Prediction with Time-Series Mixer for the S&P500 Index

Junyi Ye
Department of Computer Science
New Jersey Institute of Technology
Newark, USA
jy394@njit.edu

Jingyi Gu
Department of Computer Science
New Jersey Institute of Technology
Newark, USA
jg95@njit.edu

Ankan Dash
Department of Computer Science
New Jersey Institute of Technology
Newark, USA
ad892@njit.edu

Fadi P. Deek
Department of Informatics
New Jersey Institute of Technology
Newark, USA
fadi.deek@njit.edu

Guiling "Grace" Wang
Department of Computer Science
New Jersey Institute of Technology
Newark, USA
gwang@njit.edu

2023 IEEE 39th International Conference on Data Engineering Workshops (ICDEW) | 979-8-3503-2244-6/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICDEW58674.2023.00008

***Abstract*—As an essential US economic indicator, the S&P500 Index is used to assess the current state of market performance and gauge the economy's future course. However, stock market index prediction is challenging due to its nonlinearity and inherently volatile character. Recurrent Neural Networks (RNN) and their variants are de facto standards for sequence modeling. Recently, Convolutional Neural Networks (CNN) and attention-based networks, such as dilated casual convolutions and Transformers, have also become popular in time series forecasting. In this paper, we report on the design of a Time-Series Mixer (TS-Mixer) architecture based on MLP-Mixer, an all-MLP architecture for time series forecasting. To the best of our knowledge, this is the first implementation of MLP-Mixer-based architecture for sequence modeling. Modern deep learning models are increasingly built to handle univariate time series data. They generally pay attention to analyzing temporal dependencies while ignoring the relationship among features. The proposed architecture is specifically created for multivariate time series forecasting to capture temporal feature interactions while simultaneously learning feature correlations. To accomplish this, the proposed Time-Feature Mixer contains two types of MLP layers: feature mixer and temporal mixer. The feature mixer is applied independently to each data point to capture the correlation among features. In contrast, the temporal mixer extracts temporal dependency (trend, seasonal, cyclical, or random characteristics) of each feature across the whole input sequence. Compared to prevalent neural networks in sequence modeling, TS-Mixer exhibits competitive performance regarding S&P500 Index prediction.**

***Index Terms*—stock market prediction, time series forecasting, neural networks, multi-layer perceptron, MLP-Mixer**

I. Introduction

Stock market prediction, an interdisciplinary study on finance and data science, has become one of the most popular Fintech applications [1] [2]. The Standard and Poor's 500 (S&P500) is an essential benchmark for evaluating current market performance and forecasting the economy's direction. It includes the 500 leading companies and covers approximately 70% of the available stock market capitalization in the United States. Therefore, it is commonly used as a proxy to reflect the stock market's performance.

Many researchers aim to analyze the stock market and build effective prediction models for stock market forecasting. The stock market analysis has two basic categories: fundamental analysis and technical analysis. Fundamental analysis evaluates the stock price based on its intrinsic value. While technical analysis relies on charts and pattern recognition [3]. Recent trends point to a significant increase in the usage of experience-based technical indicators as customized features for deep neural networks and data-driven machine learning models, drawing considerable interest from both academia and the financial sector.

Much of the existing work for stock market prediction using deep learning focuses on developing complex deep learning models. Some comprise intricate structures, such as gating mechanisms, attention mechanisms, and convolutional layers with temporal designs. Though effective, these complex approaches take much training time to meaningfully investigate temporal and intercorrelation patterns, which require extensive computing resources and are inefficient during real-world deployment. Our work aims to design a simple model that does not rely on any form of costly complex attention mechanism, gates, or convolution operation, yet performs competitively compared to existing models. It is capable of fully exploring multi-dimensional dynamics between temporal and feature relationships.

Thus, we propose an all-multi-layer perceptron (All-MLP) architecture, MLP-Mixer-based deep learning model for S&P500 index forecasting, named Time-Series Mixer (TS-Mixer). To the best of our knowledge, this is the first attempt at an MLP-Mixer-based design for sequence modeling. The central architecture of the TS-Mixer is inspired by MLP-Mixer [4], a competitive but conceptually and technically simple alternative that does not use gating structure, convolutions, or self-attention. MLPs in TS-Mixer are repeatedly applied to

2473-3490/23/$31.00 ©2023 IEEE
DOI 10.1109/ICDEW58674.2023.00008

either per-location features or temporal features, which can be considered a novel method to capture temporal correlations. Like MLP-Mixers, TS-Mixer relies only on basic matrix multiplication routines, changes to data layout (reshapes and transpositions), and scalar nonlinearities. The experiments on SPX in a high-volatility period demonstrate that our proposed TS-Mixer outperforms multiple benchmarks, including traditional indicators, RNN-based models, and popular time-series models. The major contributions are summarized as follows:

- We produced an MLP-Mixer-based model for time-series stock prediction. The model is based exclusively on multi-layer perceptrons or MLPs to learn the patterns in stock index data.
- Our proposed model can learn not only temporal relationships but also inter-feature relationships. To accomplish this, we use two types of layers: first, is an all MLP layer which is applied independently to each data point within a single sub-sequence as well as their features; second, is also an all MLP layer applied to different subsequences within the whole input sequence.
- By using this approach, our model can learn the relationships between features and the temporal dependency of each feature across the whole input sequence.
- We also highlight that, even though the test dataset is very challenging since it covers the period between 2019 and 2020 (global slowdown due to covid pandemic) and has an entirely different pattern to the training and validation dataset, our model is still able to perform competitively compared to existing baselines in time-series framework.

The remainder of the paper is organized as follows: Section II summarizes existing work in stock market prediction; Section III formulates our stock price prediction problem; Section IV presents the proposed framework in detail; Section V evaluates our proposed model, compares it with baselines regarding prediction performance, and exhibits the hyper-parameter analysis. The paper concludes with Section VI discussing future directions.

II. Literature Review

Stock market forecasting is regarded as a crucial undertaking that requires careful consideration since, with the right choices, a successful forecast could result in attractive returns. Stock market forecasting has drawn interest from economists and computer scientists as a classic yet challenging topic. The data's nonlinear, noisy, and chaotic character makes stock market prediction a significantly challenging task. Popular technical analysis includes, but is not limited to, Moving Average (MA), Bollinger Bands, and Moving Average Convergence/Divergence (MACD). Linear models such as Autoregressive Integrated Moving Average (ARIMA) [5] and Generalized Autoregressive Conditional Heteroskedasticity (GARCH) [6] also show powerful predictive ability on pricing.

Machine learning and artificial intelligence have recently attracted much interest [7]–[10] for their extraordinary ability to learn nonlinear relationships and handle massive volumes of data, especially Long short-term memory (LSTM) and Gated Recurrent Unit (GRU). They have been widely used to model temporal dependency on stock prices. Qin et al. [11] used Dual-Stage Attention-Based Recurrent Neural Network (DARNN) for time-series based stock prediction tasks. DARNN is a two-stage architecture that uses attention mechanisms to capture relevant information in the input time-series data. The two-stage attention mechanism in DARNN allows the network to focus selectively on different aspects of the input data, which is particularly useful for time-series prediction tasks. Feng et al. [12] proposed a novel approach AdvLSTM to enhance the prediction accuracy of the stock movement by leveraging adversarial training. They include a novel adversarial training framework that consists of two modules: a generator network and a discriminator network. The generator network takes as input a sequence of historical stock prices and outputs the predicted stock price movements for the next day. The discriminator network is trained to differentiate between the predicted movements produced by the generator network and the actual movements. Moreover, additional emerging works apply other types of deep learning models for stock market prediction. Wang et al. [13] proposed a novel Hierarchical Adaptive Temporal-Relational Network (HATR) which employed Temporal Convolutional Network (TCN) to capture multi-scale dynamic patterns by stacking dilated causal convolutions and gating paths. Zhou et al. [14] applied an N-BEATS deep learning technique to the stock market. N-BEATS based on backward and forward residual links and a deep stack of fully connected layers achieves state-of-the-art performance on a wide range of time-series datasets [15].

Using historical data to predict stock market performance is one approach. There have been efforts to use other types of extrinsic data, such as textual data from Twitter and Reddit, macroeconomics indicators, and fundamental sector and company data to augment stock price prediction performance. Swathi et al. [16] used a deep learning-based LSTM model for stock price prediction using Twitter sentiment analysis. The model considers the sentiment of tweets related to a company or industry and the historical stock prices to predict future stock prices. According to the authors, their model can outperform the existing models in terms of prediction accuracy and reliability. Wang et al. [17] incorporated macroeconomics factors in learning macro-micro interactions. They introduced deep learning into the copula to model the coupling and influence between macro-level factors and micro-level stock prices.

III. Problem Formulation

In this section, we present the problem statement. The general time-series forecasting problem is described as follows. Given an input sequence $x_0, ..., x_{T-1}$, the goal is to predict the corresponding outputs $y_T, ..., y_{T+K-1}$. Here T is the period of observation and K is the period of prediction. Input sequence length T is a tuning hyper-parameter and K is determined by the need of the forecasting task. To predict the output y_t at

time t, we are constrained to use previously observed inputs $x_0, ..., x_{t-1}$ only. Formally, a time-series forecasting model is a function f mapping from $x_0, ..., x_{T-1}$ to $y_T, ..., y_{T+K-1}$:

$$y_T, ..., y_{T+K-1} = f(x_0, ..., x_{T-1}) \tag{1}$$

Generally, time-series prediction can be formulated into the following types of forecasting tasks:

1) One-Step-Ahead Time-Series Forecasting: One-step-ahead forecasting is described as:

$$\hat{y}_t = f(x_{t-k}, ...x_{t-1}, y_{t-k}, ..., y_{t-1}) \tag{2}$$

where $\hat{y}_t$ is the model forecast, k is the look-back window.

2) Multi-step-ahead time-series forecasting: Multi-step-ahead forecasting is described as:

$$\hat{y}_{t+1}, ...\hat{y}_{t+K} = f(x_{t-T+1}, ...x_t, y_{t-T+1}, ..., y_t) \tag{3}$$

3) Univariate Forecasting: Univariate forecasting involves the analysis of a single feature without taking into account the effect of the other features. The underlying assumption for this formulation is that the impact of other features is embodied or reflected by the target feature [18]:

$$\hat{y}_{t+1}, ...\hat{y}_{t+K} = f(y_{t-T+1}, ..., y_t) \tag{4}$$

4) Multivariate Forecasting: Multivariate forecasting analysis involves multiple features across time which assume they are independent:

$$\hat{y}_{t+1}, ...\hat{y}_{t+K} = f(x_{t-T+1}, ...x_t, y_{t-T+1}, ..., y_t) \tag{5}$$

IV. TS-Mixer Architecture

The basic structure of TS-Mixer is similar to the design of MLP-Mixer. Figure 1 (top-left) depicts the macro-structure of TS-Mixer. TS-Mixer is an All-MLP architecture containing a per-sub-sequence fully-connected layer, several TS-Mixer blocks, a global average pooling layer, and a classifier or regression head based on downstream tasks. Other components include skip connections and dropout. Compared to MLP-Mixer, layer normalization is not used in TS-Mixer based on the evaluation results of our experiments.

Firstly, the input sequence is divided into several non-overlapping sub-sequences with the same shape (<sub-sequence length, number of features>). A per-sub-sequence fully-connected layer linearly projects each sub-sequence into a 1D vector with fixed length d_{model}, which allows TS-Mixer to be compatible with time-series inputs with arbitrary feature size. These vectors can be seen as patches in images or tokens in sentences. After the linear projection, the input sequence is converted into a 2D vector with the shape of (<number of sub-sequences, d_{model} >), and the dimensionality of the input is unchanged throughout the TS-Mixer blocks. TS-Mixer blocks are designed for capturing both long-term and short-term temporal patterns from original input sequences.

A. TS-Mixer Blocks

Like MLP-Mixer, TS-Mixer blocks use two types of MLP blocks: MLP1 blocks and MLP2 blocks, shown in Figure 1 (right). We name them long-term temporal mixers and short-term temporal mixers, respectively. Both MLP1 block and MLP2 block consist of two fully-connected layers and a Gaussian Error Linear Unit (GELU) [19] nonlinear activation is illustrated in Figure 1 (bottom-left). GELU is a high-performing neural network activation function. The GELU assigns weights for input by their percentile, different from ReLU, which multiplies the input by 1 or 0 based on their sign. GELU can be considered a smoother ReLU [19].

The **short-term temporal mixer** (i.e., MLP2) allows communication among data points within a single sub-sequence and their features. It repeatedly operates on each sub-sequence independently, capturing short-term temporal dependency within individual sub-sequence. The parameters in the short-term temporal mixer can be seen as the short-term memory of TS-Mixer that focuses on learning the relationship of local data points without considering the whole input sequence. Note that the parameter size in the short-term feature mixer is only determined by d_{model}, which is a tuning hyper-parameter and is independent of the input sequence length.

The **long-term temporal mixer** (i.e., MLP1) allows communication between different sub-sequences/tokens and captures temporal dependencies across the whole input sequence. It repeatedly operates on each feature independently and extracts long-term temporal dependency within the individual feature. The parameters in the long-term temporal mixer can be seen as the long-term memory of TS-Mixer that is expected to automatically capture the global patterns (i.e., seasonal, trend, cyclical, and random characteristics) of an individual feature across the whole input sequence. For instance, the overall trend of the US stock market has kept increasing in the most recent 20 years. These two types of temporal mixers are interleaved to enable interaction of both short-term temporal and long-term temporal features, serving as an All-MLP multi-scaled temporal model without elaborate designs or complex computation units such as gate structures, convolutions, and attention layers.

We have designed three different TS-Mixer blocks with different orders of temporal mixers, as illustrated on the right side of Fig. 1.

1) **Mixer block** uses the same design as Mixer block from MLP-Mixer [4]. It has a short-term temporal mixer followed by a long-term temporal mixer.
2) **MixerReverse block** is the variant by reversing the order of short-term temporal mixer and long-term temporal mixer in Mixer block.
3) **MixerParallel block** is a variant that sums the input and the output from both short-term temporal mixer and long-term temporal mixer. Skip-connections are utilized in all three TS-Mixer blocks to help increase the depth of neural networks and ensure feature reusability.

In the extreme case, such as when the length of the sub-

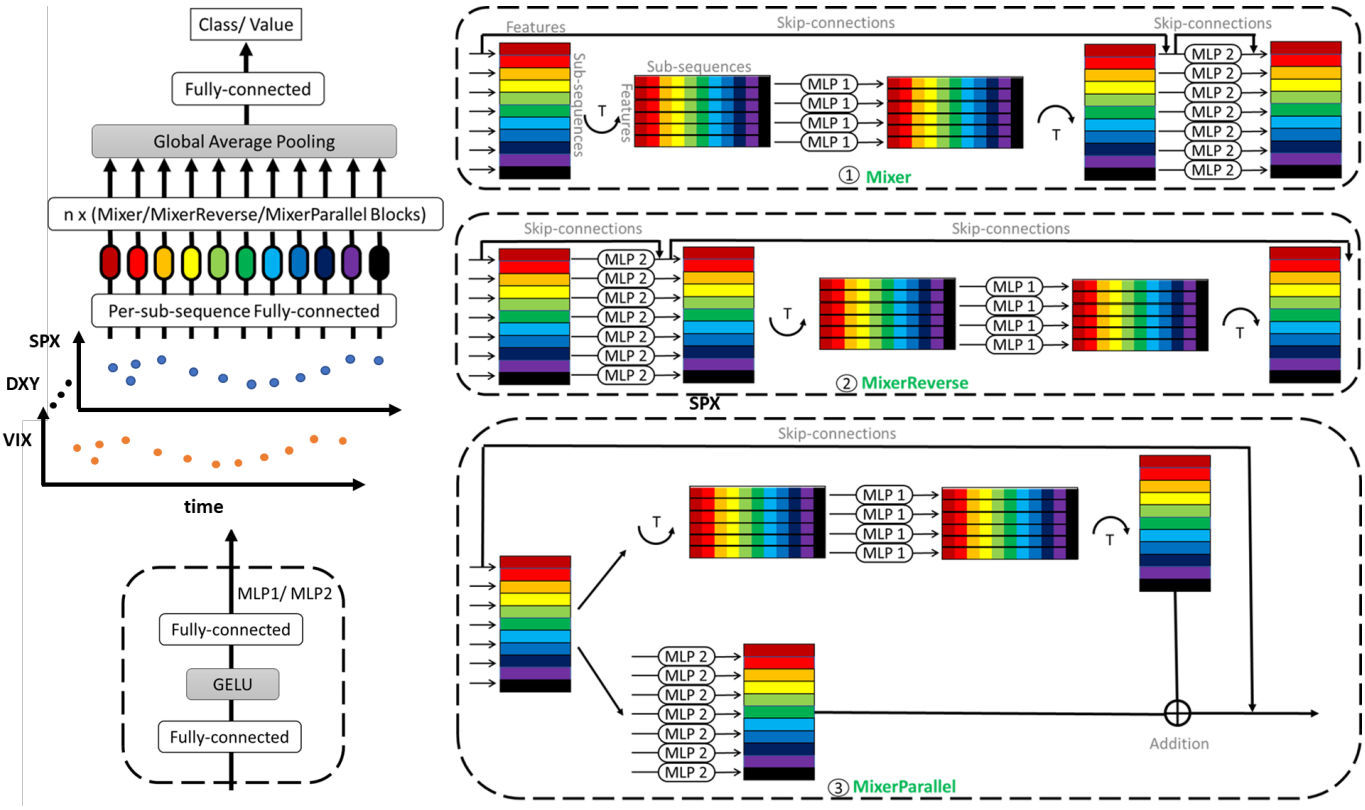

Fig. 1. The Architecture of TS-Mixer. The macro-structure of TS-Mixer (top-left). The illustration of MLP1 and MLP2 blocks (bottom-left). Three types of Mixer block design (right). ① Mixer block (the same Mixer block in MLP-Mixer); ② MixerReverse block; ③ MixerParallel block.

sequence equals one, the TS-Mixer architecture is equivalent to using 1×1 convolutions for feature mixing and single-channel depth-wise convolutions of a full receptive field and parameter sharing for temporal mixing. In other words, the short-term temporal mixer (MLP2) is simplified as a pure feature mixer without involving any temporal factors. In contrast, the long-term temporal mixer (MLP1) extracts both local and global temporal dependencies at the same time. Compared to CNN, the MLP blocks do not require additional costly matrix multiplication. After removing the layer normalization from the original MLP-Mixer architecture, the outputs of the short-term temporal mixer and long-term temporal mixer can be rewritten as

$$\begin{aligned} U_{*,i} &= X_{*,i} + W_2\sigma(W_1 X_{*,i}), i = 1, ..., C, \\ Y_{j,*} &= U_{j,*} + W_4\sigma(W_3 U_{j,*}), j = 1, ..., S. \end{aligned} \tag{6}$$

where σ is an element-wise nonlinearity GELU. $W_{1,2,3,4}$ are MLP weights. S and C are tunable hidden widths in the MLP2 and MLP1, respectively. Note that C is independent of the sequence size and the overall complexity is linear in the number of elements in the sequence.

V. Model Evaluation

A. Dataset Description

In this section, we describe the data source, selected features, and data pre-processing techniques. Our dataset consists of three components: S&P 500 Index, Dollar Index(DXY), and CBOE Volatility Index (VIX).

1) **S&P500 Index (SPX)** is considered an essential benchmark index for the U.S. stock market. It is a market-capitalization weighted index that is composed of the top 500 companies with the largest market capitalization (i.e., the total value of all a company's shares of stock) in the U.S. Predicting values of S&P500 Index is crucial because it gives investors and government officials a broad view of the economic health of the U.S. in the future. Our S&P500 Index dataset contains historical daily Opening, High, Low, and Closing prices, and Volume.
2) **U.S. dollar Index (DXY)** is a measure of the dollar's value against a basket of six world currencies. It reflects the dollar's value in global markets. The DXY dataset contains daily Opening, High, Low, and Closing prices.
3) **CBOE Volatility Index (VIX)** is an index that represents the maker's expectations for the relative strength of near-term price changes of the S&P500 Index. VIX is often used as a method to measure market sentiment (i.e., the degree of fear) among market participants. Our dataset contains daily High/Closing value and Low/Closing value of VIX.

All these features are publicly available on Yahoo Finance [20]. Except for US holidays, it has the daily data five days a

TABLE I
TRAIN, VALIDATION, TEST SETS SUMMARY

	Start Date	End Date	# of Samples
Train	2000-01-03	2016-10-19	4422
Validate	2016-10-20	2018-11-23	523
Test	2018-11-26	2020-12-30	523

week for 20 years.

The time span of our dataset is a total of 20 years, from 2000 to 2020, which encompasses US business cycle expansions and contractions. We split it into a train set (80%), a validation set (10%), and a test set (10%) across time. The detailed date range and the number of samples are illustrated in Table I. The validation set determines the best model during training for each hyper-parameter setting.

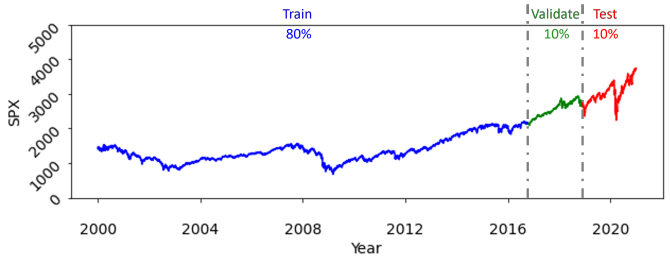

Fig. 2. SPX Dataset Chart (2000-2020).

Data normalization and data standardization are used before training the neural networks. Each feature and the target is rescaled into the range $[-1, 1]$ with the following equation.

$$x_{normalized} = \frac{2(x - min(x_{train}))}{(max(x_{train}) - min(x_{train}))} - 1 \quad (7)$$

where $min(x_{train})$ and $max(x_{train})$ are the maximum and minimum of feature x in the training set.

The dataset is segmented into sequences of five continuous trading days (i.e., weekly sequences). In other words, we predict the S&P500 Index value with data from the prior week.

B. Baseline Models

We conducted experiments on the following baseline models, including traditional technical analysis (SMA, EMA), RNN-related models (RNN, LSTM, GRU), and popular time-series methods (TCN, N-BEATS).

1) Simple Moving Average (SMA): Simple Moving Average calculates the average of a sequence of the selected range. The equation is illustrated below, where n is the number of samples in the sequence. For instance, SMA(5) forecasts the 6th variable with the average of the previous five variables. SMA(1) takes the latest value as the prediction.

$$\hat{x}_i = SMA_i(n) = \frac{x_{i-n} + x_{i-n+1} + ... + x_{i-1}}{n} \quad (8)$$

2) Exponential Moving Average (EMA): The major difference between SMA and EMA is that EMA assigns large weights to recent values, while SMA assigns equal weights to all values. Both SMA and EMA are techniques to smooth the fluctuations from sequence data. Compared with SMA, EMA is the preferred average among traders since it is more reactive to the most recent value than SMA. SMA and EMA are univariate methods that assume the future stock price can be forecasted with the historical stock prices.

$$EMA_i(n) = \frac{2 \times x_{i-1}}{1+n} + EMA_{i-1}(n) \times (1 - \frac{2}{1+n}) \quad (9)$$

where the smoothing number n typically equals 2.

3) Recurrent Neural Network (RNN): Recurrent Neural Networks, also known as RNNs, are a class of neural networks designed for time-series data. In contrast to MLPs, RNNs capture the correlation between data points and require memory to store the hidden state of previous inputs. The hidden states are calculated as follows:

$$h_t = tanh(W_{ih}x_t + b_{ih} + W_{hh}h_{t-1} + b_{hh}) \quad (10)$$

where $tanh$ is a nonlinear activation function. Ws and bs are weights and biases. For each value in the input sequence, each RNN layer computes the hidden states h_t at time t. x_t is the input at time t, and h_{t-1} is the hidden state of the previous RNN layer at time $t-1$. RNN has several advantages. (1) It can process inputs of any length. (2) The model size is independent of the input sequence length. (3) The computation considers historical information. (4) Weights in RNN are shared across time. However, RNN trains very slowly due to its chained computation graph and suffers from gradient vanishing and exploding problems [21], making it unsuitable for long sequence input. For each value in the input sequence, each RNN layer computes the hidden states h_t at time t. x_t is the input at time t, and h_{t-1} is the hidden state of the previous RNN layer at time $t-1$.

4) Long Short-Term Memory (LSTM): Long Short-Term Memory (LSTM) addresses the vanishing gradient problem in RNNs. An LSTM unit has three gates (input gate, forget gate, and output gate) and a cell structure. The cell remembers values over arbitrary time intervals, and three gates control the information flow into and out of the cell.

$$\begin{aligned} i_t &= \delta(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \\ f_t &= \delta(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \\ g_t &= tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}) \\ o_t &= \delta(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \\ c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\ h_t &= o_t \odot tanh(c_t) \end{aligned} \quad (11)$$

5) Gated Recurrent Unit (GRU): Kyunghyun et al. [22] introduced Gated Recurrent Unit (GRU) in 2014. It includes only two gate structures, update gate and reset gate in vanilla RNN. Compared with LSTM, GRU has fewer parameters and

lacks the cell unit. GRU exhibits better performance on small datasets [23].

$$\begin{aligned} r_t &= \delta(W_{ir}x_t + b_{ir} + W_{hr}h_{t-1} + b_{hr}) \\ z_t &= \delta(W_{iz}x_t + b_{iz} + W_{hz}h_{t-1} + b_{hz}) \\ n_t &= tanh(W_{in}x_t + b_{in} + r_t(W_{hn}h_{t-1} + b_{hn})) \\ h_t &= (1 - z_t)n_t + z_t h_{t-1} \end{aligned} \tag{12}$$

6) Temporal Convolutional Network (TCN): Temporal Convolutional Network (TCN) describes a family of generic convolutional architectures for sequence modeling [24]. It consists of residual connections and dilated causal convolutions layers. The dilated causal convolution is a special case of a one-dimensional fully connected layer. The dilation convolution operation employs a filter f to slide over sequential inputs x at time step t by skipping values with a certain dilation rate as follows:

$$x \circledast f_t = \sum_{s=0}^{k-1} f_s \cdot x(t - d \times s) \tag{13}$$

where d is the dilation rate to control the skipping step. When $d = 1$, it is equivalent to the one-dimensional convolution layer; as the dilation rate grows exponentially, the dilated convolution can receive a wider and deeper range of historical information. This scheme prevents information leakage from the future to the past. Compared with RNN-based models, TCN exhibits competitive performance in a large range of sequence modeling tasks [24] and allows parallel computation to improve efficiency.

7) N-BEATS: N-BEATS [15] is one of the state-of-the-art time-series deep learning models originally designed for solving univariate times-series forecasting problems. The basic building block of N-BEATs is fully-connected layers that do not use any memory units or gate structures like RNNs. It consists of a very deep stack of blocks that include stacks of fully-connected layers as well as backward and forward residual links. Each block has two outputs, backcast and forecast, where the forecast is the extracted features for the downstream task, i.e., the time-series forecasting and backcast refers to the best estimation of the input sequence. The architecture has two variants, (1) the generic architecture that substitutes the polynomial and harmonic basis for the identity basis. (2) the interpretable architecture, which projects the time series into polynomials and harmonic basis to learn trend and seasonality features. We use the generic configuration of N-BEATS as our baseline as it does not require any time-series-specific knowledge. More specifically, the backcast $\hat{x}_l$ and forecast $\hat{y}_l$ output of l-th block in generic architecture is defined as follows.

$$\hat{x}_l = V_b^l \theta_b^l + b_b^l, \hat{y}_l = V_f^l \theta_f^l + b_f^l \tag{14}$$

where V is the learned basis by the model, θ is the expansion coefficient for V and b is the bias. N-BEATS demonstrates state-of-the-art performance for two configurations on several well-known datasets, improving forecast accuracy by 11% over a statistical benchmark and by 3% over the winner, a hybrid model of RNN and Holt-Winters exponential smoothing of the M4 forecast competition.

TABLE II
HYPER-PARAMETER SUMMARY

Hyper-Parameter	Range of Values
Input Features	{S&P500, S&P500+DXY+VIX}
Data Normalization	{mean&variance, [0, 1], [-1, 1]}
Type of Mixer Blocks	{Mixer, MixerReverse, MixerParallel}
# of Mixer Blocks	{1, 2, 3, 4}
d_{model}	{32, 64, 128, 256}
Positional Encoding	{with, without}
Dropout	{0.1, 0.2, 0.3, 0.4}

TABLE III
PERFORMANCE EVALUATION

Model	MAE	RMSE
SMA(2)	29.835	45.718
SMA(3)	34.147	51.267
SMA(5)	41.914	61.542
SMA(10)	55.998	81.681
EMA(2)	29.249	45.586
EMA(3)	31.811	48.722
EMA(5)	37.752	56.207
EMA(10)	50.654	73.789
RNN	63.089	77.627
LSTM	35.989	49.940
GRU	71.410	85.220
N-BEATS	28.101	44.429
TCN	28.398	44.911
TS-Mixer	**28.014**	44.330
TS-Mixer-Reverse	28.427	**44.230**
TS-Mixer-Parallel	28.235	44.685

[1] The numbers next to SMA and EMA indicate the number of samples in the sequence, e.g., SMA(2) is calculated simple average price of the previous two days.

C. Hyper-Parameter Tuning

TS-Mixer is implemented in PyTorch. The hyper-parameters are tuned based on the validation set. The details of hyperparameters for the training process are shown in Table II.

D. Evaluation Metric

Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) are used to evaluate time-series forecasting models. All

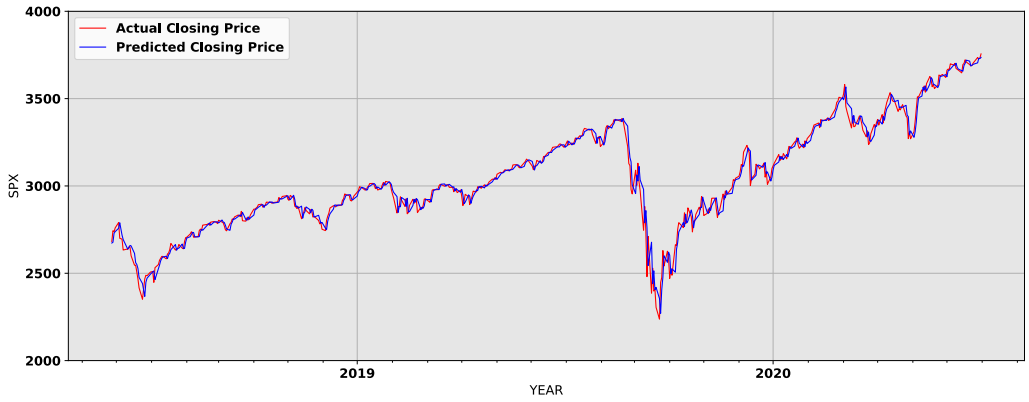

Fig. 3. Comparison of the actual closing price and predicted closing price from TS-Mixer during the testing period

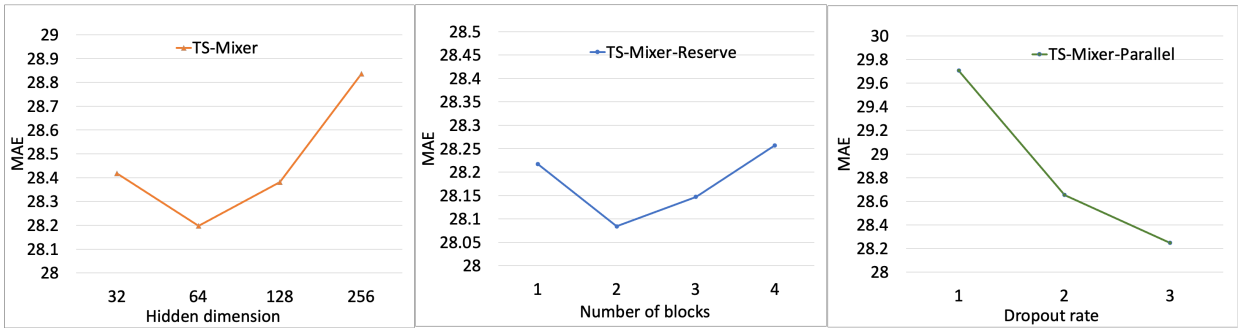

Fig. 4. Influence of Hyper-parameter on Performance.

metrics produce the average model prediction errors, which means lower values are better.

1) Root Mean Square Error (RMSE): RMSE is the standard deviation of the residuals (prediction errors). Residuals are a measure of how far from the regression line data points are; RMSE is a measure of how to spread out these residuals. In other words, it conveys how concentrated the data is around the line of best fit.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2} \tag{15}$$

where N is the total number of test samples.

2) Mean Absolute Error (MAE): MAE measures the average magnitude of the errors in a set of predictions without considering their direction. It is the average over the test samples of the absolute differences between prediction and actual observation where all individual differences have equal weight.

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|\hat{y}_i - y_i| \tag{16}$$

Since RMSE is squared before averaging, it gives a relatively high weight to large errors compared to MAE.

E. Comparison with baselines

Table III presents the numerical performance. Figure 3 shows the comparison of prediction and actual prices from TS-Mixer on SPX during the testing period. We make the following observations:

1) TS-Mixer and its variants achieve the best performance regarding both metrics. Compared with baselines, TS-Mixer outperforms EMA(2) among traditional indicators by 1.2 of MAE, and surpasses N-BEATS in neural networks by 0.9 of MAE. It validates the superior performance of the alternate MLP structure of TS-Mixer.
2) MAs based on the previous two or three days generally outperform the RNN-based model. N-BEATS and TCN, however, are competitive in spotting complicated dynamic patterns in the stock market because of their deep learning model structures and effective memory usage.
3) From Figure 3 we can observe that the predicted closing price from TS-Mixer is highly aligned with the actual target values. Even though the test data contains significantly different patterns from the training data, it is clear that the proposed model can learn the pertinent patterns and trends in the training data and perform well on the test data.

4) As opposed to the alternatives, the Mixer design, which consists of a long-term temporal mixer followed by a short-term temporal mixer, is more suitable for the price prediction task for SPX.

F. Hyper-Parameter Analysis

We further explore how our proposed model structure of TS-Mixer influences performance. Figure 4 shows the impact of various hyper-parameter values on MAE for TS-Mixer and its variants. As shown in the above figure, MAE initially decreases as the hidden dimension grows because TS-Mixer can capture more useful information. However, as the hidden dimension continues to increase, MAE begins to rise, possibly due to overly complex neural networks leading to overfitting. The middle figure exhibits a similar trend. With an increasing number of blocks, the ability of TS-Mixer-Reverse to capture both relevant temporal and feature representation, respectively, is enhanced at first, while subsequently weakened due to overfitting. The right figure demonstrates an increasing dropout rate effectively regularizes the model complexity, thereby boosting the performance of TS-Mixer-Parallel regarding MAE.

VI. Conclusion and Future Work

The S&P500 Index is a determinate US economic indicator, allowing us to evaluate the current market performance and forecast the direction of the economy. However, due to the nonlinearity and volatile nature of the stock market, index prediction is a challenging task. RNN and its variants are the de facto standards for sequence modeling. Recently, CNN and attention-based networks, such as 1D CNNs and Transformers, have also become popular in time-series analysis. In this paper, we report on the design of an MLP-Mixer-based architecture named Time-Series Mixer (TS-Mixer), an all-MLP architecture for time-series forecasting. Our literature review indicates that this is the first time that MLP-Mixer-based architecture is utilized to sequence modeling. TS-Mixer contains two types of layers: long-term and short-term temporal mixer. One with MLPs applies independently to each sub-sequence (i.e., captures temporal patterns in short-term period), and the other with MLPs applied across the whole input sequence (i.e., extracts long-term trend, seasonal, cyclical, or random characteristics). Extensive experiments on SPX in a high-volatility period validate that TS-Mixer exhibits competitive performance in predicting S&P500 Index, surpassing other popular time-series frameworks.

References

[1] S. R. Das, "The future of fintech," *Financial Management*, vol. 48, no. 4, pp. 981–1007, 2019.

[2] T. Paul, "Fintech empowers prediction of stock market index using artificial neural network," in *2021 International Conference on Artificial Intelligence and Computer Science Technology (ICAICST)*, 2021, pp. 42–46.

[3] W. Jiang, "Applications of deep learning in stock market prediction: recent progress," *Expert Systems with Applications*, vol. 184, p. 115537, 2021.

[4] I. O. Tolstikhin, N. Houlsby, A. Kolesnikov, L. Beyer, X. Zhai, T. Unterthiner, J. Yung, A. Steiner, D. Keysers, J. Uszkoreit *et al.*, "Mlp-mixer: An all-mlp architecture for vision," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[5] R. J. Hyndman and G. Athanasopoulos, *Forecasting: principles and practice*. OTexts, 2018.

[6] T. Bollerslev, "Generalized autoregressive conditional heteroskedasticity," *Journal of econometrics*, vol. 31, no. 3, pp. 307–327, 1986.

[7] R. Aguilar-Rivera, M. Valenzuela-Rendón, and J. Rodríguez-Ortiz, "Genetic algorithms and darwinian approaches in financial applications: A survey," *Expert Systems with Applications*, vol. 42, no. 21, pp. 7684–7697, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417415003954

[8] Q. Ding, S. Wu, H. Sun, J. Guo, and J. Guo, "Hierarchical multi-scale gaussian transformer for stock movement prediction," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, C. Bessiere, Ed. International Joint Conferences on Artificial Intelligence Organization, 7 2020, pp. 4640–4646, special Track on AI in FinTech. [Online]. Available: https://doi.org/10.24963/ijcai.2020/640

[9] Y. Duan, L. Wang, Q. Zhang, and J. Li, "Factorvae: A probabilistic dynamic factor model based on variational autoencoder for predicting cross-sectional stock returns," in *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*. AAAI Press, 2022, pp. 4468–4476. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/20369

[10] J. Gu, F. P. Deek, and G. Wang, "Stock broad-index trend patterns learning via domain knowledge informed generative network," *arXiv preprint arXiv:2302.14164*, 2023.

[11] Y. Qin, D. Song, H. Chen, W. Cheng, G. Jiang, and G. W. Cottrell, "A dual-stage attention-based recurrent neural network for time series prediction," *CoRR*, vol. abs/1704.02971, 2017. [Online]. Available: http://arxiv.org/abs/1704.02971

[12] F. Feng, H. Chen, X. He, J. Ding, M. Sun, and T.-S. Chua, "Enhancing stock movement prediction with adversarial training," 2018. [Online]. Available: https://arxiv.org/abs/1810.09936

[13] H. Wang, S. Li, T. Wang, and J. Zheng, "Hierarchical adaptive temporal-relational modeling for stock trend prediction." in *IJCAI*, 2021, pp. 3691–3698.

[14] Q. Zhou, H. Liu, W. Li, T. Mo, and B. Wu, "Bilateral autotrading framework for stock prediction," in *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2021, pp. 1–8.

[15] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio, "N-beats: Neural basis expansion analysis for interpretable time series forecasting," *arXiv preprint arXiv:1905.10437*, 2019.

[16] T. Swathi, N. Kasiviswanath, and A. A. Rao, "An optimal deep learning-based lstm for stock price prediction using twitter sentiment analysis," *Applied Intelligence*, vol. 52, no. 12, p. 13675–13688, sep 2022. [Online]. Available: https://doi.org/10.1007/s10489-022-03175-2

[17] G. Wang, L. Cao, H. Zhao, Q. Liu, and E. Chen, "Coupling macro-sector-micro financial indicators for learning stock representations with less uncertainty," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 5, 2021, pp. 4418–4426.

[18] I. A. Moosa, "Univariate time series techniques," in *Exchange rate forecasting: Techniques and applications*. Springer, 2000, pp. 62–97.

[19] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," *arXiv preprint arXiv:1606.08415*, 2016.

[20] "Yahoo Finance kernel description," https://finance.yahoo.com/, accessed: 2022-02-10.

[21] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International conference on machine learning*. PMLR, 2013, pp. 1310–1318.

[22] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *arXiv preprint arXiv:1409.1259*, 2014.

[23] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[24] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv preprint arXiv:1803.01271*, 2018.