Learning Systems: The Example of Computational Advertising, Journal of Machine Learning Research 14 (101) (2013) 3207–3260.
URL http://jmlr.org/papers/v14/bottou13a.html

[175] 2020 stock market crash, page Version ID: 1108282892 (Sep. 2022).
URL https://en.wikipedia.org/w/index.php?title=2020_stock_market_crash&oldid=1108282892

[176] Z. Hu, W. Liu, J. Bian, X. Liu, T.-Y. Liu, Listening to Chaotic Whispers: A Deep Learning Framework for News-oriented Stock Trend Prediction, arXiv:1712.02136 [cs, q-fin]ArXiv: 1712.02136 (Feb. 2019).
URL http://arxiv.org/abs/1712.02136

[177] V. Sakalauskas, D. Kriksciuniene, Research of the Calendar Effects in Stock Returns, in: W. Abramowicz, D. Flejter (Eds.), Business Information Systems Workshops, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 69–78.

[178] G. Bonne, J. Wang, H. Zhang, Machine Learning Factors: Capturing Non Linearities in Linear Factor Models, Tech. rep., MSCI (Mar. 2021).
URL https://www.msci.com/www/research-report/machine-learning-factors/02410413451

[179] H. Sharma, Hierarchical Clustering (Apr. 2021).
URL https://harshsharma1091996.medium.com/hierarchical-clustering-996745fe656b

[180] D. Müllner, Modern hierarchical, agglomerative clustering algorithms, arXiv:1109.2378 [cs, stat] (Sep. 2011).
URL http://arxiv.org/abs/1109.2378

[181] Knowledge-based systems, page Version ID: 1111001447 (Sep. 2022).
URL https://en.wikipedia.org/w/index.php?title=Knowledge-based_systems&oldid=1111001447

[182] F. Hayes-Roth, D. A. D. A. Waterman, D. B. Lenat, Building expert systems, Reading, Mass. : Addison-Wesley Pub. Co., 1983.
URL http://archive.org/details/buildingexpertsy00temd

[183] D. Cheng, F. Yang, X. Wang, Y. Zhang, L. Zhang, Knowledge Graph-based Event Embedding Framework for Financial Quantitative Investments, in: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '20, Association for Computing Machinery, New York, NY, USA, 2020, pp. 2221–2230. doi:10.1145/3397271.3401427.
URL http://doi.org/10.1145/3397271.3401427

[184] J. Shinavier, K. Branson, W. Zhang, S. Dastgheib, Y. Gao, B. Arsintescu, F. Özcan, E. Meij, Panel: Knowledge Graph Industry Applications, in: Companion Proceedings of The 2019 World Wide Web Conference, WWW '19, Association for Computing Machinery, New York, NY, USA, 2019, p. 676. doi:10.1145/3308560.3317711.
URL http://doi.org/10.1145/3308560.3317711

[185] A. Singhal, Introducing the Knowledge Graph: things, not strings (May 2012).
URL https://blog.google/products/search/introducing-knowledge-graph-things-not/

[186] S. Ji, S. Pan, E. Cambria, P. Marttinen, P. S. Yu, A Survey on Knowledge Graphs: Representation, Acquisition, and Applications, IEEE Transactions on Neural Networks and Learning Systems 33 (2) (2022) 494–514, conference Name: IEEE Transactions on Neural Networks and Learning Systems. doi:10.1109/TNNLS.2021.3070843.

[187] Knowledge representation and reasoning, page Version ID: 1121765398

(Nov. 2022).
URL https://en.wikipedia.org/w/index.php?title=Knowledge_representation_and_reasoning&oldid=1121765398
[188] J. Sowa, Semantic Networks (1992).
URL http://www.jfsowa.com/pubs/semnet.htm
[189] F. Lehmann, Semantic networks, Computers & Mathematics with Applications 23 (2) (1992) 1–50. doi:10.1016/0898-1221(92)90135-5.
URL https://www.sciencedirect.com/science/article/pii/0898122192901355
[190] Semantic triple, page Version ID: 1068928735 (Jan. 2022).
URL https://en.wikipedia.org/w/index.php?title=Semantic_triple&oldid=1068928735
[191] Resource Description Framework (RDF) Model and Syntax Specification.
URL https://www.w3.org/TR/PR-rdf-syntax/
[192] A. Kamath, R. Das, A Survey on Semantic Parsing, in: 1st Conference on Automated Knowledge Base Construction, AKBC 2019, Amherst, MA, USA, May 20-22, 2019, 2019. doi:10.24432/C5WC7D.
[193] Propositional Logic.
URL https://iep.utm.edu/prop-log/
[194] C. J. H. Mann, The Description Logic Handbook - Theory, Implementation and Applications, Kybernetes 32 (9/10) (2003) 1563–1563, place: London Publisher: Emerald Group Publishing Limited. doi:10.1108/k.2003.06732iae.006.
[195] First-order logic, page Version ID: 1121783215 (Nov. 2022).
URL https://en.wikipedia.org/w/index.php?title=First-order_logic&oldid=1121783215
[196] A. Newell, J. C. Shaw, H. A. Simon, Report on a general problem-solving program, in: IFIP Congress, 1959.
[197] A. Horn, On sentences which are true of direct unions of algebras1, The Journal of Symbolic Logic 16 (1) (1951) 14–21, publisher: Cambridge University Press. doi:10.2307/2268661.
URL https://www.cambridge.org/core/journals/journal-of-symbolic-logic/article/abs/on-sentences-which-are-true-of-direct-unions-of-algebras1/DF348CB269B06D6702DA3AE4DCF38C39
[198] P. Jackson, Introduction to expert systems, 3rd Edition, International computer science series., Addison-Wesley, Harlow, England ;, 1999.
[199] M. Minsky, A framework for representing knowledge, in: The Psychology of Computer Vision, McGraw-Hill, 1975.
URL https://web.media.mit.edu/~minsky/papers/Frames/frames.html
[200] J. Martin, Knowledge Engineering Environment: KEE (1988).
URL https://purl.stanford.edu/sv429hd6966
[201] W. A. Woods, J. G. Schmolze, The KL-ONE family, Computers & Mathematics with Applications 23 (2) (1992) 133–177. doi:10.1016/0898-1221(92)90139-9.
URL https://www.sciencedirect.com/science/article/pii/0898122192901399
[202] D. B. Lenat, R. V. Guha, Building Large Knowledge-Based Systems; Representation and Inference in the Cyc Project, 1st Edition, Addison-Wesley Longman Publishing Co., Inc., USA, 1989.
[203] RDF - Semantic Web Standards (Feburary 2014).
URL https://www.w3.org/RDF/
[204] OWL - Semantic Web Standards (December 2012).
URL https://www.w3.org/OWL/
[205] T. Berners-Lee, J. Hendler, O. Lassila, The Semantic Web, Scientific American 284 (5) (2001) 34–43.
URL http://www.sciam.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21
[206] What is a knowledge graph?
URL https://www.jean-delahousse.net/en/graphe-de-connaissance-ontologie-vocabulaires-controles/
[207] E. F. Kendall, Ontology engineering, Synthesis lectures on the semantic web, theory and technology ; #18, Morgan & Claypool Publishers, San Rafael, California, 2019.
[208] L. A. Galárraga, C. Teflioudi, K. Hose, F. Suchanek, AMIE: association rule mining under incomplete evidence in ontological knowledge bases, in: Proceedings of the 22nd international conference on World Wide Web, WWW '13, Association for Computing Machinery, New York, NY, USA, 2013, pp. 413–422. doi:10.1145/2488388.2488425.
URL http://doi.org/10.1145/2488388.2488425
[209] A. Neelakantan, B. Roth, A. McCallum, Compositional Vector Space Models for Knowledge Base Completion, in: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), Association for Computational Linguistics, Beijing, China, 2015, pp. 156–166. doi:10.3115/v1/P15-1016.
URL https://aclanthology.org/P15-1016
[210] R. Yangarber, R. Grishman, P. Tapanainen, S. Huttunen, Automatic Acquisition of Domain Knowledge for Information Extraction, in: COLING 2000 Volume 2: The 18th International Conference on Computational Linguistics, 2000.
URL https://aclanthology.org/C00-2136
[211] C. Welty, J. W. Murdock, Towards Knowledge Acquisition from Information Extraction, in: I. Cruz, S. Decker, D. Allemang, C. Preist, D. Schwabe, P. Mika, M. Uschold, L. M. Aroyo (Eds.), The Semantic Web - ISWC 2006, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2006, pp. 709–722. doi:10.1007/11926078_51.
[212] I. Muhammad, A. Kearney, C. Gamble, F. Coenen, P. Williamson, Open Information Extraction for Knowledge Graph Construction, in: G. Kotsis, A. M. Tjoa, I. Khalil, L. Fischer, B. Moser, A. Mashkoor, J. Sametinger, A. Fensel, J. Martinez-Gil (Eds.), Database and Expert Systems Applications, Communications in Computer and Information Science, Springer International Publishing, Cham, 2020, pp. 103–113. doi:10.1007/978-3-030-59028-4_10.
[213] V. Yadav, S. Bethard, A Survey on Recent Advances in Named Entity Recognition from Deep Learning models, in: Proceedings of the 27th International Conference on Computational Linguistics, Association for Computational Linguistics, Santa Fe, New Mexico, USA, 2018, pp. 2145–2158.
URL https://aclanthology.org/C18-1182
[214] S. Pawar, G. K. Palshikar, P. Bhattacharyya, Relation Extraction : A Survey, arXiv:1712.05191 [cs] (Dec. 2017). doi:10.48550/arXiv.1712.05191.
URL http://arxiv.org/abs/1712.05191
[215] D. Jurafsky, J. Martin, Semantic Role Labeling, in: Speech and Language Processing, 3rd Edition, 2021.
URL https://web.stanford.edu/~jurafsky/slp3/19.pdf
[216] W. Shen, J. Wang, J. Han, Entity Linking with a Knowledge Base: Issues, Techniques, and Solutions, IEEE Transactions on Knowledge and Data Engineering 27 (2) (2015) 443–460, conference Name: IEEE Transactions on Knowledge and Data Engineering. doi:10.1109/TKDE.2014.2327028.
[217] D. Zeng, K. Liu, S. Lai, G. Zhou, J. Zhao, Relation Classification via Convolutional Deep Neural Network, in: Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers, Dublin City University and Association for Computational Linguistics, Dublin, Ireland, 2014, pp. 2335–2344.
URL https://aclanthology.org/C14-1220
[218] Y. Shen, X. Huang, Attention-Based Convolutional Neural Network for Semantic Relation Extraction, in: Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, The COLING 2016 Organizing Committee, Osaka, Japan, 2016, pp. 2526–2536.
URL https://aclanthology.org/C16-1238
[219] Y. Zhang, P. Qi, C. D. Manning, Graph Convolution over Pruned Dependency Trees Improves Relation Extraction, in: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Brussels, Belgium, 2018, pp. 2205–2215. doi:10.18653/v1/D18-1244.
URL https://aclanthology.org/D18-1244
[220] M. Miwa, Y. Sasaki, Modeling Joint Entity and Relation Extraction with Table Representation, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, Doha, Qatar, 2014, pp. 1858–1869. doi:10.3115/v1/D14-1200.
URL https://aclanthology.org/D14-1200
[221] M. Miwa, M. Bansal, End-to-End Relation Extraction using LSTMs on Sequences and Tree Structures, in: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1:

Long Papers), Association for Computational Linguistics, Berlin, Germany, 2016, pp. 1105–1116. doi:10.18653/v1/P16-1105.
URL https://aclanthology.org/P16-1105

[222] A. Katiyar, C. Cardie, Going out on a limb: Joint Extraction of Entity Mentions and Relations without Dependency Trees, in: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Vancouver, Canada, 2017, pp. 917–928. doi:10.18653/v1/P17-1085.
URL https://aclanthology.org/P17-1085

[223] R. Ng, V. S. Subrahmanian, Probabilistic logic programming, Information and Computation 101 (2) (1992) 150–201. doi:10.1016/0890-5401(92)90061-J.
URL https://www.sciencedirect.com/science/article/pii/089054019290061J

[224] M. Richardson, P. Domingos, Markov logic networks, Machine Learning 62 (1) (2006) 107–136. doi:10.1007/s10994-006-5833-1.
URL https://doi.org/10.1007/s10994-006-5833-1

[225] J. Cussens, Parameter Estimation in Stochastic Logic Programs, Machine Learning 44 (3) (2001) 245–271, company: Springer Distributor: Springer Institution: Springer Label: Springer Number: 3 Publisher: Kluwer Academic Publishers. doi:10.1023/A:1010924021315.
URL http://link.springer.com/article/10.1023/A:1010924021315

[226] W. W. Cohen, TensorLog: A Differentiable Deductive Database, CoRR abs/1605.06523, arXiv: 1605.06523 (2016).
URL http://arxiv.org/abs/1605.06523

[227] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, O. Yakhnenko, Translating Embeddings for Modeling Multi-relational Data, in: Advances in Neural Information Processing Systems, Vol. 26, Curran Associates, Inc., 2013.
URL https://papers.nips.cc/paper/2013/hash/1cecc7a77928ca8133fa24680a88d2f9-Abstract.html

[228] Y. Lin, Z. Liu, M. Sun, Y. Liu, X. Zhu, Learning Entity and Relation Embeddings for Knowledge Graph Completion, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 29, 2015, number: 1. doi:10.1609/aaai.v29i1.9491.
URL https://ojs.aaai.org/index.php/AAAI/article/view/9491

[229] R. Socher, D. Chen, C. D. Manning, A. Y. Ng, Reasoning with neural tensor networks for knowledge base completion, in: Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1, NIPS'13, Curran Associates Inc., Red Hook, NY, USA, 2013, pp. 926–934.

[230] T. Trouillon, J. Welbl, S. Riedel, E. Gaussier, G. Bouchard, Complex Embeddings for Simple Link Prediction, in: M.-F. Balcan, K. Q. Weinberger (Eds.), Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016, Vol. 48 of JMLR Workshop and Conference Proceedings, JMLR.org, 2016, pp. 2071–2080.
URL http://proceedings.mlr.press/v48/trouillon16.html

[231] Z. Sun, Z.-H. Deng, J.-Y. Nie, J. Tang, RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space, in: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019, OpenReview.net, 2019.
URL https://openreview.net/forum?id=HkgEQnRqYQ

[232] S. He, K. Liu, G. Ji, J. Zhao, Learning to Represent Knowledge Graphs with Gaussian Embedding, in: Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, CIKM '15, Association for Computing Machinery, New York, NY, USA, 2015, pp. 623–632. doi:10.1145/2806416.2806502.
URL http://doi.org/10.1145/2806416.2806502

[233] H. Xiao, M. Huang, X. Zhu, TransG : A Generative Model for Knowledge Graph Embedding, in: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Berlin, Germany, 2016, pp. 2316–2325. doi:10.18653/v1/P16-1219.
URL https://aclanthology.org/P16-1219

[234] H. Xiao, M. Huang, X. Zhu, From one point to a manifold: knowledge graph embedding for precise link prediction, in: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI'16, AAAI Press, New York, New York, USA, 2016, pp. 1315–1321.

[235] T. Dettmers, P. Minervini, P. Stenetorp, S. Riedel, Convolutional 2D Knowledge Graph Embeddings, in: S. A. McIlraith, K. Q. Weinberger (Eds.), Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018, AAAI Press, 2018, pp. 1811–1818.
URL https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17366

[236] L. Guo, Z. Sun, W. Hu, Learning to Exploit Long-term Relational Dependencies in Knowledge Graphs, in: K. Chaudhuri, R. Salakhutdinov (Eds.), Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA, Vol. 97 of Proceedings of Machine Learning Research, PMLR, 2019, pp. 2505–2514.
URL http://proceedings.mlr.press/v97/guo19c.html

[237] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, M. Welling, Modeling Relational Data with Graph Convolutional Networks, in: A. Gangemi, R. Navigli, M.-E. Vidal, P. Hitzler, R. Troncy, L. Hollink, A. Tordai, M. Alam (Eds.), The Semantic Web, Lecture Notes in Computer Science, Springer International Publishing, Cham, 2018, pp. 593–607. doi:10.1007/978-3-319-93417-4_38.

[238] W. Zhang, J. Chen, J. Li, Z. Xu, J. Z. Pan, H. Chen, Knowledge Graph Reasoning with Logics and Embeddings: Survey and Perspective, arXiv:2202.07412 [cs] (Feb. 2022).
URL http://arxiv.org/abs/2202.07412

[239] Y. Lin, Z. Liu, H. Luan, M. Sun, S. Rao, S. Liu, Modeling Relation Paths for Representation Learning of Knowledge Bases, in: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Lisbon, Portugal, 2015, pp. 705–714. doi:10.18653/v1/D15-1082.
URL https://aclanthology.org/D15-1082

[240] B. Ding, Q. Wang, B. Wang, L. Guo, Improving Knowledge Graph Embedding Using Simple Constraints, in: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Melbourne, Australia, 2018, pp. 110–121. doi:10.18653/v1/P18-1011.
URL https://aclanthology.org/P18-1011

[241] S. Guo, Q. Wang, L. Wang, B. Wang, L. Guo, Jointly Embedding Knowledge Graphs and Logical Rules, in: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Austin, Texas, 2016, pp. 192–202. doi:10.18653/v1/D16-1019.
URL https://aclanthology.org/D16-1019

[242] S. Guo, Q. Wang, L. Wang, B. Wang, L. Guo, Knowledge Graph Embedding with Iterative Guidance from Soft Rules, in: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence, AAAI'18/IAAI'18/EAAI'18, AAAI Press, 2018, event-place: New Orleans, Louisiana, USA.

[243] Y. Wang, H. Wang, J. He, W. Lu, S. Gao, TAGAT: Type-Aware Graph Attention neTworks for reasoning over knowledge graphs, Knowledge-Based Systems 233 (2021) 107500. doi:10.1016/j.knosys.2021.107500.
URL https://www.sciencedirect.com/science/article/pii/S0950705121007620

[244] Z. Zhang, F. Zhuang, M. Qu, F. Lin, Q. He, Knowledge Graph Embedding with Hierarchical Relation Structure, in: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Brussels, Belgium, 2018, pp. 3198–3207. doi:10.18653/v1/D18-1358.
URL https://aclanthology.org/D18-1358

[245] K.-W. Chang, W.-t. Yih, B. Yang, C. Meek, Typed Tensor Decomposition of Knowledge Bases for Relation Extraction, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, Doha, Qatar, 2014, pp. 1568–1579. doi:10.3115/v1/D14-1165.

URL https://aclanthology.org/D14-1165
[246] M. Qu, J. Tang, Probabilistic Logic Neural Networks for Reasoning, in: Advances in Neural Information Processing Systems, Vol. 32, Curran Associates, Inc., 2019.
URL https://proceedings.neurips.cc/paper/2019/hash/13e5ebb0fa112fe1b31a1067962d74a7-Abstract.html
[247] A. P. Dempster, N. M. Laird, D. B. Rubin, Maximum Likelihood from Incomplete Data Via the EM Algorithm, Journal of the Royal Statistical Society: Series B (Methodological) 39 (1) (1977) 1–22. doi:10.1111/j.2517-6161.1977.tb01600.x.
URL https://onlinelibrary.wiley.com/doi/10.1111/j.2517-6161.1977.tb01600.x
[248] Z. Wei, J. Zhao, K. Liu, Z. Qi, Z. Sun, G. Tian, Large-scale Knowledge Base Completion: Inferring via Grounding Network Sampling over Selected Instances, in: Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, CIKM '15, Association for Computing Machinery, New York, NY, USA, 2015, pp. 1331–1340. doi:10.1145/2806416.2806513.
URL http://doi.org/10.1145/2806416.2806513
[249] T. Rocktäschel, S. Riedel, End-to-end Differentiable Proving, in: Advances in Neural Information Processing Systems, Vol. 30, Curran Associates, Inc., 2017.
URL https://papers.nips.cc/paper/2017/hash/b2ab001909a8a6f04b51920306046ce5-Abstract.html
[250] W. Y. Wang, W. W. Cohen, Learning First-Order Logic Embeddings via Matrix Factorization, in: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI'16, AAAI Press, 2016, pp. 2132–2138, event-place: New York, New York, USA.
[251] F. Yang, Z. Yang, W. W. Cohen, Differentiable Learning of Logical Rules for Knowledge Base Reasoning, in: Advances in Neural Information Processing Systems, Vol. 30, Curran Associates, Inc., 2017.
URL https://papers.nips.cc/paper/2017/hash/0e55666a4ad822e0e34299df3591d979-Abstract.html
[252] S. Deng, N. Zhang, W. Zhang, J. Chen, J. Z. Pan, H. Chen, Knowledge-Driven Stock Trend Prediction and Explanation via Temporal Convolutional Network, in: Companion Proceedings of The 2019 World Wide Web Conference, ACM, San Francisco USA, 2019, pp. 678–685. doi:10.1145/3308560.3317701.
URL https://dl.acm.org/doi/10.1145/3308560.3317701
[253] X. Ding, Y. Zhang, T. Liu, J. Duan, Knowledge-Driven Event Embedding for Stock Prediction, in: Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, The COLING 2016 Organizing Committee, Osaka, Japan, 2016, pp. 2133–2142.
URL https://www.aclweb.org/anthology/C16-1201
[254] A. Sil, A. Yates, Re-ranking for joint named-entity recognition and linking, in: Proceedings of the 22nd ACM international conference on Information & Knowledge Management, CIKM '13, Association for Computing Machinery, New York, NY, USA, 2013, pp. 2369–2374. doi:10.1145/2505515.2505601.
URL http://doi.org/10.1145/2505515.2505601
[255] S. Bai, J. Z. Kolter, V. Koltun, An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling, CoRR abs/1803.01271, arXiv: 1803.01271 (2018).
URL http://arxiv.org/abs/1803.01271
[256] J. Long, Z. Chen, W. He, T. Wu, J. Ren, An integrated framework of deep learning and knowledge graph for prediction of stock price trend: An application in Chinese stock exchange market, Applied Soft Computing 91 (2020) 106205. doi:10.1016/j.asoc.2020.106205.
URL https://linkinghub.elsevier.com/retrieve/pii/S1568494620301459
[257] A. Grover, J. Leskovec, node2vec: Scalable Feature Learning for Networks, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, Association for Computing Machinery, New York, NY, USA, 2016, pp. 855–864. doi:10.1145/2939672.2939754.
URL http://doi.org/10.1145/2939672.2939754
[258] G. Ang, E.-P. Lim, Learning Knowledge-Enriched Company Embeddings for Investment Management, in: Proceedings of the Second ACM International Conference on AI in Finance, ICAIF '21, Association for Computing Machinery, New York, NY, USA, 2021, event-place: Virtual Event. doi:10.1145/3490354.3494390.
URL https://doi-org.lib.ezproxy.ust.hk/10.1145/3490354.3494390
[259] W. Xu, W. Liu, C. Xu, J. Bian, J. Yin, T.-Y. Liu, REST: Relational Event-driven Stock Trend Forecasting, in: Proceedings of the Web Conference 2021, WWW '21, Association for Computing Machinery, New York, NY, USA, 2021, pp. 1–10. doi:10.1145/3442381.3450032.
URL http://doi.org/10.1145/3442381.3450032
[260] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauly, M. J. Franklin, S. Shenker, I. Stoica, Resilient Distributed Datasets: A {Fault-Tolerant} Abstraction for {In-Memory} Cluster Computing, in: 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12), 2012, pp. 15–28.
URL https://www.usenix.org/conference/nsdi12/technical-sessions/presentation/zaharia
[261] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, I. Stoica, Spark: Cluster Computing with Working Sets, in: 2nd USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 10), 2010.
URL https://www.usenix.org/conference/hotcloud-10/spark-cluster-computing-working-sets
[262] Q. Wang, Z. Xu, Z. Chen, Y. Wang, S. Liu, H. Qu, Visual analysis of discrimination in machine learning, IEEE Transactions on Visualization and Computer Graphics 27 (2) (2020) 1470–1480, publisher: IEEE.
[263] S. Ghemawat, H. Gobioff, S.-T. Leung, The Google File System, in: Proceedings of the 19th ACM Symposium on Operating Systems Principles, Bolton Landing, NY, 2003, pp. 20–43.
[264] K. Shvachko, H. Kuang, S. Radia, R. Chansler, The Hadoop Distributed File System, in: 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), 2010, pp. 1–10, iSSN: 2160-1968. doi:10.1109/MSST.2010.5496972.
[265] W. Fedus, B. Zoph, N. Shazeer, Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity, arXiv:2101.03961 [cs]ArXiv: 2101.03961 (Jan. 2021).
URL http://arxiv.org/abs/2101.03961
[266] G. Bender, P.-J. Kindermans, B. Zoph, V. Vasudevan, Q. Le, Understanding and Simplifying One-Shot Architecture Search, in: Proceedings of the 35th International Conference on Machine Learning, PMLR, 2018, pp. 550–559, iSSN: 2640-3498.
URL https://proceedings.mlr.press/v80/bender18a.html
[267] S. A. Weil, S. A. Brandt, E. L. Miller, D. D. E. Long, C. Maltzahn, Ceph: a scalable, high-performance distributed file system, in: Proceedings of the 7th symposium on Operating systems design and implementation, OSDI '06, USENIX Association, USA, 2006, pp. 307–320.
[268] M. Technologies, Infiniband technology overview, InfiniBand White Paper (2008).
URL https://network.nvidia.com/related-docs/whitepapers/WP_InfiniBand_Technology_Overview.pdf
[269] Z. Zhang, C. Chang, H. Lin, Y. Wang, R. Arora, X. Jin, Is Network the Bottleneck of Distributed Training?, in: Proceedings of the Workshop on Network Meets AI & ML, NetAI '20, Association for Computing Machinery, New York, NY, USA, 2020, pp. 8–13. doi:10.1145/3405671.3405810.
URL http://doi.org/10.1145/3405671.3405810
[270] E. F. Codd, A Relational Model of Data for Large Shared Data Banks, Commun. ACM 13 (6) (1970) 377–387. doi:10.1145/362384.362685.
URL http://doi.acm.org/10.1145/362384.362685
[271] D. Namiot, Time Series Databases, in: DAMDID/RCDL, 2015.
[272] S. K. Jensen, T. B. Pedersen, C. Thomsen, Time Series Management Systems: A Survey, IEEE Transactions on Knowledge and Data Engineering 29 (11) (2017) 2581–2600. doi:10.1109/TKDE.2017.2740932.
URL http://ieeexplore.ieee.org/document/8012550/
[273] F. Gessert, W. Wingerath, S. Friedrich, N. Ritter, NoSQL database systems: a survey and decision guidance, Computer Science - Research and Development 32 (3) (2017) 353–365. doi:10.1007/s00450-016-0334-3.
URL https://doi.org/10.1007/s00450-016-0334-3
[274] R. kumar Kaliyar, Graph databases: A survey, in: Communication & Automation International Conference on Computing, 2015, pp. 785–790. doi:10.1109/CCAA.2015.7148480.

[275] R. Angles, C. Gutierrez, Survey of graph database models, ACM Computing Surveys 40 (1) (2008) 1:1–1:39. doi:10.1145/1322432.1322433.
URL http://doi.org/10.1145/1322432.1322433

[276] K.-L. Tan, Q. Cai, B. C. Ooi, W.-F. Wong, C. Yao, H. Zhang, In-memory Databases: Challenges and Opportunities From Software and Hardware Perspectives, ACM SIGMOD Record 44 (2) (2015) 35–40. doi:10.1145/2814710.2814717.
URL http://doi.org/10.1145/2814710.2814717

[277] J. Dean, S. Ghemawat, MapReduce: simplified data processing on large clusters, Communications of the ACM 51 (1) (2008) 107–113. doi:10.1145/1327452.1327492.
URL http://doi.org/10.1145/1327452.1327492

[278] Apache Hadoop.
URL https://hadoop.apache.org/

[279] J. L. Hennessy, D. A. Patterson, Computer Architecture, Fifth Edition: A Quantitative Approach, 5th Edition, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2011.

[280] Y. Lu, J. Cheng, D. Yan, H. Wu, Large-scale distributed graph computing systems: an experimental evaluation, Proceedings of the VLDB Endowment 8 (3) (2014) 281–292. doi:10.14778/2735508.2735517.
URL http://doi.org/10.14778/2735508.2735517

[281] W. Xiao, J. Xue, Y. Miao, Z. Li, C. Chen, M. Wu, W. Li, L. Zhou, {Tux²}: Distributed Graph Computation for Machine Learning, 2017, pp. 669–682.
URL https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/xiao

[282] D. Yang, J. Liu, J. Lai, EDGES: An Efficient Distributed Graph Embedding System on GPU Clusters, IEEE Transactions on Parallel and Distributed Systems 32 (7) (2021) 1892–1902, conference Name: IEEE Transactions on Parallel and Distributed Systems. doi:10.1109/TPDS.2020.3041219.

[283] CUDA Toolkit Documentation (2022).
URL https://docs.nvidia.com/cuda/

[284] MPI: A message passing interface, in: Supercomputing '93:Proceedings of the 1993 ACM/IEEE Conference on Supercomputing, 1993, pp. 878–883, iSSN: 1063-9535. doi:10.1145/169627.169855.

[285] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, PyTorch: An Imperative Style, High-Performance Deep Learning Library, arXiv:1912.01703 [cs, stat]ArXiv: 1912.01703 (Dec. 2019).
URL http://arxiv.org/abs/1912.01703

[286] P. Barham, A. Chowdhery, J. Dean, S. Ghemawat, S. Hand, D. Hurt, M. Isard, H. Lim, R. Pang, S. Roy, B. Saeta, P. Schuh, R. Sepassi, L. Shafey, C. Thekkath, Y. Wu, Pathways: Asynchronous Distributed Dataflow for ML, Proceedings of Machine Learning and Systems 4 (2022) 430–449.
URL https://proceedings.mlsys.org/paper/2022/hash/98dce83da57b0395e163467c9dae521b-Abstract.html

[287] H. Jin, Q. Song, X. Hu, Auto-Keras: An Efficient Neural Architecture Search System, arXiv:1806.10282 [cs, stat] (Mar. 2019). doi:10.48550/arXiv.1806.10282.
URL http://arxiv.org/abs/1806.10282

[288] M. Shoeybi, M. Patwary, R. Puri, P. LeGresley, J. Casper, B. Catanzaro, Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism, arXiv:1909.08053 [cs] (Mar. 2020). doi:10.48550/arXiv.1909.08053.
URL http://arxiv.org/abs/1909.08053

[289] S. Rajbhandari, J. Rasley, O. Ruwase, Y. He, ZeRO: Memory Optimizations Toward Training Trillion Parameter Models, arXiv:1910.02054 [cs, stat]ArXiv: 1910.02054 (May 2020).
URL http://arxiv.org/abs/1910.02054

[290] D. Byrd, M. Hybinette, T. H. Balch, Abides: Towards high-fidelity multi-agent market simulation, in: Proceedings of the 2020 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation, SIGSIM-PADS '20, Association for Computing Machinery, New York, NY, USA, 2020, p. 11–22. doi:10.1145/3384441.3395986.
URL https://doi.org/10.1145/3384441.3395986

[291] S. Amrouni, A. Moulin, J. Vann, S. Vyetrenko, T. Balch, M. Veloso, ABIDES-Gym: Gym Environments for Multi-Agent Discrete Event Simulation and Application to Financial Markets, in: Proceedings of the Second ACM International Conference on AI in Finance, ICAIF '21, Association for Computing Machinery, New York, NY, USA, 2021, event-place: Virtual Event. doi:10.1145/3490354.3494433.
URL https://doi-org.lib.ezproxy.ust.hk/10.1145/3490354.3494433

[292] M. Karpe, J. Fang, Z. Ma, C. Wang, Multi-Agent Reinforcement Learning in a Realistic Limit Order Book Market Simulation, in: Proceedings of the First ACM International Conference on AI in Finance, ICAIF '20, Association for Computing Machinery, New York, NY, USA, 2020, event-place: New York, New York. doi:10.1145/3383455.3422570.
URL https://doi-org.lib.ezproxy.ust.hk/10.1145/3383455.3422570

[293] A. Coletta, M. Prata, M. Conti, E. Mercanti, N. Bartolini, A. Moulin, S. Vyetrenko, T. Balch, Towards Realistic Market Simulations: A Generative Adversarial Networks Approach, in: Proceedings of the Second ACM International Conference on AI in Finance, ICAIF '21, Association for Computing Machinery, New York, NY, USA, 2021, event-place: Virtual Event. doi:10.1145/3490354.3494411.
URL https://doi-org.lib.ezproxy.ust.hk/10.1145/3490354.3494411

[294] Apache Airflow, original-date: 2015-04-13T18:04:58Z (Nov. 2022).
URL https://github.com/apache/airflow

[295] S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro, E. Shelhamer, cuDNN: Efficient Primitives for Deep Learning (Oct. 2014). doi:10.48550/arXiv.1410.0759.
URL https://arxiv.org/abs/1410.0759v3

[296] E. Wang, Q. Zhang, B. Shen, G. Zhang, X. Lu, Q. Wu, Y. Wang, Intel Math Kernel Library, in: High-Performance Computing on the Intel Xeon Phi, 2014, pp. 167–188. doi:10.1007/978-3-319-06486-4_7.

[297] R. van de Geijn, K. Goto, BLAS (Basic Linear Algebra Subprograms), in: D. Padua (Ed.), Encyclopedia of Parallel Computing, Springer US, Boston, MA, 2011, pp. 157–164. doi:10.1007/978-0-387-09766-4_84.
URL https://doi.org/10.1007/978-0-387-09766-4_84

[298] N. P. Jouppi, C. Young, N. Patil, D. Patterson, A domain-specific architecture for deep neural networks, Communications of the ACM 61 (9) (2018) 50–59. doi:10.1145/3154484.
URL https://doi.org/10.1145/3154484

[299] S. Markidis, S. Chien, E. Laure, I. Peng, J. S. Vetter, NVIDIA Tensor Core Programmability, Performance & Precision, in: 2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), IEEE Computer Society, Los Alamitos, CA, USA, 2018, pp. 522–531. doi:10.1109/IPDPSW.2018.00091.
URL https://doi.ieeecomputersociety.org/10.1109/IPDPSW.2018.00091

[300] K. Rupnow, Y. Liang, Y. Li, D. Chen, A study of high-level synthesis: Promises and challenges, in: 2011 9th IEEE International Conference on ASIC, 2011, pp. 1102–1105, iSSN: 2162-755X. doi:10.1109/ASICON.2011.6157401.

[301] P. Coussy, D. Gajski, M. Meredith, A. Takach, An Introduction to High-Level Synthesis, IEEE Design & Test of Computers 26 (4) (2009) 8–17. doi:10.1109/MDT.2009.69.
URL http://ieeexplore.ieee.org/document/5209958/

[302] J. Cong, J. Lau, G. Liu, S. Neuendorffer, P. Pan, K. Vissers, Z. Zhang, FPGA HLS Today: Successes, Challenges, and Opportunities, ACM Trans. Reconfigurable Technol. Syst.Place: New York, NY, USA Publisher: Association for Computing Machinery (Apr. 2022). doi:10.1145/3530775.
URL https://doi-org.lib.ezproxy.ust.hk/10.1145/3530775

[303] S. Smith, M. Patwary, B. Norick, P. LeGresley, S. Rajbhandari, J. Casper, Z. Liu, S. Prabhumoye, G. Zerveas, V. Korthikanti, E. Zhang, R. Child, R. Y. Aminabadi, J. Bernauer, X. Song, M. Shoeybi, Y. He, M. Houston, S. Tiwary, B. Catanzaro, Using DeepSpeed and Megatron to Train Megatron-Turing NLG 530B, A Large-Scale Generative Language Model, arXiv:2201.11990 [cs] (Feb. 2022).
URL http://arxiv.org/abs/2201.11990

[304] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, L. Zettlemoyer, Deep contextualized word representations, in: Proceedings of the 2018 Conference of the North American Chapter of the As-

sociation for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), Association for Computational Linguistics, New Orleans, Louisiana, 2018, pp. 2227–2237. doi:10.18653/v1/N18-1202.
URL https://aclanthology.org/N18-1202

[305] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, E. Brynjolfsson, S. Buch, D. Card, R. Castellon, N. Chatterji, A. Chen, K. Creel, J. Q. Davis, D. Demszky, C. Donahue, M. Doumbouya, E. Durmus, S. Ermon, J. Etchemendy, K. Ethayarajh, L. Fei-Fei, C. Finn, T. Gale, L. Gillespie, K. Goel, N. Goodman, S. Grossman, N. Guha, T. Hashimoto, P. Henderson, J. Hewitt, D. E. Ho, J. Hong, K. Hsu, J. Huang, T. Icard, S. Jain, D. Jurafsky, P. Kalluri, S. Karamcheti, G. Keeling, F. Khani, O. Khattab, P. W. Koh, M. Krass, R. Krishna, R. Kuditipudi, A. Kumar, F. Ladhak, M. Lee, T. Lee, J. Leskovec, I. Levent, X. L. Li, X. Li, T. Ma, A. Malik, C. D. Manning, S. Mirchandani, E. Mitchell, Z. Munyikwa, S. Nair, A. Narayan, D. Narayanan, B. Newman, A. Nie, J. C. Niebles, H. Nilforoshan, J. Nyarko, G. Ogut, L. Orr, I. Papadimitriou, J. S. Park, C. Piech, E. Portelance, C. Potts, A. Raghunathan, R. Reich, H. Ren, F. Rong, Y. Roohani, C. Ruiz, J. Ryan, C. Ré, D. Sadigh, S. Sagawa, K. Santhanam, A. Shih, K. Srinivasan, A. Tamkin, R. Taori, A. W. Thomas, F. Tramèr, R. E. Wang, W. Wang, B. Wu, J. Wu, Y. Wu, S. M. Xie, M. Yasunaga, J. You, M. Zaharia, M. Zhang, T. Zhang, X. Zhang, Y. Zhang, L. Zheng, K. Zhou, P. Liang, On the Opportunities and Risks of Foundation Models, arXiv:2108.07258 [cs] (Aug. 2021). doi:10.48550/arXiv.2108.07258.
URL http://arxiv.org/abs/2108.07258

[306] Z.-H. Zhou, Ensemble Methods: Foundations and Algorithms, 1st Edition, Chapman & Hall/CRC, 2012.

[307] V. J. Hellendoorn, A. A. Sawant, The growing cost of deep learning for source code, Communications of the ACM 65 (1) (2021) 31–33. doi:10.1145/3501261.
URL http://doi.org/10.1145/3501261

[308] S. C. H. Hoi, D. Sahoo, J. Lu, P. Zhao, Online learning: A comprehensive survey, Neurocomputing 459 (2021) 249–289. doi:10.1016/j.neucom.2021.04.112.
URL https://www.sciencedirect.com/science/article/pii/S0925231221006706

[309] M. De Lange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, T. Tuytelaars, A Continual Learning Survey: Defying Forgetting in Classification Tasks, IEEE Transactions on Pattern Analysis and Machine Intelligence 44 (7) (2022) 3366–3385, conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence. doi:10.1109/TPAMI.2021.3057446.

[310] A. Acar, H. Aksu, A. S. Uluagac, M. Conti, A Survey on Homomorphic Encryption Schemes: Theory and Implementation, ACM Computing Surveys 51 (4) (2018) 79:1–79:35. doi:10.1145/3214303.
URL http://doi.org/10.1145/3214303

[311] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, Y. Gao, A survey on federated learning, Knowledge-Based Systems 216 (2021) 106775. doi:10.1016/j.knosys.2021.106775.
URL https://www.sciencedirect.com/science/article/pii/S0950705121000381

[312] Q. Li, Z. Wen, Z. Wu, S. Hu, N. Wang, Y. Li, X. Liu, B. He, A Survey on Federated Learning Systems: Vision, Hype and Reality for Data Privacy and Protection, IEEE Transactions on Knowledge and Data Engineering (2021) 1–1Conference Name: IEEE Transactions on Knowledge and Data Engineering. doi:10.1109/TKDE.2021.3124599.

[313] M. S. Burgin, Theory of knowledge: structures and processes, World Scientific series in information studies ; Vol. 5, World Scientific Publishing Co. Pte Ltd., Singapore, 2017.

[314] D. H. Jonassen, Structural knowledge: techniques for representing, conveying, and acquiring structural knowledge, L. Erlbaum Associates, Hillsdale, N.J, 1993.

[315] C. Pavese, Knowledge How, in: E. N. Zalta, U. Nodelman (Eds.), The Stanford Encyclopedia of Philosophy, fall 2022 Edition, Metaphysics Research Lab, Stanford University, 2022.
URL https://plato.stanford.edu/archives/fall2022/entries/knowledge-how/

[316] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, D. Amodei, Language Models are Few-Shot Learners, arXiv:2005.14165 [cs]ArXiv: 2005.14165 (Jul. 2020).
URL http://arxiv.org/abs/2005.14165

[317] D. Ha, J. Schmidhuber, World models (2018). doi:10.5281/ZENODO.1207631.
URL https://zenodo.org/record/1207631

[318] K. Alattas, A. Alkaabi, A. B. Alsaud, An Overview of Artificial General Intelligence: Recent Developments and Future Challenges, Journal of Computer Science 17 (4) (2021) 364–370, publisher: Science Publications. doi:10.3844/jcssp.2021.364.370.
URL https://thescipub.com/abstract/jcssp.2021.364.370

[319] D. Kahneman, Thinking, fast and slow, Farrar, Straus and Giroux, New York, 2011.
URL https://www.amazon.de/Thinking-Fast-Slow-Daniel-Kahneman/dp/0374275637/ref=wl_it_dp_o_pdT1_nS_nC?ie=UTF8&colid=151193SNGKJT9&coliid=I3OCESLZCVDFL7

[320] G. Marcus, The Next Decade in AI: Four Steps Towards Robust Artificial Intelligence, arXiv:2002.06177 [cs]ArXiv: 2002.06177 (Feb. 2020).
URL http://arxiv.org/abs/2002.06177

[321] Y. Bengio, GFlowNets and System 2 Deep Learning (2022).
URL https://www.microsoft.com/en-us/research/video/gflownets-and-system-2-deep-learning/

[322] Y. Bengio, S. Lahlou, T. Deleu, E. J. Hu, M. Tiwari, E. Bengio, GFlowNet Foundations, arXiv:2111.09266 [cs, stat] (Aug. 2022).
URL http://arxiv.org/abs/2111.09266

[323] L. Yao, Z. Chu, S. Li, Y. Li, J. Gao, A. Zhang, A Survey on Causal Inference, ACM Transactions on Knowledge Discovery from Data 15 (5) (2021) 74:1–74:46. doi:10.1145/3444944.
URL http://doi.org/10.1145/3444944

[324] J. Pearl, The book of why : the new science of cause and effect, Allen Lane, London, 2018, publication Title: The book of why : the new science of cause and effect.

[325] B. Schölkopf, F. Locatello, S. Bauer, N. R. Ke, N. Kalchbrenner, A. Goyal, Y. Bengio, Toward Causal Representation Learning, Proceedings of the IEEE 109 (5) (2021) 612–634, conference Name: Proceedings of the IEEE. doi:10.1109/JPROC.2021.3058954.

[326] T. J. VanderWeele, I. Shpitser, On the definition of a confounder, The Annals of Statistics 41 (1) (2013) 196–220, publisher: Institute of Mathematical Statistics. doi:10.1214/12-AOS1058.
URL https://projecteuclid.org/journals/annals-of-statistics/volume-41/issue-1/On-the-definition-of-a-confounder/10.1214/12-AOS1058.full

[327] G. Atluri, A. Karpatne, V. Kumar, Spatio-Temporal Data Mining: A Survey of Problems and Methods, ACM Comput. Surv. 51 (4), place: New York, NY, USA Publisher: Association for Computing Machinery (Aug. 2018). doi:10.1145/3161602.
URL https://doi-org.lib.ezproxy.ust.hk/10.1145/3161602

[328] S. Wang, J. Cao, P. Yu, Deep Learning for Spatio-Temporal Data Mining: A Survey, IEEE Transactions on Knowledge and Data Engineering (2020) 1–1Conference Name: IEEE Transactions on Knowledge and Data Engineering. doi:10.1109/TKDE.2020.3025580.

[329] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, arXiv:1810.04805 [cs]ArXiv: 1810.04805 (May 2019).
URL http://arxiv.org/abs/1810.04805

[330] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, I. Sutskever, Learning Transferable Visual Models From Natural Language Supervision, arXiv:2103.00020 [cs] (Feb. 2021). doi:10.48550/arXiv.2103.00020.
URL http://arxiv.org/abs/2103.00020

[331] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. d. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, A. Ray, R. Puri, G. Krueger, M. Petrov, H. Khlaaf, G. Sastry, P. Mishkin, B. Chan, S. Gray, N. Ryder, M. Pavlov, A. Power, L. Kaiser, M. Bavarian, C. Winter, P. Tillet, F. P. Such, D. Cummings, M. Plappert, F. Chantzis, E. Barnes, A. Herbert-Voss, W. H. Guss, A. Nichol, A. Paino, N. Tezak, J. Tang, I. Babuschkin, S. Balaji, S. Jain, W. Saunders, C. Hesse,

A. N. Carr, J. Leike, J. Achiam, V. Misra, E. Morikawa, A. Radford, M. Knight, M. Brundage, M. Murati, K. Mayer, P. Welinder, B. McGrew, D. Amodei, S. McCandlish, I. Sutskever, W. Zaremba, Evaluating Large Language Models Trained on Code, arXiv:2107.03374 [cs] (Jul. 2021). doi:10.48550/arXiv.2107.03374.
URL http://arxiv.org/abs/2107.03374

[332] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, I. Sutskever, Zero-Shot Text-to-Image Generation, arXiv:2102.12092 [cs] (Feb. 2021). doi:10.48550/arXiv.2102.12092.
URL http://arxiv.org/abs/2102.12092

[333] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, N. Houlsby, An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, in: International Conference on Learning Representations, 2022.
URL https://openreview.net/forum?id=YicbFdNTTy

[334] W. Wang, H. Bao, L. Dong, J. Bjorck, Z. Peng, Q. Liu, K. Aggarwal, O. K. Mohammed, S. Singhal, S. Som, F. Wei, Image as a Foreign Language: BEiT Pretraining for All Vision and Vision-Language Tasks, arXiv:2208.10442 [cs] (Aug. 2022). doi:10.48550/arXiv.2208.10442.
URL http://arxiv.org/abs/2208.10442

[335] X.-C. Zhang, C.-K. Wu, Z.-J. Yang, Z.-X. Wu, J.-C. Yi, C.-Y. Hsieh, T.-J. Hou, D.-S. Cao, MG-BERT: leveraging unsupervised atomic representation learning for molecular property prediction, Briefings in Bioinformatics 22 (6) (2021) bbab152. doi:10.1093/bib/bbab152.
URL https://doi.org/10.1093/bib/bbab152

[336] R. Guo, L. Cheng, J. Li, P. R. Hahn, H. Liu, A Survey of Learning Causality with Data: Problems and Methods, ACM Computing Surveys 53 (4) (2020) 75:1–75:37. doi:10.1145/3397269.
URL http://doi.org/10.1145/3397269

[337] Z. Shen, J. Liu, Y. He, X. Zhang, R. Xu, H. Yu, P. Cui, Towards Out-Of-Distribution Generalization: A Survey, arXiv:2108.13624 [cs] (Aug. 2021). doi:10.48550/arXiv.2108.13624.
URL http://arxiv.org/abs/2108.13624

[338] L. Breiman, Bagging predictors, Machine Learning 24 (2) (1996) 123–140. doi:10.1007/BF00058655.
URL https://doi.org/10.1007/BF00058655

[339] R. E. Schapire, The strength of weak learnability, Machine Learning 5 (2) (1990) 197–227. doi:10.1007/BF00116037.
URL https://doi.org/10.1007/BF00116037

[340] L. Breiman, Arcing classifier (with discussion and a rejoinder by the author), The Annals of Statistics 26 (3) (1998) 801–849, publisher: Institute of Mathematical Statistics. doi:10.1214/aos/1024691079.
URL https://projecteuclid.org/journals/annals-of-statistics/volume-26/issue-3/Arcing-classifier-with-discussion-and-a-rejoinder-by-the-author/10.1214/aos/1024691079.full

[341] D. H. Wolpert, Stacked generalization, Neural Networks 5 (2) (1992) 241–259. doi:10.1016/S0893-6080(05)80023-1.
URL https://www.sciencedirect.com/science/article/pii/S0893608005800231

[342] L. Breiman, Stacked regressions, Machine Learning 24 (1) (1996) 49–64. doi:10.1007/BF00117832.
URL https://doi.org/10.1007/BF00117832

[343] J. A. Hoeting, D. Madigan, A. E. Raftery, C. T. Volinsky, Bayesian Model Averaging: A Tutorial, Statistical Science 14 (4) (1999) 382–401, publisher: Institute of Mathematical Statistics.
URL http://www.jstor.org/stable/2676803

[344] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: A Simple Way to Prevent Neural Networks from Overfitting, Journal of Machine Learning Research 15 (56) (2014) 1929–1958.
URL http://jmlr.org/papers/v15/srivastava14a.html

Author Biographies

Jian Guo is currently the Executive President and a Chief Scientist at International Digital Economy Academy (IDEA). As a founding member of IDEA, he also serves as the head of IDEA Research Center of AI Finance & Deep Learning and a Professor of Practice at the Hong Kong University of Science and Technology (Guangzhou). Dr. Guo received his B.S. in mathematics from Tsinghua University, and received his Ph.D. in statistics from University of Michigan in 2011. He started his professorship (tenure-track) at Harvard University since 2011. He published a number of research papers in deep/reinforcement/statistical learning, including theory and application. Dr. Guo is one of the pioneering AI finance researchers, and is an entrepreneur in quantitative investment industry.

Saizhuo Wang is currently a Ph.D. candidate in Department of Computer Science and Engineering at the Hong Kong University of Science and Technology, under the supervision of Prof. Harry Heung-Yeung Shum and Prof. Lionel Ming-Shuan Ni and working with Prof. Jian Guo. He received his bachelor of engineering degree in computer science from Chu Kochen Honors College at Zhejiang University. His research interest mainly in the interdisciplinary field of artificial intelligence and financial technology.

Lionel M. Ni is currently the Founding President of the Hong Kong University of Science and Technology (Guangzhou) and Chair Professor in the university's Data Science and Analytics Thrust, as well as Chair Professor of Computer Science and Engineering at the Hong Kong University of Science and Technology. He is a Life Fellow of IEEE, and a Fellow of the Hong Kong Academy of Engineering Science. Prof. Ni's research includes high-performance computing, mobile computing, wireless networking, big data, and intelligent computing. He has published three books and 350+ refereed journal and conference articles. He has chaired 30+ professional conferences and has received eight awards for authoring outstanding papers. Prof. Ni received his Ph.D. in Electrical Engineering from Purdue University in 1980.

Heung-Yeung Shum is the Founding Chairman of International Digital Economy Academy (IDEA), and a Professor-at-Large at the Institute for Advanced Study, Hong Kong University of Science and Technology. He is a Foreign Member of National Academy of Engineering of the US, International Fellow of Royal Academy of Engineering of the UK, ACM Fellow and IEEE Fellow. Until March 2020, he was the Executive Vice President of Microsoft Corporation, responsible for AI and Research. Dr. Shum received his Ph.D. in Robotics from School of Computer Science at Carnegie Mellon University.