StockFormer: Learning Hybrid Trading Machines with Predictive Coding

Siyu Gao , **Yunbo Wang**[*] and **Xiaokang Yang**
MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University
{siyu.gao, yunbow, xkyang}@sjtu.edu.cn

Abstract

Typical RL-for-finance solutions directly optimize trading policies over the noisy market data, such as stock prices and trading volumes, without explicitly considering the future trends and correlations of different investment assets as we humans do. In this paper, we present StockFormer, a hybrid trading machine that integrates the forward modeling capabilities of predictive coding with the advantages of RL agents in policy flexibility. The predictive coding part consists of three Transformer branches with modified structures, which respectively extract effective latent states of long-/short-term future dynamics and asset relations. The RL agent adaptively fuses these states and then executes an actor-critic algorithm in the unified state space. The entire model is jointly trained by propagating the critic's gradients back to the predictive coding module. StockFormer significantly outperforms existing approaches across three publicly available financial datasets in terms of portfolio returns and Sharpe ratios.

1 Introduction

Reinforcement learning (RL) has shown promising results in practical applications of financial decision-making problems, such as improving stock trading strategies by identifying promising buying and selling points [Liu *et al.*, 2021; Zhong *et al.*, 2020]. A common practice is to formulate the portfolio optimization problem as a Markov decision process (MDP) [Puterman, 2014] and directly perform model-free RL algorithms in the state space represented by the observed data (*e.g.*, stock prices, trading volumes, and technical indicators). However, it makes an excessively strong assumption that the observed data is sufficiently informative and can well represent (i) *the correlations between hundreds (or thousands) of stocks* and (ii) *the underlying (or even future) dynamics in the rapidly changing financial markets*.

To address this issue, we consider the way in which humans make investment decisions. There are two factors that require special consideration, that is, the *dynamic stock correlations* and the expected returns of each stock in both *long- and short-term horizons*. We present StockFormer, a novel RL agent that learns to adaptively discover and capitalize on promising trading opportunities. It is a hybrid trading machine that integrates the forward modeling capabilities of predictive coding with the advantages of actor-critic methods for trading flexibility. Predictive coding [Elias, 1955; Spratling, 2017; Rao and Ballard, 1999] is one of the most successful self-supervised learning methods in natural language processing [Mikolov *et al.*, 2013] and computer vision [Oord *et al.*, 2018], whose core idea is to extract useful latent states from noisy market data that can maximally benefit the prediction of future or missing contextual information.

Specifically, we leverage three Transformer-like networks to respectively learn long-horizon, short-horizon, and relational latent representations from the observed market data. To ease the difficulty of concurrent time series modeling, StockFormer employs multi-head feed-forward networks in the attention block, which can maintain the diversity of temporal patterns learned from multiple concurrent market asset series (*e.g.*, trading records of hundreds of stocks in the same time period). For policy optimization, the three types of latent states are adaptively and progressively combined through a series of multi-head attention structures. StockFormer exploits the actor-critic method but only propagates the critic's analytic gradients back into the relational state encoder.

Notably, there exists another line of work that aims to improve future stock prediction accuracy with powerful time series modeling networks [Li *et al.*, 2018; Feng *et al.*, 2019; Wang *et al.*, 2021; Duan *et al.*, 2022]. They use fixed trading rules, such as "*buy-and-hold*", which recycles capital from unsuccessful stock bets to average in on stocks that have promising future returns and hold them for a fixed period of time. As shown in Table 1, despite recent success, these models are not directly designed to maximize investment returns and cannot provide flexible trading decisions.

We empirically observe that StockFormer remarkably outperforms existing stock prediction and RL-for-finance approaches across three public datasets from NASDAQ and Chinese stock markets as well as the cryptocurrency market.

2 Related Work

RL for finance. There have been many attempts to use RL methods to make trading decisions [Théate and Ernst, 2021; Weng *et al.*, 2020; Liang *et al.*, 2018; Benhamou *et al.*, 2020].

[*]Corresponding author

Model	Category	Trading policy	RL state space
FactorVAE [Duan *et al.*, 2022]	Stock prediction	Fixed (*e.g.*, buy-and-hold)	n/a
SARL [Ye *et al.*, 2020]	RL-for-finance	Learned	Observed asset prices + Asset movement signal
StockFormer	Hybrid	Learned	Temporal + Relational predictive coding (via SSL)

Table 1: StockFormer is a hybrid trading framework that is clearly different from the previous art of (a) stock prediction methods and (b) RL-for-finance methods. In SARL, a typical "asset movement signal" is the financial new embedding. "SSL": Self-supervised learning.

Main differences of these models include: the definition of the input states [Zhong *et al.*, 2020; Liu *et al.*, 2021; Weng *et al.*, 2020], the engineering of reward functions [Liang *et al.*, 2018; Hu and Lin, 2019], and the RL algorithms [Benhamou *et al.*, 2020; Suri *et al.*, 2021; Huotari *et al.*, 2020]. SARL [Ye *et al.*, 2020] proposed to learn policy in the noisy observation space and expand the space with additional asset movement signals such as financial news embeddings. FinRL [Liu *et al.*, 2021] integrates multiple off-the-shelf RL algorithms such as Soft Actor-Critic (SAC) [Haarnoja *et al.*, 2018] and DDPG [Lillicrap *et al.*, 2016]. It defines the states as a combination of the covariance matrix of the close prices of all stocks and the MACD indicators, whose dimension will sharply increase with the growth of the number of stocks. We use its SAC implementation as the baseline of StockFormer.

Stock prediction. Mainstream stock prediction approaches can be divided into three categories. CNN-based models [Hoseinzade and Haratizadeh, 2019; Wen *et al.*, 2019] take the historical data of different stocks as a set of input feature maps of a convolutional neural network. RNN-based models [Li *et al.*, 2018; Zhang *et al.*, 2017; Qin *et al.*, 2017] are better at capturing the underlying sequential trends in stocks. Other network architectures use attention mechanisms [Hu *et al.*, 2018; Li *et al.*, 2018; Wu *et al.*, 2019], dilated convolutions [Wang *et al.*, 2021; Cho *et al.*, 2019], or graph neural networks [Feng *et al.*, 2019; Wang *et al.*, 2021; Patil *et al.*, 2020] to jointly model the long-term dynamics and the relations of trading assets. Inspired by previous Transformer-based architectures [Vaswani *et al.*, 2017; Kitaev *et al.*, 2020; Zhou *et al.*, 2021; Wu *et al.*, 2021; Fedus *et al.*, 2021], StockFormer integrates Transformer in the RL-for-finance framework. It borrows the idea of self-supervised predictive coding as a representation learning method to extract useful and compact latent states from noisy and high-dimensional market data.

3 Portfolio Optimization as POMDPs

We believe that: *One of the key challenges of RL-for-finance is to extract useful states that can reflect the essential dynamics of the market from noisy, high-dimensional raw transaction records.* Therefore, unlike previous work that commonly treats portfolio optimization as an MDP problem with 5-tuples $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma)$, in this paper, we take it as partially observable Markov decision processes (POMDPs) with 7-tuples $(\mathcal{O}, \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{Z}, \mathcal{R}, \gamma)$, where $\mathcal{O}$ is the observation space of the noisy market data, $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{T}(s_{t+1}|s_t, a_t)$ denotes the state transition probability, $\mathcal{Z}(O_t|s_t)$ is the prior distribution of the observed data O_t given latent state s_t. Assuming a fixed length of P steps in each episode (*e.g.*, we use 1,000 trading days for the stock trading experiments), the goal is to learn the optimal policy π^* that can maximize the total payoff: $G_t = \mathbb{E}_{\tau\sim\pi}[\sum_{t=1}^{P} \gamma^{t-1} R_t]$, where the R_t is the immediate reward at each time step drew from the reward function $\mathcal{R}(s, a)$ and $\gamma \in (0, 1)$ is the discount factor of future rewards. We here give detailed definitions of $\mathcal{O}$, $\mathcal{S}$, $\mathcal{A}$, and $\mathcal{R}$ as follows.

Noisy observation space ($\mathcal{O}$). Let us take the stock trading problem for instance. The observed data includes:

- Raw trading records $O_t^{\text{price}} \in \mathbb{R}^{T\times N\times 5}$ that involve daily *opening, close, highest, lowest prices*, and *trading volume* during the past T days, where N is the number of stocks.
- Technical indicators $O_t^{\text{stat}} \in \mathbb{R}^{N\times K}$, where K is the number of indicators that capture the dynamics of the time series.
- A covariance matrix $O_t^{\text{cov}} \in \mathbb{R}^{N\times N}$ between sequences of daily close prices of all stocks over a fixed period before t.

In the following sections, we use $O_t^{\text{relat}} \in \mathbb{R}^{N\times(K+N)}$ to denote the concatenation of O_t^{stat} and O_t^{cov}.

Compositional state space ($\mathcal{S}$). The state space of StockFormer is composed of three types of latent states and the explicit account state $s_t^{\text{amount}} \in \mathbb{R}^{N+1}$ that represents the total account balance and the holding amount of each trading asset.

Action space ($\mathcal{A}$). We have a continuous action space such that $a_t \in \mathbb{R}^N$, indicating the amount of buying, holding, or selling shares on each trading asset. To simulate real-world trading scenarios, we discretize a_t into multiple intervals of daily trading signals when interacting with the environment.

Reward function ($\mathcal{R}$). The reward $R_t \sim \mathcal{R}(s_t, a_t)$ is defined as the daily portfolio return ratios.

4 StockFormer

As aforementioned, we have highlighted the challenge of RL-for-finance methods, that is, learning flexible policies aware of the dynamic relations between a batch of trading targets in the financial market and the future trends of each individual of them. To tackle this challenge, we introduce StockFormer, an RL agent that extracts disentangled latent states from noisy time series data through *predictive coding*, and then optimizes the trading decisions in the compositional state space. Therefore, StockFormer consists of two training phases: predictive coding and policy learning.

4.1 Predictive Coding

In this phase, three Transformer-like network branches are trained in a self-supervised manner to respectively learn the

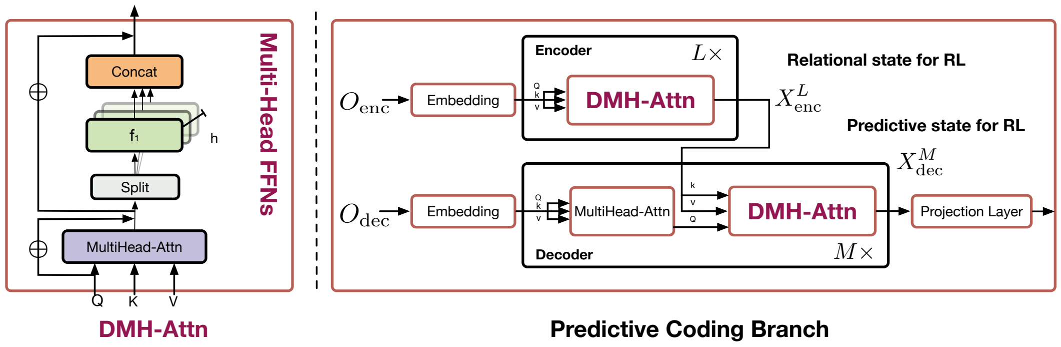


Figure 1: **Left:** *Diversified multi-head attention block* (DMH-Attn), which improves the original attention block in Transformers with multi-head feed-forward networks. It captures the diverse patterns of concurrent time series in different sub-spaces. **Right:** The general architecture of Transformer branches learned through predictive coding, which extracts useful representations for the RL agent by maximizing the likelihood of predicting the missing context or future returns of the financial market.

relational, long-horizon, and short-horizon hidden representations. The key insight of so-called *predictive coding* [Elias, 1955; Spratling, 2017; Rao and Ballard, 1999] is to extract useful representations that are maximally beneficial for predicting future, missing or contextual information. These representations jointly form the compositional state space in the next training phase for learning the investment policy. Next, we first introduce an improved Transformer architecture and then discuss the predictive coding methods for the relational and predictive representations respectively.

Building block: Diversified multi-head attention. The diversity of temporal patterns among the concurrent sequences of multiple trading assets in financial markets (*e.g.*, hundreds of stocks) greatly increases the difficulty of learning effective representations from raw data. To tackle this difficulty, as shown in Figure 1 (Left), we renovate the multi-head attention block in the original Transformer with a group of feed-forward networks (FFNs) rather than a single FFN, in which each FFN individually responds to a single head in the output of the multi-head attention layer. Without changing the overall number of parameters, such a mechanism strengthens multi-head attention's original feature decoupling ability, which facilitates modeling the diverse temporal patterns in different subspaces. We thus refer to the modified attention block as *diversified multi-head attention* (DMH-Attn). For a set of d_{model}-dimensional keys (K), values (V), and queries (Q), the process in a diversified multi-head attention block can be represented as follows. We split the output features Z of the multi-head attention layer by h along the channel dimension, where h is the number of parallel attention heads, and then apply a separate FFN to each group of separated features in Z:

$$\begin{aligned} &Z = \text{MH-Attn}(Q,K,V)+Q, \quad x_i = \text{Split}(Z) \\ &f_i = \max(0, x_i W_{1,i} + b_{1,i}) W_{2,i} + b_{2,i} \\ &\text{DMH-Attn}(Q,K,V) = \text{Concat}(f_1, ..., f_h) + Z, \end{aligned} \tag{1}$$

where "MH-Attn" denotes multi-head attention, f_i denotes the output features of each FFN head, which contains two linear transformations with the ReLU activation in between.

General predictive coding architecture. Each Transformer branch in StockFormer can be divided into encoder and decoder layers, as shown in Figure 1 (Right). Both parts are used in the predictive coding phase with different training objectives, but only the encoder layers are used during policy optimization. We have L encoder layers and M decoder layers. The representations X^L_{enc} of the final encoder layer is used as one of the inputs of each decoder layer. The computation in the l^{th} encoder layer and the m^{th} decoder layer can be presented as follows:

$$\begin{aligned} &\text{Encoder Layer:} \\ &\quad X^l_{\text{enc}} = \text{DMH-Attn}(X^{l-1}_{\text{enc}}, X^{l-1}_{\text{enc}}, X^{l-1}_{\text{enc}}) \\ &\text{Decoder Layer:} \\ &\quad F^{m-1}_{\text{dec}} = \text{MH-Attn}(X^{m-1}_{\text{dec}}, X^{m-1}_{\text{dec}}, X^{m-1}_{\text{dec}}) + X^{m-1}_{\text{dec}} \\ &\quad X^m_{\text{dec}} = \text{DMH-Attn}(F^{m-1}_{\text{dec}}, X^L_{\text{enc}}, X^L_{\text{enc}}), \end{aligned} \tag{2}$$

where X^l_{enc} and X^m_{dec} are the output of the encoder and decoder layer respectively. Specifically, X^0_{enc} and X^0_{dec} are the inputs of the the first encoder and decoder layers, which are the *positional embedding* [Vaswani *et al.*, 2017] of the raw input data O_{enc} and O_{dec}. X^M_{dec} by the last decoder layer are then fed into a projection layer to generate the final output in each predictive coding task, that is, predicting future returns or reasoning about the missing (masked) information in the financial market at this moment.

Relation inference module (1st Transformer branch). As shown in Figure 2, the relation inference module is used to capture the dynamic correlations among concurrent time series, *e.g.*, different stocks. At time step t, we use the same input for the encoder and the decoder, $O^{\text{relat}}_{\text{enc},t} = O^{\text{relat}}_{\text{dec},t} \in \mathbb{R}^{N\times(K+N)}$, where N is the number of concurrent trading assets and K is the number of statistics that capture the dynamics of the time series data. Particularly for the stock market datasets, we use common technical indicators as the statistics, such as MACD, RSI, and SMA. During the training phase, $O^{\text{relat}}_{\text{enc},t}$ can be divided into two parts:

- The covariance matrix $O^{\text{cov}}_t \in \mathbb{R}^{N\times N}$ between sequences of daily close prices of all concurrent trading assets over a fixed period of time before t.

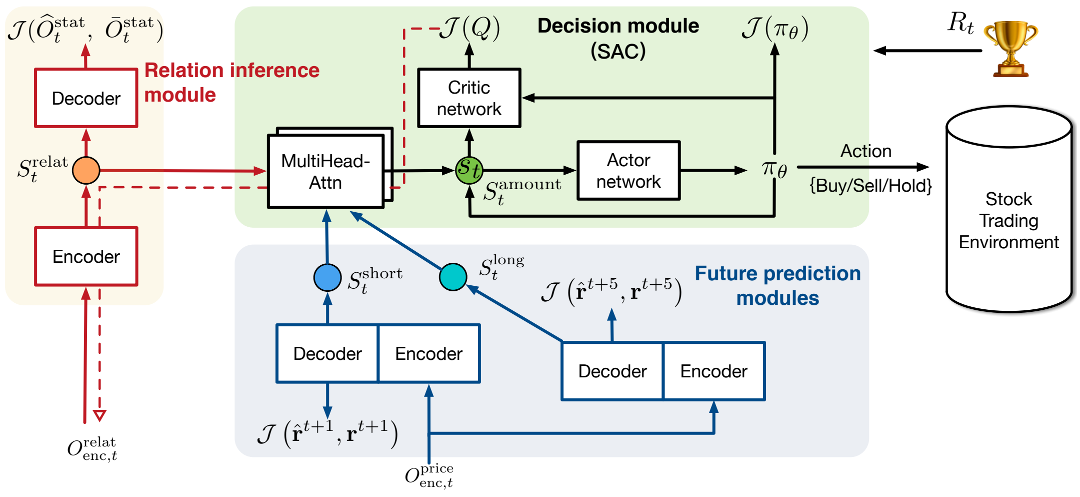

$\mathcal{J}(\widehat{O}_t^{\text{stat}}, \bar{O}_t^{\text{stat}})$
Decoder
Relation inference module
S_t^{relat}
Encoder
$O_{\text{enc},t}^{\text{relat}}$
$\mathcal{J}(Q)$
Decision module (SAC)
$\mathcal{J}(\pi_\theta)$
Critic network
MultiHead-Attn
s_t
S_t^{amount}
Actor network
π_θ
R_t
Action
{Buy/Sell/Hold}
Stock Trading Environment
S_t^{short}
S_t^{long}
$\mathcal{J}(\hat{\mathbf{r}}^{t+5}, \mathbf{r}^{t+5})$
Future prediction modules
Decoder
Encoder
Decoder
Encoder
$\mathcal{J}(\hat{\mathbf{r}}^{t+1}, \mathbf{r}^{t+1})$
$O_{\text{enc},t}^{\text{price}}$

Figure 2: Unlike previous RL-for-finance methods, StockFormer builds the decision module upon learned representations provided by a relation inference module and two future prediction modules. The decision module contains a couple of multi-head attention layers that integrate the compositional representations, an actor network, and a critic network. In particular, the critic propagates its gradients of state values back into the relation inference module (Solid arrow: the data flow; Dashed arrow: the gradient flow of the critic loss).

- The *masked* statistics $O_t^{\text{stat}} \in \mathbb{R}^{N\times K}$, in which we randomly select half of the time series and mask their input statistics by zero. It is worth noting that in the test phase, we use complete data without masks as input to the module.

The task of the relation inference module is to reconstruct the masked statistics (*i.e.*, technical indicators) given O_t^{cov} and the remaining visible statistics. The self-supervised loss function in this module can be represented as:

$$\mathcal{J}_t^{\text{relat}} = \mathcal{J}(\widehat{O}_t^{\text{stat}}, \bar{O}_t^{\text{stat}}) = \frac{1}{N}\sum_n ||\widehat{O}_t^{\text{stat}}, \bar{O}_t^{\text{stat}}||^2, \quad (3)$$

where $\widehat{O}_t^{\text{stat}}$ and $\bar{O}_t^{\text{stat}}$ are the reconstructed values of the masked statistics and their ground truth. Intuitively, such a predictive coding method drives the Transformer encoder to capture the dependencies among the concurrent time series (*i.e.*, stocks). The relation inference module provides its final encoding features (*i.e.*, X_{enc}^L in the above descriptions of the general architecture) for the subsequent decision module, as parts of the state space of the RL agent, denoted by s_t^{relat}.

Future prediction modules (2nd&3rd branches). In the short-term prediction module, the task is to predict the return ratio of each stock[1] for the next day ($H = 1$). In this module, we use different inputs for the encoder and the decoder. The inputs of the encoder are $O_{\text{enc},t}^{\text{price}} \in \mathbb{R}^{T\times 5}$, which involve daily *opening, close, highest, lowest prices*, and *trading volumes* during the past T days. We feed the transaction records at step t to the decoder, *i.e.*, $O_{\text{dec},t}^{\text{price}} \in \mathbb{R}^{1\times 5}$. Similar to the short-term prediction module, the long-term prediction module is to predict the return ratios in a longer future horizon (*e.g.*, $H = 5$ for daily stock trading). The long-term prediction task encourages the Transformer branch to capture future dynamics from a longer perspective. We adopt the regression loss and stock ranking loss from [Feng *et al.*, 2019]:

$$\mathcal{J}_t^{\text{future}} = \mathcal{J}(\hat{\mathbf{r}}^{t+H}, \mathbf{r}^{t+H}) = ||\hat{\mathbf{r}}^{t+H} - \mathbf{r}^{t+H}||^2 \quad + \\ \alpha \sum_{i=1}^{N}\sum_{j=1}^{N} \max\left(0, -\left(\hat{r}_i^{t+H} - \hat{r}_j^{t+H}\right)\left(r_i^{t+H} - r_j^{t+H}\right)\right), \quad (4)$$

where $\mathbf{r}^{t+H} = [r_1^{t+H}, \dots, r_N^{t+H}]$ denotes the true future return ratios of all stocks, $r_i^{t+H} = (p_i^{t+H} - p_i^t)/p_i^t$, p_i^t is the closing price of stock i on day t, $\hat{\mathbf{r}}^{t+H} = [\hat{r}_1^{t+H}, \dots, \hat{r}_N^{t+H}]$ is the predicted return ratios of all stocks, and α is a hyperparameter used to balance the weight of the two loss terms. The second term encourages the predicted ranking of any pair of stocks to maintain the same order as the ground truth. We here use the output features of the decoders (*i.e.*, X_{dec}^M before the projection layer) in the future prediction modules as parts of the states of the RL agent (*i.e.*, s_t^{long} and s_t^{short}).

4.2 Policy Optimization

In the second training phase, StockFormer integrates the three types of latent representations into a unified state space $\mathcal{S} \subseteq \mathbb{R}^{N\times(D+1)}$ through a series of multi-head attention layers and then exploits an actor-critic method to learn the trading policy. StockFormer learns flexible trading strategies by taking full advantage of the future trends of each time series in the long/short time horizon, as well as the dynamic relationships between dozens of concurrent trading assets.

Latent states integration. With predictive coding, we obtain informative latent representations from three Transformer branches, denoted by $s_t^{\text{relat}} \in \mathbb{R}^{N\times D}$, $s_t^{\text{long}} \in \mathbb{R}^{N\times D}$, and $s_t^{\text{short}} \in \mathbb{R}^{N\times D}$. Therefore, the first question in the decision-making module is how to integrate different types of predictive embeddings into a unified state space. We use a cascaded multi-head attention mechanism as follows so that the decision

[1]We here take daily stock trading as an example.

module can be informed with the predicted future information in the financial market, as well as the dynamic relations between different trading targets.

$$\begin{aligned} s_t^{\text{future}} &= \text{MultiHead-Attn}(s_t^{\text{long}}, s_t^{\text{short}}, s_t^{\text{short}}) + s_t^{\text{long}} \\ s_t^{h} &= \text{MultiHead-Attn}(s_t^{\text{future}}, s_t^{\text{relat}}, s_t^{\text{relat}}) + s_t^{\text{future}} \\ s_t &= \text{Concat}(s_t^{h}, s_t^{\text{hold}}), \end{aligned} \tag{5}$$

where s_t^{future} and s_t^h are the outputs from each multi-head attention layer, s_t^{long} is used as the query in the first MH-Attn unit as it is less vulnerable to short-term noise than s_t^{short} and s_t^{future} is used as the query to extract more effective relational features. We combine s_t^h with the shares of all trading targets that we hold at a certain time step, $s_t^{\text{hold}} \in \mathbb{R}^{N\times 1}$, to form the final states $s_t \in \mathbb{R}^{N\times(D+1)}$, and then feed s_t into the actor network and the critic network. In the experiments, we empirically show the effectiveness of the proposed architecture of the decision module: Exploiting a series of multi-head attention layers better integrates different sources of latent representations into a unified state space, which can eventually benefit policy optimization.

Joint learning with RL objectives. We use the soft actor-critic (SAC) [Haarnoja *et al.*, 2018] to learn the trading policy in the unified state space. The critic network learns the parametric soft Q-function by minimizing the following soft Bellman residual and propagates the analytic gradients of state values back into the relation inference module which is adopted from previous actor-critic approaches with visual inputs [Lee *et al.*, 2020; Yarats *et al.*, 2021]. In other words, we allow for the joint training of the two training phases of predictive coding and policy learning in this stage, so that the critic's evaluation of the state values can further help to mine the correlations between the trading assets from noisy and high-dimensional observation data. The advantage of learning a hybrid trading machine is that the two training phases can be deeply bound, which integrates the forward modeling capabilities of predictive coding with the advantages of Rl-for-finance methods for the flexibility of the trading policy. Specifically, we optimize the critic loss $\mathcal{J}(Q)$ by

$$\begin{aligned} &\min_{\phi,\psi} \mathbb{E}_{(s_t,a_t)\sim\mathcal{D}}[1/2(Q(s_t,a_t) - \widehat{Q}(s_t,a_t))^2], \text{ where} \\ &\widehat{Q}(s_t,a_t) = R_t + \gamma\left(Q_{\phi'}\left(s_{t+1},a_{t+1}\right) - \lambda \log \pi_\theta\left(a_{t+1} \mid s_{t+1}\right)\right), \end{aligned} \tag{6}$$

where θ, ϕ, and ψ represent the parameters from the actor network, the critic network, and the relation inference module respectively, $\mathcal{D}$ is the replay buffer, $Q_{\phi'}$ is the target Q network and $a_{t+1} \sim \pi_\theta(\cdot \mid s_{t+1})$. For the actor loss $\mathcal{J}(\pi_\theta)$, we have

$$\min_\theta \mathbb{E}_{s_t\sim\mathcal{D}}\left[D_{\text{KL}}(\pi_\theta(\cdot|s_t) \parallel \exp(Q_\phi(s_t,\cdot))/Z_\phi(s_t))\right], \tag{7}$$

where Z_ϕ is a normalization factor.

5 Experiments

5.1 Compared Methods

We compare StockFormer with the following methods:

- The "*market benchmarks*" in terms of CSI-300 Index and NASDAQ Composite Index.
- Min-variance portfolio allocation strategy (Min-Var) [Basak and Chabakauri, 2010] that balances returns and risks.
- Recent advances for stock prediction [Wang *et al.*, 2021; Feng *et al.*, 2019; Duan *et al.*, 2022] and a cutting-edge Transformer model for general time series prediction [Wu *et al.*, 2021]. We here use the "*buy-and-hold*" strategy, in which we buy a stock each day that has the highest estimated return in the next 5 days and sell it 5 days later .
- RL methods, including SAC [Haarnoja *et al.*, 2018], DDPG [Lillicrap *et al.*, 2016], and SARL [Ye *et al.*, 2020]. The first two models are implemented on the FinRL platform [Liu *et al.*, 2021] and directly perform policy learning on the technical indicators and the covariance matrix between the trading targets. We reproduce the SARL and feed the current asset prices (rather than the financial news) to its external state encoder for a fair comparison.

Market	# Assets	# Train Days	# Test Days
CSI-300	88	1935	728
NASDAQ-100	86	2000	756
Cryptocurrency	27	1108	258

Table 2: Statistical details of the finance investment datasets.

All models are experimented with the transaction costs and each RL method is performed three times with different seeds.

5.2 CSI-300 Stock Dataset

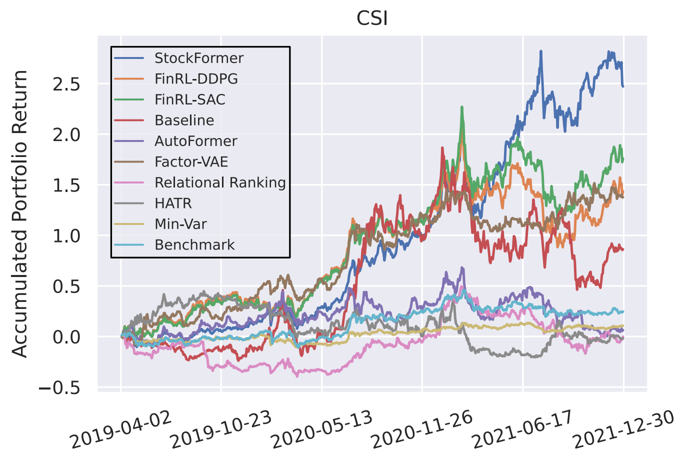
This dataset contains stock data from the CSI-300 Composite Index from 01/17/2011 to 12/30/2021. As shown in Table 2, the dataset is divided into the training and test splits containing the basic stock price-volume information in 1,935 days and 728 trading days respectively. Furthermore, we follow [Feng *et al.*, 2019] to retain the stocks that have been traded on more than 98% training days since 01/17/2011. Our investment pool contains 88 stocks. If a stock in the training set is suspended from trading, we interpolate the missing data in using the daily rate of change of the CSI-300 Composite Index.

Figure 3(Left) shows qualitative results of the accumulated portfolio return on the CSI-300 test set. It shows that StockFormer outperforms other compared models by large margins, including both stock prediction models trading with the "*buy-and-hold*" strategy, as well as the RL-for-finance methods. Notably, the red curve presents the results of the proposed long-term prediction module without involving any other branches or the decision module in StockFormer, which is trained to predict future returns in 5 days and works with the "*buy-and-hold*" strategy as other stock prediction models. The remarkable advantage of this baseline model compared with existing approaches, including HATR [Wang *et al.*, 2021], Relational Ranking [Feng *et al.*, 2019], and AutoFormer [Wu *et al.*, 2021], demonstrates the powerful dynamics modeling capability of the proposed Transformer architecture with diversified multi-head attention.

Table 3 provides corresponding quantitative results. Besides the total *portfolio return* (PR), we follow the work of Factor-VAE [Duan *et al.*, 2022] to include *annual return* (AR), *Sharpe*

Method	CSI-300				NASDAQ-100				Cryptocurrency	
	PR$^{\uparrow}$	AR$^{\uparrow}$	SR$^{\uparrow}$	MDD$^{\downarrow}$	PR$^{\uparrow}$	AR$^{\uparrow}$	SR$^{\uparrow}$	MDD$^{\downarrow}$	PR$^{\uparrow}$	SR$^{\uparrow}$
Market benchmark	0.24	0.08	0.51	0.18	1.05	0.30	1.16	0.30	-	-
Min-Var [Basak and Chabakauri, 2010]	0.11	0.04	0.38	**0.13**	0.59	0.18	0.99	0.26	-0.09	0.02
HATR [Wang *et al.*, 2021]	-0.02	-0.01	0.14	0.45	0.08	0.28	0.25	0.41	-0.65	-0.66
Relational Ranking [Feng *et al.*, 2019]	-0.01	-0.01	0.17	0.45	0.96	0.28	0.95	0.30	-	-
FactorVAE [Duan *et al.*, 2022]	1.37	0.38	1.27	0.17	1.20	0.33	0.99	**0.24**	-	-
AutoFormer [Wu *et al.*, 2021]	0.08	0.03	0.25	0.40	-0.25	-0.10	-0.21	0.42	-0.27	-0.16
Our future prediction module ($t+5$)	0.89	0.27	0.73	0.49	0.66	0.20	0.64	0.44	-0.40	-0.71
SARL [Ye *et al.*, 2020]	1.59	0.39	1.38	0.31	1.22	0.30	0.99	0.34	0.06	0.43
FinRL-SAC [Liu *et al.*, 2021]	1.76	0.42	1.41	0.34	1.38	0.34	1.24	0.33	0.10	0.55
FinRL-DDPG [Liu *et al.*, 2021]	1.43	0.36	1.23	0.39	0.83	0.22	0.87	0.33	0.15	0.60
StockFormer	**2.47**	**0.54**	**1.73**	<u>0.31</u>	**1.71**	**0.40**	**1.39**	<u>0.31</u>	**0.24**	**0.75**

Table 3: Quantitative results in portfolio return (**PR**), annual return (**AR**), Sharpe ratio (**SR**), and maximum drawdown (**MDD**) on the test splits of the stock market datasets. Transaction costs are included in buying and selling actions. For stock prediction methods (Rows 3-7), we use the "*buy-and-hold*" strategy. In the last two columns, we show the results on the cryptocurrency dataset.



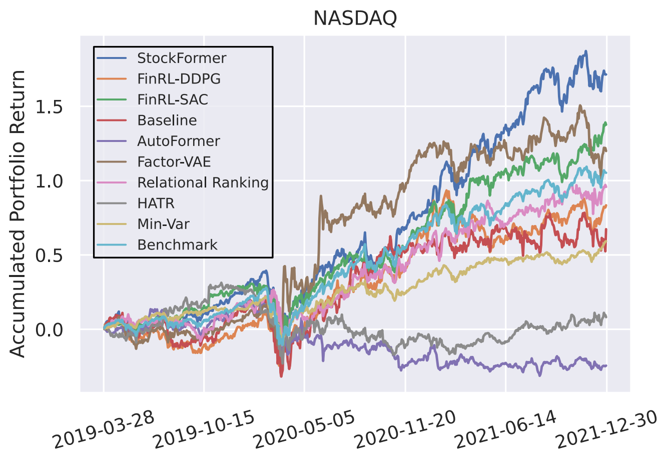


Figure 3: Accumulated portfolio returns on the CSI (Left) and NASDAQ (Right) test sets. We use the "*buy-and-hold*" strategy for the stock prediction models, including HATR, FactorVAE, and our baseline model (*i.e.*, the long-term prediction Transformer branch in StockFormer).

ratio (SR), and *maximum drawdown* (MDD) as the evaluation metrics. It is worth noting that the final StockFormer further improves the future prediction baseline (Row 7) and significantly outperforms all of the stock prediction models by learning flexible RL strategies. StockFormer also outperforms the vanilla SAC by 40.3% ($1.76 \rightarrow 2.47$) in the portfolio return and by 22.7% ($1.41 \rightarrow 1.73$) in the Sharpe ratio. Such a profit boost comes from executing policy optimization over the extracted relational and predictive states.

5.3 NASDAQ-100 Stock Dataset

For the NASDAQ stock dataset, we collect the daily price-volume records and related technical indicators between 01/17/2011 and 12/30/2021 from Yahoo Finance. Like in CSI-300, we use the 98% criteria to filter stocks, which derives an investment pool of 86 stocks, and then fill in the missing data based on the daily rate of change of the NASDAQ 100 Index. We construct the training and test datasets that involve 2,000 and 756 trading days respectively. As shown in Table 3 and Figure 3(Right), on the NASDAQ-100 test set, due to the effectiveness of predictive coding, StockFormer improves the vanilla SAC by 23.9% ($1.38 \rightarrow 1.71$) in the portfolio return and by 12.1% ($1.24 \rightarrow 1.39$) in Sharpe ratio.

5.4 Cryptocurrency Dataset

We further evaluate StockFormer by using it to make trading decisions in the cryptocurrency market. With the 98% filtering criteria, we select 27 cryptocurrencies and collect their daily records between 11/01/2017 and 05/01/2022 from Yahoo Finance. We split the data into a training set of 1,108 trading days and a test set of 258 trading days.

An interesting result in Table 3 is that most stock prediction models present negative return ratios on the test set, even worse than the Min-Var baseline. We suspect that this is due to the complexity and the non-stationarity of the cryptocurrency data. In other words, it is difficult to model the temporal dynamics accurately in such data scenarios, which further highlights the advantage of the model-free RL methods. For FinRL, SARL, and StockFormer, they do not explicitly model the transition function in the state space, and can flexibly produce more

Input latent states of RL agents	PR	SR
W/o relational states	1.42	1.43
W/o short-term predictive states	1.53	1.43
W/o long-term predictive states	1.45	1.34
StockFormer (Final)	**2.47**	**1.73**

Table 4: Ablation study of the necessity of individual branches in StockFormer on CSI. We remove each branch separately while maintaining the other two branches for the decision module.

Relation module	Future prediction modules	PR	SR
FFN	FFN	1.29	1.27
Multi-head FFN	FFN	1.48	1.29
FFN	Multi-head FFN	1.52	1.31
Multi-head FFN	Multi-head FFN (**StockFormer**)	**1.71**	**1.39**

Table 5: Ablation study of the proposed multi-head FFN in attention blocks in StockFormer on NASDAQ, compared with the original attention block [Vaswani *et al.*, 2017] with conventional FFNs.

conservative trading policies (as shown by the Sharpe ratios).

Another interesting fact in Table 3 is that while the future prediction module in StockFormer results in a negative portfolio return when working alone, its predictive coding states can still benefit the hybrid trading machine when integrated into StockFormer. We can see that StockFormer achieves positive and the best performance, which has a 140.0% $(0.10 \rightarrow 0.24)$ improvement over the vanilla SAC method in the portfolio return and a 36.4% $(0.55 \rightarrow 0.75)$ improvement in Sharpe ratio, showing the ability to control the investment risks.

5.5 Ablation Study

Necessity of individual Transformer branches. To verify the performance of the relation inference module and future prediction modules, we compare three variants of StockFormer on the CSI dataset. From Table 4, we can see that removing any encoding branch leads to a remarkable performance drop, and each branch makes essential contributions to the final performance of our final approach. The results demonstrate that the relation module and prediction modules provide complementary representations to the decision module from different perspectives of concurrent time series modeling.

Performance gain of the modified attention block. Table 5 compares the results of three StockFormer variants, in which we may use the original Transformer architecture without the proposed multi-head FFNs in any one of the three predictive coding branches. These results show that multi-head FFNs can effectively improve the attention block in Transformer by mitigating the difficulty of concurrent time series modeling.

Design of the decision module. An important question is how to integrate different types of predictive embeddings into a unified state space. To answer this question, we study different network structures for fusing the relational and long-/short-term predictive states in the decision module and show the results in Table 6. By replacing the multi-head attention layers in StockFormer with two fully connected layers (denoted by "2-layer FFN"), we observe that the portfolio return drops from 1.71 to 1.62 and the Sharpe ratio drops from 1.39 to 1.31. The results validate the effectiveness of the proposed architecture of the decision module: Exploiting a series of multi-head attention layers better integrates different sources of latent representations into a unified state space, which can eventually benefit policy optimization.

State fusion in decision module	PR	SR
2-layer FFN	1.62	1.31
2-layer multi-head attention (**StockFormer**)	**1.71**	**1.39**

Table 6: Evaluation on NASDAQ of different methods that integrate the relational states and the predictive states in the decision module.

Actor gradient	Critic gradient	PR	SR
✗	✗	0.03	0.35
✓	✗	0.02	0.35
✗	✓(**StockFormer**)	**0.24**	**0.75**

Table 7: Ablation study of back-propagating the actor-critic gradients to the relation inference module on the cryptocurrency dataset.

Joint training vs. Two separate training stages. The two training phases, predictive coding and policy learning, are closely related. In Table 7, we experiment with different gradient back-propagation solutions in the policy learning phase of the actor-critic method. We observe that StockFormer achieves the best results by passing the gradients of the critic loss back into the relation inference module. It significantly improves the baseline model with two separate training phases (PR: $0.03 \rightarrow 0.24$), in which the RL agent does not propagate any gradients back to the predictive coding modules. We believe that the critic's evaluation of the state values can further help to mine the correlations between the trading assets from noisy and high-dimensional observation data. These results show the superiority of the joint learning mechanism.

6 Conclusions

This paper presents StockFormer, an RL approach for financial market decision-making. It has two main contributions. First, it incorporates predictive coding in the actor-critic RL framework, which integrates the advantages of existing stock prediction models in learning future dynamics and the advantages of RL-for-finance methods in learning more flexible polices. Second, we proposed to use three branches of the modified Transformers to learn a mixture of long–/short-term predictive states and relational states. These three types of states are then combined adaptively and progressively through a series of attention structures in the decision-making module, such that the RL agent can optimize the decisions in a unified and meaningful state space. StockFormer shows competitive results on three finance datasets compared with a wide range of deep learning approaches, including both stock prediction and RL-for-finance models.

Acknowledgements

This work was supported by NSFC (62250062, U19B2035, 62106144), Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102), the Fundamental Research Funds for the Central Universities, and Shanghai Sailing Program (21Z510202133) from the Science and Technology Commission of Shanghai Municipality.

References

[Basak and Chabakauri, 2010] Suleyman Basak and Georgy Chabakauri. Dynamic mean-variance asset allocation. *The Review of Financial Studies*, 23(8):2970–3016, 2010.

[Benhamou *et al.*, 2020] Eric Benhamou, David Saltiel, Jean Jacques Ohana, Jamal Atif, and Rida Laraki. Deep reinforcement learning (DRL) for portfolio allocation. In *ECMLPKDD*, 2020.

[Cho *et al.*, 2019] Chun-Hung Cho, Guan-Yi Lee, Yueh-Lin Tsai, and Kun-Chan Lan. Toward stock price prediction using deep learning. In *UCC*, 2019.

[Duan *et al.*, 2022] Yitong Duan, Lei Wang, Qizhong Zhang, and Jian Li. FactorVAE: A probabilistic dynamic factor model based on variational autoencoder for predicting cross-sectional stock returns. In *AAAI*, 2022.

[Elias, 1955] Peter Elias. Predictive coding–I. *IRE transactions on information theory*, 1(1):16–24, 1955.

[Fedus *et al.*, 2021] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *arXiv preprint arXiv:2101.03961*, 2021.

[Feng *et al.*, 2019] Fuli Feng, Xiangnan He, Xiang Wang, Cheng Luo, Yiqun Liu, and Tat-Seng Chua. Temporal relational ranking for stock prediction. *ACM Transactions on Information Systems*, 37(2):1–30, 2019.

[Haarnoja *et al.*, 2018] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

[Hoseinzade and Haratizadeh, 2019] Ehsan Hoseinzade and Saman Haratizadeh. CNNpred: CNN-based stock market prediction using a diverse set of variables. *Expert Systems with Applications*, 129:273–285, 2019.

[Hu and Lin, 2019] Yuh-Jong Hu and Shang-Jen Lin. Deep reinforcement learning for optimizing finance portfolio management. In *AICAI*, 2019.

[Hu *et al.*, 2018] Ziniu Hu, Weiqing Liu, Jiang Bian, Xuanzhe Liu, and Tie-Yan Liu. Listening to chaotic whispers: A deep learning framework for news-oriented stock trend prediction. In *WSDM*, 2018.

[Huotari *et al.*, 2020] Tommi Huotari, Jyrki Savolainen, and Mikael Collan. Deep reinforcement learning agent for S&P 500 stock selection. *Axioms*, 9(4):130, 2020.

[Kitaev *et al.*, 2020] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *ICLR*, 2020.

[Lee *et al.*, 2020] Alex X Lee, Anusha Nagabandi, Pieter Abbeel, and Sergey Levine. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. In *NeurIPS*, 2020.

[Li *et al.*, 2018] Hao Li, Yanyan Shen, and Yanmin Zhu. Stock price prediction using attention-based multi-input LSTM. In *ACML*, 2018.

[Liang *et al.*, 2018] Zhipeng Liang, Hao Chen, Junhao Zhu, Kangkang Jiang, and Yanran Li. Adversarial deep reinforcement learning in portfolio management. *arXiv preprint arXiv:1808.09940*, 2018.

[Lillicrap *et al.*, 2016] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *ICLR*, 2016.

[Liu *et al.*, 2021] Xiao-Yang Liu, Hongyang Yang, Jiechao Gao, and Christina Dan Wang. FinRL: Deep reinforcement learning framework to automate trading in quantitative finance. In *ICAIF*, 2021.

[Mikolov *et al.*, 2013] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *ICLR*, 2013.

[Oord *et al.*, 2018] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[Patil *et al.*, 2020] Pratik Patil, Ching-Seh Mike Wu, Katerina Potika, and Marjan Orang. Stock market prediction using ensemble of graph theory, machine learning and deep learning models. In *ICSIM*, 2020.

[Puterman, 2014] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[Qin *et al.*, 2017] Yao Qin, Dongjin Song, Haifeng Chen, Wei Cheng, Guofei Jiang, and Garrison Cottrell. A dual-stage attention-based recurrent neural network for time series prediction. *arXiv preprint arXiv:1704.02971*, 2017.

[Rao and Ballard, 1999] Rajesh PN Rao and Dana H Ballard. Predictive coding in the visual cortex: A functional interpretation of some extra-classical receptive-field effects. *Nature neuroscience*, 2(1):79–87, 1999.

[Spratling, 2017] Michael W Spratling. A review of predictive coding algorithms. *Brain and cognition*, 112:92–97, 2017.

[Suri *et al.*, 2021] Karush Suri, Xiao Qi Shi, Konstantinos Plataniotis, and Yuri Lawryshyn. TradeR: Practical deep hierarchical reinforcement learning for trade execution. *arXiv preprint arXiv:2104.00620*, 2021.

[Théate and Ernst, 2021] Thibaut Théate and Damien Ernst. An application of deep reinforcement learning to algorithmic trading. *Expert Systems with Applications*, 173:114632, 2021.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

[Wang *et al.*, 2021] Heyuan Wang, Shun Li, Tengjiao Wang, and Jiayi Zheng. Hierarchical adaptive temporal-relational modeling for stock trend prediction. In *IJCAI*, 2021.

[Wen *et al.*, 2019] Min Wen, Ping Li, Lingfei Zhang, and Yan Chen. Stock market trend prediction using high-order information of time series. *IEEE Access*, 7:28299–28308, 2019.

[Weng *et al.*, 2020] Liguo Weng, Xudong Sun, Min Xia, Jia Liu, and Yiqing Xu. Portfolio trading system of digital currencies: A deep reinforcement learning with multidimensional attention gating mechanism. *Neurocomputing*, 402:171–182, 2020.

[Wu *et al.*, 2019] Jheng-Long Wu, Chi-Sheng Yang, Kai-Hsuan Liu, and Min-Tzu Huang. A deep learning model for dimensional valencearousal intensity prediction in stock market. In *iCAST*, 2019.

[Wu *et al.*, 2021] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In *NeurIPS*, 2021.

[Yarats *et al.*, 2021] Denis Yarats, Amy Zhang, Ilya Kostrikov, Brandon Amos, Joelle Pineau, and Rob Fergus. Improving sample efficiency in model-free reinforcement learning from images. In *AAAI*, 2021.

[Ye *et al.*, 2020] Yunan Ye, Hengzhi Pei, Boxin Wang, Pin-Yu Chen, Yada Zhu, Ju Xiao, and Bo Li. Reinforcement-learning based portfolio management with augmented asset movement prediction states. In *AAAI*, 2020.

[Zhang *et al.*, 2017] Liheng Zhang, Charu Aggarwal, and Guo-Jun Qi. Stock price prediction via discovering multi-frequency trading patterns. In *KDD*, 2017.

[Zhong *et al.*, 2020] Yueyang Zhong, YeeMan Bergstrom, and Amy R Ward. Data-driven market-making via model-free learning. In *IJCAI*, 2020.

[Zhou *et al.*, 2021] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *AAAI*, 2021.