# Which Data Tells the Truth? A Multimodal Deep Learning Framework for Stock Price Movement Prediction

September 8, 2024

**Abstract**

This paper introduces a novel multimodal data framework for stock price movement prediction, combining stock graphical, market, and text modalities. Our model addresses two key challenges in return prediction about how to extract signals from different types of data and which type of data dominates others. Using a 20-year US stock market dataset, we show that deep learning and language models efficiently capture critical features, with time series data proving more influential than graphical and text modalities. The framework's attention mechanisms and weight allocation effectively reduce conflicts between modalities. Our best-performing model, Fusion(AW), achieves over 70% balanced accuracy and a Sharpe ratio of 4.33 annually, outperforming single- and dual-modal approaches. Out-of-sample tests during recessions demonstrate the model's resilience in volatile markets. This research has broad implications for investment decision-making and paves the way for further exploration of multimodal data in financial modeling.

*Keywords:* Deep Learning, Large Language Models, Attention Mechanism, Return Prediction, Multimodal Data, Machine Learning

## 1 Introduction

Return trend prediction has long intrigued investors. How to identify effective signals in complex market noise has always been a difficult problem. Recently, we have entered in the era of data explosion with abundant data sources, which implies that we can predict market trends without relying solely on past return data. Except for traditional time series data, the application of other types of data, textual data like financial news and image data like

price patterns, has gained increasing attention from researchers in financial forecasting. How to effectively utilize these alternative data has attracted everyone's attention. Recently, with the continuous development of artificial intelligence and quantitative finance, machine learning, especially deep learning, has become increasingly in-depth. Compared with traditional statistical methods, machine learning methods bring unique advantages to handling high-dimensional data in capturing complex non-linear relationships and feature selection (Nagel 2021). Moreover, the "black-box" method has shown its strong ability to analyze alternative data like images and text which is extremely hard to be used with statistical methods in traditional financial forecasting.

For a long time in the past, market return prediction lies in traditional statistical methods, particularly time series models like Autoregressive (AR) and Autoregressive Integrated Moving Average (ARIMA) models. These models have been widely used due to their simplicity and interpretability. Box et al. (2015) provided a comprehensive framework for ARIMA models, which became a cornerstone in time series analysis. The model has shown its huge success in time series prediction in many fields. One of the early applications of ARIMA models in financial markets was documented by Hamilton (2020), who demonstrated their effectiveness in forecasting economic and financial time series. However, despite their widespread use, ARIMA models have limitations, particularly in their ability to capture non-linear relationships inherent in financial data.

In recent years, the non-linear limitations of traditional statistical models in financial data have paved the way for the adoption of machine learning techniques, which can learn patterns from large datasets and handle high-dimensional data. Kim (2003) applied SVMs to predict stock prices and found that they outperformed traditional models like ARIMA in certain contexts. Khaidem et al. (2016) demonstrated the effectiveness of Random Forests in predicting stock market movements, showing that they could outperform traditional models by capturing complex interactions between variables. As financial markets became

more complex, the need for models that could capture intricate patterns and long-term dependencies in the data led to the adoption of deep learning techniques. Neural Networks (NNs) have been explored in finance since the 1980s, with Zhang (2003) providing a comprehensive review of their application in time series forecasting. Gu et al. (2020) show deep neural network can explain more structure in stock returns because of their ability to fit flexible functional forms with many covariates. Caner and Daniele (2023) develop a deep neural network model to get a consistent estimator of the precision matrix of asset returns in large portfolios based on deep learning residuals formed from non-linear factor models, and the model works well even in low signal-to-noise environment.Chen et al. (2024) applied a Long Short Term Memory (LSTM) structure network into generative adversarial network (GAN) to predict stock returns and their model outperforms out-of-sample all benchmark approaches in both statistical and economics indicators. The increasing use of deep learning models reflects the recognition that the complexity in capturing signals of financial markets. This also means that relying solely on traditional market data may not fully identify market signals.

Patterns like Candlestick patterns or K-line charts are usually used in technical analysis. Besides exploring the historical trends, technical analysis can reflect not only the changing balance between supply and demand (Bessembinder and Chan (1998)),but also the sentiment of the investors in the market (Marshall et al. (2006)). Several studies argue for the potential predictability of price movement in technical analysis. Nti et al. (2020) showed that more than 60% of stock market prediction documents they reviewed were based on technical analysis. However, according to Barberis (2018) and Jiang et al. (2023), prices contain subtle and complex patterns that can be difficult to detect with the traditional methods of empirical finance. Udagawa (2018) uses the Candlestick Chart to capture price trend information and portrays it in the form of a single candlestick or a combination of candlesticks with varying lengths. Birogul et al. (2020) encoded charts into 2D charts

and learned the morphological features through deep neural networks. Chen and Tsai (2020) proposed a two-step approach for automatically recognizing eight candlestick patterns, with an average accuracy surpassing that of some deep learning models like LSTM. They also employed reinforcement learning methods to achieve online adaptive control of parameters within unfamiliar environments, ultimately enabling the implementation of high-frequency transaction strategies. Wang and Wang (2019) evaluated the effectiveness of several renowned candlestick patterns, employing recent data from 20 American stocks to estimate stock price movement. Hu et al. (2018) use price plot Convolutional Neural Networks (CNN) 's to cluster individual stocks; Cohen et al. (2020) classify images with some specific technical pattens (crossings of Bollinger Bands, MACD, Relative Strength Index). However, those literatures don't predict returns directly. To test whether the images could be used for returns prediction, Jiang et al. (2023) used CNN to analyze stock-level price charts and identify predictive price patterns for return predictions. They are the first to in performing a large-scale, transparent analysis of return prediction for individual stocks. They employ short time scales (daily data) and long-time scales (monthly data) OHLC (open, high, low, and close) charts as predictor data, which allows the feature extraction from the price patterns. Their empirical results show that image-based CNN predictions are powerful and robust predictors of future returns.

Besides market data and technical analysis, the use of textual data for asset pricing is an emerging topic and remains in its earliest stages because of lack of regular structure and highly complex models. The purpose behind financial sentiment analysis is usually guessing how the markets will react with the information presented in the text. Jegadeesh and Wu (2013) promoted a supervised estimation of sentiment word weights using stock returns. Loughran and McDonald (2016) presents a thorough survey of works on financial text analysis utilizing machine learning with "bag-of-words" approach or lexicon-based methods. In, Pagolu et al. (2016), where n-grams from tweets with financial information

are fed into supervised machine learning algorithms to detect the sentiment regarding the financial entity mentioned. Kraus and Feuerriegel (2017) apply an LSTM neural network to ad-hoc company announcements to predict stock-market movements and show that method to be more accurate than traditional machine learning approaches. Ke et al. (2019) used a supervised learning framework to construct a score for return prediction. They isolated a list of terms via predictive screening first, assigned prediction weights to these words via topic modeling, and aggregated terms into an article-level predictive score via penalized likelihood.

Recently, the birth of language models seems to have opened the door to using news for market returns prediction. Language modeling is the task of predicting the next word in a given piece of text. The realization that a trained language model can be successfully fine-tuned for most downstream tasks with small modifications. These models are usually trained on very large corpora, and then with addition of suitable task-specific layers fine-tuned on the target dataset. The method have shown great success in many industries. In Chen et al. (2022), they used the state-of-the-art large language models (ChatGPT, LLama2, BERT, RoBERTa) in natural language processing to predict returns. They first use tokens to assess the tone of articles for sentiment analysis and then use the output for return prediction. Their work test results from 16 different markets to analyze the sentimental scores and shows LLMs' strong ability to capture and incorporate contextual information, surpassing the limitations of traditional word-based approaches.

Intuitively, according to above works, with more and different types of data, we can extract more diverse and useful information or signals for prediction. The idea has been proven in many other fields like Sentiment Analysis (Zhang et al. 2023, Rahman et al. 2020, Yu et al. 2020), Medical Domain (Zhang et al. 2022, Huang et al. 2021, Wang et al. 2022), Autonomous Driving (Feng et al. 2020, Huang et al. 2022). However, using different modalities or types of data will have some challenges. First, each modality may reveal

different information characteristics. Features extracted from data of different modalities may conflict because of differences in Information Density (eg. Time series data can convey clear return information in a short list, while image data may require detailed analysis to extract information of the same degree.) and differences in Information Expression (eg. Even when expressing the same concept, different modalities might use completely different methods. Text describes directly through words, but images present through visual elements). Besides, not all information in each modality is useful information, and indiscriminate introduction will cause noises when extracting signals. Up till now, the existing literature usually gives equal weights for different types of inputs. The action may introduce noises from different types of data into the model directly and lead to a wrong prediction. If we can find a dominant type of data and use it to guide extracting features from other types of data, noises will be eliminated. Moreover, how to fuse features extracted from different types of data is also a challenge. Finding a method that can combine all useful features but not lose much information is also one of our contributions in this paper. Therefore, we aim to solve the following problems when using different types of data.

(1) How can signals or features be identified independently from different types of data, such as time series, images, or text?

(2) Which type of data tends to contribute most to improved predictions, and can one modality dominate others to help guide other types in extracting meaningful factors? (eg. Time series returns may dominate image data and textual data to get a better prediction.)

(3)What are the most efficient methods for integrating signals from different data types, and how do they handle conflicts? Indiscriminately combining data can lead to reduced model accuracy, as inconsistencies or contradictions between modalities may emerge, harming the overall predictive performance.

Our primary contribution lies in utilizing various state-of-the-art deep learning techniques to extract meaningful signals from diverse data types, each suited to a specific

modality. Specifically, we employ CNNs to effectively extract features from OHLC charts, capitalizing on their ability to capture spatial patterns and structures present in these financial time series images. Additionally, we leverage LSTM networks and traditional Neural Networks for handling time series data, given their strength in identifying temporal dependencies and patterns over time. Furthermore, for textual data, we apply advanced language models such as BERT to derive features based on sentiment, keyword analysis, and context. Through this combination of methods, we are able to identify distinct signals from each data modality, ensuring a comprehensive understanding of various data sources in our prediction models.

Our second contribution is the discovery that time series data holds significant predictive power compared to other types of data when forecasting market returns. By conducting extensive experiments, we find that the temporal dynamics and sequential dependencies embedded in time series data provide more accurate signals for return predictions. This insight highlights the importance of time-based data, as financial markets are inherently time-sensitive. Despite leveraging sophisticated techniques to extract features from OHLC charts and textual data, we observe that these features often serve as complementary inputs. They enhance the performance of time series models but rarely surpass the dominance of pure time series features in predictive accuracy. This finding suggests that while other data types provide valuable context, time series data should be prioritized in return prediction frameworks.

Our third contribution introduces an innovative method that leverages the attention mechanism to integrate features from different data modalities, enhancing the overall prediction accuracy. Traditionally, the simplest approach for multi-modal feature integration was to assign equal weights to all features from different data sources and concatenate them for prediction. However, this method often overlooks the varying importance of individual data types. Our proposed method uses an attention mechanism to dynamically assign

weights based on the relevance of features. This allows the model to focus on the most informative and non-redundant aspects of each data modality. Specifically, we calculate attention scores between features generated from different data types—such as time series, OHLC charts, and textual data—to identify and filter out conflicting or redundant information. After this filtering process, we concatenate the most relevant features and use them for final return predictions. This approach ensures a more tailored and context-sensitive integration of data, leading to improved predictive performance.

Our model was applied to analyze 20 years of U.S. equity returns using three diverse datasets: market data, OHLC (Open, High, Low, Close) charts, and financial news. By integrating these different data sources, we aimed to capture a comprehensive range of market signals. The results demonstrate that our model significantly outperforms traditional benchmarks, including single deep learning models such as CNN, NN, and LSTM, as well as advanced language models applied to textual data. This superiority is evident across multiple out-of-sample evaluations, showcasing the robustness and reliability of our approach. Statistically, our model achieved remarkable performance with an F1 score of 87.47% and an accuracy rate of 70.53%, surpassing the results of individual deep learning and language models. Economically, the model also delivered exceptional outcomes, producing the highest annualized Sharpe ratios: 4.23 for equal-weighted long-short portfolios and 2.03 for long-only portfolios, indicating strong risk-adjusted returns.

An important finding from our analysis is that time series data, such as market returns, dominate other types of alternative data like OHLC charts and financial news. When we excluded time series data, the model's accuracy dropped sharply from 70.53% to 55.32%, highlighting the crucial role that temporal information plays in financial return predictions. Furthermore, the introduction of an attention mechanism to weight the importance of different data types led to significant improvements. By using attention to focus on the most relevant features, we achieved nearly a 10% increase in accuracy and over a 10% improve-

ment in the Sharpe ratio. This dynamic feature weighting ensures the model prioritizes key signals while filtering out less important or redundant data, further enhancing its predictive power. Finally, we tested the efficiency and robustness of our model during periods of economic recession. Even under these challenging conditions, the model maintained superior predictive performance, outperforming individual deep learning models and language models. This resilience demonstrates the model's ability to adapt and perform in both stable and volatile market environments, confirming its broad applicability and reliability in financial forecasting.

The rest of the paper is organized as follows. Section 2 introduces the approaches we employ for different types of data and the method of weights allocation for different modalities. Section 3 shows our model structure. Section 4 presents an empirical analysis of return prediction in US market. Section 5 concludes.

# 2 Methodology

In this section, we introduce machine learning and deep learning methods for different types of data: Neural Networks and Recurrent Neural Networks for time series data, Convolutional Neural Network for patterns and language models for textual data.

## 2.1 Methods of time series data

### 2.1.1 Neural networks (NNs)

Neural networks are considered the most potent tool in machine learning, hailed as "universal approximators" that can model any smooth relationship (Hornik et al. 1989, Cybenko 1989). They are favored for tackling complex challenges in computer vision, natural language processing, and automated games like chess and go. Known as "deep learning," their strength lies in their layered architecture of nonlinear interactions. However, their elaborate

structure also makes them some of the least transparent and most parameter-heavy tools in machine learning. In our model, we first use the fundamental and traditional neural network, feed-forward networks (FFNs). FFNs consist of an input layer of raw predictors, one or more hidden layers that interact and nonlinearly transform the predictors, and an output layer that aggregates hidden layers into an outcome prediction. Analogous to axons in a biological brain, layers of the networks represent groups of neurons with each layer connected by "synapses" that transmit signals among neurons of different layers. The method has been proven efficient in asset pricing theoretically and empirically. (Gu et al. 2020, Caner and Daniele 2023) The structure of FFN works as follows: the output $y_t$ of classic NN or FFN with L hidden layers can be written using the following recursive formula:

$$x_t^l = g\left(W^{l-1}x_t^{l-1} + b^{l-1}\right) \tag{1}$$

$$y_t = W^L x_t^L + b^L \tag{2}$$

where $l = 1, \ldots, L$ denotes the number of layers. $W^{l-1}$ is weight parameter and $b^{l-1}$ is the bias parameter. $g(\cdot)$ is the activation function. During the training process, each layer of a feed-forward network can learn to recognize various features of the input data. We refer Gu et al. (2020) to use RELU function as $g(\cdot)$. Figure 1 shows the feedforward neural networks (FFNs) structure with one hidden layer.

### 2.1.2 Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM)

FFNs generally assume that input data are independent and identically distributed. However, stock returns are usually time series data and temporal dependency needs to be considered. RNNs are specifically designed to handle time series data and LSTMs are the most common and widely used type of RNNs. LSTMs have a memory mechanism that allows them to remember and carry forward significant information through long sequences,

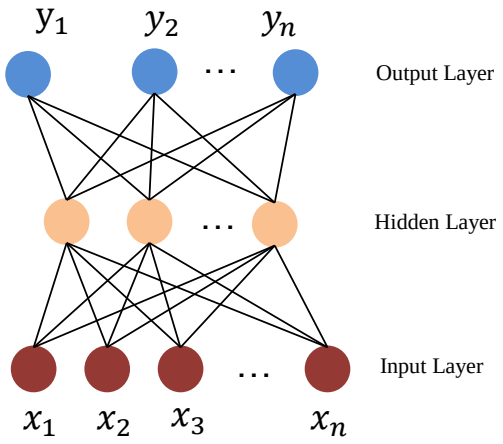


Figure 1: Feed Forward Network with one hidden layer

making them highly effective for tasks involving sequential data such as natural language processing, time-series analysis, and more. One key advantage of LSTMs over FFNs is their ability to capture temporal dependencies and patterns over time. FFNs, with their static architecture, lack the mechanism to process sequences in the context of their order, making them unsuitable for tasks where the sequence and timing of inputs significantly influence the output. LSTMs, on the other hand, can maintain a state or context that evolves, allowing them to remember past inputs and use this information to influence future predictions. This capability enables LSTMs to perform exceptionally well on tasks like language modeling, speech recognition, and sequence prediction, where understanding the temporal dynamics is crucial. It also shows success in asset pricing (Chen et al. 2024, Ghosh et al. 2022, Kim and Won 2018).

The structure of LSTM can be described as follows. In LSTM, it has three gates: the input gate $\Gamma_i$, the forget gate $\Gamma_f$, and the output gate $\Gamma_o$ and one cell state $c_t$, which is used to control the long-term memory.

The forget gate $\Gamma_f$ controls which information from the previous cell state should be kept or discarded. Essentially, it decides what portion of the past information is relevant

enough to be carried forward to the next time step.

$$\Gamma_f = \sigma\left(W_f x_t + U_f h_{t-1} + b_f\right) \tag{3}$$

Where $x_t$ is the current input of market data, $h_{t-1}$ is the previous hidden state which serves as a kind of memory that captures and encodes important aspects of the sequence data up to the previous time step. $W_f$ and $U_f$ represent the weights associated with the current input and previous state, respectively. $\sigma(\cdot)$ is the sigmoid function that outputs a value between 0 and 1. If the output of the forget gate is close to 1, it means the corresponding information is important and should be kept. If it's close to 0, the information is considered less important and can be forgotten.

Similarly, an input gate is used to decide which new information to store in the cell state and the new information $\widetilde{c_t}$ is a function of a hidden state at the previous timestamp $t-1$ and input at timestamp $t$.

$$\Gamma_i = \sigma\left(W_i x_t + U_i h_{t-1} + b_i\right) \tag{4}$$

$$\widetilde{c_t} = \tanh\left(W_c x_t + U_c h_{t-1} + b_c\right) \tag{5}$$

Where $x_t$ is still the current input, $h_{t-1}$ is the previous hidden state. $W_i$ and $U_i$ represent the weights associated with the current input and previous state, respectively. $\sigma(\cdot)$ is the sigmoid function. $\widetilde{c_t}$ represents the potential new content that could be added to the cell state.

After that, we can update the cell date $c_t$

$$c_t = \Gamma_f \times c_{t-1} + \Gamma_i \times \widetilde{c_t} \tag{6}$$

Finally, the LSTM needs to decide what to output. This output will be based on the cell state, but will be a filtered version. First, a sigmoid layer decides which parts of the cell state $c_t$ to output. Then, the cell state is put through tanh (to push the values to be

between $-1$ and 1) and multiplied by the output of the sigmoid gate, so that only the parts we decided to output are actually output.

$$\Gamma_o = \sigma\left(W_o x_t + U_o h_{t-1} + b_o\right) \tag{7}$$

$$h_t = \Gamma_o \times \tanh\left(c_t\right) \tag{8}$$

The LSTM's ability to add or remove information to the cell state, carefully regulated by the gates, allows it to handle temporal dependencies in the input data. Figure 2 show the LSTM structure.

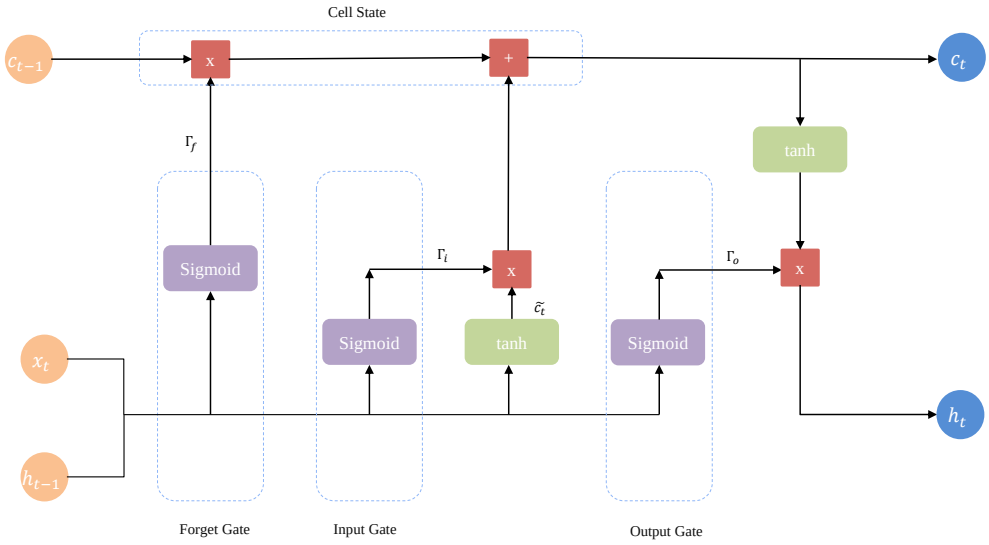


Figure 2: LSTM structure

## 2.2 Methods of Image Data

### 2.2.1 Convolutional Neural Network (CNN)

In this section, we show how to use the CNN to deal with image datasets. CNN is a class of deep neural networks whose applications focus on image analytics. In detail, most CNNs are made of several convolution layers and pooling layers. Normally, convolution layers are responsible for extracting high-level features in the images through padding operation. This

will pack information in a small field into a single feature rather than a grid of features. The pooling layer can help decrease the computational power required to process through the pooling operation, which divides the entire graph into several sub-segments and an extra feature for each segment. In this way, dimensionality reduction can be achieved. Meanwhile, the pooling layer can also help extract dominant features. The details of each component in CNN are introduced as follows. The convolution layer contains several convolution kernels with each cell of the convolution kernel corresponding to a bias vector and a weight coefficient. While the convolution kernel is working, the input features will be scanned regularly, and the input features will be summed and multiplied by the matrix elements in the receptive field. The output of the convolution layer will be forwarded to the pooling layer for filtering and feature selection. This layer employs predefined pooling functions to summarize a feature map's neighborhood into a single statistic, effectively reducing data dimensions. The selection of pooling regions mirrors the convolution process, dictated by parameters such as pooling size, stride, and padding. Figure 3 describes a CNN structure with only one convolution and pooling layer.

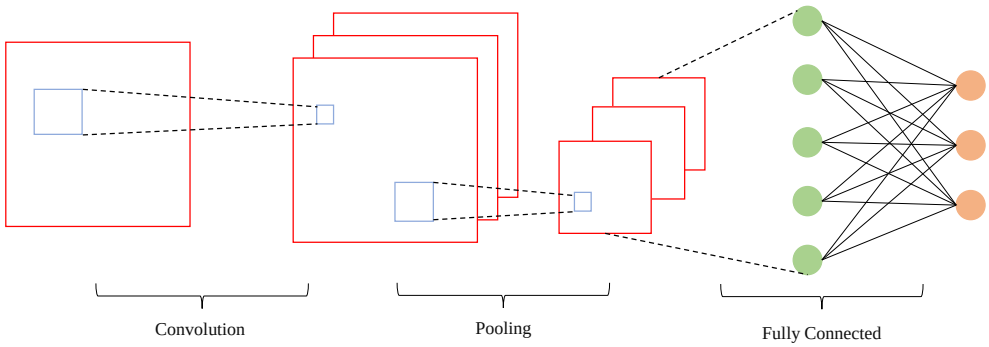


Figure 3: CNN with one convolution and pooling layer

The mathematical process to extract features by CNN is similar to the FFN. The main difference is that the input of CNN is an image not a list of data. The selected feature $y_t$ of one CNN block (one convolution layer and one pooling layer) with input image $x$ can be

written using the following formulas:

$$x_{conv} = g_{conv}\left(W_{conv}x + b_{conv}\right) \tag{9}$$

$$x_{pooling} = pooling\left(x_{conv}\right) \tag{10}$$

$$y_{output} = g_{fcon}\left(W_{fcon}x_{pooling} + b_{fcon}\right) \tag{11}$$

Where $x$ is the image dataset, $x_{conv}$ is the output of the convolution layer, $x_{pooling}$ is the output of the pooling layer, $W_{conv}$ and $W_{fcon}$ are weight parameters and $b_{conv}$,$b_{fcon}$ are the bias parameters. $g_{conv}$ is the activation function, which is the LeakReLu function, to compute the convolution (Ke et al. 2019).

### 2.2.2 Why we not use 1D CNN

It's important to clarify the concepts and differences between 1D and 2D CNNs, as these describe how convolutional filters in a CNN traverse the data.

1D CNNs are typically employed for sequential data, such as time series, signals, or sequences of words in natural language processing. In a 1D CNN, the convolution operation moves along a single dimension, capturing patterns over time or within a sequence. Here, the data are structured as a matrix, with time corresponding to rows and variables to columns, making 1D CNNs particularly effective for tasks like sentiment analysis, speech recognition, or any application involving data with a clear sequential or temporal structure. Some scholars have used 1D CNNs to predict the stock market and demonstrated its effectiveness (Hoseinzade and Haratizadeh 2019, Selvin et al. 2017).

In contrast, 2D CNNs are designed to handle spatial data, such as images or videos. In these networks, the convolution operation moves across two dimensions—usually height and width in the case of images. This allows the network to capture spatial hierarchies and features like edges, textures, and patterns, which are essential for tasks such as image

classification, object detection, and facial recognition. In our research, we propose embedding time series data as images before applying predictive analysis. According to Ke et al. (2019), representing the data as an image enables convolutional filters to capture non-linear spatial relationships among various price curves, offering a potential advantage in predictive accuracy than using the data as a sequence.

## 2.3 Methods of Textual Data

### 2.3.1 Large Language Models (LLM)

In this section, we show how to use Large Language Models (LLM) in textual analysis. According to Chen et al. (2022) and Ke et al. (2019), textual analyses are usually processed in two steps. The first step is a text representation step which decides on the numerical representation of the text data and the second step is to input the output from the first step into that an econometric model to calculate some variables (eg. Return, volatility). The output of step 1 is a $D \times$P numerical matrix, where $D$ is the number of documents in the text corpus, and $P$ is the number of topics. In the past, the first step is usually finished with "Bag of Words", which collapses each document observation into a high dimensional vector of counts spanning all unique terms in the full corpus of documents. However, this method is overly simplistic and only accesses the information in text that is conveyable by term usage frequency. It sacrifices nearly all information that is conveyed through word ordering or contextual relationships between terms. Besides, the number of parameters and the corpus-specific dimension-reduced representations are also challenges faced with traditional methods.

Currently, state-of-the-art LLMs have been dominating performance benchmarks across various NLP tasks, primarily due to their expansive scale. They are often pre-trained on enterprise-level platforms, some of which have made their pre-trained models publicly

available. LLMs delegate Step one to the handful of people in the world who can best execute it. The output of LLMs is a numerical vector representation (or "embedding") of a document. The main benefit of an LLM in Step one is that it provides more sophisticated and well-trained text representations. Step two will use the output of LLMs with little or no modifications.

### 2.3.2 Tokenization

The first part to use LLMs for text analysis is tokenization. Tokenization is the process of breaking up a piece of text into smaller components, called tokens. These tokens can be words, phrases, symbols, or other meaningful elements. The purpose of tokenization is to simplify and prepare text for further processing. In LLMs, tokens are often subwords, a method derived from word-based tokenization that splits text into words using specific delimiters. Subword tokenization breaks down rare words into smaller, meaningful units, reducing data sparsity by increasing token frequency and keeping vocabulary sizes manageable. This approach is especially useful in languages with rich morphology, where diverse word forms abound.

Here is an example from a piece of news of Tesla: "Tesla plans to increase production of its Model 3 vehicles to meet high demand by the end of 2017." A tokenizer can break down the sentence into a sequence of ordered tokens. The results will be [‘Tesla’, ‘plans’, ‘to’, ‘increase’, ‘production’, ‘of’, ‘its’, ‘Model’, ‘3’, ‘vehicles’, ‘meet’, ‘high’, ‘demand’, ‘by’, ‘the’, ‘end’, ‘of’, ‘2017’, ‘.’] This tokenization breaks down complex words and retains important elements like splitting "Amazon.com" to handle the domain separately. LLM's ability to break words into subwords allows it to handle a wide variety of words it might not have seen during training, making it robust for processing diverse texts.

### 2.3.3 Transformer Encoder Structure

The next step is to transfer tokens into numerical representation. Each word is converted into a unique numeric ID within the tokenizer. These numeric IDs are subsequently transformed into embedding vectors by the embedding layer. This layer essentially functions as a lookup table, mapping each ID to a predefined vector. These embedding vectors are then fed into several Transformer layers of BERT. Within these layers, the vectors undergo a series of complex transformations, enabling each vector to accumulate and integrate information from the entire input sequence. After processing, the output for each input token is a vector, typically 768 dimensions for the BERT-Base model. These vectors contain rich semantic and contextual information and can be used for a variety of downstream tasks like prediction.

The core element of BERT is the Transformer Encoder. Transformer was introduced as an alternative to recurrent neural networks for modeling sequential information. Originally part of a sequence-to-sequence model with both encoder and decoder components, BERT utilizes only the encoder. This encoder comprises multiple identical layers, each with a multi-headed self-attention mechanism and a fully connected feed-forward network. In self-attention, embeddings generate three mappings—key, query, and value. A dot product between a token's key and all queries computes similarity scores, which then modify the value vectors to reformulate the token's representation. This multi-perspective approach allows the model to analyze sequences comprehensively. The output passes through feed-forward networks with shared parameters, enhancing processing efficiency. Vaswani et al. (2017) highlighted that unlike RNNs, Transformers avoid sequential computation, facilitating parallelization on GPUs and better handling long-range dependencies within sequences, overcoming RNNs' limitations in sustaining information across many steps. BERT marked a pivotal shift in language models within NLP by introducing a deeply bidirectional system,

enhancing text understanding by analyzing both preceding and succeeding contexts. This innovation spurred further research, leading to the development of advanced models like ChaGPT-4, revolutionizing how models interpret and generate language. RoBERTa and FinBERT are both BERT variants, enhancing their performance through training adjustments.

### 2.3.4 Pre-training and Fine tuning in LLM

The training phase in transfer learning, often termed pre-training, involves learning a large number of model parameters from vast, diverse datasets like Wikipedia and Common Crawl. This allows the model to grasp language syntax, semantics, and context. For example, BERT's pre-training includes two unsupervised tasks: Masked Language Model (MLM) and Next Sentence Prediction (NSP). In MLM, 15% of tokens are masked, and the model predicts them, enabling it to understand context from both directions. NSP determines if two sentences are consecutive. RoBERTa improves BERT by removing NSP, increasing the training data and batch sizes, and introducing dynamic masking for more robust training.

The fine-tuning stage follows pre-training, where the model is adapted to specific tasks like text classification or question answering using supervised learning on smaller, labeled datasets. This stage is faster since the model already has general language understanding from pre-training. Pre-training offers a broad foundation, while fine-tuning hones the model's skills for a specific task.

Inspired by Peters et al. (2018), we use a feature extraction approach by inputting new text into the pre-trained model, generating token representations as vectors that capture contextual meaning. These vectors are then used for downstream tasks without updating pre-trained parameters. This method reduces computational efforts, allowing efficient use of pre-trained models while maintaining their language understanding for specific applications.

### 2.3.5 Choice of LLMs

Our work builds upon the approach of Chen et al. (2022) by incorporating three prominent large language models (LLMs): Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al. 2018), Robustly Optimized BERT Pre-training Approach (RoBERTa) (Liu et al. 2019), and BERT for Financial Text Mining (FinBERT) (Araci 2019), specifically chosen for their proven effectiveness in financial text analysis and prediction tasks.

While models like ChatGPT and LLaMA2 utilize "next token prediction" to predict the next word in a sequence, such as predicting "rate" after "The Fed raised interest," this helps the model learn contextual language understanding. However, using general-purpose LLMs like ChatGPT, LLaMA, or LLaMA2 presents specific challenges when applied to investment-related tasks.

Firstly, ChatGPT, LLaMA, and LLaMA2 are primarily designed for conversational tasks and general language understanding, not for the specific demands of financial forecasting. These models lack built-in mechanisms to handle key financial variables like expected returns, firm fundamentals, or macroeconomic trends, which are critical for investment strategies. Secondly, these models may struggle with generating consistent, comparable sentiment scores across a wide range of financial news articles, making it difficult to extract actionable insights for trading purposes.

Most importantly, ChatGPT, LLaMA, and LLaMA2 do not provide any form of measurement for expected returns or quantitative market predictions. These models lack the specific financial training necessary to predict stock movements or integrate financial indicators with precision. While they offer strong general language capabilities, they are not optimized for tasks requiring deep financial expertise.

In contrast, BERT-based models like FinBERT are designed for text mining and are

more suited to financial-specific tasks. By focusing on these models, we ensure that our approach is tailored to handle the complexities of financial data while leveraging state-of-the-art language modeling techniques for better prediction outcomes.

## 2.4 Methods of weights allocation

In predicting financial markets, the integration of multimodal data—combining diverse types of information such as numerical, textual, and visual data—has been a challenge. It is hard to decide optimal weights for each type of data. Therefore, equal weights, the most straightforward approaches to combining multiple modalities of different modalities, are used in previous researches (Lee and Yoo 2020, Zhang et al. 2024). However, this method is not accurate because we cannot just give equal weights to each type of data. With advanced machine learning methods have been used in recent years, it is possible for us to find weights for different types of data at each time stamp. We consider to induce attention mechanism, which has prove its efficiency in other industries.

### 2.4.1 Attention Mechanism

Attention mechanism is a computational mechanism that dynamically assigns varying degrees of significance to different part, enabling the model to capture mutual relationships between features from different types of data for enhanced understanding. It does this by calculating attention scores that determine how much focus to place on other parts of the input when generating an output. The attention structure are shown as follows: for each input embedding, there are three vectors: the Query vector ($Q$) which determines how to attend to other embeddings, the Key vector ($K$) which represents each embedding in a way that can be matched by queries, and the Value vector ($V$) which contains the actual content of the embeddings that will be used to compute the output. The three vectors can

be obtained by multiplying three different weight matrices $W_Q$,$W_K$, and $W_V$.

$$Q_i = F_i W_Q,\ K_i = F_i W_K,\ V_i = F_i W_V \tag{12}$$

Where $i$ is image, time or text, $F_i$ is the input features from different types of data.

To obtain the relevance between features, attention score could be obtained by the following process. First, a dot product $Q_i \times K_j$ could be calculated and this matrix represents how much each element of the sequence should attend to every other element. Then the matrix are scaled down by the square root of the dimension of the key vectors $d_k$ to avoid very large values of the dot product growing disproportionately. The employment of the softmax function converts the scores into a probability distribution, where higher scores increase the weight of the corresponding value in the output, and ensure that the weights sum to one. The final step in the computation of the attention output is to multiply the attention weights by the values. This step produces the output of the attention layer, where each output element is a weighted sum of the values, with weights specified by the attention scores.

$$\widetilde{F_i} = \ Attention\,(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{13}$$

# 3 Model Description

## 3.1 Model Structure

Our network model is composed of two parts: feature extraction and weights allocation. In feature extraction part, We denote three different types of datasets $d_i$ ($d_{image}$, $d_{time}$, and $d_{text}$). For different datasets, the feature extraction process will be

$$F_i = extractor_i\,(d_i,\ \theta_i) \tag{14}$$

Where $F_i$ is the extracted features from different datasets. $\theta_i$ includes all parameters from $i$ dataset.

Specially, for image dataset, the Convolutional Neural Network (CNN) are employed for feature selection. The process is as follows:

$$d_{con\_image} = g_1 \left( W_1 d_{image} + b_1 \right) \tag{15}$$

$$\widetilde{d}_{image} = pooling \left( d_{con\_image} \right) \tag{16}$$

$$F_{image} = g_2 \left( W_2 \widetilde{d}_{image} + b_2 \right) \tag{17}$$

where $F_{image}$ represents final extracted features from CNN model, $d_{image}$ represents the image dataset, $d_{con\_image}$ is the output of the convolution layer, $\widetilde{d}_{image}$ is the output of the pooling layer. $W_1$ and $W_2$ are weights parameters and $b_1$ and $b_2$ are bias parameters in CNN. $g_1$ and $g_2$ are activation functions.

For time series market dataset, the feature selection process is as follows:

$$\widetilde{d}_{time} = g_3 \left( W_3 d_{time} + b_3 \right) \tag{18}$$

$$F_{time} = g_4 \left( W_4 \widetilde{d}_{time} + b_4 \right) \tag{19}$$

where $F_{time}$ represents final output of extracted features with NN or LSTM model from time series dataset, $d_{time}$ represents the original dataset, $\widetilde{d}_{time}$ is the output of the intermediate layer. $W_3$ and $W_4$ are weights parameters and $b_3$ and $b_4$ are bias parameters in Neural Networks.

For text data, some state-of-art language models (BERT, RoBERTa, FinBERT) are used for feature extraction process.

$$F_{text} = LLM \left( d_{text} \right) \tag{20}$$

where $F_{text}$ represents final output of extracted features with language models, $d_{text}$ represents the financial news dataset.

Through different extractors, we can get unique features from different datasets. Then we use extracted features to allocate weights to combine our model. In weights allocation

part, we consider two different methods: the first will be the most straightforward method, which gives equal weights to all three types of data.

$$F_{concat} = F_{time} + F_{image} + F_{text} \tag{21}$$

The second method will use attention mechanism for weights allocation, to calculate attention score between different type of dataset, especially the score $\alpha$ between image data and time series data and the attention score $\beta$ between text data and time series data. Figure 4 shows the attention layer structure. Through this method, we can use the dominate time series data to guide other type of data and filter noises created by different modalities.

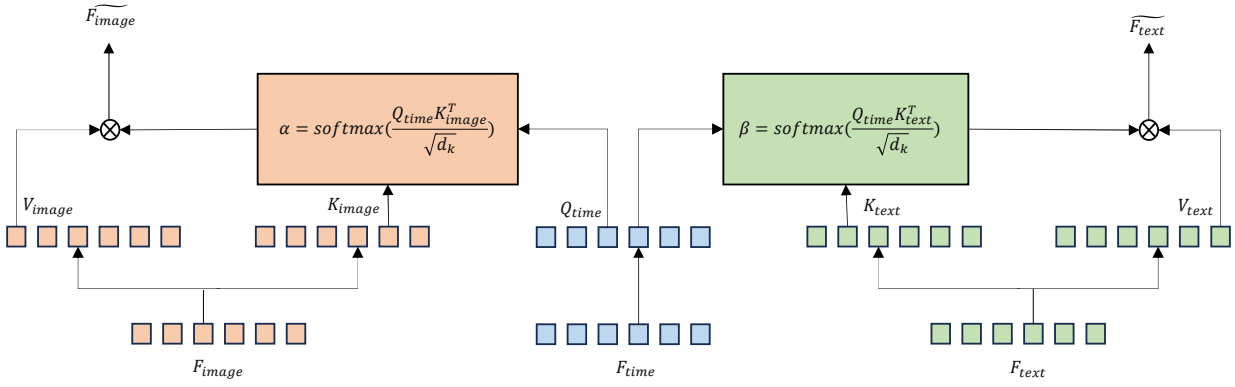


Figure 4: Attention Layer structure

**Note:** $\alpha$ represents the similarity matrix between graphic features and time series numerical features, while $\beta$ represents the similarity matrix between time series numerical features and textual features.

$$\alpha = softmax\left(\frac{Q_{time}K_{image}^T}{\sqrt{d_k}}\right) \tag{22}$$

$$\beta = softmax\left(\frac{Q_{time}K_{text}^T}{\sqrt{d_k}}\right) \tag{23}$$

The new features extracted from image data via the score $\alpha$ is:

$$\widetilde{F_{image}} = \alpha V_{image} \tag{24}$$

The new features extracted from text data via the score $\beta$

$$\widetilde{F_{text}} = \beta V_{text} \tag{25}$$

Then we concatenate three features.

$$F_{concat} = F_{time} + \widetilde{F_{image}} + \widetilde{F_{text}} \tag{26}$$

Figure 5 shows our model structure.

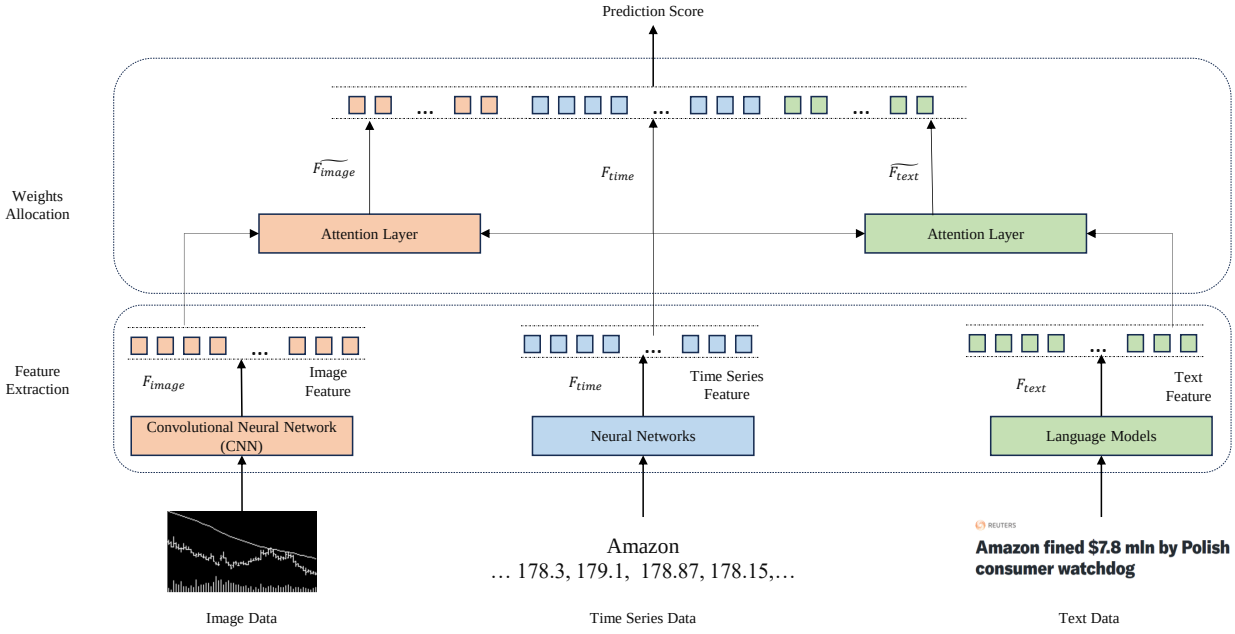


Figure 5: Multimodality Model structure

## 3.2 Estimation

For classicfication problem, our aim is to optimize the following function:

$$L(y, \hat{y}) = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y}) \tag{27}$$

where $\hat{y}$ is the output from the last step of the model.If the predicted probability exactly corresponds with the label, $\hat{y} = y$, then the loss function is zero, otherwise the loss is positive.

To optimize the objective function, gradient descent (GD), which minimizes a function by iteratively moving in the direction of the steepest descent, is the common method. However, GD requires the entire dataset to compute the gradient at each step, it can be very slow and impractical for large datasets. To solve above problem, we apply stochastic gradient descent (SGD), which improves upon the idea of GD by using only one or a few training examples at a time to calculate the gradient. This means SGD doesn't require the entire dataset to update the model's parameters. We utilize the adaptive moment estimation (Adam) algorithm Kingma and Ba (2014) for gradient evaluation, applying it to minibatches of size $M$ from the full data in each iteration. The Adam algorithm combines ideas from two other extensions of SGD to compute adaptive learning rates for each parameter. It avoids fluctuation when using standard SGD.

# 4 Empirical Analysis

## 4.1 Data Collection and Processing

In our dataset, we utilized real-world datasets encompassing financial news, market data, and OHLC charts spanning from January 2000 to June 2019, encompassing a about 20-year period. For return data, we obtain stock daily return data from CRSP for all firms listed on NYSE, AMEX, and NASDAQ for the corresponding dates. For image data, we refer Jiang et al. (2023) to generate OHLC images (Open, High, Low, Close and Volume) by collecting data from CRSP within the same time period. For financial news dataset, we refer Chen et al. (2022) and Ke et al. (2019) to use news from London Stock Exchange (LSEG) Data & Analytics, formerly Refinitiv, whose dataset encompasses global news from Thomson Reuters Real-time News Feed (RTRS), with each article containing a title and publication date.

In preparing the news database for analysis, we first employ the publication date to

align the articles with the daily market data. The number of news titles per trading day varied; hence, we aggregated all the titles for a given day for one company into a single extended sentence and employed Language Models to encode the textual data into feature vectors. Consequently, we obtained a single sentence embedding vector for each trading day. Besides, we have implemented following filters. First, in order to align the time dimension with the images and market transaction data, we only retain news about individual companies that appeared on the trading day. This means news that occurs on non-trading days (such as weekends) will not be counted. Next, we refer Chen et al. (2022) to take measures to remove redundant articles that essentially replicate the content of preceding stories. Redundancy has been assessed through the computation of a novelty score, derived from cosine similarity calculations based on the bag-of-words representations of any pair of articles. An article is classified as redundant if it attains a cosine similarity score of 0.8 or higher when compared with another article published within the preceding five business days. This process ensures the diversity of the dataset and also safeguards the novelty of the content, resulting in a substantial reduction of superfluous repetition. It is important to note that the removal of such repetition also enhances the signal-to-noise ratio, which is critical as we utilize firm returns as labels in our tasks.

For image data, our price trend analysis focuses on returns adjusted for corporate actions by using return to construct a price series. In each image, we normalize the first day closing price to one, and construct each subsequent daily close from returns. Each day's opening, high, low price levels are scaled in proportion to that day's closing price level.

## 4.2 Model Training

For each dataset, we initially employ an 60 - 20 - 20% split for training, validation and testing for prediction. Following prior works, we used several evaluation metrics like classification accuracy and F1 Score to evaluate our model.

First, we divide the entire sample into training, validation, and testing samples. In this twenty year sample, we randomly select 60% dataset for training and 20% for validation. Randomly selecting the training and validation sample helps balance positive and negative labels in our classification problem, which attenuates a potential bias in classification due to extended periods of bullish or bearish market swings. The resulting training and validation dataset have approximately 50% up and 50% down labels in all scenarios we consider. The remaining four years of data comprise the out-of-sample test data set. We treat the prediction analysis as a classification problem. In particular, the label is defined as 1 if the subsequent return is positive and 0 otherwise. The training step minimizes the standard objective function for classification problems, a cross-entropy loss.

## 4.3 Model comparison

In our empirical analysis, we compare eight models: CNN, NN, LSTM, BERT, RoBERTa, FinBERT, Fusion(EW) and Fusion (AW).The first three are deep learning models. CNN is the deep learning model to deal with graphical data and it has been proven efficient in financial prediction by Jiang et al. (2023). NN is the traditional neural network, which has been commonly used in stock prediction (Gu et al. 2020, Kraus and Feuerriegel 2017). LSTM structure is designed for identifying temporal dependency among time series data. BERT, RoBERTa and FinBERT are large language model. BERT (Devlin et al. 2018) and RoBERTa (Liu et al. 2019) have shown its success in stock prediction in Chen et al. (2022), while FinBERT, developed by Araci (2019), is a pre-trained language model specifically designed for financial sentiment analysis. Fusion(EW) and Fusion (AW) are our models combining features from all three different types of datasets. Specially, both fusion models use graphical features from CNN, numerical features from NN, and textual features from RoBERTa. The difference between Fusion(EW) and Fusion (AW) lies in the way features are assigned. Fusion(EW) divides the three features equally, while Fusion (AW) uses the

attention mechanism to adjust the weights.

## 4.4 Empirical Result: Classification Result

Our model aims to predict a binary outcome: 1 indicating a positive return and 0 signifying otherwise. The fitted value of the logistic regression is an estimate for the probability of a positive outcome. A true positive (TP) or true negative (TN) occurs when a predicted probability of greater than 50% coincides with a positive realized return and a probability less than 50% coincides with a negative return. The threshold of 50% is used as a natural cutoff for positive sentiment score. False positives and negatives (FP and FN) are the complementary outcomes. We calculate classification accuracy as follows:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{28}$$

Besides, we apply F1 score when considering the imbalanced situation where one class is much more frequent than the other. In these cases, accuracy can be misleading because a model could achieve high accuracy by simply predicting the majority class, ignoring the minority class altogether. The process is as follows:

$$\text{F1 Score} = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{29}$$

Table 1 shows results of Accuracy, F1 Score and function loss for all Models. We have following conculsions: First, All models can identify valid market signals from their specific data sets. They consistently outperform a random guess (50%) in terms of average accuracy over these years. This implies all deep learning and language models can identify useful signals from noisy market data for prediction. Second, our three language models (BERT, RoBERTa, FinBERT) exhibit higher overall accuracy and F1 score than traditional deep learning models (CNN, NN, LSTM) in the classicfication problem, with CNN is the only model which accuracy is lower than 60% (57.80%). It is notable that NN model can achieve

64.56% average accuracy, which is nearly equal to the result of BERT. Compared with language models, NN model requires significantly less computing power than the language model, which means that the NN model is a more feasible choice when computing power is limited. In our three language models, RoBERTa achieves the highest accuracy of 67.38%, with FinBERT and BERT achieving 66.23% and 65.40%, respectively. Third, fusion models with Attention mechanism (Fusion (AW)) achieve the highest results with 87.47% F1 score and 70.53% Accuracy, While Fusion models with equally weights (Fusion (EW)) just shows a result with 65.84%. The result is just slightly higher than BERT's result, but lower than the other two language models (RoBERTa and FinBERT). By comparison, this shows that each dataset of different modalities mines unique signal features, and the conflict of multi-modal features can be effectively removed through the Attention mechanism, thereby improving the accuracy of model prediction.

Table 1: F1 Score and Accuracy for All Models

| **Model** | **F1 Score** | **Accuracy** | **Loss** |
|---|---|---|---|
| CNN | 78.70% | 57.80% | 0.45 |
| NN | 83.25% | 64.56% | 0.39 |
| LSTM | 81.73% | 62.62% | 0.41 |
| BERT | 83.60% | 65.40% | 0.36 |
| RoBERTa | 84.77% | 67.38% | 0.31 |
| FinBERT | 84.26% | 66.23% | 0.33 |
| Fusion (EW) | 83.68% | 65.84% | 0.35 |
| Fusion (AW) | 87.47% | 70.53% | 0.28 |

## 4.5 Which data tell the truth

To better understand the influence of each modality in the our model, Table 2 reports the ablation results of the subtraction of each modality to the Fusion(AW) on the dataset. We note that after removing the image inputs, the performance of Fusion model remains relatively high, with 85.96% F1 score and 68.58% average accuracy. Its performance is even better than Fusion(EW) and all three language models. This implies that the features extracted from images account for a small proportion in our model, and are likely to conflict with features extracted from other modal data. The worst case is to eliminate the numerical market data. The accuracy in this situation is just a little bit higher than 50%. This proves the market data's importance in prediction and conflicts between features from image and financial news datasets. In addition, after removing financial news dataset, the fusion model still shows a higher F1 score, but the low accuracy. This may imply financial news could offer valid market signals. Therefore, we argue that eliminating the irrelevant information that appears in auxiliary modalities (like image) and improving the contribution of auxiliary modalities in performance should be paid more attention to.

Table 2: Comparison of Accuracy and F1 for Different modality

| **Method** | **Result** | | |
|---|---|---|---|
| | **F1** | **Accuracy** | **Loss** |
| **Fusion(AW)** | **87.47** | **70.53** | **0.28** |
| w/o image | 85.96 | 68.58 | 0.30 |
| w/o time series | 73.52 | 55.32 | 0.58 |
| w/o financial news | 82.56 | 63.37 | 0.43 |

## 4.6 Empirical Result:Performance of Portfolios

We follow Chen et al. (2022) to use a similar method to construct portfolios. Our approach is straightforward: we construct a zero-net-investment portfolio by taking long positions in the top quintile (10%) of stocks with the most positive scores and short positions in the bottom quintile (10%) of stocks with the most negative scores.

When forming the long and short sides of the strategy, we consider both equal-weighted and value-weighted schemes. Equal weighting provides a straightforward and robust method to assess the predictive power of sentiment across the spectrum of firm sizes and aligns with common practices in hedge funds' news text-based portfolio construction. On the other hand, value weighting places greater emphasis on large-cap stocks, which can be justified for economic reasons (assigning more weight to more productive firms) and practical trade implementation reasons (such as managing transaction costs).

We form portfolios solely at the market open each day, primarily for two reasons. Firstly, acting on overnight news before the morning open is often impractical as it's when most traders gain access to the market. Secondly, unless specialized in high-frequency trading, most funds don't continuously adjust their positions in response to intraday news due to their investment styles and process limitations.

Table 3 shows portfolios performance from all models. We notice that Fusion models and language models perform better than deep learning models in both returns and sharpe ratio. Fusion (AW) achieves the best performance with 4.33 sharpe ratio for equal-weighted long-short portfolios and 1.35 sharpe ratio for value-weighted long-short portfolios. For language models, RoBERTa has the best prformance with 4.08 sharpe ratio for equally-weighted long-short portfolios, followed by BERT model with 3.75 sharpe ratio and FinBERT model with 3.54 sharpe ratio. For deep learning models, it is interesting to notice that Neural Network is the best deep learning model compared with CNN and LSTM. It achieves over

30% return and 3.36 sharpe ratio for equally-weighted long-short portfolios, which is just slightly lower than language models.

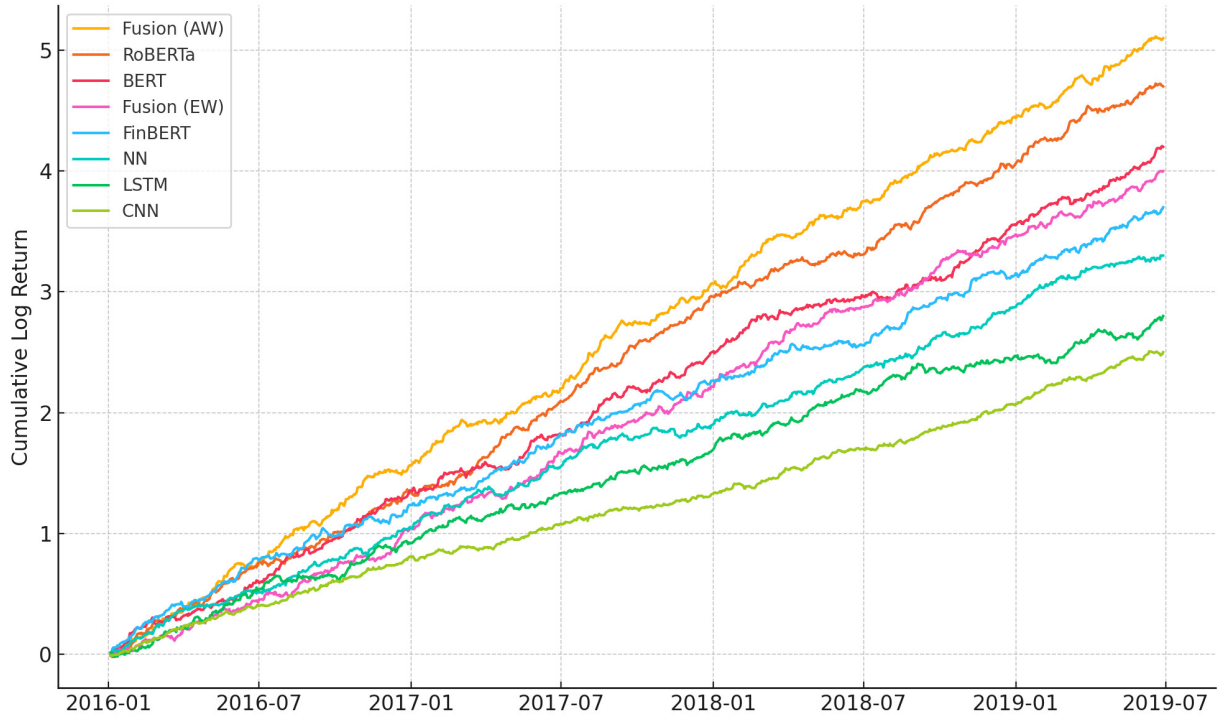


Figure 6: Cumulative log returns of US Equal-Weighted Portfolios

**Note:** US equal-weighted long-short portfolio cumulative log returns for portfolios sorted on sentiment scores. The portfolios are built on the basis of Fusion(AW),Fusion(EW), RoBERTa, BERT, FinBERT, LSTM, NN and CNN models, respectively.

## 4.7 Transaction Cost

In the previous sections, our evaluations primarily focused on examining the predictive strength of each model within an economic context, without considering transaction costs. However, to evaluate the practical feasibility of our trading strategies—especially given their high turnover—it is crucial to account for trading costs in the performance assessment.

To provide a more accurate representation of the net performance, we assume a daily transaction cost of 20 basis points (bps) for stocks. Furthermore, we refer Chen et al.

Table 3: Performance of Portfolios for all models

| | CNN | | | | NN | | | |
|---|---|---|---|---|---|---|---|---|
| | EW | | VW | | EW | | VW | |
| | Long | L-S | Long | L-S | Long | L-S | Long | L-S |
| Return | 0.11 | 0.07 | 0.06 | 0.05 | 0.30 | 0.32 | 0.22 | 0.10 |
| Std | 0.20 | 0.11 | 0.19 | 0.11 | 0.21 | 0.10 | 0.19 | 0.11 |
| Sharpe Ratio | 0.53 | 0.32 | 0.25 | 0.21 | 1.56 | 3.36 | 1.12 | 0.89 |
| | LSTM | | | | BERT | | | |
| | EW | | VW | | EW | | VW | |
| | Long | L-S | Long | L-S | Long | L-S | Long | L-S |
| Return | 0.22 | 0.14 | 0.09 | 0.02 | 0.37 | 0.37 | 0.21 | 0.12 |
| Std | 0.22 | 0.11 | 0.22 | 0.10 | 0.20 | 0.10 | 0.19 | 0.11 |
| Sharpe Ratio | 0.87 | 1.44 | 0.77 | 0.22 | 1.82 | 3.75 | 1.12 | 1.15 |
| | RoBERTa | | | | FinBERT | | | |
| | EW | | VW | | EW | | VW | |
| | Long | L-S | Long | L-S | Long | L-S | Long | L-S |
| Return | 0.39 | 0.42 | 0.22 | 0.13 | 0.35 | 0.35 | 0.20 | 0.10 |
| Std | 0.21 | 0.10 | 0.19 | 0.11 | 0.21 | 0.10 | 0.18 | 0.10 |
| Sharpe Ratio | 1.91 | 4.18 | 1.16 | 1.21 | 1.64 | 3.54 | 1.07 | 1.10 |
| | Fusion(EW) | | | | Fusion(AW) | | | |
| | EW | | VW | | EW | | VW | |
| | Long | L-S | Long | L-S | Long | L-S | Long | L-S |
| Return | 0.34 | 0.35 | 0.21 | 0.09 | 0.42 | 0.50 | 0.21 | 0.15 |
| Std | 0.22 | 0.10 | 0.19 | 0.12 | 0.20 | 0.11 | 0.19 | 0.11 |
| Sharpe Ratio | 1.69 | 3.61 | 1.15 | 0.81 | 2.03 | 4.33 | 1.14 | 1.35 |

(2022) and Ke et al. (2019) introduce a turnover reduction strategy termed Exponentially-Weighted Calendar Time (EWCT). This approach limits portfolio turnover to a fixed proportion each period, while assigning exponentially decaying weights to stocks based on how recently they appeared in the news, effectively extending their holding period.

Table 4 highlights the performance of EWCT portfolios under varying turnover constraints ($\gamma = 0.1$ to $\gamma = 0.9$). Gross returns exhibit a consistent upward trajectory, increasing from 3.32 bps at $\gamma = 0.1$ to 28.94 bps at $\gamma = 0.9$, illustrating that higher turnover allows for greater responsiveness to predictive signals. Similarly, net returns rise from 1.00 bps at $\gamma = 0.1$ to 7.51 bps at $\gamma = 0.9$, although the rate of increase is tempered by transaction costs.

The gross Sharpe ratio follows a gradual improvement, starting at 4.30 for $\gamma = 0.1$ and reaching 4.92 at $\gamma = 0.9$, indicating that while more frequent trading improves efficiency, these gains are moderate. The net Sharpe ratio, which accounts for transaction costs, peaks at 1.46 for $\gamma = 0.4$ before declining slightly to 1.44 at $\gamma = 0.5$, reflecting the increasing influence of transaction costs as turnover rises, the results is consistent with Chen et al. (2022).

The results suggest that moderate turnover levels (around $\gamma = 0.4$) provide an optimal balance between trading responsiveness and cost efficiency. Beyond this point, while gross returns and Sharpe ratios continue to improve, the net Sharpe ratio begins to decline, indicating that the marginal benefit of higher turnover is increasingly offset by transaction costs. Thus, while higher turnover facilitates greater capture of predictive signals, the associated costs ultimately constrain risk-adjusted performance.

## 4.8 Model signification

In this section, we explore the effectiveness of the model strategy from a statistical perspective. We use the Jobson and Korkie (1986) significance test with Memmel (2003) correction

Table 4: Performance Analysis of Trading Strategies with Transaction Cost

| Turnover | Gross Return | Gross Sharpe Ratio | Net Return | Net Sharpe Ratio |
|---|---|---|---|---|
| 0.10 | 3.32 | 4.30 | 1.00 | 1.34 |
| 0.20 | 6.68 | 4.52 | 1.99 | 1.43 |
| 0.30 | 9.97 | 4.61 | 2.92 | 1.46 |
| 0.40 | 13.23 | 4.73 | 3.81 | 1.46 |
| 0.50 | 16.45 | 4.75 | 4.65 | 1.44 |
| 0.60 | 19.62 | 4.79 | 5.45 | 1.42 |
| 0.70 | 22.78 | 4.85 | 6.20 | 1.40 |
| 0.80 | 25.89 | 4.88 | 6.89 | 1.37 |
| 0.90 | 28.94 | 4.92 | 7.51 | 1.32 |

to compare all models. The test is also used in Ao et al. (2018), Caner and Daniele (2023). The null and alternative hypothesis are as follows:

$$H_0 : SR_{fus} \leq SR_0$$

$$H_1 : SR_{fus} > SR_0$$

where $SR_{fus}$ is the Sharpe ratio of our model and $SR_0$ represents remaining models. We set Fusion(AW) as the benchmark to test against all other methods since it generally shows the best Sharpe ratios in both long-short and long-only portfolios. Table 5 shows p-value of our significance test.

We also analyze the pairwise correlations of returns from different strategies, visualized in the heatmap figure 7. The results show that traditional deep learning models exhibit moderate to low correlations with each other, suggesting that these models approach market predictions from different angles. In contrast, correlations among fusion models (Fusion(AW) and Fusion(EW)) are notably high, both exceeding 0.8, suggesting these models share similar characteristics or structures. The correlations between lan-

Table 5: P-value of the Jobson and Korkie test with Memmel correction for all models

| | Long-short portfolios | Long-only portfolios |
|---|---|---|
| Fusion(AW) | - | - |
| CNN | 0.026 | 0.002 |
| NN | 0.442 | 0.306 |
| LSTM | 0.282 | 0.027 |
| BERT | 0.584 | 0.640 |
| RoBERTa | 0.652 | 0.402 |
| FinBERT | 0.256 | 0.352 |
| Fusion(EW) | 0.196 | 0.016 |

guage models (FinBERT, BERT, and RoBERTa) and other strategies remain moderate, highlighting the ability of NLP-based models to provide unique insights, particularly in understanding textual context in market movements. The correlation between LLM-based strategies and traditional machine learning models like CNN and LSTM is generally lower, emphasizing the different methodologies these models use to interpret data, indicating a divergence in how they interpret the market. Besides, it is interesting to notice that LSTM and NN has a correlation with 0.66, which indicates that they both capture the temporal correlation of time series.

## 4.9 Empirical results from different time period

To further assess the effectiveness and robustness of our strategy, we transform the test set to be during a recession and implement our strategy. We set up the training time period from January 2000 to Decemeber 2007 and test time period from January 2008 to December 2009.

Table 6 presents the portfolio performance of portfolios across different models. All

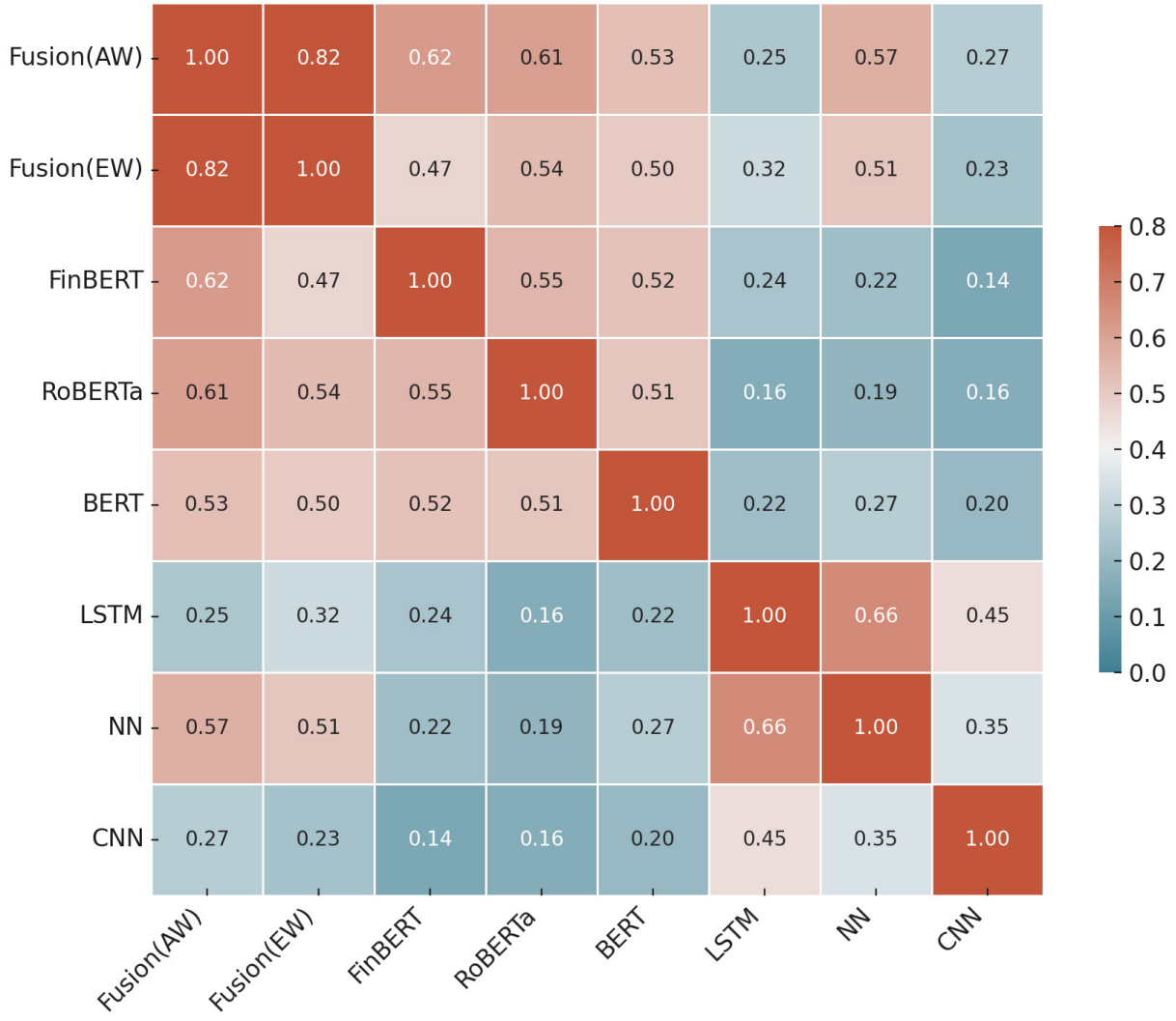


Figure 7: Correlation of Returns for sentiment portfolios

Table 6: Performance of Portfolios for all models during recession

| | CNN | | NN | | LSTM | | BERT | |
|---|---|---|---|---|---|---|---|---|
| | EW | VW | EW | VW | EW | VW | EW | VW |
| Sharpe Ratio | 0.23 | -0.22 | 0.44 | 0.09 | 0.37 | -0.07 | 1.27 | 0.50 |
| | RoBERTa | | FinBERT | | Fusion(EW) | | Fusion(AW) | |
| | EW | VW | EW | VW | EW | VW | EW | VW |
| Sharpe Ratio | 1.34 | 0.55 | 1.12 | 0.21 | 1.15 | 0.26 | 1.37 | 0.60 |

models' sharpe ratios decrease a lot. Among the models, Fusion(AW) still reaches a Sharpe Ratio of 1.37 under EW and 0.60 under VW, outperforming other deep learning models while staying competitive with LLMs. Language models, especially BERT and RoBERTa, stand out with notably high Sharpe Ratios. BERT exhibits a Sharpe Ratio of 1.27 under the equally weighted scheme, while RoBERTa achieves an even higher ratio of 1.34. This indicates that language models can capture nuances, trends, and sentiment from text, which becomes crucial during a recession when markets are influenced by uncertainty and sentiment shifts.

In contrast, consistent with Gu et al. (2020), deep learning models typically work on structured numerical data and rely primarily on past price. They may miss out on the subtleties found in economic or financial text data. In our results, CNN and LSTM show much lower performance. CNN's Sharpe Ratios are 0.23 (EW) and -0.22 (VW), while LSTM achieves 0.37 (EW) and -0.07 (VW), indicating that these models struggle to generate competitive returns during recessions, especially under the value-weighted scheme.

# 5 Conclusion

We introduce an innovative multimodal data framework for stock price movement prediction, incorporating a stock graphical modality alongside market and text modalities. Our model addresses critical challenges in return prediction, specifically how to extract meaningful signals from large-scale, heterogeneous datasets and how to prioritize the most dominant sources of information. By leveraging a 20-year dataset from the US stock market, our analysis demonstrates that deep learning techniques and language models effectively extract key features from diverse data sources, with time series data emerging as more influential than graphical and text modalities. Utilizing an efficient weight allocation method, attention mechanisms, we mitigate conflicts between modalities. Notably, our study rep-

resents a pioneering approach to applying multimodal data in stock market prediction, outperforming single-modal approaches in both accuracy and returns, and achieving superior results over dual-modal models. Our top-performing model, Fusion (AW), achieves over 70% balanced accuracy and a Sharpe ratio of 4.33 annually, surpassing all other deep learning and language models. Furthermore, through rigorous out-of-sample testing during recessions, we provide robust evidence of the model's resilience and superior performance even in economic downturns.

We have not only demonstrated the efficacy of deep learning in extracting features from traditional data sources but also leveraged the capabilities of large language models (LLMs) in natural language processing (NLP) to generate contextualized embeddings from news text. This marks a significant advancement over conventional approaches that rely on traditional statistical and technical analysis, providing a more nuanced and comprehensive view of market dynamics. However, our LLM-based model, with its large number of parameters, requires extensive training, making it dependent on the size and quality of the training datasets. As datasets continue to grow and model complexity increases, the difficulty of training these models escalates exponentially, potentially leading to diminishing returns in performance improvements compared to other state-of-the-art methods.

Overall, our research enhances the accuracy of stock movement prediction and deepens the understanding of multimodal data's role in financial predictive modeling. These findings have important implications for investment risk management and decision-making. Furthermore, our work paves the way for future studies, including the integration of additional data sources and the application of more advanced machine learning and deep learning techniques to further improve predictive performance. Future research could explore these directions to bolster the robustness and applicability of the multimodal data-driven machine learning framework.

# References


Ao, M., Yingying, L., and Zheng, X. (2018). Approaching Mean-Variance Efficiency for Large Portfolios. *The Review of Financial Studies*, 32(7):2890–2919.

Araci, D. (2019). Finbert: Financial sentiment analysis with pre-trained language models. *arXiv preprint arXiv:1908.10063*.

Barberis, N. (2018). Psychology-based models of asset prices and trading volume. 1:79–175.

Bessembinder, H. and Chan, K. (1998). Market efficiency and the returns to technical analysis. *Financial Management*, 27(2):5–17.

Birogul, S., Temür, G., and Kose, U. (2020). Yolo object recognition algorithm and "buy-sell decision" model over 2d candlestick charts. *IEEE access*, 8:91894–91915.

Box, G. E., Jenkins, G. M., Reinsel, G. C., and Ljung, G. M. (2015). *Time series analysis: forecasting and control*. John Wiley & Sons.

Caner, M. and Daniele, M. (2023). Deep learning based residuals in non-linear factor models: Precision matrix estimation of returns with low signal-to-noise ratio. Available at https://arxiv.org/abs/2209.04512.

Chen, J.-H. and Tsai, Y.-C. (2020). Encoding candlesticks as images for pattern classification using convolutional neural networks. *Financial Innovation*, 6(1):26.

Chen, L., Pelger, M., and Zhu, J. (2024). Deep learning in asset pricing. *Management Science*, 70(2):714–750.

Chen, Y., Kelly, B. T., and Xiu, D. (2022). Expected returns and large language models. *Available at SSRN 4416687*.

Cohen, N., Balch, T., and Veloso, M. (2020). Trading via image classification. In *Proceedings of the first ACM international conference on AI in finance*, pages 1–6.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Feng, D., Haase-Schütz, C., Rosenbaum, L., Hertlein, H., Glaeser, C., Timm, F., Wiesbeck, W., and Dietmayer, K. (2020). Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges. *IEEE Transactions on Intelligent Transportation Systems*, 22(3):1341–1360.

Ghosh, P., Neufeld, A., and Sahoo, J. K. (2022). Forecasting directional movements of stock prices for intraday trading using lstm and random forests. *Finance Research Letters*, 46:102280.

Gu, S., Kelly, B., and Xiu, D. (2020). Empirical Asset Pricing via Machine Learning. *The Review of Financial Studies*, 33(5):2223–2273.

Hamilton, J. D. (2020). *Time series analysis*. Princeton university press.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366.

Hoseinzade, E. and Haratizadeh, S. (2019). Cnnpred: Cnn-based stock market prediction using a diverse set of variables. *Expert Systems with Applications*, 129:273–285.

Hu, G., Hu, Y., Yang, K., Yu, Z., Sung, F., Zhang, Z., Xie, F., Liu, J., Robertson, N., Hospedales, T., et al. (2018). Deep stock representation learning: From candlestick charts

to investment decisions. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 2706–2710. IEEE.

Huang, K., Shi, B., Li, X., Li, X., Huang, S., and Li, Y. (2022). Multi-modal sensor fusion for auto driving perception: A survey. *arXiv preprint arXiv:2202.02703*.

Huang, S.-C., Shen, L., Lungren, M. P., and Yeung, S. (2021). Gloria: A multimodal global-local representation learning framework for label-efficient medical image recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3942–3951.

Jegadeesh, N. and Wu, D. (2013). Word power: A new approach for content analysis. *Journal of financial economics*, 110(3):712–729.

Jiang, J., Kelly, B., and Xiu, D. (2023). (re-) imag (in) ing price trends. *The Journal of Finance*, 78(6):3193–3249.

Jobson, J. D. and Korkie, B. (1986). Performance hypothesis testing with the sharpe and treynor measures: A comment. *Journal of Finance*, 41(5):1175–1176.

Ke, Z. T., Kelly, B. T., and Xiu, D. (2019). Predicting returns with text data. Technical report, National Bureau of Economic Research.

Khaidem, L., Saha, S., and Dey, S. R. (2016). Predicting the direction of stock market prices using random forest. *arXiv preprint arXiv:1605.00003*.

Kim, H. Y. and Won, C. H. (2018). Forecasting the volatility of stock price index: A hybrid model integrating lstm with multiple garch-type models. *Expert Systems with Applications*, 103:25–37.

Kim, K.-j. (2003). Financial time series forecasting using support vector machines. *Neurocomputing*, 55(1-2):307–319.

Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *Computer Science*. Available at https://arxiv.org/abs/1412.6980.

Kraus, M. and Feuerriegel, S. (2017). Decision support from financial disclosures with deep neural networks and transfer learning. *Decision Support Systems*, 104:38–48.

Lee, S. I. and Yoo, S. J. (2020). Multimodal deep learning for finance: integrating and forecasting international stock markets. *The Journal of Supercomputing*, 76:8294–8312.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Loughran, T. and McDonald, B. (2016). Textual analysis in accounting and finance: A survey. *Journal of Accounting Research*, 54(4):1187–1230.

Marshall, B. R., Young, M. R., and Rose, L. C. (2006). Candlestick technical trading strategies: Can they create value for investors? *Journal of Banking Finance*, 30(8):2303–2323.

Memmel, C. (2003). Performance hypothesis testing with the sharpe ratio. *Finance Letters*, 1. Available at AvailableatSSRN412588.

Nagel, S. (2021). *Machine learning in asset pricing*, volume 8. Princeton University Press.

Nti, I. K., Adekoya, A. F., and Weyori, B. A. (2020). A systematic review of fundamental and technical analysis of stock market predictions. *Artificial Intelligence Review*, 53(4):3007–3057.

Pagolu, V. S., Reddy, K. N., Panda, G., and Majhi, B. (2016). Sentiment analysis of twitter data for predicting stock market movements. In *2016 international conference on signal*

*processing, communication, power and embedded system (SCOPES)*, pages 1345–1350. IEEE.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. In Walker, M., Ji, H., and Stent, A., editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Rahman, W., Hasan, M. K., Lee, S., Zadeh, A., Mao, C., Morency, L.-P., and Hoque, E. (2020). Integrating multimodal information in large pretrained transformers. In *Proceedings of the conference. Association for Computational Linguistics. Meeting*, volume 2020, page 2359. NIH Public Access.

Selvin, S., Vinayakumar, R., Gopalakrishnan, E., Menon, V. K., and Soman, K. (2017). Stock price prediction using lstm, rnn and cnn-sliding window model. In *2017 international conference on advances in computing, communications and informatics (icacci)*, pages 1643–1647. IEEE.

Udagawa, Y. (2018). Predicting stock price trend using candlestick chart blending technique. pages 4162–4168.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Wang, M. and Wang, Y. (2019). Evaluating the effectiveness of candlestick analysis in forecasting us stock market. In *Proceedings of the 2019 3rd International Conference on Compute and Data Analysis*, pages 98–101.

Wang, Z., Wu, Z., Agarwal, D., and Sun, J. (2022). Medclip: Contrastive learning from unpaired medical images and text. *arXiv preprint arXiv:2210.10163*.

Yu, W., Xu, H., Meng, F., Zhu, Y., Ma, Y., Wu, J., Zou, J., and Yang, K. (2020). Ch-sims: A chinese multimodal sentiment analysis dataset with fine-grained annotation of modality. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 3718–3727.

Zhang, G. P. (2003). Time series forecasting using a hybrid arima and neural network model. *Neurocomputing*, 50:159–175.

Zhang, H., Wang, Y., Yin, G., Liu, K., Liu, Y., and Yu, T. (2023). Learning language-guided adaptive hyper-modality representation for multimodal sentiment analysis. *arXiv preprint arXiv:2310.05804*.

Zhang, W., Zhao, L., Xia, H., Sun, S., Sun, J., Qin, M., Li, X., Zhao, Y., Zhao, Y., Cai, X., et al. (2024). A multimodal foundation agent for financial trading: Tool-augmented, diversified, and generalist. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 4314–4325.

Zhang, Y., Jiang, H., Miura, Y., Manning, C. D., and Langlotz, C. P. (2022). Contrastive learning of medical visual representations from paired images and text. In *Machine Learning for Healthcare Conference*, pages 2–25. PMLR.