Which Data Tells the Truth? A Multi-modal Deep Learning Framework for Stock Price Movement Prediction

January 5, 2026

Abstract

This paper introduces a novel multimodal data framework for stock price movement prediction, combining stock graphical, market, and text modalities with state-of-the-art natural language processing models such as BERT. 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 a higher balanced accuracy and a Sharpe ratio of 3.05 annually, outperforming single- and dual-modal approaches. Moreover, our model shows its ability and robustness in three recent periods—the COVID-19 crash (2020), the Russia–Ukraine shock (2022–2023), and the AI-driven expansion (2023–present). 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, Natural Language Processing, Large Language Models, Attention Mechanism, Return Prediction, Multimodal Data

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, deep learning and language models, have become increasingly in-depth. Compared with traditional statistical methods, those methods bring unique advantages to handling high-dimensional data in capturing complex non-linear relationships and feature selection (Nagel 2021). Moreover, those "black-box" method has shown their 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, 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 (1994), 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. Deep Neural Networks (DNNs) 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 (2025) 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, LLama, 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.

Beyond these methodological contributions, our study offers a further novel contribution by providing a systematic comparison of multimodal deep learning models across multiple historically distinct market regimes. Specifically, we evaluate our framework during three economically significant periods—the COVID–19 crash in 2020, the Russia–Ukraine geopolitical shock in 2022–2023, and the AI–driven expansion beginning in 2023. These episodes embody fundamentally different sources of market dislocation and return predictability, ranging from a global macroeconomic collapse to a geopolitical and commodity–driven shock and a technology–induced secular boom. By examining model performance across these heterogeneous environments, we demonstrate that multimodal architectures—particularly those incorporating language models and attention–based fusion—exhibit superior robustness and maintain strong predictive power despite substantial shifts in the underlying return–generating process. This cross–period evidence highlights the structural adaptability of our approach and confirms that its predictive advantages persist across crisis, transition, and expansion phases.

Our model was applied to analyze 20 years of U.S. equity returns using three diverse datasets: market data, OHLCV (Open, High, Low, Close, Volume) 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, DNN, 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 67.47% and an accuracy rate of 60.53%, surpassing the results of individual deep learning and language models. Economically, the model also delivered exceptional outcomes, producing the highest annualized Sharpe ratios: 3.05 for equal-weighted long-short portfolios and 1.68 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 OHLCV charts and financial news. When we excluded time series data, the model's accuracy dropped sharply from 60.53% to 47.32%, 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 a over 10% increase in accuracy and 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 evaluate the robustness of our framework during episodes of severe market stress. Even in recessionary or crisis–driven environments—where return dynamics are highly unstable and traditional models tend to deteriorate—our approach maintains superior predictive performance relative to individual deep learning architectures and stand-alone language models. The strategy continues to generate accurate forecasts and economically meaningful risk-adjusted returns, underscoring its ability to adapt to both volatile and tranquil market conditions. This sustained performance across adverse regimes provides strong evidence of the model's structural resilience and highlights its broad applicability in real-world financial forecasting settings.

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, deep learning and large language models 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 Deep Neural networks (DNNs)

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 2025)

We construct Neural Network models as follows:

$$x_t^{(0)} = x_t, \tag{1}$$

$$x_t^{(l)} = g_1\left(b^{(l-1)} + W^{(l-1)} x_t^{(l-1)}\right), \quad l = 1, \ldots, L_f \tag{2}$$

$$y^{(\text{num})} = b^{(L_f)} + W^{(L_f)} x_t^{(L_f)}. \tag{3}$$

Equation (1) initializes the network with the vector of individual asset returns, $x_t \in R^N$. Equation (2) represents a transformation of the asset return vector as it propagates through the hidden layers, where $x_t^{(l-1)} \in R^{K^{(l-1)}}$ are inputs from previous layer, $W^{(l-1)} \in R^{K^{(l)} \times K^{(l-1)}}$ is the weight matrix mapping from layer $l-1$ to l, $b^{(l-1)}$ is the bias vector for layer l, $K^{(l)}$ denotes the number of neurons in layer l. Equation (3) defines the final set of features extracted at the output layer. The vector $y^{(\text{num})} \in R^K$ is the feature from numerical data, and $g_1(\cdot)$ is the activation function.

We refer to Gu et al. (2020), who use a RELU function as $g_1(\cdot)$. Figure 1 shows the feedforward neural networks (FFNs) structure with one hidden layer.

2.1.2 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,

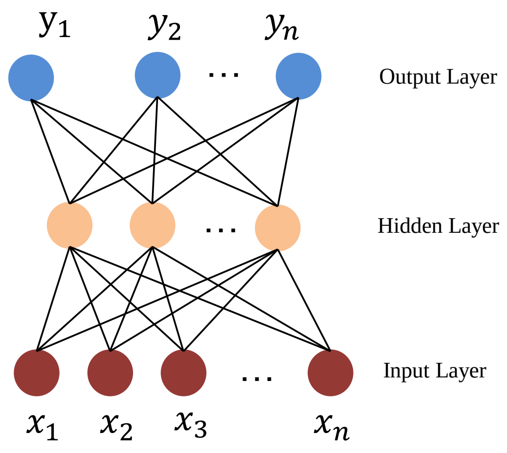


Figure 1: Feed Forward Network with one hidden layer

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 Long Short-Term Memory (LSTM) network is a type of neural network designed to process sequential data. It improves on traditional recurrent neural networks (RNNs) by incorporating mechanisms to remember long-term dependencies while selectively forgetting irrelevant information.

An LSTM consists of three main components, known as gates: the input gate inputgate_t, the forget gate forgetgate_t, and the output gate outputgate_t, along with a cell state c_t, which stores long-term memory.

At each time step t, the forget gate forgetgate_t determines how much of the previous memory (c_{t-1}) should be retained or discarded.

$$\text{forgetgate}_t = \sigma\left(W_f x_t + U_f h_{t-1} + b_f\right) \tag{4}$$

where $x_t \in R^N$ is the stock returns. $h_{t-1} \in R^{d_h}$ is the hidden state from the previous time step, capturing past sequence information. $W_f \in R^{d_h \times N}$ maps the input to the forget gate. $U_f \in R^{d_h \times d_h}$ maps the previous hidden state to the forget gate. $b_f \in R^{d_h}$ is a bias term. $\sigma(\cdot)$ is the sigmoid activation function, which outputs values between 0 and 1.

The input gate inputgate_t controls how much of the new input should be added to the cell state. Additionally, a candidate cell state $\tilde{c}_t$ is computed as a potential update to the memory.

$$\text{inputgate}_t = \sigma\left(W_i x_t + U_i h_{t-1} + b_i\right) \tag{5}$$

$$\tilde{c}_t = \tanh\left(W_c x_t + U_c h_{t-1} + b_c\right) \tag{6}$$

where $W_i \in R^{d_h \times N}$, $U_i \in R^{d_h \times d_h}$, $b_i \in R^{d_h}$. $W_c \in R^{d_h \times N}$, $U_c \in R^{d_h \times d_h}$, $b_c \in R^{d_h}$. $\tilde{c}_t \in R^{d_h}$ represents the new candidate information to be stored in memory. $\tanh(\cdot)$ ensures that the candidate values are between -1 and 1.

Once the forget and input gates decide what information to discard and what to add, the cell state is updated as:

$$c_t = \text{forgetgate}_t \times c_{t-1} + \text{inputgate}_t \times \tilde{c}_t \tag{7}$$

where $c_t \in R^{d_h}$ is the updated memory, and $c_{t-1} \in R^{d_h}$ is the previous memory.

The final step determines what part of the updated cell state should be used as the hidden state h_t, which carries short-term information to the next time step and is used for making predictions.

$$\text{outputgate}_t = \sigma\left(W_o x_t + U_o h_{t-1} + b_o\right) \tag{8}$$

$$h_t = \text{outputgate}_t \times \tanh\left(c_t\right) \tag{9}$$

where $W_o \in R^{d_h \times N}$, $U_o \in R^{d_h \times d_h}$, $b_o \in R^{d_h}$, $h_t \in R^{d_h}$ is the new hidden state.

The last step is to link the hidden state h_t to an extracted feature output $y^{(\text{num})} \in R^K$. This is typically done using a fully connected layer:

$$y^{(\text{num})} = W_y h_t + b_y \tag{10}$$

where, $W_y \in R^{K \times d_h}$ maps the hidden state to the output, $b_y \in R^K$ is a bias term.

The LSTM's ability use forget gate decides what past information to retain, input gate which determines what new information should be stored, cell state integrates past and new knowledge and output gate which controls what information is passed forward to give us final extracted feature y_t from the hidden state h_t. Figure 2 show the LSTM structure.

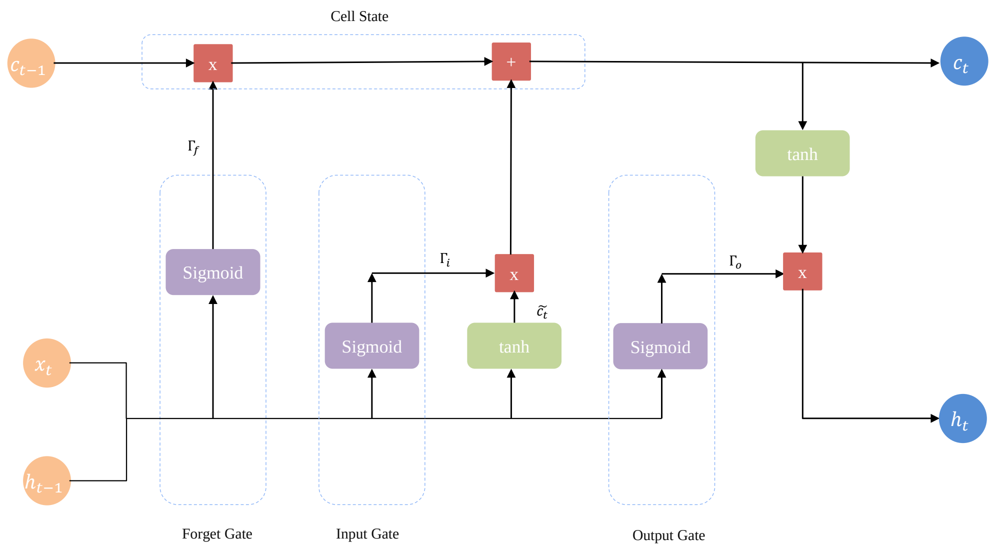


Figure 2: LSTM structure

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.

When we use CNN to processes 2D structured data(e.g., patterns), there are three key dimensions: Height (H) which is the number of rows in the input feature matrix, Width (W) which is the number of columns in the input feature matrix, and the number of Channels (C) which is the depth of the input (number of feature maps). Unlike physical

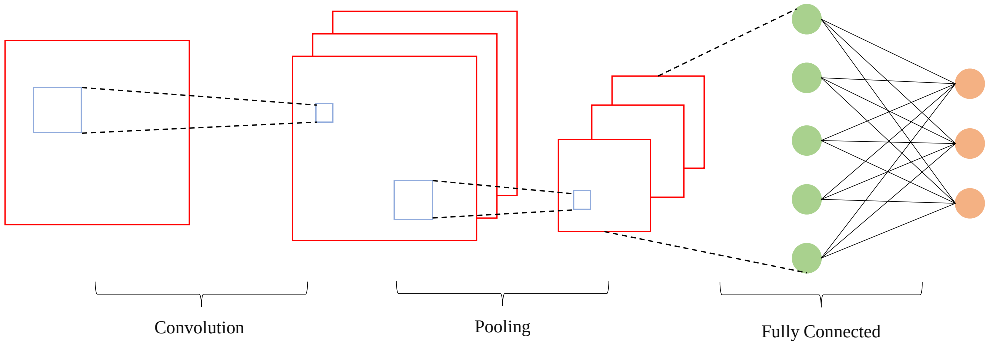


Figure 3: CNN with one convolution and pooling layer

images where height and width may be measured in centimeters (cm) or meters (m), in CNNs, these values are purely integer counts of rows and columns.

If the input data consists of structured numerical values (such as a 2D feature matrix), its shape is $x \in R^{H \times W \times C}$ where H is the number of rows, W is the number of columns, C is the number of feature maps.

The simplified mathematical process to extract features by CNN is similar to the FFN. A more detailed process about CNN process will be shown in Appendix 2A. The main difference is that the input of CNN is an image not a list of data. The selected feature $y \in R^K$ of one CNN block (one convolution layer and one pooling layer) with input image $x \in R^{H \times W \times C}$ can be written using the following formulas:

$$x_{conv} = g_{conv}\left(W_{conv}x + b_{conv}\right) \tag{11}$$

$$x_{pooling} = pooling\left(x_{conv}\right) \tag{12}$$

$$y^{(\text{patterns})} = g_{fcon}\left(W_{fcon}x_{pooling} + b_{fcon}\right) \tag{13}$$

Where x is the pattern input, 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, we follow (Ke et al. 2019) to use the LeakReLu function for g_{conv}.

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.

2.3.6 Extracting Features from Text Using LLMs

Large Language Models (LLMs) provide a structured way to transform unstructured textual data into numerical representations that can be further used for financial forecasting and econometric modeling. The process can be simplified as follows:

$$y^{(\text{text})} = LLM(x^{(\text{text})}) \tag{14}$$

where $x^{(\text{text})}$ represents the input textual data (e.g., financial news), and $y^{(\text{text})}$ represents the extracted feature representations. The function $LLM(\cdot)$ encapsulates the transformation process through a pre-trained language model.

Given a document $x^{(\text{text})}$, the first step is to tokenize it into subword units:

$$\mathbf{T} = Tokenizer(x^{(\text{text})}) \tag{15}$$

where $\mathbf{T} = t_1, t_2, \ldots, t_N$ represents the sequence of N tokens. These tokens are then mapped to their respective numeric IDs and embedded into a high-dimensional vector space:

$$\mathbf{E} = \text{Embedding}(\mathbf{T}) \tag{16}$$

where $\mathbf{E} \in R^{N \times d}$, with d being the embedding dimension.

To derive document-level features, the token embeddings are passed through multiple layers of a Transformer encoder to capture deep contextual relationships:

$$y^{(\text{text})} = \text{Transformer}(\mathbf{E}) \tag{17}$$

where $\mathbf{z}$ represents the final hidden states of all tokens, capturing rich semantic and contextual features from the input text. The extracted features $y^{(\text{text})}$ are then used in financial forecasting models. We follow Chen et al. (2022) to use BERT, RoBERTa, and FinBERT to denote the final extracted feature as:

$$\mathbf{y}^{(\text{text})} \in R^{768} \tag{18}$$

which represents the 768-dimensional feature vector obtained from above models.

2.4 Methods of weights allocation

In predicting financial markets, the integration of multimodal data—combining diverse types of information has been a challenge. It is hard to decide optimal weights for 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, there are three vectors: the Query vector (Q) which determines how to attend to other inputs, the Key vector (K) which represents each input in a way that can be matched by queries, and the Value vector (V) which contains the actual content of the inputs 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.

The attention mechanism is a computational method that dynamically assigns different importance levels to various input elements, enabling the model to capture relationships between features extracted from different data types (numerical data, patterns).

In this setting, the extracted features from different modalities have distinct dimensions:

$$y^{(\text{num})} \in R^{K_{\text{num}}}, \quad y^{(\text{patterns})} \in R^{K_{\text{patterns}}}, \quad y^{(\text{text})} \in R^{K_{\text{text}}}, \tag{19}$$

where $y^{(\text{num})}$ represents the extracted numerical features, $y^{(\text{patterns})}$ represents the extracted pattern-based features, $y^{(\text{text})}$ represents the extracted text-based features. Since these features have different dimensions, we must align them into a common space before applying attention.

To process multi-modal inputs in a unified attention framework, we apply linear transformations to project both feature types into a shared space of dimension K_{shared}:

For each modality, we compute separate queries, keys, and values using its own weight matrices:

$$Q^{(\text{num})} = y^{(\text{num})} W_Q^{(\text{num})}, \quad K^{(\text{num})} = y^{(\text{num})} W_K^{(\text{num})}, \quad V^{(\text{num})} = y^{(\text{num})} W_V^{(\text{num})} \tag{20}$$

$$Q^{(\text{patterns})} = y^{(\text{patterns})} W_Q^{(\text{patterns})}, \quad K^{(\text{patterns})} = y^{(\text{patterns})} W_K^{(\text{patterns})}, \quad V^{(\text{patterns})} = y^{(\text{patterns})} W_V^{(\text{patterns})} \tag{21}$$

$$Q^{(\text{text})} = y^{(\text{text})} W_Q^{(\text{text})}, \quad K^{(\text{text})} = y^{(\text{text})} W_K^{(\text{text})}, \quad V^{(\text{text})} = y^{(\text{text})} W_V^{(\text{text})} \tag{22}$$

where $W_Q^{(\text{num})} \in R^{K_{\text{num}} \times d_k}$, $W_K^{(\text{num})} \in R^{K_{\text{num}} \times d_k}$, $W_V^{(\text{num})} \in R^{K_{\text{num}} \times d_v}$, $W_Q^{(\text{patterns})} \in R^{K_{\text{patterns}} \times d_k}$, $W_K^{(\text{patterns})} \in R^{K_{\text{patterns}} \times d_k}$, $W_V^{(\text{patterns})} \in R^{K_{\text{patterns}} \times d_v}$, $W_Q^{(\text{text})} \in R^{K_{\text{text}} \times d_k}$, $W_K^{(\text{text})} \in R^{K_{\text{text}} \times d_k}$, $W_V^{(\text{text})} \in R^{K_{\text{text}} \times d_v}$.

Weights of queries ($W_Q^{(\text{num})}$, $W_Q^{(\text{patterns})}$, $W_Q^{(\text{text})}$) transform the features into a query representation, which will later search for relevant information. Weights of Keys ($W_K^{(\text{num})}$, $W_K^{(\text{patterns})}$, $W_K^{(\text{text})}$) transform the features into a key representation, which will be compared to the queries. Weights of values($W_V^{(\text{num})}$, $W_V^{(\text{patterns})}$,$W_V^{(\text{text})}$) transform the features into a value representation, which contains the actual feature information that will be aggregated.

Next step is to get the attention score between two different data types. We apply the softmax function[1] to convert attention scores into a probability distribution. The employment of the softmax function converts the scores into a probability distribution,

[1]The softmax funtion is is used in attention mechanisms to normalize raw attention scores into a probability distribution, ensuring the weights sum to 1 and highlighting the most relevant features. This prevents unstable gradients and helps the model focus on important information while filtering out noise. Unlike other activation functions, softmax guarantees proper weighting, making attention interpretable and effective. The mathematical formula for a softmax function is as follows:

$$\text{softmax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)} \tag{23}$$

where higher scores increase the weight of the corresponding value in the output, and ensure that the weights sum to one.

We use numerical data (num) and image data (patterns) as an example:

$$\text{AttentionScore}_{\text{num}\to\text{patterns}} = g\left(\frac{Q^{(\text{num})}\left(K^{(\text{patterns})}\right)^{T}}{\sqrt{d_k}}\right) \tag{24}$$

This measures how much attention numerical features should pay to image-based features. The softmax function $g(\cdot)$ ensures that the scores sum to 1, making them interpretable as probability weights.

Once we have the attention scores, we compute the final attention-weighted representation by multiplying these scores with the corresponding value vectors. The final attention-weighted feature representation is:

$$\tilde{y}^{(\text{patterns})} = \text{AttentionScore}_{\text{num}\to\text{patterns}} V^{(\text{patterns})} \tag{25}$$

This produces a refined numerical representation that selectively integrates relevant image information, guided by the numerical dataset. All Q, K, V for the two different data formats (numerical and pattern-based) are trainable parameters.

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 apply three different types of datasets: numerical datset $x_t^{(\text{num})}$, patterns dataset $x_t^{(\text{patterns})}$, textual dataset $x_t^{(\text{text})}$. For different datasets, we use different deep learning methods to extract features.

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

$$x^{(\text{con\_patterns})} = g_1\left(W_1 x^{(\text{patterns})} + b_1\right) \tag{26}$$

$$x^{(\text{pool\_patterns})} = pooling\left(x^{(\text{con\_patterns})}\right) \tag{27}$$

$$y^{(\text{patterns})} = g_2\left(W_2 x^{(\text{pool\_patterns})} + b_2\right) \tag{28}$$

where $y^{(\text{patterns})}$ represents final extracted features from CNN model, $x^{(\text{patterns})}$ represents the input patterns, $x^{(\text{con\_patterns})}$ is the output of the convolution layer, $x^{(\text{pool\_patterns})}$ 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 numerical dataset, we show the feature selection process with one layer neural network:

$$\tilde{x}^{(\text{num})} = g_3\left(W_3 x^{(\text{num})} + b_3\right) \tag{29}$$

$$y^{(\text{num})} = g_4\left(W_4 \tilde{x}^{(\text{num})} + b_4\right) \tag{30}$$

where $y^{(\text{num})} \in R^{K_{\text{num}}}$ represents final output of extracted features from numerical dataset, $x_t^{(\text{num})} \in R^N$ represents the input of numerical data, $\tilde{x}_{num}$ 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 textual dataset, state-of-art language models are used for feature extraction process.

$$y^{(\text{text})} = LLM\left(x^{(\text{text})}\right) \tag{31}$$

where $y^{(\text{text})} \in R^{K_{\text{text}}}$ 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 concatenates all extracted features together. The concatenation operation combines these vectors along a new axis, typically along the feature dimension.

$$y^{(\text{concat})} = concat(y^{(\text{patterns})}, y^{(\text{num})}) \tag{32}$$

where $y^{(\text{concat})} \in R^{K_{\text{patterns}}+K_{\text{num}}}$.

The second method will use attention mechanism for weights allocation, to calculate attention score between different type of dataset, especially the score α between image data and time series data and the attention score β 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.

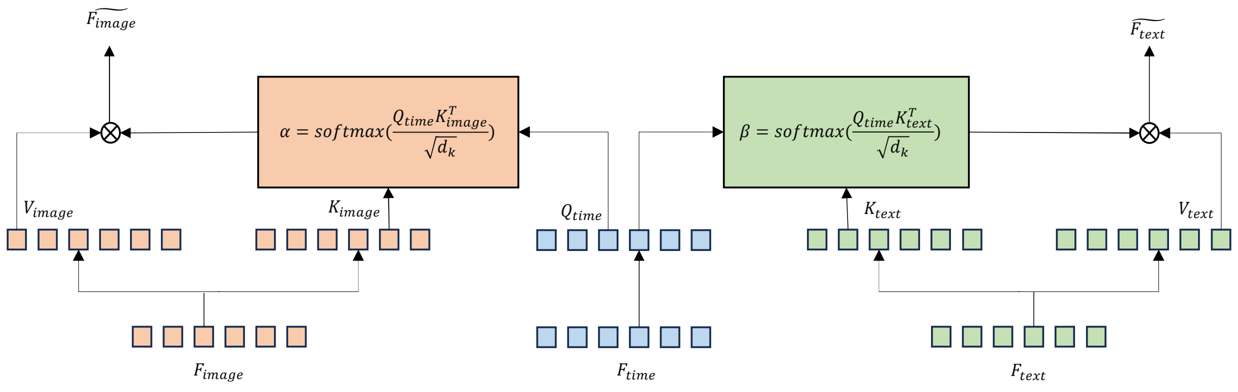


Figure 4: Attention Layer structure

Note: α represents the similarity matrix between graphic features and time series numerical features, while β represents the similarity matrix between time series numerical features and textual features.

$$\alpha = softmax\left(\frac{Q^{(\text{num})}(K^{(\text{patterns})})^T}{\sqrt{d_k}}\right) \tag{33}$$

$$\beta = softmax\left(\frac{Q^{(\text{num})}(K^{(\text{text})})^T}{\sqrt{d_k}}\right) \tag{34}$$

The new features extracted from images via the score α is:

$$\tilde{y}^{(\text{patterns})} = \alpha V_{patterns} \tag{35}$$

The new features extracted from text data via the score β

$$\tilde{y}^{(\text{text})} = \beta V_{text} \tag{36}$$

Then we concatenate two features.

$$\tilde{y}^{(\text{concat})} = concat(\tilde{y}^{(\text{patterns})}, y^{(\text{num})}, \tilde{y}^{(\text{text})}) \tag{37}$$

After we have obtained the new concatenated features from two different data formats, our last step is to obtain the prediction score based on concatenated features. We achieve this using a fully connected layer,

$$\hat{y} = \sigma(W\tilde{y}^{(\text{concat})} + b) \tag{38}$$

where $\hat{y}$ is the prediction score, $\sigma(\cdot)$ is the sigmoid function, which is $\sigma(z) = \frac{1}{1+e^{-z}}$ to output values between 0 and 1, interpreted as a probability. The fully connected layer performs a linear transformation on the concatenated features before applying the sigmoid function to produce the final prediction score. Figure 5 shows our model structure.

3.2 Estimation

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

$$L(y, \hat{y}) = -y\log(\hat{y}) - (1-y)\log(1-\hat{y}) \tag{39}$$

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.

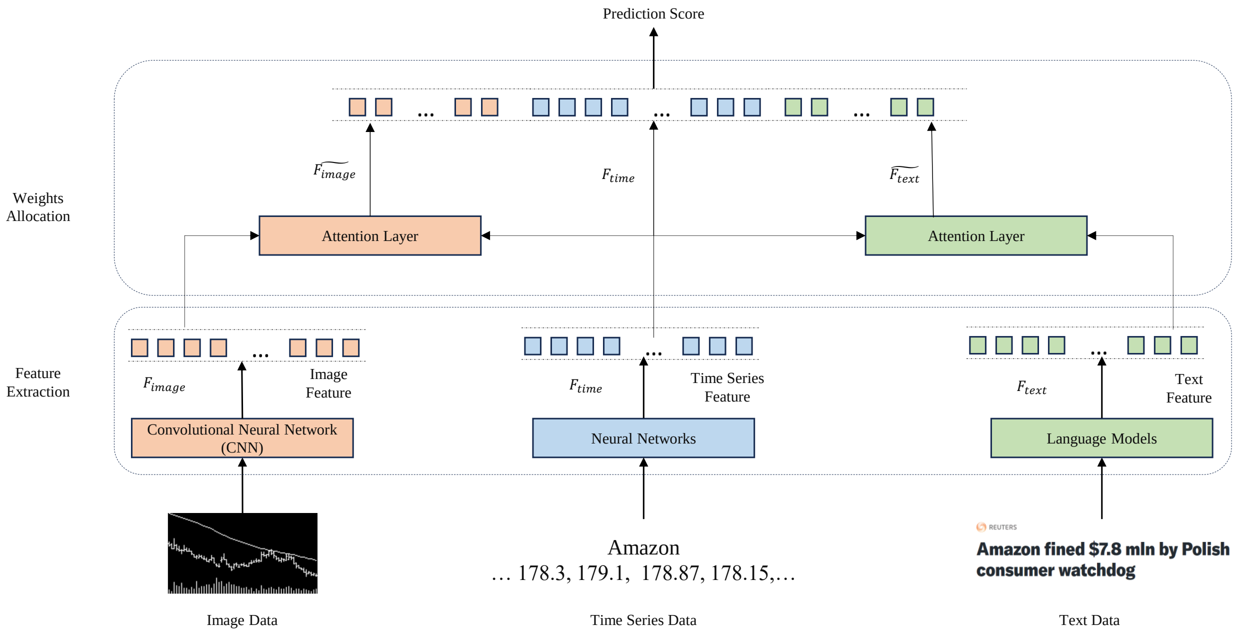


Figure 5: Multimodality Model structure

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 2004 to June 2024, 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 OHLCV images (Open, High, Low, Close and Volume) by collecting data from CRSP within the same time period. For financial news dataset, we use news from Refinitiv, which is rebranded as Massive now, whose dataset encompasses company's news, with each 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 news 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 Rolling-Window Model Training

To evaluate the predictive performance of each model in a time-consistent manner, we employ an month rolling-window estimation framework. This design ensures that at every point in the sample, the model is trained exclusively on data available up to that date and is evaluated on a strictly out-of-sample future period, thereby eliminating look-ahead bias.

Each rolling window spans a total of 36 months. The first thirty months of the window are used for in-sample estimation, consisting of a 24-month training subperiod followed by a 6-month validation subperiod. The subsequent 6-month period is held out as the out-of-sample test set for that window.

Formally, let t_k denote the starting date of rolling window k. The in-sample region is defined as

$$[t_k,\, t_k + 30 \text{ months}) = \underbrace{[t_k,\, t_k + 24 \text{ months})}_{\text{training}} \cup \underbrace{[t_k + 24 \text{ months},\, t_k + 30 \text{ months})}_{\text{validation}}.$$

Model parameters are estimated on the training subsample using the standard classification objective and are subsequently tuned using the 6-month validation segment. Hyperparameters such as learning rate, number of trees, maximum depth, and regularization parameters are selected based on validation performance.

After this in-sample phase is completed, the model is evaluated on the 6-month out-of-sample segment:

$$[t_k + 30 \text{ months}, \, t_k + 36 \text{ months}),$$

which contains observations strictly later than those used for model estimation or tuning.

Once the 6-month test set is exhausted, the entire rolling window advances by one calendar month, forming the next window $k + 1$. The model is reestimated and retuned using the updated 30-month in-sample interval, and new 6-month interval is used for out-of-sample testing. This procedure repeats until the end of the dataset is reached. This long evaluation horizon allows us to assess the robustness and stability of the models across different market regimes, including both bull and bear periods.

4.3 Model comparison

In our empirical analysis, we compare eight models: CNN, DNN, 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). DNN is the traditional deep 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 DNN, 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:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{40}$$

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:

$$\text{F1 Score} = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{41}$$

Table 1 shows results of Accuracy, F1 Score and function loss for all Models. We have following conculsions: First, All models except CNN 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 most deep learning and language models can identify useful signals from noisy market data for prediction. CNN is the only model with accuracy lower than 50% (48.80%). Second, our three language models (BERT, RoBERTa, FinBERT) exhibit higher overall accuracy and F1 score than traditional deep learning models (CNN, DNN, LSTM) in the classicfication problem. It is notable that DNN model can achieve 54.56% average accuracy, which is nearly equal to the result of BERT. Compared with language models, DNN model requires significantly less computing power than the language model, which means that the DNN model is a more feasible choice when computing power is limited. In our three language models, RoBERTa achieves the highest accuracy of 56.38%, with FinBERT and BERT achieving 55.23% and 55.40%, respectively. Third, fusion models with Attention mechanism (Fusion (AW)) achieve the highest results with 67.47% F1 score and 60.53% Accuracy, While Fusion models with equally weights (Fusion (EW)) just shows a result with 54.84%. 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, Accuracy and Loss for All Models

Model	**F1 Score**	**Accuracy**	**Loss**
CNN	58.70%	48.80%	0.45
DNN	63.25%	54.56%	0.39
LSTM	61.73%	53.62%	0.41
BERT	63.60%	55.40%	0.36
RoBERTa	64.77%	56.38%	0.31
FinBERT	64.26%	55.23%	0.33
Fusion (EW)	63.68%	54.84%	0.35
Fusion (AW)	67.47%	60.53%	0.28

4.5 Which data tell the truth

To evaluate the relative contribution of each modality within our multimodal framework, Table 2 presents an ablation analysis in which each input source is removed from the Fusion(AW) model. The results reveal several important patterns regarding the informativeness and complementarity of the modalities.

First, removing the image modality produces only a modest decline in performance: the model still attains an F1 score of 60.96% and an accuracy of 55.58%, outperforming Fusion(EW) and even most models in accuracy. This suggests that image-based features contribute relatively little incremental predictive power and may introduce noise or conflicts when combined with other modalities.

Second, the largest deterioration occurs when numerical time-series data are removed. Accuracy drops to 47.32%, barely above random guessing, highlighting that market data constitute the dominant and indispensable information source for price movement prediction. This finding further indicates that numerical signals form the backbone of the multimodal architecture and that other modalities alone cannot compensate for their absence.

Third, eliminating the financial news modality leads to a moderate performance decline. While the model still maintains a reasonable F1 score (57.56%), accuracy falls notably. This indicates that textual information enhances predictive stability and helps refine directional signals extracted from numerical data, even though it does not dominate the model's overall performance.

Taken together, the ablation results demonstrate that (i) numerical market data carry the most fundamental predictive content, (ii) financial news provides valuable complementary signals, and (iii) image-based representations offer limited marginal benefit in their current form and may introduce cross-modal conflicts. These insights suggest that future research should focus on improving the quality and integration of auxiliary modalities—particularly image-based features—while preserving the central role of numerical and textual information.

Table 2: Comparison of Accuracy and F1 for Different modality

Method	**Result**		
	F1	**Accuracy**	**Loss**
Fusion(AW)	**67.47**	**60.53**	**0.28**
w/o image	60.96	55.58	0.30
w/o time series	55.52	47.32	0.68
w/o financial news	57.56	52.37	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 3.05 sharpe ratio for equal-weighted long-short portfolios and 0.94 sharpe ratio for value-weighted long-short portfolios. For language models, RoBERTa has the best prformance with 2.76 sharpe ratio for equally-weighted long-short portfolios, followed by FinBERT model with 2.20 sharpe ratio and BERT model with 2.12 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 1.91 sharpe ratio for equally-weighted long-short portfolios, which is just slightly lower than language models.

4.7 Transaction Cost

The previous sections assessed model performance from a purely predictive perspective, abstracting from real-world trading frictions. Given that our strategies generate relatively high turnover, it is essential to incorporate transaction costs to evaluate their economic viability. To this end, we adopt the Exponentially-Weighted Calendar Time (EWCT) turnover-reduction mechanism proposed by Chen et al. (2022) and Ke et al. (2019), which constrains rebalancing intensity while allowing positions to decay smoothly over time. This framework provides a realistic approximation of how institutional investors attenuate excessive trading without discarding valuable predictive signals.

Table 4 reports the gross and net performance of EWCT portfolios under turnover limits ranging from $\gamma = 0.1$ to $\gamma = 0.9$. Several salient patterns emerge. First, gross returns increase monotonically with the turnover constraint, rising from 3.32 bps at $\gamma = 0.1$ to

Table 3: Performance of Portfolios for all models

	CNN				DNN			
	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	0.25	0.16	0.24	0.16	0.26	0.15	0.24	0.16
Sharpe Ratio	0.45	0.63	0.27	0.33	1.15	1.91	0.90	0.63

	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	0.35	0.21	0.12
Std	0.27	0.16	0.27	0.15	0.25	0.15	0.24	0.16
Sharpe Ratio	0.79	0.86	0.33	0.14	1.48	2.12	0.88	0.74

	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	0.30	0.35	0.20	0.11
Std	0.26	0.15	0.24	0.16	0.26	0.15	0.23	0.15
Sharpe Ratio	1.48	2.76	0.89	0.82	1.12	2.20	0.85	0.69

	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	0.10	0.42	0.50	0.21	0.15
Std	0.27	0.15	0.24	0.17	0.25	0.16	0.24	0.16
Sharpe Ratio	1.22	2.26	0.87	0.59	1.68	3.05	0.89	0.94

28.94 bps at $\gamma = 0.9$. This pattern reflects the intuitive fact that higher turnover enables more aggressive exploitation of short-lived predictive signals. Gross Sharpe ratios follow a similar but more muted trend, improving from 2.30 to 2.75 before stabilizing in the range of 2.38–2.75.

Second, once transaction costs are incorporated, the relationship between turnover and performance becomes non-linear. Net returns increase from 1.00 bps at $\gamma = 0.1$ to 7.51 bps at $\gamma = 0.9$, but the net Sharpe ratio peaks at $\gamma = 0.4$ with a value of 1.21 before declining. Specifically, net Sharpe decreases from 1.21 at $\gamma = 0.4$ to 1.07 at $\gamma = 0.8$, indicating that marginal gains from additional signal responsiveness are progressively offset by rising trading costs. This inverted-U pattern is consistent with the theoretical predictions in Chen et al. (2022), whereby excessive turnover erodes risk-adjusted profitability even when raw predictive power remains strong.

Taken together, these results suggest that moderate turnover constraints (around $\gamma = 0.4$) strike the optimal balance between harvesting predictive signals and mitigating trading frictions. While higher turnover continues to boost gross performance, the diminishing net Sharpe ratios highlight the practical limitations imposed by real-world trading costs. Therefore, incorporating turnover-control mechanisms such as EWCT is essential for translating statistical predictive accuracy into economically meaningful investment 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 to compare all models. The test is also used in Ao et al. (2018), Caner and Daniele (2025).

Table 4: Performance Analysis of Trading Strategies with Transaction Cost

Turnover Limit	**Turnover**	**Gross Return**	**Gross Sharpe**	**Net Return**	**Net Sharpe**
0.10	10.25	3.32	2.30	1.00	1.04
0.20	20.95	6.68	2.52	1.99	1.13
0.30	32.07	9.97	2.61	2.92	1.16
0.40	43.78	13.23	2.73	3.81	1.21
0.50	54.53	16.45	2.75	4.65	1.19
0.60	63.76	19.62	2.69	5.45	1.13
0.70	78.62	22.78	2.55	6.20	1.09
0.80	87.17	25.89	2.38	6.89	1.07
0.90	96.54	28.94	2.42	7.51	1.12

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 6. 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-

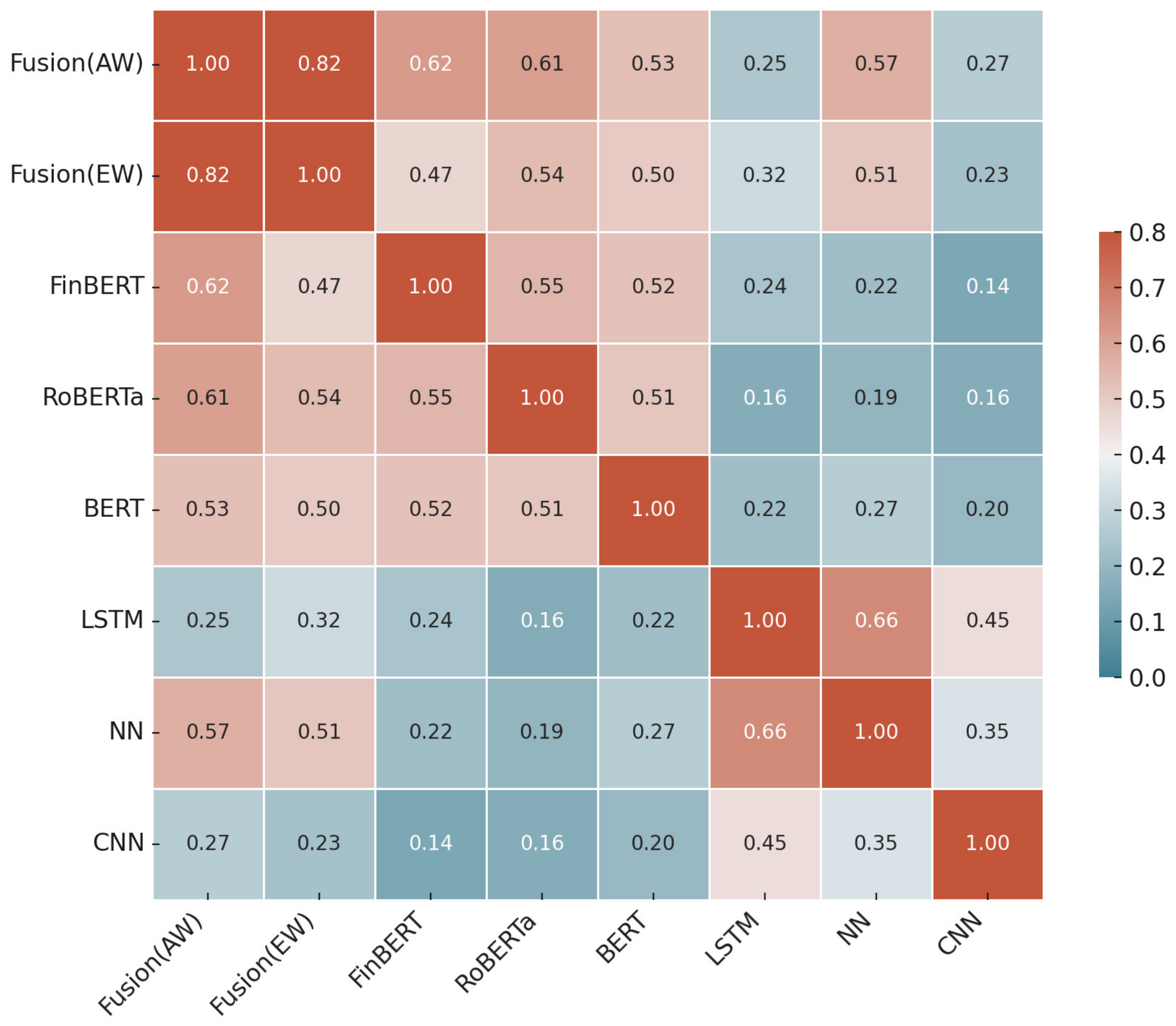
Table 5: P-value of the Jobson and Korkie test with Memmel correction for all models

	Long-short portfolios	Long-only portfolios
Fusion(AW)	-	-
CNN	0.026	0.002
DNN	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 DNN has a correlation with 0.66, which indicates that they both capture the temporal correlation of time series.

4.9 Cross–Regime Performance Comparison

To further evaluate the effectiveness and robustness of our proposed strategy, we extend our long-short strategy by examining its performance across several distinct market regimes. Specifically, we consider three recent and economically significant subperiods: (i) the COVID–19 pandemic downturn from February to October 2020, (ii) the geopolitical shock following Russia's invasion of Ukraine from January 2022 to December 2023, and



	Fusion(AW)	Fusion(EW)	FinBERT	RoBERTa	BERT	LSTM	NN	CNN
Fusion(AW)	1.00	0.82	0.62	0.61	0.53	0.25	0.57	0.27
Fusion(EW)	0.82	1.00	0.47	0.54	0.50	0.32	0.51	0.23
FinBERT	0.62	0.47	1.00	0.55	0.52	0.24	0.22	0.14
RoBERTa	0.61	0.54	0.55	1.00	0.51	0.16	0.19	0.16
BERT	0.53	0.50	0.52	0.51	1.00	0.22	0.27	0.20
LSTM	0.25	0.32	0.24	0.16	0.22	1.00	0.66	0.45
NN	0.57	0.51	0.22	0.19	0.27	0.66	1.00	0.35
CNN	0.27	0.23	0.14	0.16	0.20	0.45	0.35	1.00

Figure 6: Correlation of Returns for sentiment portfolios

(iii) the AI–driven expansion from January 2023 to December 2024. This setup ensures credible performance evaluation across heterogeneous market environments. The empirical results corresponding to these regime–specific evaluations are summarized as follows:

4.9.1 The COVID–19 pandemic downturn

Table 6: Performance of Portfolios during the COVID–19 pandemic downturn in 2020

	CNN		DNN		LSTM		BERT	
	EW	VW	EW	VW	EW	VW	EW	VW
Return	-0.14	-0.26	-0.09	-0.11	-0.07	-0.12	0.09	0.05
Std	0.47	0.45	0.45	0.42	0.51	0.47	0.42	0.39
Sharpe Ratio	-0.23	-0.42	-0.17	-0.11	-0.15	-0.26	0.26	0.17

	RoBERTa		FinBERT		Fusion(EW)		Fusion(AW)	
	EW	VW	EW	VW	EW	VW	EW	VW
Return	0.17	0.12	0.08	0.04	0.11	0.07	0.15	0.12
Std	0.46	0.45	0.45	0.42	0.51	0.47	0.42	0.39
Sharpe Ratio	0.34	0.25	0.19	0.07	0.26	0.14	0.30	0.22

Table 6 reports the portfolio performance of various models during the COVID–19 pandemic downturn in 2020, a period characterized by extreme volatility and rapid information flow. As expected, traditional deep learning models that rely primarily on price or image-based features—such as CNN, DNN, and LSTM—exhibit negative returns with uniformly negative Sharpe ratios, indicating limited ability to navigate abrupt market dislocations driven by macroeconomic uncertainty.

In contrast, language-model–based strategies demonstrate markedly stronger resilience. Both BERT and FinBERT generate positive returns under equal- and value-weighted schemes, suggesting that textual information provides more timely signals of market sentiment and risk. Notably, RoBERTa delivers the strongest overall performance, achieving a positive Sharpe ratio of 0.34 under the equal-weighted portfolio, the highest among all models considered. This superior performance underscores the advantage of pretrained language models in capturing real-time market dynamics during crisis conditions.

Furthermore, fusion models that combine multimodal information also perform competitively, with Fusion(EW) and Fusion(AW) producing Sharpe ratios comparable to or exceeding those of individual language models. These findings collectively indicate that news-derived textual features carry substantial incremental predictive power during systemic stress episodes, whereas models relying solely on historical prices or images are more vulnerable to trend breakdowns and regime shifts.

4.9.2 Russia's invasion of Ukraine in 2022-2023

Table 7: Performance of Portfolios during the Russia's invasion of Ukraine in 2022-2023

	CNN		DNN		LSTM		BERT	
	EW	VW	EW	VW	EW	VW	EW	VW
Return	0.22	0.17	0.28	0.19	0.27	0.18	0.31	0.25
Std	0.28	0.25	0.24	0.21	0.27	0.27	0.23	0.24
Sharpe Ratio	0.88	0.63	1.04	0.82	0.91	0.75	1.20	1.04
	RoBERTa		FinBERT		Fusion(EW)		Fusion(AW)	
	EW	VW	EW	VW	EW	VW	EW	VW
Return	0.36	0.24	0.40	0.27	0.41	0.30	0.47	0.33
Std	0.26	0.22	0.24	0.23	0.31	0.32	0.24	0.26
Sharpe Ratio	1.39	1.11	1.45	1.15	1.34	0.96	1.87	1.22

Table 7 presents the portfolio performance of all models during the period surrounding Russia's invasion of Ukraine in 2022–2023, a regime marked by heightened geopolitical risk, supply-chain disruptions, and persistent commodity-price shocks. Across all model classes, returns are positive and Sharpe ratios are substantially higher than during the COVID–19 downturn, reflecting a market environment in which directional trends and risk premia were more stable despite elevated uncertainty.

Among price- and sequence-based deep learning approaches, DNN and LSTM exhibit stronger performance than CNN, achieving Sharpe ratios above 1.0 under the equal-weighted specification. This suggests that models incorporating temporal dependencies are better suited to capturing the medium-term drift present during this geopolitical shock.

Language-model–based strategies again outperform their image- and time-series–based counterparts. BERT, RoBERTa, and FinBERT all generate Sharpe ratios exceeding 1.1 under at least one weighting scheme, with FinBERT slightly outperforming RoBERTa in equal-weighted returns. These results imply that text-derived sentiment and news-flow indicators provide valuable real-time signals during periods of rapid geopolitical developments.

The strongest performance is delivered by the multimodal fusion models, particularly Fusion(AW), which attains a Sharpe ratio of 1.87 under the equal-weighted portfolio—the highest among all models. This superior outcome highlights the benefit of integrating heterogeneous sources of information, as the fusion architecture captures not only the evolving narrative embedded in news text but also the accompanying movements in market prices.

Overall, the evidence indicates that during the Russia–Ukraine conflict, models leveraging textual information or combining multiple modalities were particularly effective in navigating market conditions shaped by geopolitical uncertainty.

Table 8: Performance of Portfolios during AI–driven expansion since 2023

	CNN		DNN		LSTM		BERT	
	EW	VW	EW	VW	EW	VW	EW	VW
Return	0.40	0.26	0.37	0.27	0.42	0.35	0.52	0.40
Std	0.23	0.22	0.22	0.21	0.23	0.20	0.21	0.22
Sharpe Ratio	1.84	1.26	1.68	1.23	2.01	1.65	2.91	1.97
	RoBERTa		FinBERT		Fusion(EW)		Fusion(AW)	
	EW	VW	EW	VW	EW	VW	EW	VW
Return	0.65	0.45	0.62	0.47	0.53	0.42	0.70	0.51
Std	0.21	0.20	0.22	0.22	0.21	0.22	0.21	0.21
Sharpe Ratio	3.03	2.31	2.76	2.12	2.54	1.98	3.26	2.37

4.9.3 AI–driven expansion since 2023

Table 8 summarizes the model performance during the AI–driven market expansion since 2023, a period characterized by strong secular momentum in technology and semiconductor sectors, record earnings growth among AI–exposed firms, and persistent investor optimism surrounding generative AI advancements. Compared with earlier market regimes, all models deliver substantially higher returns and Sharpe ratios, reflecting the trend–dominated nature of this environment.

Within the class of traditional deep learning models, LSTM exhibits the strongest performance (Sharpe ratio of 2.01 under equal weighting), consistent with its ability to capture persistent upward drift and long–horizon price trends that typified the AI boom. CNN and DNN also show improved performance, though their Sharpe ratios remain below those of sequential or language–based models.

Language models again outperform other model families by a considerable margin. BERT, RoBERTa, and FinBERT achieve exceptionally strong Sharpe ratios—all above

2.0 under at least one weighting scheme—indicating that news–based textual signals have become increasingly informative in an environment where corporate announcements, AI product launches, and regulatory commentary exert immediate influence on market expectations.

Fusion models produce highly competitive outcomes, with the attention–weighted fusion model (Fusion(AW)) delivering the highest Sharpe ratio of 3.26 under equal weighting. This superior performance highlights the benefits of integrating heterogeneous data sources during a period when both market prices and textual narratives jointly shaped investor sentiment and asset valuations. The evidence suggests that during the AI–driven expansion, strategies leveraging language models and multimodal architectures were particularly effective in exploiting sustained sectoral momentum and fast–evolving information flows.

Across the three major market regimes examined—the COVID–19 crash (2020), the Russia–Ukraine geopolitical shock (2022–2023), and the AI–driven expansion (2023–present), our results reveal a consistent hierarchy in model performance. Price- and image–based deep learning models (CNN, DNN, LSTM) perform poorly during abrupt regime shifts such as the COVID–19 downturn but recover notably once market trends stabilize, with our model showing the greatest adaptability across environments. Language–model strategies (BERT, RoBERTa, FinBERT) display far stronger robustness, uniquely producing positive returns during the COVID–19 crash and maintaining superior Sharpe ratios throughout the geopolitical shock and the AI boom, underscoring the value of news–based signals for capturing rapid shifts in sentiment. Multimodal fusion models consistently deliver the strongest performance overall, with Fusion(AW) outperforming all standalone modalities across periods, demonstrating that integrating heterogeneous information sources yields complementary predictive power resilient to diverse market conditions. Collectively, these findings highlight the structural robustness of our multimodal and language–based framework, which performs favorably in crisis environments, geopolitical stress episodes, and sustained bull markets alike.

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 represents 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.

Besides, our model's performance across three major market regimes: the COVID-19 crash, the Russia–Ukraine conflict, and the AI-driven expansion. These out-of-sample regime analyses reveal a persistent hierarchy in model effectiveness: price- and image-based deep learning models struggle during abrupt regime shifts, while language-model–based and multimodal fusion approaches exhibit strong resilience in both crisis conditions and trend-dominated markets. The superior robustness of multimodal architectures underscores the importance of integrating diverse data sources to capture rapidly evolving market dynamics.

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. (2025). Deep learning based residuals in non-linear factor models: Precision matrix estimation of returns with low signal-to-noise ratio. *Journal of Econometrics*, 251:106083.

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. (1994). Time series analysis.

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.

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, 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.