Junior Management Science

www.jums.academy
ISSN: 2942-1861

Predicting Stock Market Trends Using Convolutional Neural Networks: A Deep Learning Approach

Ege Özkul

Technical University of Munich

Abstract

Technical analysis aims to predict stock returns based on price and volume patterns and has seen growing adoption of machine learning methods. However, most approaches rely on hand-crafted features. This paper investigates whether deep learning applied to stock chart images can predict future returns without manual feature engineering. Building on Jiang et al. (2023), this study applies Convolutional Neural Networks (CNNs), a computer vision architecture, to predict stock returns from price charts and extends their approach by implementing Vision Transformers, specifically the Class-Attention in Image Transformer (CaiT). Stock prices, volumes, and moving averages are encoded into images, which are used to train models that classify future stock returns as either positive or negative. Results show that both CNN and CaiT models outperform traditional technical indicators such as momentum and reversal strategies when applied to US stocks. Moreover, combining the two models yields incremental predictive power. An investment strategy based on their joint predictions achieves higher returns and Sharpe ratios than either model alone.

Keywords: technical analysis; stock return prediction; deep learning; convolutional neural networks; vision transformers

1 Introduction

Technical analysis is a method used to forecast future asset price movements based on the examination of past market data, primarily price and volume (Brock et al., 1992, p. 1731). Rather than relying on economic fundamentals, technical analysis identifies patterns and trends in historical price behavior, assuming that market movements exhibit recurring and predictable structures. This approach can be applied qualitatively, through the interpretation of recurring price formations on charts, or quantitatively, using technical indicators such as moving averages, which smooth out price fluctuations over time to reveal underlying trends (Menkhoff & Taylor, 2007, p. 939). This reliance on historical price patterns and market trends directly contradicts the Efficient Market Hypothesis (EMH), particularly its weak form (Murray et al., 2024, p. 1). The weak form EMH suggests that price changes follow a random walk, meaning that past price movements do not provide any predictive power for future returns (Fama, 1970, p. 389). In contrast, technical analysis assumes that historical price behavior influences future price movements, which implies a degree of market inefficiency.

Technical analysis is widely used in the financial industry (Zhu & Zhou, 2009, p. 519). Most brokerages provide charting analysis, making technical tools more accessible to market participants (Han et al., 2021, p. 1). Menkhoff and Taylor (2007) highlight its widespread adoption among foreign exchange traders, while (Lo & Hasanhodzic, 2010) examine how professional practitioners implement technical strategies in real-world trading. However, despite its extensive use in the industry, academics have traditionally doubted the validity of technical analysis, often questioning its reliability in predicting asset prices. Some even compare it to astrology or dismiss it as "voodoo finance," emphasizing the skepticism surrounding its legitimacy (Lo et al., 2000, p. 1705). Nevertheless, several studies have provided strong empirical evidence supporting technical analysis such as (Brock et al., 1992; Jegadeesh & Titman, 1993; Lo et al., 2000) and (Zhu & Zhou, 2009).

More recently, machine learning models have been increasingly applied in technical analysis. One of the biggest challenges in utilizing these techniques is the need for feature engineering, as predictive signals must be manually designed before they can be effectively used (Jiang et al., 2023, p. 3193). To address this limitation, (Jiang et al., 2023), in their paper "(Re-)Imag(in)ing Price Trends" propose a novel approach that converts stock data into images and applies Convolutional Neural Networks (CNNs) to automatically learn predictive features without manual intervention. Their findings show that image-based predictive features significantly enhance return predictions compared to traditional trend signals. The CNN models identified previously unexplored price patterns that proved

I would like to thank Professor Dr. Christoph Kaserer and Minghui Chen for their valuable guidance and support throughout the preparation of this thesis.

DOI: https://doi.org/10.5282/jums/v11i1pp195-226

more effective in forecasting returns, ultimately leading to superior portfolio performance.

This paper replicates their study by implementing the proposed CNN models and extends it by integrating a state-of-the-art image recognition architecture: Vision Transformers (ViTs; (Dosovitskiy et al., 2020)). ViTs are built on the Transformer architecture, which is also used in OpenAI's ChatGPT (Raschka, 2024, p. 6). Unlike CNNs, which process images by detecting local patterns such as edges and textures, ViTs divide images into patches and use a self-attention mechanism to learn relationships between different regions of the image. This allows ViTs to capture long-range dependencies and global context within an image, making them particularly effective for tasks requiring a broader understanding of spatial relationships (Maurício et al., 2023, p. 13).

Jiang et al. (2023) argue that one of the reasons for representing time series as images is that it allows models to capture structural relationships that may be challenging to identify using conventional time-series approaches, much like how humans recognize trends more easily in visual graphs than in raw numerical data. By utilizing this structured format, an advanced pattern detection model such as CNN can process the entire visual representation, enabling it to uncover hidden dependencies and trends that might otherwise go unnoticed Jiang et al. (2023, p. 3194). Building on this logic, a model designed to capture broader contextual relationships across an entire image may extract even more meaningful insights. Given that CNNs specialize in detecting local features such as edges and textures, they excel at identifying small-scale visual patterns within stock price images. However, ViTs may further enhance pattern detection in stock charts. By analyzing the image holistically rather than focusing on localized features, ViTs could improve the model's ability to recognize overarching trends and structural formations in stock charts. This paper tests whether the image-based approach introduced by (Jiang et al., 2023) can be further enhanced using ViTs.

Following (Jiang et al., 2023), I converted US stock data from 1993 to 2019 into 5-day, 20-day, and 60-day images. Each image is in grayscale format, incorporating price data, trading volume, and moving averages. For the ViT models, I adapted the images to an RGB format, where each color channel represents either price data, volume, or moving averages. Jiang et al. (2023) trained models on three image types (5-day, 20-day, and 60-day) to predict whether stock returns would be positive or negative over the subsequent week, month, or quarter, resulting in nine models. In my implementation, I trained eight models, omitting the 60-day image model for weekly return predictions due to storage constraints. For the ViTs, I trained only three models: two predicting weekly returns using 5-day and 20-day images, and one predicting quarterly returns using 5-day images. Given the high computational cost of training ViTs, I limited their implementation to these three cases. I utilized the Class-Attention in Image Transformers (CaiT) architecture proposed by (Touvron et al., 2021), a specialized variant of ViT that allows for deeper network architectures. Unlike standard ViTs, CaiT first learns the relationships within the image and then leverages these learned relationships to make predictions (Touvron et al., 2021). Additionally, ViTs lack inductive bias which makes them difficult to train from scratch or with small datasets. To address this challenge, I applied two techniques proposed by (S. H. Lee et al., 2021).

Both CNN and CaiT models generate probability estimates indicating the likelihood of a stock having a positive return over the prediction period. Following (Jiang et al., 2023), stocks are sorted into decile portfolios based on these predictions, and the portfolios are rebalanced at the same frequency as the prediction horizon to ensure consistency in evaluation. The performance of CNN-based strategies is assessed against traditional trend following signals, including 2-12 momentum (MOM), monthly short-term reversal (STR), and weekly short-term reversal (WSTR). Strategy performance is measured using a High-Low (H-L) portfolio and Sharpe ratios. Furthermore, I compare the performance of CaiT-based strategies with their CNN-based counterparts.

After comparing the performance of CaiT and CNN strategies, I examined their potential for integration. Since each model may focus on different features within the same image, combining them could yield superior results. To test this hypothesis, I first calculated the simple average of both models' probability estimates and sorted stocks into deciles based on these averages. This averaging process introduces an element of agreement in the High (highest decile or decile 10) and Low (lowest decile or decile 1) deciles, ensuring that stocks assigned a high probability by both models are positioned in the High decile, while those with low probabilities fall into the Low decile. Analyzing these deciles along with the H-L portfolio provides insights into the combined behavior of the models.

Building on this consensus-based approach, I propose an investment strategy that leverages both models by selecting stocks where they agree most strongly. I evaluate the strategy by reporting annualized returns and Sharpe ratios, assessing its resilience to transaction costs, and analyzing its cumulative returns over the out-of-sample period. This study also investigates whether CNNs and ViTs can be effectively integrated to enhance stock return predictions. To the best of my knowledge, no prior research has explored the use of stock charts as inputs for ViTs to predict stock returns or attempted to combine ViTs and CNNs for improved return prediction performance.

The remainder of this thesis is structured as follows: Section 2 reviews the relevant literature on Technical Analysis and Computer Vision. Section 3 introduces the fundamental concepts of CNNs and ViTs, detailing the architectural design choices. Section 4 outlines the data sources and methodology employed in implementing both architectures. Section 5 presents the empirical results for CNN and CaiT models, including the effects of averaging their predictions. Section 6 proposes the investment strategy derived from these models. Section 7 discusses the limitations of the study and provides a critical evaluation of the findings. Finally, Section 8 concludes with key insights.

2 Literature Review

This paper explores the intersection of technical analysis, Convolutional Neural Networks, and Vision Transformers. It investigates how CNN-based image analysis can enhance technical analysis and then applies ViTs to assess whether stock prediction can be further improved using an alternative image recognition algorithm. Accordingly, this section reviews the relevant literature from two perspectives: first, technical analysis, and second, computer vision.

Technical analysis has a long history, with its origins traced back to Charles Dow in the late 1800s in the United States Brock et al. (1992, p. 1731). Dow developed what became known as Dow Theory, which aimed to predict market trends by identifying upward or downward price movements Zhu and Zhou (2009, p. 521). However, early empirical studies cast doubt on the effectiveness of technical analysis. Cowles (1933) conducted one of the first major empirical tests, analyzing the forecasts of William Peter Hamilton, a leading Dow Theory advocate. Cowles (1933) found that Hamilton's predictions between 1904 and 1929 underperformed a simple buy-and-hold strategy, suggesting that technical forecasting lacked predictive power. Later studies reinforced this skepticism—(Fama & Blume, 1966) tested 24 filter rules and found that they did not outperform passive investing, while (Jensen & Benington, 1970) examined Levy (1967)'s Relative Strength Index (RSI) and found that it did not yield superior returns once transaction costs were accounted for. The Efficient Market Hypothesis, introduced by (Fama, 1970), further challenged technical analysis by asserting that past prices contain no useful information for predicting future movements—a direct contradiction to the principles of technical analysis. As a result, early studies were generally skeptical of technical analysis, casting doubt on its ability to generate consistent excess returns.

Despite early skepticism, more recent research has provided empirical evidence supporting the effectiveness of technical analysis in financial markets. Lo et al. (2000) developed a quantitative framework to systematically detect technical patterns and rigorously tested them, finding that technical analysis can offer incremental information beyond random price movements, suggesting that market inefficiencies may exist in the short term. Jegadeesh and Titman (2001) found that momentum strategies remain highly profitable and are not a product of data snooping. Their results indicate that systematic mispricing, driven by behavioral biases, can create exploitable trading opportunities. Similarly, (Neely et al., 2014) examined the relationship between technical indicators and the equity risk premium, finding that technical analysis and macroeconomic variables capture different aspects of market dynamics. Their research shows that combining the two yields superior out-of-sample forecasts, challenging the traditional view that only fundamental data is valuable. More recently, (Murray et al., 2024) tested the weak-form Efficient Market Hypothesis using machine learning (ML) models trained on historical price data. Their findings suggest that ML-based technical analysis strategies generate significant excess returns, which cannot be fully explained by risk factors, further supporting the idea that technical patterns may contain predictive power. Collectively, these studies indicate that there is strong empirical evidence supporting the effectiveness of technical analysis.

Moving averages (MAs) serve as the foundation of trend-following strategies, which are widely used in financial markets Han et al. (2021, p. 1). One of the first major studies on MAs, conducted by (Brock et al., 1992), analyzed 90 years of Dow Jones Industrial Average data and found that MAs and Trading Range Breakouts exhibited statistically significant predictive power. However, (Sullivan et al., 1999) revisited this finding, testing whether MAs truly predict stock returns or if their past performance was due to data-snooping bias. Their results showed that while MAs appeared profitable in historical data, they lost predictive power in out-of-sample tests. Despite this skepticism, later studies still found empirical support for MAs. Zhu and Zhou (2009) examined MAs from an asset allocation perspective and concluded that they help investors make better allocation decisions, especially in uncertain market conditions. Han et al. (2016) introduced the Trend Factor, a model that integrates MAs across different time horizons to capture persistent price trends, making them particularly effective during market uncertainty. More recently, (Detzel et al., 2021) examined MAs in markets where fundamental valuation is difficult, such as Bitcoin and speculative stocks, and found that MA-based strategies generate significant excess returns, particularly in cryptocurrencies. These studies suggest that MAs remain a valuable tool for traders and investors.

In addition to traditional technical analysis methods, recent research has increasingly incorporated advanced machine learning techniques such as Support Vector Machines (SVMs) and Genetic Algorithms (GAs) to improve predictive accuracy in financial markets. SVMs, first introduced by (Noble, 2006), classify data by finding an optimal hyperplane, making them effective in financial forecasting. Cao and Tay (2003) demonstrated that SVMs with adaptive parameters outperform Backpropagation Neural Networks when predicting futures contracts from the Chicago Mercantile Market. M. Lee (2009) further refined SVM-based forecasting by integrating a hybrid feature selection method with financial data from NASDAQ, 20 futures contracts, and 9 spot indexes, achieving superior accuracy and generalization. Additionally, (Kercheval & Zhang, 2015) leveraged high-frequency limit order book data from NASDAQ to build multi-class SVM models, successfully predicting short-term price movements and generating profitable trading signals in real-time. Beyond SVMs, Genetic Algorithms, first developed by (Holland, 1975), offer an evolutionary approach to financial forecasting by optimizing trading strategies through natural selection principles. Allen and Karjalainen (1999) pioneered the use of GAs in technical analysis, applying them to develop trading rules for the S&P 500, though results indicated they did not consistently outperform a buy-and-hold strategy after transaction costs. More recent studies have improved GA-based approaches: (Brown et al.,

2013) introduced the Dynamic-Radius Species-Conserving Genetic Algorithm to select stocks from the Dow Jones Index, significantly outperforming the index in returns. Y. Hu et al. (2015) combined GAs with the eXtended Classifier System in their eTrend model, which adapts trading rules dynamically, outperforming buy-and-hold strategies with superior risk-adjusted returns.

Neural networks have become a widely used approach in financial forecasting due to their ability to model complex, nonlinear relationships in stock price data. Tsang et al. (2007) developed NN5, a Backpropagation Neural Network to predict HSBC stock price movements in the Hong Kong market, achieving over 70% accuracy, though profitability was limited by transaction costs. Similarly, (Guresen et al., 2011) compared various Artificial Neural Network architectures for stock market index prediction, and found that the Multi-Layer Perceptron produced the most accurate and stable forecasts for NASDAQ index movements. Further advancing the field, (Nelson et al., 2017) applied Long Short-Term Memory (LSTM) networks to predict stock price movements on the Brazilian stock exchange using historical price data and technical indicators, demonstrating LSTM's superiority over traditional machine learning models, with an accuracy of up to 55.9% in short-term price direction prediction. These studies highlight the growing role of neural networks in technical analysis, offering improved predictive power and adaptability compared to conventional methods.

Recent research has explored the application of CNNs in financial market prediction, with some studies integrating them with other models while others use CNNs independently. Kim and Kim (2019) developed a hybrid LSTM-CNN model, showing that combining numerical and image-based stock data outperforms standalone CNNs or LSTMs. Similarly, (Di Persio & Honchar, 2016) introduced a Wavelet-CNN model, demonstrating that wavelet preprocessing enhances feature extraction and that ensemble learning further improves the accuracy of stock market price prediction. J. Lee et al. (2019) applied CNNs within a Deep Q-Network, demonstrating that U.S.-trained CNNs generalize well to global markets, while (Hoseinzade & Haratizadeh, 2019) found that multi-market CNNs outperform single-market models for stock index movement prediction. In contrast, other studies relied solely on CNNs, using either raw numerical data or transformed financial images. Gunduz et al. (2017) enhanced CNN accuracy by structuring input features according to their correlation, and (G. Hu et al., 2018) used Convolutional AutoEncoders to extract deep features from candlestick charts, demonstrating better stock clustering for portfolio optimization. Similarly, (Cohen et al., 2020) used CNN-based image classification to replicate human technical analysis, achieving 95% accuracy in detecting buy signals, while (Chen et al., 2016) showed that CNNs outperformed traditional market models in time-series forecasting. However, (Jiang et al., 2023) provided the most comprehensive demonstration of CNNs' predictive power, transforming stock price trends into images and proving that CNNs outperform traditional momentum and reversal strategies, generalize across time scales, and successfully transfer from U.S. to global markets.

Convolutional Neural Networks have played a crucial role in the advancement of image recognition. Introduced by (Lecun & Bengio, 1995), CNNs leverage spatial hierarchies and local connectivity to effectively process image data. Over the years, several architectures have significantly contributed to the field. AlexNet, proposed by (Krizhevsky et al., 2012), was a groundbreaking deep CNN that won the ImageNet competition in 2012, demonstrating the power of deep learning in large-scale image classification. It featured multiple convolutional layers, max-pooling, and the use of ReLU activations to speed up training and improve accuracy. GoogleNet, developed by (Szegedy et al., 2014), introduced the Inception module, which optimized computational efficiency while increasing network depth and width, allowing for multi-scale feature extraction. ResNet, proposed by (He et al., 2016a), addressed the vanishing gradient problem in deep networks by introducing residual connections, enabling the training of networks with over 100 layers while maintaining high accuracy. These architectures have set benchmarks for deep learning-based image analysis and continue to inspire modern developments in computer vision.

More recently, Vision Transformers have emerged as a powerful alternative to CNNs for image recognition. Inspired by the Transformer architecture originally proposed by (Vaswani et al., 2017) for natural language processing, ViTs were introduced by (Dosovitskiy et al., 2020) as a model that treats images as sequences of patches, similar to words in a text sequence. Unlike CNNs, which rely on convolutional layers to extract local features, ViTs use self-attention mechanisms to model global dependencies within an image, allowing them to capture long-range interactions Maurício et al. (2023, p. 13). However, early ViTs required large-scale datasets for effective training, which led to the development of optimized versions such as Data-efficient Image Transformers (DeiT) by (Touvron et al., 2020), which introduced a knowledge distillation approach to make ViTs more accessible with limited data. Other variations like Pooling-based Vision Transformers (Heo et al., 2021) incorporated spatial dimension reduction inspired by CNNs to improve efficiency, and Class-Attention in Image Transformers by (Touvron et al., 2021) introduced specialized attention mechanisms to enhance classification accuracy. These advancements have solidified ViTs as a competitive alternative to traditional CNNs in image analysis tasks, pushing the boundaries of deep learning in computer vision.

To address the limitations of ViTs on small datasets, various strategies have been proposed to enhance their performance without extensive pre-training on large-scale datasets. One such approach by (S. H. Lee et al., 2021) introduces Shifted Patch Tokenization and Locality Self-Attention to improve locality inductive bias, making ViTs more effective for small-scale datasets. SPT increases the receptive field of tokens, allowing ViTs to capture more spatial relationships between neighboring pixels, while LSA

refines the attention mechanism to focus more on relevant local features rather than distributing attention uniformly. Similarly, (Shao & Bi, 2022) propose a hybrid transformer model that integrates Convolutional Parameter Sharing Multi-Head Attention and Local Feed-Forward Networks, effectively combining the strengths of CNNs and transformers. Their method incorporates convolutional layers within the transformer structure, improving local feature extraction while maintaining the long-range dependencies of self-attention. Additionally, (Gani et al., 2022) present a self-supervised weight initialization approach, leveraging low-resolution feature prediction to pre-train ViTs on small datasets without relying on external large-scale datasets. These advancements contribute to making ViTs more practical for small-scale vision tasks, reducing dependency on massive datasets while improving performance and efficiency.

3 Theoretical Background

In this section, I will discuss the theoretical background of CNNs and ViTs. This section has two main purposes: first, to introduce these algorithms, explain their fundamental mechanics, and highlight the key ideas behind them; second, to illustrate how they apply to stock return prediction and the rationale behind my design choices.

I will begin with CNNs, explaining their general structure and core concepts. Next, I will discuss Transformers, the foundation of Vision Transformers, and explain the self-attention mechanism from a natural language processing perspective. Then, I will describe how this mechanism is adapted to images by outlining the standard ViT architecture and its potential to improve pattern recognition in stock charts. Following this, I will introduce the specific CaiT architecture and justify its selection for this task. Finally, the chapter will conclude with an explanation of two techniques that enable ViTs to be trained from scratch on stock image datasets, making their application feasible for this study.

3.1 CNN Background

Convolutional Neural Networks are a specialized class of neural networks designed to process data with grid-like structures, such as images Goodfellow et al. (2016, p. 330). In a standard CNN layer, there are three key stages: the convolution stage, involves applying learned filters to small sections of the input data; the activation stage, which applies a non-linear function to introduce non-linearity; and the pooling stage, groups nearby values together, simplifying the feature maps and helping the network ignore small changes in the input Goodfellow et al. (2016, p. 339). Furthermore, CNNs leverage two fundamental principles: parameter sharing, which allows the same filter to be used across different regions, reducing both the number of parameters and the amount of computation required; and translation equivariance, meaning that shifts in the input lead to corresponding shifts in the output, thereby maintaining the spatial consistency of the detected features Jiang et al. (2023, p. 3240).

In the convolution operation, a small matrix of learned parameters, known as a kernel or filter, slides over the input data to compute weighted sums at every position, thereby detecting local patterns such as lines or edges Goodfellow et al. (2016, p. 335). This process is used to transform raw input into a more informative representation that highlights important features while preserving spatial relationships Jiang et al. (2023, p. 3238). Figure 1 demonstrates this process: each element of the kernel (labeled w, x, y, z) is multiplied by corresponding input values (labeled a, b, c, etc.) within a specific patch, and these products are summed to yield a single output value. As the kernel moves across the entire input, it produces an output map that indicates where the feature appears and with what intensity. Importantly, the weights of these filters are learned during training, enabling the network to adapt to the specific patterns present in the data, and in practice, many different filters are applied—not just one—to capture a wide variety of features essential for effective analysis (Goodfellow et al., 2016, pp. 333–336).

Following the convolution operation, an activation function is applied element-wise to the resulting feature maps to introduce non-linearity Jiang et al. (2023, p. 3240). This non-linearity is crucial because, without it, the entire network would collapse into a simple linear model, irrespective of its depth Goodfellow et al. (2016, p. 172). By incorporating non-linearity, neural networks gain the ability to approximate any function, enabling them to capture intricate relationships between inputs and outputs when given sufficient layers. This transformation allows networks to convert linear input data into highly complex, non-linear outputs, making them effective for modeling complex structures that would be unattainable with a purely linear approach Goodfellow et al. (2016, p. 198). One of the most common activation functions is the Rectified Linear Unit (ReLU; Goodfellow et al. (2016, p. 193)), which is defined as:

$$f(x) = \max(0, x) \tag{1}$$

Following the convolution and activation steps, the pooling stage aggregates information from local regions of the activation maps to simplify the overall representation. In particular, max pooling partitions the feature maps into small, non-overlapping regions and selects the highest value from each region Goodfellow et al. (2016, p. 339). This operation reduces the spatial dimensions of the data, lowering computational complexity for subsequent layers, while also emphasizing the most prominent features. By doing so, max pooling makes the network more robust to slight shifts or distortions in the input, ensuring that key patterns are retained even if their exact locations change slightly Jiang et al. (2023, p. 3241).

As mentioned earlier, the convolution operation involves sliding a filter over the input data, and the same filter is used

[1] taken from Goodfellow et al. (2016, p. 334)

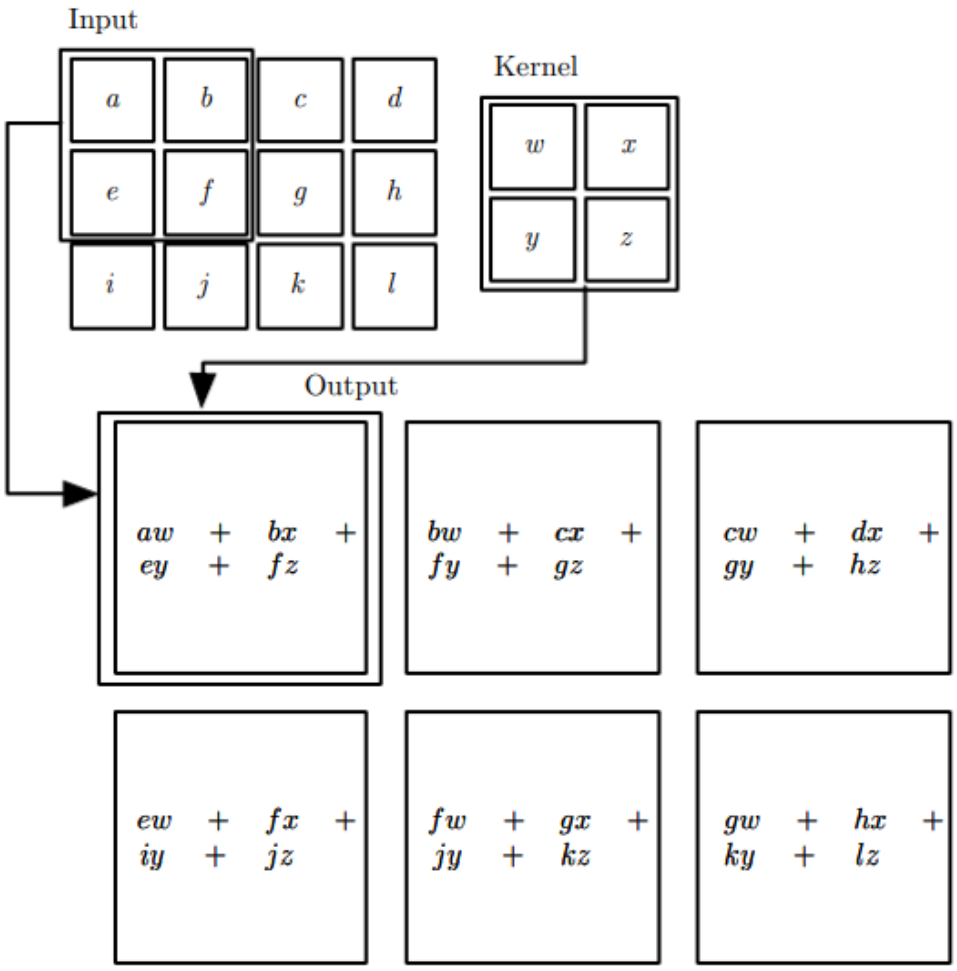

Figure 1: Convolution Operation Example[1]

Description. This figure illustrates the convolution operation in a neural network, where a 2D input matrix is processed using a small kernel (filter) to produce an output feature map. The input consists of a 4×3 matrix, while the kernel is a 2×2 matrix. The kernel slides over the input matrix, computing element-wise multiplications followed by summations to generate output values. Each resulting value represents a weighted sum of overlapping regions between the input and the kernel.

at every location. This technique is known as parameter sharing. The reuse of filter weights means that the network does not have to learn separate parameters for every region of the input, which significantly reduces the total number of parameters Goodfellow et al. (2016, p. 335). Not only does this make the model more memory- and computation-efficient, but it also reinforces translation equivariance—meaning that when an input is shifted, the output shifts in the same way Goodfellow et al. (2016, p. 338). This property ensures that if a feature is detected in one part of the image, it will still be recognized even if it appears in a different location, which is highly desirable for tasks like image recognition. Translation equivariance contributes to the robustness and generalization of the network by enabling consistent feature detection regardless of the location Jiang et al. (2023, p. 3240).

Now that we have examined key concepts of CNNs, an important consideration is the distinction between local and global features in images. Through the convolution operation, CNNs excel at detecting local features by analyzing small parts of an image, and due to translation equivariance, they can recognize these features—like eyes—regardless of their position Goodfellow et al. (2016, p. 342). However, capturing global features is more challenging. In a typical CNN, different sections of an image are not directly connected; instead, they are gradually linked through successive convolution layers Peng et al. (2023, p. 9455). Each convolution reduces the size of the output relative to its input, and although subsequent convolutions cover larger areas of the image, this information is aggregated indirectly Goodfellow et al. (2016, p. 337). Moreover, the application of max pooling further diminishes spatial information, as it selects only the maximum value from a given region, thereby losing precise details about where that value originated Goodfellow et al. (2016, p. 347). For traditional image recognition tasks,

such as face recognition, this trade-off in spatial precision is often acceptable. Instead of recognizing an entire face as a structured entity, CNNs can identify and classify faces based on the presence of key local features, such as eyes, nose, and mouth, and their approximate locations. The network does not require an exact representation of a face to correctly classify an image as containing a face Goodfellow et al. (2016, p. 342). However, in the context of stock charts, where the location of a feature also represents a specific time, preserving this spatial (and temporal) information becomes crucial. For example, from a technical analysis perspective, it is really important to detect the intersection of the moving average line and close price, but also where this intersection happened is also important. Recognizing different parts of a chart, such as the beginning and end of a period, is crucial, as they may convey distinct information. This understanding led me to implement an alternative image recognition model to better capture global patterns.

3.2 Transformer Background

Vision Transformers are image analysis models built on the Transformer architecture, sharing the fundamental mechanism of self-attention. While Transformers were originally developed for natural language processing (NLP), ViTs adapt this architecture to process images as input. Beyond this key difference, the core principles remain largely the same. Therefore, in this section, I will first introduce Transformers and the self-attention mechanism from a natural language perspective. This foundational understanding will serve as a basis for the following section, where I will explain Vision Transformers by building upon the concepts introduced here.

Transformers represent a major breakthrough in deep learning, having fundamentally reshaped the field of NLP since their introduction by (Vaswani et al., 2017). Through an attention mechanism, the Transformer architecture combines the computational parallelism of Convolutional Neural Networks with the ability to model long-range dependencies and process variable-length sequences, which provides an advantage for the long texts—a strength traditionally associated with Recurrent Neural Networks (RNNs). This innovative approach not only boosts efficiency during training but also leads to superior performance across various tasks Kamath et al. (2022, p. 19).

Before a Transformer model can process text, it must first convert raw language into a numerical format that neural networks can understand. Text cannot be directly processed mathematically for training deep learning models because it consists of categorical data. To bridge this gap, words and symbols are represented as continuous-valued vectors through a process known as embedding Raschka (2024, p. 25).

Before embedding occurs, an essential preprocessing step called tokenization is performed. Tokenization breaks the input text into discrete units, known as tokens, which can be individual words, subwords, or special characters, including punctuation Raschka (2024, p. 27). These tokens are then converted into numerical representations, typically integers, which serve as a unique ID and will be the input to the embedding layer Raschka (2024, p. 33).

At its core, an embedding maps discrete tokens into a continuous vector space. In the embedding layer, each token is associated with a high-dimensional vector, drawn from a structured matrix known as the embedding matrix Kamath et al. (2022, p. 20). This matrix consists of a set number of vectors, each corresponding to a unique token in the model's vocabulary Raschka (2024, p. 34). The dimensionality of these vectors is predefined and chosen based on the model's design requirements. OpenAI's GPT-3, for example, uses embeddings with 12,288 dimensions Raschka (2024, p. 28). Higher-dimensional embeddings allow for more nuanced distinctions between words, capturing deeper semantic relationships Courant et al. (2023, p. 199). When a set of tokens enters the embedding layer, the model uses their unique IDs to locate their corresponding vectors within the embedding matrix, returning the appropriate embeddings Raschka (2024, p. 56). Initially, the values in this matrix, known as weights, are assigned randomly. As training progresses, these values are iteratively refined through backpropagation, allowing the model to adjust embeddings based on conceptual relationships found in the data Raschka (2024, p. 54). This optimization process ensures that words frequently appearing in similar conceptual categories develop embeddings that move closer together in the vector space. For example, as training continues, vectors for "king" and "queen" become more similar, as do those for "dog" and "cat," reflecting their semantic connections Raschka (2024, p. 28).

One important characteristic of Transformers is that they are permutation invariant, meaning they do not inherently recognize word order Lin et al. (2022, p. 122). For example, the phrases "Fox jumps over dog" and "Dog jumps over fox" would appear identical to a Transformer Raschka (2024, p. 57). To address this, positional embeddings are added to token embeddings, encoding the position of each token in a sequence, such as a sentence, to provide a sense of order Lin et al. (2022, p. 122).

By converting discrete tokens into embedding vectors, Transformers can process text in a mathematically meaningful way, forming the foundation for all subsequent learning. Since these embeddings are trainable, they evolve over time, allowing the model to continuously improve its language understanding as it learns from more data Raschka (2024, p. 68).

The self-attention mechanism is the most important component of a Transformer. It enables the model to focus on long sequences of text and capture contextual relationships between words effectively. Unlike traditional sequence-based models, which process tokens sequentially, self-attention allows each token to analyze its relationship with all other tokens in the sequence simultaneously Raschka (2024, p. 70).

To achieve this, the self-attention mechanism relies on three weight matrices: query mapping, key mapping, and value mapping. These matrices contain learnable parameters that are updated during training Courant et al. (2023, p. 195). For each token in a sequence, its embedding vector

is multiplied by these matrices, producing the query, key, and value representations. The query vector of a token is then compared with the key vectors of all tokens, including itself, to compute similarity scores using dot products Courant et al. (2023, p. 195). A high dot product value indicates a strong relationship, whereas a lower value suggests a weaker connection Raschka (2024, p. 73).

These similarity scores, often referred to as attention scores, determine how much focus each token should place on others in the sequence. The attention scores are then normalized using a softmax function, converting them into attention weights. These weights dictate the influence each token's value vector has on the final representation of the token being processed (Raschka, 2024, pp. 72–74).

To illustrate this process, consider the sentence: "Apple is a fruit." This sentence consists of four tokens: "Apple", "is", "a", "fruit". To calculate the final representation of "Apple" and determine how much it is influenced by other words, we first compute its attention scores by taking the dot product of the query vector of "Apple" with the key vectors of all tokens in the sequence, including itself. This results in four distinct attention scores for "Apple". These attention scores are then converted into attention weights using the softmax function Raschka (2024, p. 74).

Once the attention weights are obtained, they are used to compute the final representation of "Apple". Each attention weight is multiplied by the value vector of the corresponding token: the attention weight between "Apple" and "fruit" is multiplied by the value vector of "fruit", the attention weight between "Apple" and "is" is multiplied by the value vector of "is", and so on. The weighted value vectors are then summed, forming the final representation of "Apple", also referred as context vector, which represents how it interacts with the entire sequence. This process is performed simultaneously for every token in the sentence, allowing each word to build a context-aware representation Raschka (2024, p. 7).

By leveraging self-attention, Transformers efficiently capture dependencies between words, regardless of their distance in the sequence. This ability to dynamically adjust attention weights makes them highly effective in understanding language and processing complex sequences Courant et al. (2023, p. 195).

While self-attention allows Transformers to capture relationships between tokens, multi-head self-attention extends this capability by enabling the model to learn multiple types of relationships simultaneously. Instead of computing a single self-attention, multi-head self-attention applies several self-attention mechanisms in parallel, each with its own set of query, key, and value matrices. These independent attention heads allow the model to focus on different aspects of the input sequence Kamath et al. (2022, p. 24). After processing, the outputs of all attention heads are concatenated, ensuring that the model retains diverse contextual information. This mechanism enhances the Transformer's ability to understand complex language structures, improving its performance across various natural language processing tasks Raschka (2024, p. 101).

Multi-head self-attention is a crucial concept in the Transformer architecture, enabling the model to focus on different aspects of a sequence simultaneously. After multi-head self-attention, the output passes through layer normalization, a technique that stabilizes training by normalizing activations across the layer. This helps control the variance of inputs, ensuring that the network learns more efficiently and avoids issues like exploding or vanishing gradients Raschka (2024, p. 120).

Following layer normalization, the transformed embeddings are processed by a fully connected feedforward neural network (FNN). Unlike self-attention, which focuses on interactions between tokens, the FNN refines each token's representation independently Lin et al. (2022, p. 113). It consists of two linear layers with a non-linear activation function, allowing the model to capture complex features and improve its expressiveness. The FNN ensures that the token representations remain useful for deeper layers while maintaining the same input and output dimensions for seamless stacking of other layers Raschka (2024, p. 130).

Finally, residual connections are applied to both the self-attention and FNN outputs. These connections help prevent information loss and enable better gradient flow, making it easier to train deep architectures. By adding the original input back to the transformed output, residual connections ensure that important features are preserved while still benefiting from the transformations introduced by self-attention and the FNN (Raschka, 2024, pp. 132–133).

Together, these components—multi-head self-attention, layer normalization, the feedforward network, and residual connections—work together to form a Transformer block. Multiple Transformer blocks can be stacked on top of each other Raschka (2024, p. 140).

3.3 Vision Transformer Background

Now that I have explained the key mechanisms behind Transformers, I will introduce the fundamental aspects of Vision Transformers. Vision Transformers share a similar architecture to standard Transformers but include modifications tailored for image processing. ViTs were first proposed by (Dosovitskiy et al., 2020) that introduced adaptations necessary for handling visual data. In this section, I will outline the key improvements made for processing images by explaining the basic structure of ViT as proposed in (Dosovitskiy et al., 2020). While the architecture I implemented differs slightly from the original design, I will discuss my modifications and design choices in the next chapter.

In the previous chapter, I explained how Transformers for NLP process words by converting them into tokens, where each token is assigned a unique ID. The model then uses these IDs to retrieve corresponding embedding vectors from an embedding matrix, which consists of a fixed number of vectors, each representing a word in the training data. Since words

[2] taken from Dosovitskiy et al. (2020, p. 3)

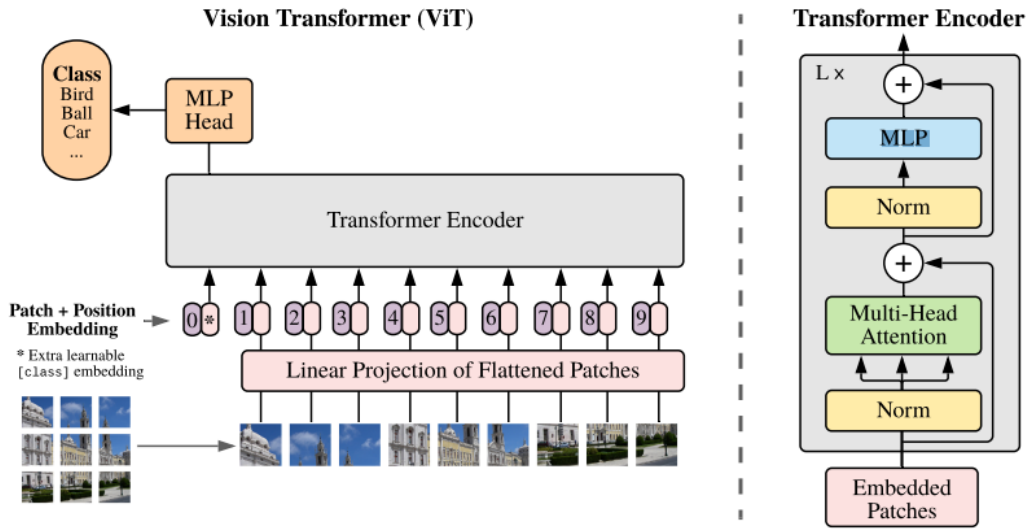

Figure 2: Vision Transformer Architecture[2]

Description. The figure illustrates the ViT pipeline on the left and the Transformer Encoder on the right. In the left part, an input image is divided into non-overlapping patches, flattened, and linearly projected into embeddings with positional encodings and a learnable class token. These embeddings are processed by the Transformer Encoder, and the class token output is passed to an Multi-Layer Perceptron (MLP) Head for classification. The right part shows the Transformer Encoder, where each layer follows a structured sequence: Layer Normalization (Norm) is applied first, followed by Multi-Head Self-Attention and a residual connection. Another Layer Normalization is then applied before passing through a MLP block, which is followed by a second residual connection.

are categorical, they always map to the same column in the embedding matrix, ensuring that a word like "eye" receives the same embedding regardless of the sentence it appears in. However, this approach does not directly apply to images, as there are countless pixel combinations that could resemble an eye. Unlike words, which have a predefined vocabulary, images do not have a fixed set of possible representations that can be directly mapped to an embedding matrix. Instead, Vision Transformers introduce patch embedding, where an image is divided into smaller patches rather than being processed as a whole Courant et al. (2023, p. 203). Figure 2 shows the basic ViT architecture that was proposed by (Dosovitskiy et al., 2020) and how the images are divided into patches. First the image is divided into nine patches. Each patch consists of raw pixel values that are first flattened into a one-dimensional vector. Instead of using an ID to retrieve an embedding, these flattened pixel vectors are mapped to a fixed-dimensional space through a trainable linear projection Dosovitskiy et al. (2020, p. 3). This projection functions similarly to an embedding matrix, ensuring that the transformed pixel values are in the appropriate format for the self-attention mechanism. During training, the model optimizes this projection, learning how to represent pixel information in a way that enhances its ability to process relationships between patches effectively Dosovitskiy et al. (2020, p. 4).

After generating patch embeddings, positional embeddings are added to retain spatial information, similar to how word positions are incorporated in traditional Transformers Dosovitskiy et al. (2020, p. 3). Just as positional embeddings help indicate word order in a sentence, they ensure that the model understands the relative arrangement of patches within the image. Additionally, a learnable class embedding, known as class token, is prepended to the sequence of patch embeddings Dosovitskiy et al. (2020, p. 3). Since this class token will also participate in the self-attention mechanism later, it will learn relationships between different patches, gradually aggregating information from the entire image. This token is designed to serve as a comprehensive representation of the entire image, making it particularly useful for classification tasks Courant et al. (2023, p. 203).

The prepared embeddings, consisting of the class token, patch embeddings, and positional embeddings, are then passed into a Transformer encoder Dosovitskiy et al. (2020, p. 3). The term "encoder" originates from the original Transformer architecture, which was designed for sequence-to-sequence tasks, such as language translation Vaswani et al. (2017, p. 3). Within the encoder, self-attention, specifically multi-head self-attention, plays a crucial role in capturing dependencies between different patches. Instead of treating patches independently, self-attention enables the model to analyze how different regions relate to one another along with the class token, effectively learning a global understanding of the image Touvron et al. (2020, p. 3). Just as self-attention helps capture context in sentences or paragraphs, it enables ViTs to extract meaningful relationships between distant patches in an image (Li et al. 2023, p. 1; Touvron et al. 2020, p. 3). This could be particularly beneficial for stock chart analysis, where the relationships

between different parts of the chart may contribute to accurate classification.

As seen in Figure 2, after self-attention learns relationships between patches, a Multilayer Perceptron refines the extracted information. Unlike traditional Transformer blocks, where layer normalization is applied after attention and feedforward layers, ViTs apply layer normalization before both multi-head self-attention and MLP layers. Additionally, residual connections are included to ensure stable gradient flow and efficient training Dosovitskiy et al. (2020, p. 3). Layer normalization, multi-head self-attention, MLP, and residual connections form the Transformer encoder in ViT. Dosovitskiy et al. proposed stacking 12 Transformer encoder layers as a baseline. Throughout these layers, the class token continuously updates by integrating information from important patches, ultimately forming a global representation of the entire image. After passing through all Transformer encoder layers, the final class token is sent to the MLP head, which generates the classification prediction Courant et al. (2023, p. 203).

To conclude, in ViTs, an image is divided into patches, and the pixel values of each patch are flattened. Through a trainable linear projection, the model learns how to represent these pixels as patch embeddings. The positions of the patches are also encoded and, along with their embeddings, are fed into the multi-head self-attention mechanism. Each self-attention head captures different relationships between the patches. For example, in the sentence 'I ate pizza, and I like it,' the self-attention mechanism learns that the word 'it' refers to 'pizza.' Similarly, in a stock chart image, different parts of the image learn how they relate to one another and whether those relationships influence stock returns. This ability to model dependencies across different regions of the image allows ViTs to potentially uncover patterns that CNNs might overlook.

3.4 Class-Attention in Image Transformer

In the previous chapter, I introduced the standard Vision Transformer architecture, which I will refer to as the base ViT. However, the architecture I implemented is Class-Attention in Image Transformers (CaiT), proposed by (Touvron et al., 2021). CaiT builds upon the base ViT with two key improvements: LayerScale and Class-Attention. These modifications address the limitations of base ViT, enabling deeper network training and improving classification efficiency.

Dosovitskiy et al. (2020) and (Touvron et al., 2020) have shown that increasing the number of layers does not necessarily enhance performance due to optimization challenges. CaiT overcomes this with LayerScale, a learnable diagonal matrix applied to residual connections Touvron et al. (2021, p. 4). By starting with small initial values and gradually learning the appropriate scaling, LayerScale stabilizes training, making it possible to train much deeper Transformers without convergence issues Touvron et al. (2021, p. 2).

The second improvement, Class-Attention, refines how the class token interacts with image patches. In base ViT, the class token attends to all patches at every layer, forcing the model to simultaneously learn both inter-patch relationships and classification features Touvron et al. (2021, p. 5). CaiT restructures this by first allowing self-attention layers to process only patch embeddings, learning spatial relationships without interference. The class token is introduced later in a dedicated class-attention stage, where it aggregates information from the learned patch representations Touvron et al. (2021, p. 5–6). This separation improves classification by ensuring that the class token captures global rather than prematurely biased local information (Touvron et al., 2021, pp. 20–21).

I chose CaiT for stock chart classification because its architectural modifications align with my assumptions about the nature of stock charts. Specifically, I assume that stock charts contain abstract patterns that hold predictive power over returns. In the base ViT, the class token is present from the beginning, aggregating information from different parts of the image. While this approach works well for natural images—where distinct visual features, such as a cat's eyes and ears, are immediately recognizable and crucial for classification—stock charts lack such obvious key elements. Instead, the patterns that influence returns are more subtle and complex. By postponing the introduction of the class token, CaiT allows the model to first focus on learning these abstract patterns within the image before associating them with the final prediction. Once the model has captured these patterns, the class-attention stage may then predict the return direction based on the fully learned representations. Given these assumptions, CaiT is a suitable choice. Additionally, in the appendix, I provide results from experiments with other architectures to compare their effectiveness.

3.5 Lack of Inductive Bias in ViT

Inductive bias refers to the set of assumptions a learning algorithm makes to generalize from training data to unseen data. Since a model cannot encounter all possible examples during training, it relies on educated guesses to make predictions. Inductive bias helps guide the model toward patterns or rules that are likely to work well, rather than treating all possible explanations equally. Without inductive bias, a model would be no better than random guessing when encountering new data Haussler (1988, p. 178).

CNNs have strong inductive biases, such as locality, where small filters focus on capturing local patterns, and translation equivariance, ensuring features are recognized regardless of their position in the image Battaglia et al. (2018, p. 6). However, these inductive biases are weaker in Vision Transformers compared to CNNs. In ViTs, self-attention layers are global, meaning spatially close patches do not inherently have stronger relationships—any two patches can be related, regardless of their position (Dosovitskiy et al., 2020, p. 4). Only MLP layers provide some degree of locality since they process each token independently. Additionally, MLP layers

[3] Modified, taken from S. H. Lee et al. (2021, p. 3).

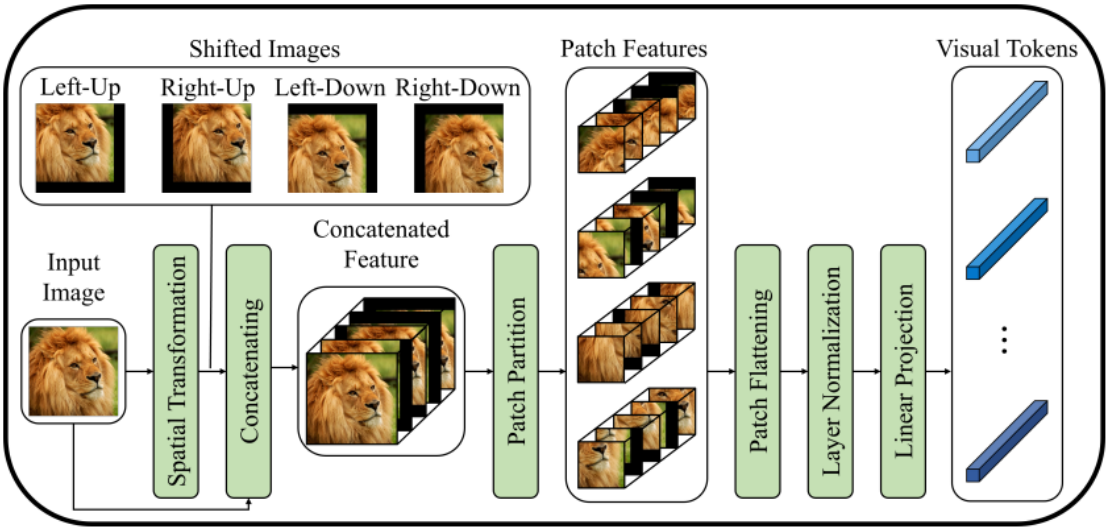

Figure 3: Shifted Patch Tokenization[3]

Description. The figure illustrates the Shifted Patch Tokenization (SPT) process. The input image is first shifted in four directions—Left-Up, Right-Up, Left-Down, and Right-Down—creating multiple spatially transformed versions. These shifted images are then concatenated with the original image before being divided into patches. Each patch is flattened, normalized, and passed through a linear projection layer, converting it into visual tokens for the Vision Transformer model.

are the only components that are translationally equivariant, as the same transformation is applied to all patches, regardless of their location (Dosovitskiy et al., 2020, p. 4). Due to this lack of inductive bias, training ViTs from scratch requires large datasets, or alternatively, pretrained models are needed to ensure effective learning (S. H. Lee et al., 2021, p. 1). Training the model from scratch with stock chart data is challenging in this case.

S. H. Lee et al. (2021) identify two key factors in Vision Transformers that contribute to their lack of locality inductive bias: poor tokenization and a weak attention mechanism for locality. In ViTs, images are divided into non-overlapping patches, meaning each patch is processed independently without considering neighboring pixels. In contrast, CNNs use sliding filters that capture not only the pixels in a specific region but also how surrounding pixels interact, increasing the receptive field and enhancing local feature extraction. Since ViTs treat each patch separately, they lack this spatial continuity, resulting in a weaker ability to encode fine-grained spatial relationships (S. H. Lee et al., 2021, p. 2). The second issue is the inefficiency of the attention mechanism when applied to image data. Unlike text-based Transformers, where the number of tokens is relatively small, ViTs must process hundreds of patches, significantly increasing the feature dimension and making self-attention computationally expensive. This leads to smooth attention scores, where attention is distributed too evenly across all patches instead of focusing on important regions. As a result, ViTs struggle to concentrate on critical parts of an image, often placing unnecessary attention on background details rather than the target object (S. H. Lee et al., 2021, p. 2).

To address these issues, (S. H. Lee et al., 2021) propose two solutions: Shifted Patch Tokenization (SPT) and Locality Self-Attention (LSA). SPT improves the locality of tokenization by shifting the input image in multiple directions before dividing it into patches. Figure 3 represents the shifting process of images and how they are transformed into visual tokens. This process embeds spatial relationships between neighboring pixels into the patch tokens, increasing local feature extraction. The shifted images are then concatenated with the original image, and standard patch partitioning is applied. By incorporating additional spatial information into the patch embeddings, SPT effectively expands the receptive field of ViTs, making them better at capturing local structures S. H. Lee et al. (2021, p. 3). Meanwhile, LSA enhances the attention mechanism by sharpening the attention score distribution. Instead of attention spreading too evenly across all patches, LSA learns a temperature parameter that controls the sharpness of the softmax function, allowing ViT to focus more on relevant regions. Additionally, diagonal masking is applied to remove self-token relations, ensuring that attention is directed towards meaningful patch interactions rather than being distributed uniformly S. H. Lee et al. (2021, p. 3). Together, SPT and LSA enhance the locality inductive bias of ViTs, improving their ability to recognize fine-g rained spatial details and focus on important image regions S. H. Lee et al. (2021, p. 1). I incorporated these two methods into the CaiT architecture, following (S. H. Lee et al., 2021), to enable CaiT to learn the stock image relationships from scratch.

4 Data and Methodology

In this paper, I replicate the study conducted by (Jiang et al., 2023) in their work "*(Re-)Imag(in)ing Price Trends*"

and extend it by replacing the Convolutional Neural Network with a Vision Transformer architecture, CaiT. Jiang et al. (2023) construct stock chart images from 5-day, 20-day, and 60-day periods to predict the subsequent 5-day, 20-day, and 60-day returns using CNN, resulting in a total of nine CNN models trained for different time horizons. Following their approach, this section describes the data used, the transformation process for generating stock chart images, and the modifications made to adapt these images for CaiT. I then provide an overview of the CNN and CaiT architectures, detailing their specifications and structures. Finally, I outline the training process, including optimization methods, loss functions, and techniques used to enhance model performance and generalization.

4.1 Data

Following (Jiang et al., 2023), the daily stock data is sourced from the Center for Research in Security Prices (CRSP) database and includes opening, high, low, and closing prices for stocks listed on the NYSE, AMEX, and NASDAQ exchanges along with volume and CRSP adjusted returns Jiang et al. (2023, p. 3205).

I adjusted the prices using the methodology of (Jiang et al., 2023) to represent price movements in a standardized format. First, data is structured into 5-day, 20-day, and 60-day periods, with each period consisting of consecutive trading days. The first closing price in each period is normalized to 1, and subsequent closing prices are iteratively calculated using CRSP-adjusted returns. Once the adjusted closing prices are determined, the opening, high, and low prices for each day are scaled proportionally based on their original relationship to that day's actual closing price Jiang et al. (2023, p. 3206). This step ensures price consistency across different stocks while also accounting for stock splits and dividends Jiang et al. (2023, p. 3199). Following (Jiang et al., 2023), moving average prices are calculated for each day in the period based on its length. For example, if 5-day images are created, moving average prices for the past 5 days are calculated Jiang et al. (2023, p. 3200).

Additionally, as described by (Jiang et al., 2023), if the return of any stock on the first day of a constructed 5-, 20-, or 60-day period is missing, it is excluded. Moreover, if the IPO or delisting of a stock occurs within a period, it is also excluded. However, if any other price or volume information is missing within those periods, the stock and period are included, but the missing values are left blank in the images Jiang et al. (2023, p. 3199).

4.2 Creating images of the stock charts

After processing the raw price data, I used the transformed data to generate the images following the methodology of (Jiang et al., 2023). The final image dataset consists of three different image sizes: 32×15, 64×60, and 96×180, corresponding to the 5-day, 20-day, and 60-day periods, respectively Jiang et al. (2023, p. 3202). The width of each image is always three times the period length, as each trading day spans three pixels. Within this structure, the first pixel column of each day represents the opening price, the second column represents the high and low prices, and the last column represents the closing price Jiang et al. (2023, p. 3199).

For the height of the price information, both the opening and closing prices are represented as single-pixel heights, while the high and low prices are marked with a single pixel at their respective levels, with the pixels in between filled to form a vertical bar Jiang et al. (2023, p. 3198). In addition to price data, the moving average line, derived from the calculated moving averages, is also included in the image. The price data and moving average line occupy the upper four-fifths of the image, while the lower part is reserved for volume bars, visually separating price trends from trading activity Jiang et al. (2023, p. 3198). The entire image is constructed on a black background, with all information—prices, moving averages, and volume bars—displayed in white Jiang et al. (2023, p. 3193). Figure 4 presents an example of the resulting 60-day images.

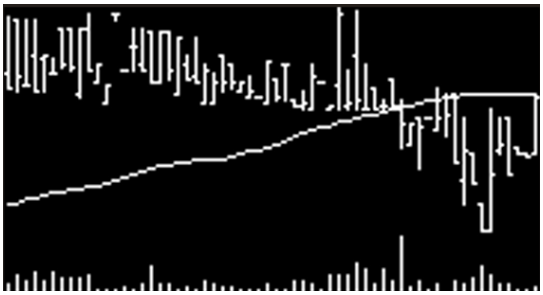

Figure 4: Example of 60-day Image for CNN

Description. This figure shows a 60-day stock chart image for CNN analysis. The lower part represents trading volume, while the upper part displays opening, high, low, and closing prices. The moving average line runs through the price section.

As described by (Jiang et al., 2023), in each stock chart, a consistent image height is ensured by scaling prices and volumes relative to their maximum and minimum values. For price and moving average lines, the values are rescaled so that their heights within the price section of the image fall between the maximum and minimum of either the opening, high, low, or close prices. Each price or moving average value is then adjusted according to this new height Jiang et al. (2023, p. 3198). Similarly, the volume bars are scaled within their designated area based on the maximum and minimum volume values Jiang et al. (2023, p. 3200). This scaling process standardizes stock price representations, making them comparable across different stocks and ensuring that they can be effectively processed by a Convolutional Neural Network Jiang et al. (2023, p. 3194).

I generated the images for CNN following the methodology of (Jiang et al., 2023); however, these images differ from those commonly used in Vision Transformer models, such as those in (Dosovitskiy et al., 2020; Touvron et al., 2021), and (S. H. Lee et al., 2021). The first key difference is that these Vision Transformer models use RGB images, meaning they

have three channels, whereas the images I created were originally grayscale, represented by a single channel. The second difference is in the image shape—standard ViT implementations use square patches, meaning each patch has equal width and height, while the images I created have rectangular shapes, which cannot be divided into square patches without distortion.

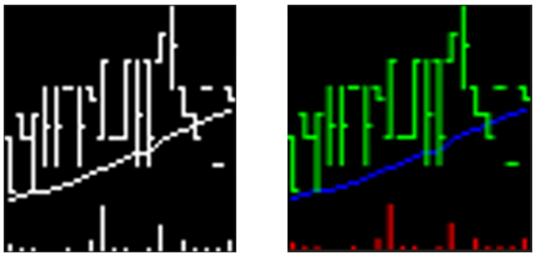

Figure 5: Grayscale and RGB Images of Stock Data

Description. The figure compares grayscale and RGB images of 20-day stock data. The left image (64×60) is a grayscale version used for CNN models, where stock prices, moving averages, and volume are represented in white on a black background. The right image (64×64) is the RGB-transformed and resized version for CaiT, with price data in green, the moving average line in blue, and volume bars in red, encoding different financial components into separate color channels.

To adapt the images for CaiT, I first converted them into RGB format, where each channel represents a specific type of information from the grayscale images. In this transformation, price data is represented in green, the moving average line in blue, and volume bars in red. Figure 5 shows an example of a 20-day stock image used for CNN (left) along with the RGB image (right) of the same stock. The original grayscale images I generated for CNNs are essentially matrices of the same shape as the image, where black pixels correspond to 0 and white pixels correspond to 255 Jiang et al. (2023, p. 8). In contrast, the RGB images used for CaiT consist of three separate matrices, each corresponding to a different color channel. These matrices are layered on top of each other, with red at the front, green in the middle, and blue at the back. In each channel matrix, the black areas are encoded as 0, while the colored regions are set to 255, effectively encoding different financial data components into separate channels for use in CaiT models.[4] Subsequently, following (Jiang et al., 2023), the pixel values of both grayscale and RGB images are normalized.

Secondly, to ensure compatibility with square patch partitioning in Vision Transformers, I transformed the images into a square format. Specifically, 32×15 images were resized to 32×32, and 64×60 images were adjusted to 64×64. This resizing was performed using bilinear interpolation, a method that smooths pixel values by averaging nearby pixels (Darji, 2021). The color variation in the right image of Figure 5 is a result of the resizing process. The transformation slightly alters pixel intensities, causing minor shifts in color representation. Although the original images maintained a fixed three-pixel width per trading day structure, this arrangement no longer applies explicitly after resizing. However, this information is still indirectly preserved, as bilinear interpolation was applied consistently across all images. Since every image in a given period was resized the same way, the resulting representations retain the encoded structure of the original format, ensuring that the attention mechanism in Vision Transformers can still capture meaningful relationships between patches. I opted for resizing instead of incorporating images with different time horizons into a single model to ensure a direct comparison between CNN and CaiT predictions. By applying the same preprocessing steps to both models, they learn from identical input data, ensuring that any observed performance differences arise from architectural distinctions rather than disparities in data structure.

4.3 CNN Architecture Specifications

The CNN architecture used in this paper is proposed by (Jiang et al., 2023) and designed to process stock price charts of different time horizons—5-day, 20-day, and 60-day—by progressively increasing network depth as the input size grows. For the 5-day input image with a resolution of 32×15, the network consists of two convolutional layers with 5×3 filters, where the first layer has 64 channels and the second has 128 channels. Each convolutional layer is followed by a 2×1 max-pooling layer, and the extracted features are passed through a fully connected layer of size 15,360 before classification via a softmax layer Jiang et al. (2023, p. 3202).

For the 20-day input image with a resolution of 64×60, the model becomes deeper with three convolutional layers, all using 5×3 filters, with the number of channels increasing from 64 to 128 to 256. Each convolutional layer is followed by a 2×1 max-pooling layer, reducing the spatial dimensions while preserving key features. The flattened feature representation is then passed through a fully connected layer of size 46,080, and classification is performed using a softmax function Jiang et al. (2023, p. 3202).

The 60-day input image, with a resolution of 96×180, is processed by an even deeper CNN. This version of the model consists of four convolutional layers with 5×3 filters, where the number of channels increases from 64 to 128, then to 256, and finally to 512. As in the previous models, each convolutional layer is followed by a 2×1 max-pooling layer, which progressively reduces dimensionality while preserving essential trend-related features. The extracted features are then passed through a fully connected layer of size 184,320, before classification with a softmax layer Jiang et al. (2023, p. 3202).

Following (Jiang et al., 2023), all versions of the CNN employ leaky ReLU, a variant of ReLU proposed by (Maas et al., 2013), as the activation function after each convolutional layer. This choice introduces non-linearity while mitigating the issue of dead neurons Jiang et al. (2023, p. 3239).

[4] In Figure 5, the price bars appear to intersect with and obscure parts of the moving average line. However, the moving average information is still encoded within its respective channel matrix; it is simply not visible in this representation.

The deeper architectures used for longer time horizons allow the model to capture increasingly complex patterns in stock price movements, with the max-pooling layers ensuring that important spatial relationships in the input images are preserved while reducing computational complexity (Jiang et al., 2023, pp. 3240–3242). The fully connected layers aggregate extracted features before the final classification stage, enabling the model to make predictions about future price movements based on historical chart patterns Jiang et al. (2023, p. 3241).

4.4 CaiT Architecture Specifications

Following (S. H. Lee et al., 2021), I used the CaiT architecture as proposed by (Touvron et al., 2021). The structure of CaiT is implemented as outlined by (Touvron et al., 2021) with two modifications: Locality Self-Attention and Shifted Patch Tokenization, as suggested by (S. H. Lee et al., 2021).

Following (Touvron et al., 2021), my CaiT model consists of 24 Transformer layers for processing patch embeddings, followed by 2 layers dedicated to class attention. Each multi-head self-attention layer has 4 attention heads, with each head having a dimension of 64, resulting in a total attention dimension of 256 per layer. The model uses a patch size of 4×4, ensuring fine-grained tokenization of the image. Each patch is projected into an embedding dimension of 192, serving as the feature representation for further processing. The MLP expansion ratio is set to 2, meaning the hidden dimension of the feedforward layers is 384. Similar to the CNN model, the CaiT model is designed for a binary classification task with two output classes, corresponding to predicting whether stock returns will go rise or fall.

Following (Touvron et al., 2021), within each Transformer layer, Layer Normalization (Ba et al., 2016) is applied before both the self-attention and MLP layers to maintain stable activations and ensure efficient learning. Residual connections (He et al., 2016b) are included in each Transformer layer, facilitating for better gradient flow and preventing vanishing gradients. To further enhance training stability in this deep architecture, LayerScale as proposed by (Touvron et al., 2021) is applied to residual connections. In MLP layers, GeLU (Hendrycks & Gimpel, 2016) activation function is used.

Following (S. H. Lee et al., 2021), Locality Self-Attention is applied to each self-attention mechanism in both Patch and Class Transformer layers, sharpening attention score distributions through a learnable temperature parameter. In the tokenization step, Shifted Patch Tokenization is used, where the input image is shifted in multiple directions before being divided into patches, ensuring that more spatial relationships are embedded into each visual token.

With these design choices, the model is optimized for learning complex spatial relationships within stock charts, allowing it to capture subtle patterns and improve classification performance. The model specifications or structure don't change with respect to the image size, as in CNN models.

4.5 Training

Now, I will describe the training process for both CNN and CaiT. For the CNN models, I follow the methodology outlined by (Jiang et al., 2023), which is itself based on the training procedure introduced by (Gu et al., 2020). The dataset spans from 1993 to 2000 for training, while the period from 2001 to 2019 is designated as the out-of-sample dataset for evaluation. Within the training dataset (1993–2000), the data is further split into 70% training and 30% validation sets, ensuring proper model tuning and preventing overfitting Jiang et al. (2023, p. 3204). These training and validation samples are selected randomly while maintaining a balanced classification task, accounting for both bullish and bearish market trends to avoid bias in model learning Jiang et al. (2023, p. 3204). I applied this same structured workflow to the CaiT models, allowing for a direct comparison between the CNN and Vision Transformer architectures in predicting stock price movements.

The training process begins by feeding the training dataset into the neural network, where the model makes predictions based on the input images. Once the model has been trained on this data, the validation set is passed through the network to evaluate its performance on unseen data Goodfellow et al. (2016, p. 204). Following (Jiang et al., 2023), the prediction task is formulated as a binary classification problem, where stock returns are categorized into two classes: up or down. Labels are assigned based on the actual return values, with positive returns labeled as 1 and negative returns labeled as 0 Jiang et al. (2023, p. 3204). The loss function, which the model attempts to minimize, is set to cross-entropy loss Jiang et al. (2023, p. 3205) in both CNN and CaiT.

Again, I followed the regularization techniques outlined by (Jiang et al., 2023), which are based on (Gu et al., 2020). To train the model, the Adam optimization algorithm (Kingma & Ba, 2017) is used, dynamically adjusting learning rates for efficient weight updates. The training begins with an initial learning rate of 1×10^{-5} and the batch size is 128 Jiang et al. (2023, p. 3205). To stabilize learning, batch normalization (Ioffe & Szegedy, 2015) is applied between convolutional layers and activation functions, ensuring a consistent distribution of activations and reducing internal covariate shift Jiang et al. (2023, p. 3205). In addition, dropout regularization (Srivastava et al., 2014) is used to further prevent overfitting, with 50% of neurons in the fully connected layer randomly deactivated during training. However, dropout is not applied to convolutional layers, as their lower parameter count naturally limits overfitting Jiang et al. (2023, p. 3205). For weight initialization, I use the Xavier initializer (Glorot & Bengio, 2010), which assigns initial weight values to facilitate faster convergence by maintaining balanced variance across layers. Furthermore, early stopping is implemented with a patience of two epochs, meaning training stops if the validation loss does not improve for two consecutive epochs, preventing excessive training and reducing the risk of overfitting Jiang et al. (2023, p. 3205). Finally, following (Gu et al., 2020) and

(Jiang et al., 2023), five independent models are trained, and their predictions are averaged to form the final output, ensuring a more stable and reliable classification result.

For the CaiT models, following Lee et al. (2022), I applied label smoothing (Szegedy et al., 2016) to the cross-entropy loss function. Label smoothing is a regularization technique that prevents the model from becoming overconfident in its predictions by slightly adjusting the true class probabilities. Instead of assigning a hard label of 0 or 1, it redistributes a small portion of the probability mass to all classes Goodfellow et al. (2016, p. 243). This technique helps reduce overfitting and improves generalization, particularly in deep networks where extreme confidence in predictions can lead to poor robustness Szegedy et al. (2016, p. 2823). Following (S. H. Lee et al., 2021), AdamW (Loshchilov & Hutter, 2019) optimizer is chosen with a weight decay of 0.05 and a warm-up period is set to 5. Stochastic depth (Huang et al., 2016) is also applied. For learning rate settings, I set the initial learning rate to 1×10^{-5} for 20-day images and to 1×10^{-4} for 5-day images, following a structured adjustment for different time horizons. Batch size is set to 128, similar to CNN. Although (S. H. Lee et al., 2021) propose several data augmentation techniques, I did not incorporate these methods in my study. Since my task involves stock charts rather than natural images, I was uncertain about their effectiveness for time-series-based graphs. Unlike object classification tasks where mixing pixels or cutting portions of an image may still preserve semantic meaning, altering stock chart images in such a manner could disrupt the underlying financial patterns crucial for prediction.

5 Results

In this section, I present the performance results of the models over the 19-year out-of-sample period. Following (Jiang et al., 2023), stocks are first sorted into deciles based on the probability outputs of their respective models. Then, the annualized average returns for each holding period, along with their annualized Sharpe ratios to assess risk-adjusted performance, are calculated. Additionally, long-short portfolios (H-L) are constructed by longing the highest decile (Decile 10 or High) and shorting the lowest decile (Decile 1 or Low) to examine the return spread between extreme predictions.

For each model name, I use a notation format where the number of days the images cover is followed by "D", and the prediction horizon for returns is followed by "R". For example, "20D/5R" indicates that 20-day images are used to predict the next 5-day returns. Following (Jiang et al., 2023), portfolios are rebalanced based on the prediction period, meaning models that predict 5-day, 20-day, and 60-day returns result in weekly, monthly, and quarterly rebalancing, respectively. Following (Jiang et al., 2023), I compute the monthly turnover for each model, as described by (Gu et al., 2020). For strategies rebalanced weekly or quarterly, turnover is appropriately scaled to a monthly basis. A 200% turnover indicates a complete replacement of stocks within one period, while 0% turnover means no changes were made Jiang et al. (2023, p. 17).

I begin by presenting the results for the CNN models, alongside comparisons with Momentum, Short-Term Reversal, and Weekly Short-Term Reversal strategies to provide additional benchmarks following (Jiang et al., 2023). Next, I report the results for the CaiT models, evaluating their predictive effectiveness in the same framework. Finally, I take the simple average of the probability outputs from both CNN and CaiT models, sort stocks based on these average probabilities and analyze the results. This combined approach, referred to as the "Average" model, offers insight into whether the integration of both architectures improves predictive accuracy and portfolio performance.

5.1 CNN Results

In this section, I discuss the results of CNN models that predict returns over 5-, 20-, and 60-day horizons, starting with the weekly rebalanced strategies. Following (Jiang et al., 2023), Table 1 focuses on short-horizon strategies 5D/5R and 20D/5R compared to traditional momentum MOM, monthly short-term reversal STR, and weekly short-term reversal WSTR. In the equal-weighted panel (top), the image-based models begin with negative Sharpe ratios in the lowest decile -1.91 for 5D/5R and -2.06 for 20D/5R but rise monotonically to 2.91 and 2.70 in the highest decile, underscoring a robust positive relationship between the CNN up probabilities and subsequent returns. Moreover, the long–short H–L portfolios deliver exceptionally high annualized Sharpe ratios of 7.20 for 5D/5R and 6.96 for 20D/5R, far exceeding the Sharpe ratios of 0.12, 1.75 and 2.91 for MOM, STR and WSTR, respectively. A similar pattern appears in the value-weighted panel (below), though at a lower level, where the H–L Sharpe ratios are 1.54 for 5D/5R and 1.64 for 20D/5R, again outperforming the benchmark signals by a wide margin. Finally, Table 1 indicates that these short-horizon image-based strategies involve relatively high monthly turnover. In the equal-weight panel, 5D/5R and 20D/5R each exceed 660% turnover, matching WSTR at 660% and contrasting with STR at 341% and MOM at 124%. The value-weighted panel follows a similar pattern, with the image-based strategies exceeding 700%, short-term reversal near 433%, and momentum at 164%.

At a monthly rebalancing frequency, image-based predictions continue to demonstrate their superiority over traditional signals. Following (Jiang et al., 2023), Table 2 depicts the monthly rebalanced portfolios using 5D/20R, 20D/20R and 60D/20R, again benchmarked against MOM, STR and WSTR. For equal-weighted portfolios, 5D/20R and 20D/20R yield H-L Sharpe ratios of 2.44 and 2.19, respectively, which is significantly higher than the highest traditional trend signal, WSTR, with the H-L Sharpe ratio of 1.29. The 60D/20R model, which relies on 60-day images, achieves a Sharpe ratio of 1.04. Traditional short-term reversal (STR) lags significantly with a Sharpe ratio of 0.54, and the momentum strategy performs the weakest at 0.32. Furthermore, the image-based strategies trade at relatively high turnover, roughly

Table 1: One Week Portfolio Performance

Equal Weight										
	5D/5R		20D/5R		MOM/5R		STR/5R		WSTR/5R	
	ret	SR	ret	SR	ret	SR	ret	SR	ret	SR
Low	-0.27	-1.91	-0.33	-2.06	0.13	0.40	-0.01	-0.05	-0.09	-0.41
2	-0.05	-0.27	-0.04	-0.20	0.10	0.43	0.06	0.37	0.04	0.22
3	0.02	0.10	0.04	0.20	0.11	0.53	0.09	0.58	0.08	0.49
4	0.07	0.38	0.09	0.45	0.11	0.59	0.10	0.67	0.09	0.59
5	0.10	0.49	0.14	0.71	0.12	0.71	0.11	0.72	0.09	0.60
6	0.13	0.67	0.16	0.83	0.12	0.79	0.11	0.71	0.11	0.68
7	0.17	0.85	0.19	1.00	0.13	0.87	0.12	0.68	0.13	0.76
8	0.22	1.09	0.22	1.16	0.15	0.95	0.12	0.62	0.15	0.78
9	0.30	1.48	0.27	1.42	0.15	0.90	0.16	0.68	0.19	0.83
High	0.54	2.91	0.50	2.70	0.16	0.77	0.37	1.17	0.45	1.55
H-L	0.82	7.20	0.83	6.96	0.03	0.12	0.38	1.75	0.54	2.91
Turnover	681%		663%		124%		341%		660%	
Value Weight										
	5D/5R		20D/5R		MOM/5R		STR/5R		WSTR/5R	
	ret	SR	ret	SR	ret	SR	ret	SR	ret	SR
Low	-0.03	-0.17	-0.05	-0.26	0.02	0.05	0.02	0.10	-0.05	-0.21
2	0.01	0.03	0.02	0.11	0.03	0.11	0.03	0.18	0.01	0.07
3	0.03	0.19	0.03	0.18	0.08	0.32	0.06	0.37	0.06	0.34
4	0.07	0.45	0.07	0.46	0.07	0.38	0.08	0.53	0.07	0.48
5	0.06	0.37	0.06	0.38	0.08	0.52	0.08	0.60	0.08	0.60
6	0.08	0.54	0.07	0.45	0.08	0.50	0.09	0.61	0.09	0.64
7	0.08	0.51	0.09	0.62	0.07	0.43	0.11	0.68	0.11	0.77
8	0.08	0.51	0.08	0.51	0.08	0.53	0.10	0.58	0.10	0.59
9	0.10	0.64	0.08	0.54	0.08	0.54	0.08	0.39	0.10	0.54
High	0.11	0.71	0.09	0.55	0.09	0.68	0.05	0.18	0.08	0.29
H-L	0.06	0.47	0.02	0.16	0.04	0.28	0.02	0.09	0.07	0.36
Turnover	188%		182%		164%		184%		185%	

Description. The table reports annualized average weekly returns (ret), annualized Sharpe ratios (SR), and monthly turnover for Equal-Weighted (top) and Value-Weighted (bottom) portfolios, comparing different models across deciles, with long-short (H-L) portfolios capturing return spreads. Modified, taken from Jiang et al. (2023, p. 3207).

ranging between 150% and 175%, close to STR turnover of 168% and WSTR turnover of 167%, while MOM trades at a turnover of 64%. Similar to weekly strategies, as we move up the decile rankings, the risk-adjusted returns increase consistently, showcasing a strong monotonic relationship between the CNN's predicted probability and actual returns. In the value-weighted context, the 5D/20R model stands out as the strongest performer with a H-L Sharpe ratio of 0.47, followed by MOM and WSTR with H-L Sharpe ratios of 0.36.

Following (Jiang et al., 2023), Table 3 shows the performance of quarterly rebalancing strategies, still outperforming traditional signals but only with one strategy. In the equal-weighted portfolios, 5D/60R gives the best H-L Sharpe ratio with 1.16, followed by WSTR with a Sharpe ratio of 0.69. 20D/60R and 60D/60R have Sharpe ratios of 0.45 and 0.40, respectively, falling behind the WSTR. MOM and STR give the least Sharpe ratios with 0.12 and 0.32, respectively. Monthly turnover for image-based strategies along with STR and WSTR falls between 57% and 60%, whereas MOM has a slightly lower turnover of 38%. 20D/60R and 60D/60R still show a monotonic increase going from the Low decile to High decile.

5.2 CaiT Results

In this section, the performance of CaiT predictions is analyzed in detail. Due to computational cost, only three CaiT models are trained to assess their effectiveness in stock return prediction. The first model utilizes 5-day images to predict 60-day returns. This model was selected because it has a more limited dataset compared to models predicting 5-day or 20-day returns, as it only considers stock data at 60-day intervals. Additionally, it employs the shortest image length (5-day images), encoding minimal information. This serves as an essential benchmark to evaluate the feasibility of applying Vision Transformers to stock classification tasks. If the performance of CaiT is comparable to that of CNN, despite the restricted data and limited encoded knowledge, it may suggest that CaiT could be effectively utilized for other image sizes and prediction windows.

The other two models predict 5-day returns, with one using 5-day images and the other using 20-day images. Vision Transformers employ a global attention mechanism to identify relationships between different regions of an image. Intuitively, using a longer image period (20 days) while analyzing the same stocks should enable the model to capture more meaningful relationships compared to shorter-period images (5 days). The objective of these two models is to assess how performance is influenced by incorporating additional historical data into the images of the same stocks. For comparison, CNN models with the same configurations were chosen as benchmarks rather than traditional price signals to highlight the differences between the two image recognition architectures. The same notation used for CNN models is maintained, with a prefix of "CaiT-" or "CNN-" to distinguish between the two. For example, "CaiT-5D/5R" represents a CaiT model that uses 5-day images to predict 5-day returns, whereas "CNN-5D/5R" denotes the corresponding CNN model.

Table 4 presents the portfolio performance of the CaiT models benchmarked against their corresponding CNN models. For the 5D/60R configuration, both models exhibit comparable performance. In equal-weighted portfolios, CaiT-5D/60R achieves a H-L portfolio return of 0.08 with a Sharpe ratio of 1.23, while CNN-5D/60R yields the same return with a slightly lower Sharpe ratio of 1.16. Additionally, the performance of the Low and High deciles is similar between the two models. However, CaiT-5D/60R demonstrates a more distinct monotonic increase from the Low to High decile, suggesting a stronger ability to differentiate between stock performance categories. Both models exhibit identical turnover rates. The similar performance could be attributed to the limited information contained in the small images and the Shifted Patch Tokenization mechanism described in Section 3.5, which may allow CaiT patches to function similarly to convolutional filters, resulting in the extraction of comparable local features.

For the 5D/5R configuration, the performance similarity between CaiT and CNN persists, with the main difference observed in the High decile. In equal-weighted portfolios, both models yield a Sharpe ratio of -1.91 in the Low decile with nearly identical returns. In the High decile, both models achieve a return of 0.54; however, the Sharpe ratio is slightly higher for CNN-5D/5R (2.91) compared to CaiT-5D/5R (2.78). Consequently, the H-L portfolio return remains consistent at 0.82, with Sharpe ratios of 7.06 for CaiT-5D/5R and 7.20 for CNN-5D/5R. This suggests that, for the same level of risk, CNN delivers slightly higher returns. Regarding turnover, CaiT-5D/5R exhibits a lower turnover rate (677%) compared to CNN-5D/5R (681%).

For the 20D/5R configuration, the performance dynamics shift. CaiT-20D/5R achieves an H-L Sharpe ratio of 7.20, matching that of CNN-5D/5R, while CNN-20D/5R records a slightly lower Sharpe ratio of 6.96. In the low decile, CNN-20D/5R has a Sharpe ratio of -2.06, indicating that it captures more negative returns per unit of risk compared to CaiT. In the high decile, both models yield a Sharpe ratio of approximately 2.7. However, this does not translate to a higher Sharpe ratio in the H-L portfolio. Despite having a lower H-L return of 0.75, CaiT-20D/5R is able to predict higher returns for the same level of risk as CNN. The turnover for CaiT-20D/5R remains slightly lower than that of CNN-20D/5R.

In conclusion, for equal-weighted portfolios, CaiT and CNN 5D/60R models produce similar returns and Sharpe ratios. In the 5D/5R configuration, both models exhibit comparable returns; however, CNN outperforms CaiT in terms of risk-adjusted return in the High decile, leading to a superior risk-adjusted return for the H-L portfolio. When larger images are utilized, the performance dynamic shifts in favor of CaiT. CaiT-20D/5R does not generate negative risk-adjusted returns to the same extent as CNN in the Low decile, and both models achieve equivalent risk-adjusted returns in the high decile. However, in the H-L portfolio, CaiT-20D/5R demonstrates a higher return for the same level of risk, highlighting

Table 2: One Month Portfolio Performance

Equal Weight												
	5D/20R		20D/20R		60D/20R		MOM/20R		STR/20R		WSTR/20R	
	ret	SR	ret	SR	ret	SR	ret	SR	ret	SR	ret	SR
Low	-0.01	-0.05	-0.02	-0.09	0.00	-0.01	0.05	0.15	0.05	0.21	-0.01	-0.02
2	0.05	0.28	0.05	0.27	0.06	0.29	0.07	0.28	0.08	0.46	0.06	0.32
3	0.07	0.39	0.08	0.43	0.09	0.45	0.09	0.45	0.10	0.62	0.09	0.57
4	0.08	0.42	0.10	0.54	0.10	0.52	0.09	0.52	0.10	0.70	0.10	0.67
5	0.09	0.51	0.10	0.57	0.11	0.61	0.10	0.63	0.11	0.76	0.11	0.75
6	0.11	0.59	0.11	0.63	0.12	0.67	0.11	0.81	0.11	0.69	0.12	0.77
7	0.12	0.62	0.13	0.73	0.13	0.75	0.13	0.93	0.12	0.74	0.11	0.71
8	0.13	0.70	0.13	0.75	0.13	0.79	0.13	0.97	0.11	0.58	0.12	0.68
9	0.17	0.90	0.15	0.90	0.14	0.88	0.14	0.90	0.09	0.42	0.13	0.62
High	0.20	1.10	0.17	1.01	0.14	1.01	0.14	0.73	0.15	0.49	0.19	0.65
H-L	0.21	2.44	0.19	2.19	0.14	1.04	0.09	0.32	0.10	0.54	0.19	1.29
Turnover	176%		174%		152%		64%		168%		167%	
Value Weight												
	5D/20R		20D/20R		60D/20R		MOM/20R		STR/20R		WSTR/20R	
	ret	SR	ret	SR	ret	SR	ret	SR	ret	SR	ret	SR
Low	0.05	0.33	0.07	0.38	0.06	0.27	0.01	0.02	0.03	0.14	0.01	0.05
2	0.04	0.24	0.06	0.37	0.08	0.43	0.03	0.12	0.06	0.35	0.03	0.18
3	0.05	0.33	0.06	0.35	0.05	0.28	0.05	0.24	0.06	0.38	0.05	0.35
4	0.07	0.45	0.07	0.46	0.07	0.38	0.07	0.38	0.08	0.53	0.07	0.48
5	0.07	0.36	0.07	0.39	0.10	0.54	0.08	0.45	0.10	0.63	0.10	0.62
6	0.09	0.48	0.08	0.39	0.07	0.39	0.10	0.64	0.09	0.53	0.08	0.49
7	0.08	0.44	0.07	0.37	0.06	0.33	0.08	0.57	0.10	0.58	0.09	0.57
8	0.10	0.56	0.07	0.43	0.07	0.39	0.07	0.50	0.09	0.46	0.09	0.46
9	0.08	0.53	0.09	0.56	0.08	0.51	0.08	0.51	0.08	0.35	0.09	0.43
High	0.10	0.58	0.10	0.64	0.09	0.55	0.10	0.48	0.07	0.21	0.06	0.22
H-L	0.04	0.44	0.04	0.36	0.02	0.14	0.03	0.09	0.03	0.13	0.05	0.24
Turnover	62%		60%		59%		43%		61%		62%	

Description. The table reports annualized average monthly returns (ret), annualized Sharpe ratios (SR), and monthly turnover for Equal-Weighted (top) and Value-Weighted (bottom) portfolios, comparing different models across deciles, with long-short (H-L) portfolios capturing return spreads. Modified, taken from Jiang et al. (2023, p. 3207).

Table 3: One Quarter Portfolio Performance

Equal Weight												
	5D/60R		20D/60R		60D/60R		MOM/60R		STR/60R		WSTR/60R	
	ret	SR	ret	SR	ret	SR	ret	SR	ret	SR	ret	SR
Low	0.06	0.25	0.06	0.22	0.06	0.21	0.07	0.17	0.05	0.19	0.04	0.13
2	0.08	0.34	0.08	0.32	0.08	0.30	0.09	0.28	0.09	0.39	0.09	0.38
3	0.09	0.38	0.09	0.36	0.10	0.39	0.11	0.43	0.11	0.53	0.10	0.46
4	0.09	0.41	0.11	0.45	0.10	0.45	0.11	0.51	0.10	0.54	0.10	0.53
5	0.11	0.47	0.11	0.47	0.11	0.46	0.10	0.55	0.11	0.62	0.11	0.63
6	0.10	0.46	0.11	0.48	0.10	0.45	0.11	0.63	0.11	0.58	0.11	0.60
7	0.10	0.47	0.11	0.51	0.11	0.53	0.11	0.61	0.12	0.61	0.11	0.54
8	0.13	0.57	0.12	0.55	0.12	0.56	0.11	0.65	0.11	0.47	0.11	0.50
9	0.11	0.52	0.12	0.59	0.12	0.62	0.12	0.63	0.10	0.36	0.11	0.43
High	0.14	0.66	0.12	0.64	0.12	0.69	0.11	0.47	0.12	0.33	0.14	0.40
H-L	0.08	1.16	0.06	0.45	0.06	0.40	0.04	0.12	0.07	0.32	0.10	0.69
Turnover	60%		59%		58%		38%		57%		57%	
Value Weight												
	5D/60R		20D/60R		60D/60R		MOM/60R		STR/60R		WSTR/60R	
	ret	SR	ret	SR	ret	SR	ret	SR	ret	SR	ret	SR
Low	0.05	0.27	0.05	0.24	0.07	0.29	0.07	0.15	0.04	0.13	0.02	0.06
2	0.06	0.30	0.06	0.29	0.07	0.34	0.03	0.10	0.07	0.36	0.05	0.25
3	0.06	0.34	0.07	0.33	0.09	0.41	0.09	0.35	0.07	0.39	0.07	0.39
4	0.07	0.38	0.10	0.49	0.08	0.40	0.08	0.38	0.07	0.45	0.09	0.55
5	0.07	0.36	0.07	0.39	0.10	0.54	0.08	0.45	0.10	0.63	0.10	0.62
6	0.09	0.48	0.08	0.39	0.07	0.39	0.10	0.64	0.09	0.53	0.08	0.49
7	0.08	0.44	0.07	0.37	0.06	0.33	0.08	0.57	0.10	0.58	0.09	0.57
8	0.10	0.56	0.07	0.43	0.07	0.39	0.07	0.50	0.09	0.46	0.09	0.46
9	0.08	0.53	0.09	0.56	0.08	0.51	0.08	0.51	0.08	0.35	0.09	0.43
High	0.10	0.58	0.10	0.64	0.09	0.55	0.10	0.48	0.07	0.21	0.06	0.22
H-L	0.04	0.44	0.04	0.36	0.02	0.14	0.03	0.09	0.03	0.13	0.05	0.24
Turnover	62%		60%		59%		43%		61%		62%	

Description. The table reports annualized average quarterly returns (ret), annualized Sharpe ratios (SR), and monthly turnover for Equal-Weighted (top) and Value-Weighted (bottom) portfolios, comparing different models across deciles, with long-short (H-L) portfolios capturing return spreads. Modified, taken from Jiang et al. (2023, p. 3207).

Table 4: Comparison of CaiT and CNN-based Portfolio Performance

Equal Weight												
	CaiT Only						CNN Only					
	5D/5R		20D/5R		5D/60R		5D/5R		20D/5R		5D/60R	
	ret	SR	ret	SR	ret	SR	ret	SR	ret	SR	ret	SR
Low	-0.28	-1.91	-0.29	-1.69	0.05	0.22	-0.27	-1.91	-0.33	-2.06	0.06	0.25
2	-0.03	-0.18	-0.01	-0.03	0.08	0.34	-0.05	-0.27	-0.04	-0.20	0.08	0.34
3	0.03	0.18	0.06	0.32	0.09	0.38	0.02	0.10	0.04	0.20	0.09	0.38
4	0.07	0.38	0.11	0.55	0.10	0.44	0.07	0.38	0.09	0.45	0.09	0.41
5	0.11	0.56	0.13	0.67	0.11	0.46	0.10	0.49	0.14	0.71	0.11	0.47
6	0.13	0.64	0.15	0.80	0.11	0.49	0.13	0.67	0.16	0.83	0.10	0.46
7	0.17	0.87	0.18	0.92	0.11	0.50	0.17	0.85	0.19	1.00	0.10	0.47
8	0.21	1.06	0.20	1.08	0.11	0.51	0.22	1.09	0.22	1.16	0.13	0.57
9	0.29	1.43	0.25	1.35	0.12	0.55	0.30	1.48	0.27	1.42	0.11	0.52
High	0.54	2.78	0.46	2.69	0.14	0.63	0.54	2.91	0.50	2.70	0.14	0.66
H-L	0.82	7.06	0.75	7.20	0.08	1.23	0.82	7.20	0.83	6.96	0.08	1.16
Turnover	677%		636%		60%		681%		663%		60%	
Value Weight												
	CaiT Only						CNN Only					
	5D/5R		20D/5R		5D/60R		5D/5R		20D/5R		5D/60R	
	ret	SR	ret	SR	ret	SR	ret	SR	ret	SR	ret	SR
Low	-0.02	-0.13	-0.02	-0.12	0.05	0.25	-0.03	-0.17	-0.05	-0.26	0.05	0.27
2	0.03	0.15	0.04	0.20	0.08	0.38	0.01	0.03	0.02	0.11	0.06	0.30
3	0.05	0.28	0.04	0.23	0.07	0.36	0.03	0.19	0.03	0.18	0.06	0.34
4	0.04	0.24	0.06	0.35	0.06	0.33	0.06	0.33	0.05	0.28	0.07	0.38
5	0.07	0.40	0.07	0.40	0.08	0.40	0.08	0.43	0.07	0.38	0.07	0.36
6	0.07	0.42	0.07	0.37	0.07	0.38	0.06	0.36	0.07	0.39	0.09	0.48
7	0.09	0.52	0.09	0.48	0.08	0.43	0.12	0.63	0.10	0.59	0.08	0.44
8	0.11	0.58	0.10	0.58	0.08	0.46	0.14	0.74	0.11	0.59	0.10	0.56
9	0.14	0.76	0.12	0.66	0.10	0.58	0.16	0.78	0.12	0.68	0.08	0.53
High	0.21	1.00	0.15	0.87	0.10	0.59	0.21	0.95	0.17	0.88	0.10	0.58
H-L	0.23	1.64	0.17	1.48	0.05	0.51	0.24	1.54	0.22	1.64	0.04	0.44
Turnover	760%		712%		61%		758%		731%		62%	

Description. The table compares CaiT-based and CNN-based portfolio strategies, with the left side showing CaiT results and the right side showing CNN results for both equal-weighted and value-weighted portfolios. It presents annualized average period returns (ret) and annualized Sharpe ratios (SR) across decile portfolios. The High-Low (H-L) spread represents the performance difference between the highest and lowest deciles. Monthly turnover rates are also reported at the bottom.

its potential advantages when incorporating more extensive historical data.

Table 5 presents the transaction cost-adjusted performance of equal-weighted CNN-based portfolios, reporting returns and Sharpe ratios for the Low and High deciles, as well as the H-L portfolio, across different prediction horizons and image sizes. The two highest Sharpe ratios, 4.28 for 5D/5R and 4.25 for 20D/5R, correspond to the strategies that also exhibited the highest Sharpe ratios before transaction costs, indicating that short-term strategies remain robust even after accounting for trading costs. Among monthly strategies, the highest Sharpe ratio of 1.61 is observed for 5D/20R, while for quarterly strategies, 5D/60R achieves the best Sharpe ratio of 0.81. These results confirm that CNN-based strategies withstand transaction costs and continue to generate positive returns and Sharpe ratios.

5.3 Transaction Costs

In Sections 5.1 and 5.2, I demonstrated that both CNN- and CaiT-based strategies generate high Sharpe ratios alongside high turnovers. In this section, I present the prediction results after incorporating transaction costs into portfolio construction. To account for these costs, I follow the approach of (Jiang et al., 2023), which is based on the methodology outlined by (Ke et al., 2021). Ke et al. (2021) assume that each asset incurs a daily transaction cost of 10 basis points (bps) for large stocks (those exceeding the 80th percentile of the NYSE size distribution) and 20 bps for smaller stocks (those below the 80th percentile; Ke et al. (2021, p. 31). This approach is motivated by (Frazzini et al., 2018), who analyze $1.7 trillion in live institutional trades and find average trading costs of 9.93 bps for large stocks and 20.18 bps for small stocks Frazzini et al. (2018, p. 53). Therefore, I adopt these transaction cost adjustments in my replication to ensure consistency with (Jiang et al., 2023) and prior literature on empirical trading cost estimation.

Table 5 presents the transaction cost-adjusted performance of equal-weighted CNN-based portfolios, reporting returns and Sharpe ratios for the Low and High deciles, as well as the H-L portfolio, across different prediction horizons and image sizes. The two highest Sharpe ratios, 4.28 for 5D/5R and 4.25 for 20D/5R, correspond to the strategies that also exhibited the highest Sharpe ratios before transaction costs, indicating that short-term strategies remain robust even after accounting for trading costs. Among monthly strategies, the highest Sharpe ratio of 1.61 is observed for 5D/20R, while for quarterly strategies, 5D/60R achieves the best Sharpe ratio of 0.81. These results confirm that CNN-based strategies withstand transaction costs and continue to generate positive returns and Sharpe ratios.

Table 6 presents the transaction cost-adjusted performance of CaiT-based strategies, structured in the same format as Table 5 for CNN. The results show that all CaiT-based strategies survive transaction costs, continuing to generate positive returns and Sharpe ratios. CaiT-5D/5R has a Sharpe ratio of 4.18, and CaiT-20D/5R has a Sharpe ratio of 4.05, both of which are slightly lower than their CNN counterparts. In contrast, CaiT-5D/60R strategy achieves a Sharpe ratio of 0.87, which is slightly higher than the CNN-based 5D/60R.

5.4 Average model results

As previously explained, CNN models generate probability estimates for individual stocks at each date, indicating the likelihood of a positive return. Similarly, CaiT models compute the same probability for the same stocks on the same dates, making a direct comparison between their outputs possible. Due to this similarity, their probabilities can also be combined. For CNN models, five separate models are trained on the same dataset, and their resulting probabilities are averaged for each stock at each date. These final averaged probabilities are then used to construct portfolios throughout the paper. Following the same approach, I take the simple average of the final CNN probabilities (average of 5 models) and the CaiT probabilities to create a new set of probabilities. In this section, I present the performance of portfolios constructed using these simple average probabilities. To ensure clarity, and consistency with Section 5.2, I prefix these models with "Average-" to distinguish them from standalone CNN and CaiT results. Additionally, I compare these "Average" results with CNN-based portfolios to assess the impact of integrating CaiT probabilities with CNN outputs.

The objective of this analysis is to assess the level of agreement between the CNN and CaiT models, particularly in the extreme decile portfolios. To illustrate this concept, consider a simple example with three stocks: A, B, and C. Suppose the CNN model estimates the probabilities of a positive return as 0.5 for A, 0.6 for B, and 0.9 for C, while the CaiT model assigns probabilities of 0.9 for A, 0.7 for B, and 0.8 for C. If we compute the average probabilities for each stock, we obtain 0.7 for A, 0.65 for B, and 0.85 for C. When ranked, Stock C remains the highest, as both models assign it a relatively high probability, suggesting strong agreement. Stock A ranks second, as it receives a high probability from at least one model. Stock B ranks lowest, since it does not receive consistently high probabilities from either model.

Extending this logic to the full stock universe, at each date, stocks that receive high probabilities from both models are expected to be positioned in the higher deciles, as decile portfolios are sorted based on the average probabilities of the stocks. Stocks that receive a high probability from one model but only a moderate probability from the other are likely to follow. Similarly, stocks assigned low probabilities by both models will be concentrated in the lower deciles. By averaging the probabilities from the two models, this approach provides a framework to evaluate how the degree of agreement between CNN and CaiT affects decile portfolio performance. Agreement in this context refers to both models assigning consistently high or low probabilities to the same stocks. Therefore, this analysis will focus primarily on

[5] modified, taken from Jiang et al. (2023, p. 3207)

Table 5: Performance of CNN-based Strategies After Transaction Costs[5]

	5D/5R		20D/5R			
	Ret	SR	Ret	SR		
Low	-0.10	-0.70	-0.16	-0.98		
High	0.38	2.04	0.35	1.88		
H-L	0.48	4.28	0.51	4.25		
Turnover	681%		663%			
	5D/20R		20D/20R		60D/20R	
	Ret	SR	Ret	SR	Ret	SR
Low	0.03	0.19	0.02	0.10	0.04	0.16
High	0.17	0.91	0.14	0.83	0.11	0.80
H-L	0.14	1.61	0.12	1.38	0.07	0.53
Turnover	176%		174%		152%	
	5D/60R		20D/60R		60D/60R	
	Ret	SR	Ret	SR	Ret	SR
Low	0.07	0.30	0.07	0.27	0.07	0.26
High	0.13	0.61	0.11	0.59	0.11	0.63
H-L	0.06	0.81	0.04	0.27	0.04	0.25
Turnover	60%		59%		58%	

Description. The table shows the performance of CNN-based strategies after transaction costs, with weekly strategies at the top, monthly in the middle, and quarterly at the bottom. It reports returns (Ret), Sharpe ratios (SR), for High, Low and H-L portfolios for each configuration. Monthly turnover rates are also reported.

Table 6: Performance of CaiT-based Strategies After Transaction Costs

	5D/5R		20D/5R		5D/60R	
	Ret	SR	Ret	SR	Ret	SR
Low	-0.10	-0.68	-0.11	-0.63	0.07	0.27
High	0.38	1.96	0.31	1.82	0.12	0.58
H-L	0.48	4.18	0.42	4.05	0.06	0.87
Turnover	677%		636%		60%	

Description. The table shows the performance of CaiT-based strategies after transaction costs. It reports returns (Ret), Sharpe ratios (SR), for High, Low and H-L portfolios for each configuration. Monthly turnover rates are also reported.

the High and Low deciles, as well as the H-L decile, to determine whether stocks with extreme probabilities in both models exhibit stronger predictive power compared to CNN, which relies on a single probability estimate.

Table 7 presents the decile portfolio performance of the Average model for 5D/5R, 20D/5R, and 5D/60R, comparing it with the respective CNN-only results. The findings indicate that, in both equal-weighted and value-weighted portfolios, the Average model consistently exhibits slightly higher H-L returns and Sharpe ratios compared to CNN-only portfolios.

For Average-5D/5R in equal-weighted portfolios, the H-L portfolio achieves a return of 0.86 and a Sharpe ratio of 7.25, slightly outperforming CNN-5D/5R, which records an H-L return of 0.82 and a Sharpe ratio of 7.20. The primary differences between the Average model and CNN-only portfolios are concentrated in the Low and High deciles, where the Average model tends to improve the performance of High decile stocks while slightly reducing the performance of Low decile stocks.

A similar trend is observed in Average-20D/5R, where the H-L return and Sharpe ratio are marginally higher than those of CNN-only portfolios. The differences are again most pronounced in the extreme deciles, with the Low decile portfolio showing a slightly lower return and Sharpe ratio, while the High decile portfolio exhibits a marginally stronger performance compared to CNN-only results.

For Average-5D/60R, the H-L return and Sharpe ratio are again slightly higher than those of CNN, although the differences are minimal. Within the High decile portfolio, the returns and Sharpe ratios remain identical to CNN-only results, whereas the Low decile portfolio exhibits a slightly lower Sharpe ratio.

To assess whether the observed differences between the CNN-only H-L portfolios and the Average H-L portfolios are statistically significant or merely due to randomness, I conducted a paired t-test for 5D/60R, 5D/5R, and 20D/5R. The results indicate varying levels of statistical significance across different prediction horizons.

For 5D/60R, the t-statistic of 1.19 and p-value of 0.24 suggest that the difference between the CNN-only and Average

Table 7: Comparison of Average Model and CNN-based Portfolio Performance

Equal Weight												
	Average model						CNN only					
	5D/5R		20D/5R		5D/60R		5D/5R		20D/5R		5D/60R	
	ret	SR	ret	SR	ret	SR	ret	SR	ret	SR	ret	SR
Low	-0.30	-2.08	-0.35	-2.14	0.05	0.22	-0.27	-1.91	-0.33	-2.06	0.06	0.25
2	-0.04	-0.24	-0.03	-0.16	0.07	0.32	-0.05	-0.27	-0.04	-0.20	0.08	0.34
3	0.02	0.11	0.04	0.23	0.09	0.39	0.02	0.10	0.04	0.20	0.09	0.38
4	0.07	0.40	0.11	0.55	0.10	0.43	0.07	0.38	0.09	0.45	0.09	0.41
5	0.10	0.50	0.13	0.68	0.10	0.42	0.10	0.49	0.14	0.71	0.11	0.47
6	0.14	0.69	0.16	0.83	0.11	0.51	0.13	0.67	0.16	0.83	0.10	0.46
7	0.17	0.83	0.18	0.96	0.11	0.50	0.17	0.85	0.19	1.00	0.10	0.47
8	0.22	1.09	0.22	1.14	0.12	0.54	0.22	1.09	0.22	1.16	0.13	0.57
9	0.30	1.46	0.27	1.42	0.12	0.54	0.30	1.48	0.27	1.42	0.11	0.52
High	0.57	2.96	0.51	2.84	0.14	0.66	0.54	2.91	0.50	2.70	0.14	0.66
H-L	0.86	7.25	0.86	7.31	0.09	1.21	0.82	7.20	0.83	6.96	0.08	1.16
Turnover	674%		648%		60%		681%		663%		60%	
Value Weight												
	Average model						CNN only					
	5D/5R		20D/5R		5D/60R		5D/5R		20D/5R		5D/60R	
	ret	SR	ret	SR	ret	SR	ret	SR	ret	SR	ret	SR
Low	-0.03	-0.19	-0.06	-0.36	0.05	0.25	-0.03	-0.17	-0.05	-0.26	0.05	0.27
2	0.00	0.02	0.01	0.06	0.05	0.25	0.01	0.03	0.02	0.11	0.06	0.30
3	0.03	0.18	0.04	0.19	0.07	0.37	0.03	0.19	0.03	0.18	0.06	0.34
4	0.05	0.29	0.05	0.30	0.08	0.41	0.06	0.33	0.05	0.28	0.07	0.38
5	0.07	0.41	0.06	0.32	0.07	0.38	0.08	0.43	0.07	0.38	0.07	0.36
6	0.07	0.40	0.06	0.35	0.08	0.40	0.06	0.36	0.07	0.39	0.09	0.48
7	0.10	0.54	0.09	0.51	0.09	0.47	0.12	0.63	0.10	0.59	0.08	0.44
8	0.12	0.66	0.11	0.63	0.08	0.50	0.14	0.74	0.11	0.59	0.10	0.56
9	0.17	0.85	0.13	0.70	0.09	0.52	0.16	0.78	0.12	0.68	0.08	0.53
High	0.22	1.01	0.17	0.90	0.10	0.61	0.21	0.95	0.17	0.88	0.10	0.58
H-L	0.25	1.63	0.23	1.79	0.05	0.52	0.24	1.54	0.22	1.64	0.04	0.44
Turnover	759%		726%		62%		758%		731%		62%	

Description. This table compares the performance of the Average model and CNN-based portfolios for equal-weighted and value-weighted strategies across different configurations. It reports average period returns (ret), Sharpe ratios (SR), High-Low (H-L) portfolio, and turnover rates.

H-L portfolios is not statistically significant at conventional levels. This implies that incorporating CaiT probabilities does not lead to a meaningful improvement over CNN for 5D/60R.

In contrast, for 5D/5R, the results are highly significant, with a t-statistic of 8.33 and a p-value of 2.72×10^{-16}, indicating a strong rejection of the null hypothesis. This suggests that averaging CNN and CaiT probabilities leads to a statistically significant improvement in portfolio performance, reinforcing the effectiveness of the Average model in short-term prediction horizons. Similarly, for 20D/5R, the t-statistic of 4.43 and p-value of 1.05×10^{-5} confirm that the Average model significantly outperforms the CNN-only model, albeit with slightly lower statistical strength than in the 5D/5R case.

To conclude, I took the simple average of CNN and CaiT probability estimates and sorted them into deciles. This approach does not necessarily provide insight into the effect of CaiT probabilities in the middle deciles, as changes in decile portfolio performance cannot be directly attributed to either CaiT or CNN. However, for the High and Low deciles, the impact of CaiT is more observable because stocks have only two possible directions—remaining in the decile or being replaced. At each portfolio formation date, CNN assigns certain stocks to the High and Low deciles. In the High decile, when probabilities are averaged, CaiT either confirms the selection by also assigning high probabilities or replaces some stocks with new ones. The same logic applies to the Low decile, where CaiT may either reinforce the selection of low-probability stocks or introduce new ones. Based on this, we can hypothesize that for Average-5D/5R and Average-20D/5R, the higher returns and Sharpe ratios compared to CNN-only portfolios suggest that the stocks introduced by CaiT have higher average returns than the stocks it removes. Similarly, in the Low decile, the reverse occurs, where the stocks CaiT replaces tend to have lower returns than those originally selected by CNN.

This effect also applies to CaiT itself. When comparing the results from Section 5.2, we observe that the H-L portfolio of the Average Model outperforms the CaiT-only portfolio, indicating that adding CNN probabilities has a positive impact on the returns and Sharpe ratio of the CaiT-only H-L portfolio.

To summarize, averaging CNN and CaiT probabilities results in H-L returns that are slightly higher than those of CNN-only portfolios. While the differences in returns are small, for 5D/5R and 20D/5R, these differences are highly statistically significant.

6 Investment strategy

In this section, I introduce the investment strategy developed using both CNN and CaiT predictions. The strategy involves going long on stocks that both models identify as most likely to increase in value and shorting stocks that both models predict as most likely to decline. In Section 6.1, I will discuss the investment thesis, explaining the rationale behind this strategy and how it was formulated based on the findings from previous sections, particularly how the agreement between the two models strengthens stock selection and enhances portfolio construction. In Section 6.2, I will evaluate the out-of-sample performance of the strategy, analyzing its effectiveness beyond the training period and assessing its robustness in real-world market conditions.

6.1 Investment Thesis

In Section 5.2, I compared CaiT-only results with CNN-only results. For 5D/5R, both models had the same H-L returns, but CNN had a higher Sharpe ratio. For 20D/5R, CaiT had lower H-L returns but a higher Sharpe ratio. These indicate that CNN and CaiT capture and leverage different characteristics when analyzing the same stock data. In Section 5.4, I averaged the probability estimates of both models and sorted stocks into deciles based on these average probabilities. The goal was to assess how the portfolios performed when there was some degree of agreement between the models, though this agreement was only implicitly defined. By taking the simple average, agreement was inferred by stocks with high probabilities from both models being placed in higher deciles, while stocks with low probabilities from both models were sorted into lower deciles. The results showed that Average models consistently produced higher H-L returns and Sharpe ratios than CNN-only results across all three model configurations. This suggests that combining both models' probability estimates has the potential to improve portfolio performance. To summarize, my results show that CNN and CaiT construct portfolios differently, and combining their outputs can improve performance. To achieve this, I will integrate both models' outputs by selecting stocks where both models strongly agree. However, directly using the probability estimates from both models is not ideal for two key reasons.

The first reason is label smoothing (Szegedy et al., 2016), a regularization technique applied in the CaiT architecture. As explained in Section 4.5, both models optimize their probability estimates by minimizing the same loss function, cross-entropy. However, in the CaiT model, label smoothing redistributes a small portion of the predicted probability mass from the most confident class to the other class, preventing overconfidence and improving model convergence. This means that CaiT outputs are inherently less extreme than CNN outputs, as label smoothing systematically reduces the probability assigned to the predicted class. Because of this regularization, the probability estimates produced by CaiT are not directly comparable to those from CNN, which does not use label smoothing. While CNN may generate more extreme probabilities, CaiT produces smoother, more calibrated probabilities. As a result, directly averaging or combining these probabilities without considering this difference could introduce bias in portfolio construction. Therefore, an alternative approach is necessary to properly integrate both models' outputs while accounting for this systematic difference in probability distributions.

The second reason why directly using the probability estimates from both models is not ideal is related to the concept

of confidence in these probabilities. Both CaiT and CNN architectures output two probability estimates: one representing the likelihood of the stock going up and the other representing the likelihood of the stock going down. Since these two probabilities sum to one, the structure inherently makes 0.5 the weakest confidence signal. If a model assigns a 0.5 probability for a stock to go up, it also assigns 0.5 probability for it to go down, indicating no clear preference and a high degree of uncertainty in the prediction. In contrast, a higher up probability, such as 0.7, represents a stronger confidence that the stock will increase in value compared to 0.6, just as a lower up probability, such as 0.3, represents stronger confidence in a decline compared to 0.4. This is because a 0.3 up probability corresponds to a 0.7 down probability, whereas a 0.4 up probability corresponds to a 0.6 down probability.

Therefore, directly using probability estimates without considering confidence means incorporating weak signals into the decision-making process. Throughout this paper, as well as in (Jiang et al., 2023), portfolio construction is based on sorting stocks into deciles using up probabilities. Under this approach, a stock with an up probability of 0.5 would be ranked above a stock with an up probability of 0.1, despite the fact that 0.5 actually represents maximum uncertainty rather than confidence in an upward movement. This misinterpretation of weak signals can affect portfolio performance. To better understand this impact, I conducted an analysis examining how portfolio performance changes when only more confident signals are used.

Chen et al. (2016) applied CNNs for time series analysis of the Taiwan Stock Index Futures, setting a confidence threshold of 0.65, where only predictions exceeding this threshold were considered valid for classification, while those below were discarded due to insufficient confidence Chen et al. (2016, p. 89). Inspired by this approach, I also incorporate confidence-based filtering but, instead of applying a fixed threshold, I use percentile-based filtering to observe how portfolio performance changes gradually as less confident signals are removed.

To achieve this, I first analyze the entire probability distribution for 5D/60R, 5D/5R, and 20D/5R across the out-of-sample period so there is a look-ahead bias in these results. I then iteratively remove probabilities between specific percentiles, starting with the 45th to 55th percentile, then expanding the exclusion range to 40th to 60th percentile, continuing this process until only the most confident signals—those in the 5th and 95th percentiles—remain. For each filtered dataset, I sort stocks into deciles using both CNN probabilities and the average of CNN and CaiT probabilities. This allows me to first examine how restricting the dataset to more confident CNN signals affect portfolio performance, and then evaluate the impact of incorporating CaiT probabilities into these subsets of confident CNN signals.

Figure 6 visualizes the H-L returns and Sharpe ratios for 5D/60R, 5D/5R, and 20D/5R under different percentile filtering thresholds. The figure also compares CNN portfolios with those constructed using average CNN and CaiT probabilities, providing insights into how increasing confidence thresholds impact portfolio returns and whether integrating CaiT probabilities enhances performance across varying confidence levels.

For 5D/60R, CNN H-L returns start at 0.08 and exhibit a gradual increase up to 0.13 at the 15th-85th percentile filter. However, after this point, returns decline slightly, ending at 0.125 in the 5th-95th percentile. In contrast, when CNN probabilities are averaged with CaiT probabilities, returns increase more consistently at each filtering step, starting at 0.085 and reaching 0.16 at the 5th-95th percentile. Regarding Sharpe ratios, CNN results do not show a clear upward trend, with most portfolios falling below the initial Sharpe ratio, except for the jump at 15th-85th. Meanwhile, the Average model's Sharpe ratios are mostly above the starting value and consistently outperform their CNN-only counterparts, though there is a slight decline at the final filtering level.

For 5D/5R and 20D/5R, the trend is notably different. Both exhibit a clear and consistent increase in H-L returns and Sharpe ratios for CNN and the Average model. The Average model consistently outperforms CNN portfolios at every filtering level. For 5D/5R, CNN starts with an H-L return of 0.82 and a Sharpe ratio of 7.2, which gradually increases to 1.7 in returns and 9.2 in Sharpe ratio. The Average model starts at 0.86 with a Sharpe ratio of 7.25 and increases to 1.90 in returns and 9.92 in Sharpe ratio, maintaining higher values than CNN-only portfolios at each filtering step. A similar pattern is observed for 20D/5R, where CNN begins with an H-L return of 0.83 and the Average model starts at 0.86. As filtering progresses, their returns increase to 1.99 and 2.17, respectively, while their Sharpe ratios rise from 6.95 and 7.30 to 9.19 and 9.98, respectively.

These findings are important because they demonstrate that gradually filtering out weaker probability signals leads to improved H-L returns and Sharpe ratios. Additionally, the Average model consistently outperforms CNN, reinforcing the value of integrating CaiT predictions. This further supports the hypothesis that removing uncertain predictions enhances model effectiveness and that combining CNN and CaiT probabilities produces superior portfolio performance across various confidence thresholds.

Based on these findings, I propose an investment strategy that selectively integrates confident signals from both CNN and CaiT models, using their probability estimates. This strategy is designed to identify strong agreement between the models while filtering out less confident signals. At each portfolio creation date, the predicted probabilities from both CNN and CaiT models are obtained. Each model independently ranks all stocks, sorting them into deciles based on their probability estimates. After this sorting, stocks that fall into the highest decile in both models, indicating that both CNN and CaiT assign them among the highest probabilities for that date, are selected for long positions. Similarly, stocks that fall into the lowest decile in both models, meaning that both models classify them as among the least likely to perform well, are selected for short positions. All selected stocks are equally weighted within the portfolio.

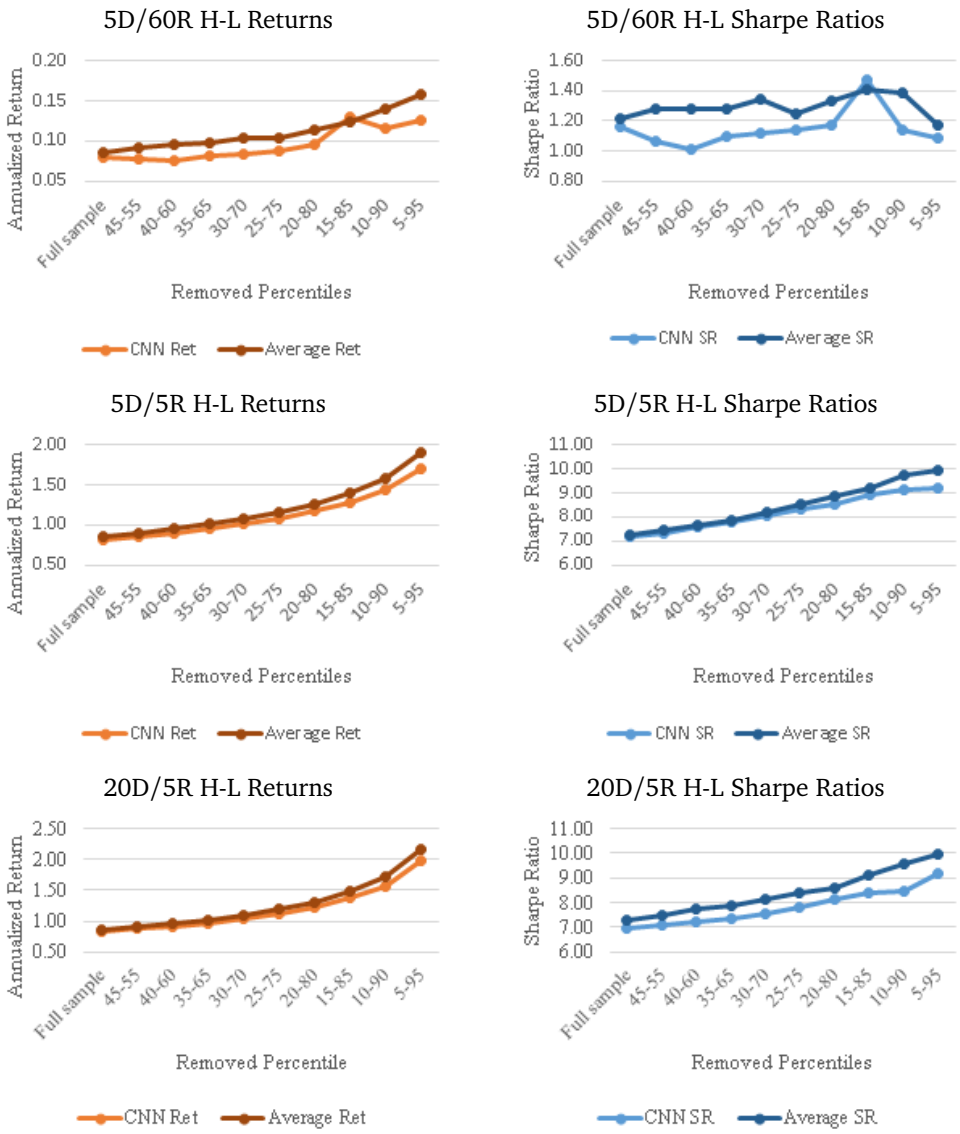

Figure 6: Effect of Confidence Filtering on H-L Returns and Sharpe Ratios

Description. The figure shows how H-L returns and Sharpe ratios change as low-confidence CNN probabilities are filtered. The top, middle, and bottom rows represent 5D/60R, 5D/5R, and 20D/5R, respectively. The CNN model is compared to the Average model, which applies filtering based on CNN probabilities. The x-axis shows removed probability percentiles, with "Full Sample" meaning all stocks are included. Percentile ranges indicate the stocks within those ranges are filtered out.

This approach offers two key advantages. First, it ensures that both models strongly agree on the highest and lowest deciles, making it possible to effectively combine their predictive strengths. At each rebalancing date, CNN captures local features while CaiT identifies global patterns, and only stocks that both models consistently classify as having the highest probability to rise or fall are included in the portfolio. This creates a stronger agreement than simple probability averaging, as it directly aligns the models' most confident predictions.

Second, this strategy acts as an implicit confidence filter. While it does not impose an absolute probability threshold, it inherently removes weaker signals by requiring that stocks be ranked in the top or bottom 10% of probabilities within both models. This ensures that even if a signal is not exceptionally strong, it must still be among the most confident signals relative to the full stock universe. Additionally, since probabilities are ranked within each model, label smoothing does not distort rankings, as a confident signal remains highly ranked even if its absolute probability estimate is slightly adjusted.

This investment strategy aligns with my empirical results, which demonstrate that removing weaker probability signals improves performance, and that combining CNN and CaiT probabilities leads to superior portfolio returns and Sharpe ratios. By ensuring that only high-confidence, mutually agreed-upon stocks are included, this strategy leverages the

complementary strengths of both models, leading to a more robust and reliable stock selection process.

6.2 Investment Performance

In this section, I analyze the performance of the proposed investment strategy by evaluating its gross returns, the impact of transaction costs, and cumulative performance over time. The strategy leverages both CNN and CaiT models to identify high-confidence long and short positions, ensuring that only stocks with strong agreement between the models are included. The results demonstrate that this approach consistently outperforms CNN portfolios and remains robust even after accounting for transaction costs, while its cumulative performance trajectory shows sustained long-term profitability.

Table 8 presents the raw performance of the investment strategy across different configurations (5D/5R, 20D/5R, and 5D/60R). The strategy delivers significant positive returns and high Sharpe ratios across all horizons. The strategy portfolio returns are 1.05 for 5D/5R, 1.14 for 20D/5R, and 0.11 for 5D/60R, with corresponding Sharpe ratios of 8.00, 8.29, and 1.33, respectively. These figures indicate that the strategy consistently generates strong risk-adjusted returns, with shorter-horizon models (5D/5R and 20D/5R) delivering the highest performance. Compared to CNN portfolios, these results show a clear improvement in both return levels and Sharpe ratios, highlighting the advantage of integrating CaiT's global feature extraction with CNN's local pattern recognition.

Table 9 assesses the strategy's performance after accounting for transaction costs. Despite the high turnover rates of 708% for 5D/5R and 700% for 20D/5R, the strategy remains highly profitable. The strategy portfolio returns after transaction costs are 0.70 for 5D/5R, 0.80 for 20D/5R, and 0.08 for 5D/60R, with Sharpe ratios of 5.37, 5.82, and 1.04, respectively. While transaction costs reduce returns, the strategy still outperforms CNN and maintains strong positive Sharpe ratios, demonstrating its resilience to trading frictions.

Figure 7 provides further validation by displaying the cumulative volatility-adjusted log returns of the strategy compared to various benchmarks, including CNN H-L portfolios, Momentum, Short-term reversal and Weekly Short-term Reversal strategies, and the S&P 500 (SPY). The proposed strategy exhibits the highest cumulative return trajectory, consistently outperforming both CNN models and other trend-based and market-based benchmarks. This sustained outperformance reinforces the effectiveness of combining CNN and CaiT models in a structured investment framework.

Overall, these results confirm that the proposed strategy delivers superior returns and risk-adjusted performance compared to CNN models. Even after incorporating transaction costs, the investment strategy remains highly profitable, highlighting its practical feasibility in real-world trading environments. The combination of deep learning-based stock selection and confidence-based filtering enhances portfolio efficiency, making it a robust and effective approach to quantitative investing.

[6] Modified, taken from Jiang et al. (2023, p. 3208)

7 Discussion and Limitations

In this section, I will discuss my findings and acknowledge the study's limitations. This paper aims to achieve three main objectives. The first objective is to replicate the study by (Jiang et al., 2023) to evaluate whether CNNs can autonomously extract meaningful features from stock chart images and whether these features possess predictive power. The second objective is to assess whether implementing a Vision Transformer architecture, specifically CaiT, using the same data and structure as the CNN model, can produce superior results. The final objective is to determine whether combining CNN and Vision Transformer predictions in an investment strategy can enhance the performance beyond that of standalone CNN or CaiT models. The discussion will be structured around these three key points.

Jiang et al. (2023) find that CNN-based strategies exhibit strong predictive power, particularly over short-term horizons, such as on a weekly basis. Their results indicate that the lowest decile portfolios tend to have substantially negative Sharpe ratios, while the highest decile portfolios demonstrate significantly positive values. Furthermore, they observe a monotonic increase in Sharpe ratios from the lowest to the highest decile Jiang et al. (2023, p. 3207).

My findings corroborate these results. In weekly strategies, CNN-based models exhibit Sharpe ratios of approximately -2.0 in the lowest decile, while the highest decile exceeds 2.7. Additionally, I observe a similar monotonic increase across deciles. Jiang et al. (2023) also find that image-based strategies generate highly effective trading signals, outperforming traditional price-based signals, albeit at the cost of significantly higher turnover Jiang et al. (2023, p. 3210). My results confirm that weekly CNN-based models achieve superior portfolio performance, with the highest H-L Sharpe ratio for equal-weighted portfolios reaching 7.2, compared to 2.91 for the best-performing traditional trend-following strategy (WSTR). However, these high returns are accompanied by considerable portfolio turnover, which is slightly higher than that of WSTR, nearly four times greater than MOM, and twice that of the STR.

To determine whether the outperformance of CNN-based strategies is primarily driven by high turnover, (Jiang et al., 2023) extend their analysis to longer trading horizons, examining monthly and quarterly portfolios. Their results indicate that CNN-based strategies continue to outperform traditional signals even at reduced turnover levels. Furthermore, they find that CNN-based models achieve higher Sharpe ratios than momentum strategies with comparable turnover levels Jiang et al. (2023, p. 3211). My findings align with these conclusions. Monthly and quarterly CNN-based strategies—particularly 5D/20R, 20D/20R, and 5D/60R—consistently outperform traditional approaches over extended investment horizons. Additionally, quarterly CNN strategies maintain

Table 8: Investment Strategy Performance

	5D/5R		20D/5R		5D/60R	
	ret	SR	ret	SR	ret	SR
Short	-0.40	-2.86	-0.50	-3.09	0.04	0.18
Long	0.65	3.42	0.64	3.62	0.15	0.68
Strategy	1.05	8.00	1.14	8.29	0.11	1.33
Turnover	708%		700%		63%	

Description. The table presents the performance metrics of the investment strategy across three configurations: 5D/5R, 20D/5R, and 5D/60R. It reports returns (ret) and Sharpe ratios (SR) for short positions, long positions, and the strategy, which is formed as a long-short portfolio. Additionally, turnover rates are provided for each configuration, indicating the level of trading activity within the strategy.

Table 9: Investment Strategy Performance After Transaction Costs

	5D/5R		20D/5R		5D/60R	
	ret	SR	ret	SR	ret	SR
Short	-0.21	-1.55	-0.31	-1.93	0.06	0.23
Long	0.48	2.55	0.49	2.74	0.14	0.64
Strategy	0.70	5.37	0.80	5.82	0.08	1.04
Turnover	708%		700%		63%	

Description. The table presents the performance metrics of the investment strategy after accounting for transaction costs across three configurations: 5D/5R, 20D/5R, and 5D/60R. It reports returns (ret) and Sharpe ratios (SR) for short positions, long positions, and the strategy, which is structured as a long-short portfolio. Additionally, turnover rates are provided for each configuration, reflecting the level of trading activity after incorporating transaction costs.

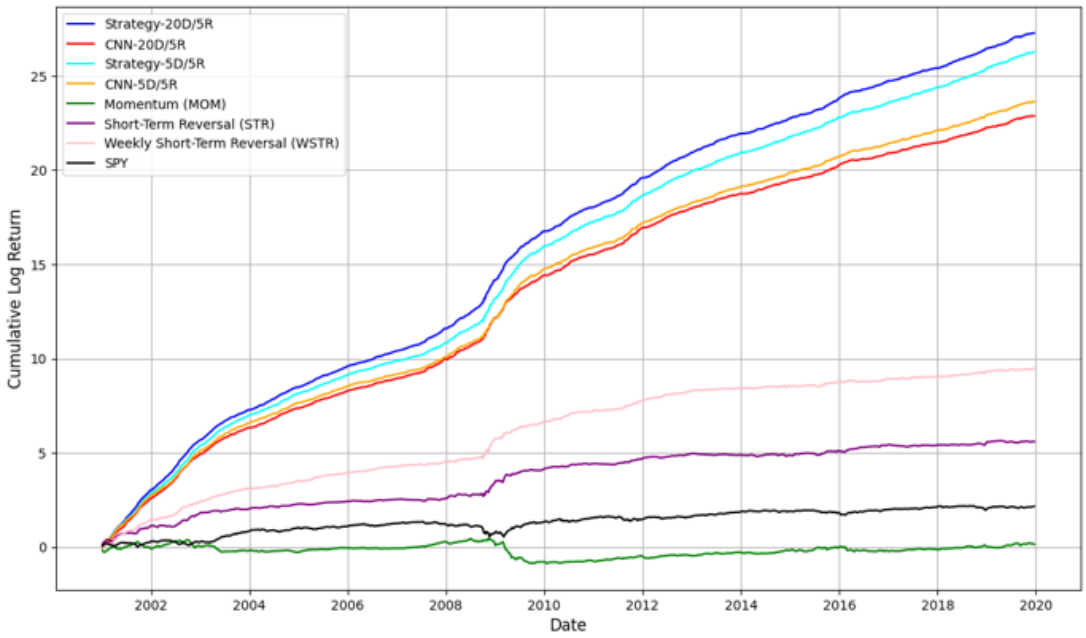

Figure 7: Cumulative Volatility Adjusted Returns of Equal-Weighted Portfolios[6]

Description. The figure presents the cumulative volatility-adjusted log returns of equal-weighted H-L portfolios for Strategy-20D/5R, CNN-20D/5R, Strategy-5D/5R, CNN-5D/5R, Momentum (MOM), Short-Term Reversal (STR), and Weekly Short-Term Reversal (WSTR). The SPY index is included for reference. All strategies have been adjusted to maintain the same volatility level as SPY over the test period.

turnover rates similar to those of monthly momentum strategies while still delivering superior Sharpe ratios for the H-L portfolio. Moreover, (Jiang et al., 2023) analyze transaction costs and cumulative log returns, concluding that CNN-based strategies remain profitable after accounting for transaction costs and continue to outperform traditional signals over time. My findings support this conclusion, as CNN-based strategies maintain their effectiveness even after transaction

cost adjustments and exhibit sustained cumulative outperformance compared to traditional approaches, as shown in Table 5 and Figure 7.

It is important to note that the primary objective of (Jiang et al., 2023) is not to develop a highly accurate return prediction model but rather to assess the predictive capabilities of image-based machine learning models in financial markets Jiang et al. (2023, p. 3204). These findings reinforce the conclusion that CNN-based strategies exhibit strong predictive power across both short-term and long-term horizons.

Now, I turn to the CaiT perspective to evaluate whether Vision Transformers enhance predictive performance compared to CNNs. The motivation for implementing a Vision Transformer architecture stems from its ability to capture global context, whereas CNNs are more effective at extracting local features. Given this distinction, CaiT may be better suited for identifying complex relationships in larger image contexts. To test this hypothesis, I specifically examine 5D/5R and 20D/5R configurations, as both models analyze the same stocks on the same dates but differ in the length of historical data included in their images. This setup results in four distinct models—CNN-5D/5R, CNN-20D/5R, CaiT-5D/5R, and CaiT-20D/5R—all making predictions on the exact same stocks. Therefore, I will compare their respective strengths in processing the same stocks. In addition to discussing returns and Sharpe ratios, I will also mention standard deviations to provide a more comprehensive assessment of risk. Furthermore, my discussion will focus exclusively on equal-weighted portfolios.

In Table 4, in the Low decile, CNN-20D/5R predicts the lowest returns and exhibits the lowest Sharpe ratio of -2.06, whereas CaiT-20D/5R achieves the highest Sharpe ratio among the models at -1.69. This suggests that, for the same set of stocks, CNN provides the most effective predictions in the lowest decile, when utilizing 20-day images.

For the High decile, CNN-5D/5R delivers the highest returns along with the highest Sharpe ratio, indicating that CNN remains the most effective predictor for stocks with strong upward momentum. However, an analysis of standard deviations in the High decile reveals that CaiT-20D/5R exhibits the lowest volatility among the four models. This suggests that while CaiT does not necessarily yield higher returns, it may be more effective at capturing global context, leading to more stable predictions with reduced variability. This ability to categorize stocks that move together more effectively could be one of CaiT's key strengths.

When evaluating Sharpe ratios of H-L portfolios, CNN-5D/5R and CaiT-20D/5R emerge as the top-performing models. CNN-5D/5R excels in short-term assessments by effectively capturing local patterns, resulting in the highest returns. In contrast, CaiT-20D/5R demonstrates an ability to group stocks with similar movement patterns more effectively in larger images, contributing to a more stable portfolio composition. This highlights the complementary nature of the two architectures—CNN excels in short-term feature extraction, while CaiT enhances global pattern recognition over extended time frames.

Recognizing that CNN and CaiT extract complementary information from stock chart images—with CNN focusing on predictive patterns that drive returns and CaiT emphasizing volatility reduction and stability—leads to a critical question: Can these two architectures be effectively combined to create an investment strategy that leverages their respective strengths? A key insight comes from simple probability averaging. In Table 7, the Average-20D/5R model achieved a higher H-L Sharpe ratio (7.31) compared to Average-5D/5R. This improved Sharpe ratio appears to stem from two distinct effects: a lower Sharpe ratio in the Low decile, likely attributed to CNN's ability to identify underperforming stocks, and a lower standard deviation in the High decile, potentially driven by CaiT's strength in capturing global patterns and reducing volatility. These observations suggest that combining CNN and CaiT predictions may create a more balanced and effective investment strategy, leveraging both architectures to optimize return potential while maintaining stability.

Another important aspect to consider is the role of confident signals. As demonstrated in Figure 6, filtering out less confident signals results in higher returns and Sharpe ratios. This effect may be attributed to two possible explanations. First, both CNN and CaiT architectures may struggle to accurately assess certain images, indicating a potential limitation in their ability to extract meaningful patterns from all stock charts. This could suggest that some price movements are inherently more difficult to predict using these models. Alternatively, a more speculative but intriguing possibility is that not all stock charts contain recognizable predictive patterns. Traditional technical analysis assumes that price patterns provide signals about future movements. However, the results may suggest that such patterns only appear in a specific subset of images, while others lack predictive features altogether. If this is the case, filtering out less confident predictions could serve as a way to identify and isolate the most informative patterns, improving overall strategy performance.

By combining both architectures, I develop an investment strategy that selects stocks classified in the High and Low deciles by both models. In Table 8, for the 20D/5R strategy, the portfolio exhibits higher returns on the Short leg and significantly lower standard deviation on the Long leg. This mirrors the patterns observed in Section 5.2 when evaluating individual models, in Section 5.4 when averaging their predictions. However, a key distinction arises in the investment strategy. The improved performance is not driven solely by one architecture but rather by the agreement between both models. Unlike previous observations, where I suggested that the effect might be attributed to a single model, in this case, both models independently select the same stocks. This alignment, rather than the influence of any individual model, is what ultimately drives the higher performance.

Yet, this also highlights certain limitations of the strategy. One key issue is that the agreement between models is purely mechanical—it is not integrated into the training process or loss function. A more advanced approach would involve designing an architecture that combines CNNs and Vision Transformers, allowing both models to influence the

learning process and jointly optimize parameters. Another limitation is that this strategy is inherently relative rather than absolute. It does not evaluate individual stocks in isolation but instead ranks them against each other. As a result, the strategy is ineffective in scenarios where the number of investable stocks is extremely limited—for instance, if only one stock were available, the strategy would not function. Finally, this study focuses on 5-day and 20-day images for CaiT, but 60-day images could provide new insights as the model may benefit from capturing longer-term dependencies in stock movements.

8 Conclusion

This study explores the predictive power of deep learning models applied to stock chart images, comparing Convolutional Neural Networks with Vision Transformers and analyzing their integration into an investment strategy. The research follows three main objectives: first, replicating (Jiang et al., 2023) to evaluate whether CNNs can autonomously extract predictive features from stock images; second, assessing whether CaiT, as a Vision Transformer model, can improve predictive performance by capturing global dependencies; and third, investigating whether combining CNN and CaiT predictions can enhance portfolio performance beyond standalone models. Using a dataset of stock images with 5-day, 20-day, and 60-day lookback periods, I trained multiple CNN and CaiT architectures, evaluated their returns, and Sharpe ratios, and examined their performance at different investment horizons.

For CNN-based strategies, my findings confirm those of (Jiang et al., 2023). CNN models effectively extract local features from stock images, generating profitable trading signals and strong Sharpe ratios, particularly in short-term horizons, while still maintaining predictive power in longer-term horizons, albeit to a lesser extent.

The study also examines CaiT's predictive capability, leveraging its strength in capturing global dependencies within stock chart images. While CNN excels at identifying high-return stocks, CaiT enhances portfolio stability by reducing volatility. Both models achieve the highest Sharpe ratios, but through different mechanisms: CNN focuses on shorter images, identifying stocks with strong return potential, whereas CaiT processes longer images, minimizing standard deviation. This suggests that CaiT may be more effective at identifying stocks that move together, contributing to lower portfolio volatility. These findings indicate that CNN and CaiT extract complementary features, supporting the hypothesis that combining both models could enhance predictive performance.

Building on these insights, I develop an investment strategy that integrates CNN and CaiT predictions by selecting stocks classified in the High and Low deciles by both models. This agreement-based approach effectively filters out weaker signals, leading to higher Sharpe ratios compared to using CNN or CaiT alone. Even after accounting for transaction costs, the strategy remains highly profitable, reinforcing its practical viability in real-world trading applications.

However, this strategy has limitations. The agreement between models is rule-based rather than learned during training. Additionally, the strategy is relative rather than absolute, meaning that it ranks stocks rather than predicting their absolute returns. This structure may limit its applicability in markets with fewer investable assets.

Overall, this research highlights the effectiveness of deep learning in financial prediction and demonstrates that CNN and CaiT offer strong predictive power focusing on complementary things. Future research could explore more sophisticated architectures that integrate both models into a unified framework, potentially improving performance beyond the agreement-based approach used here.

References

Allen, F., & Karjalainen, R. (1999). Using Genetic Algorithms to Find Technical Trading Rules. *Journal of Financial Economics*, *51*(2), 245–271. https://doi.org/10.1016/S0304-405X(98)00052-X

Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016). Layer Normalization. https://doi.org/10.48550/arXiv.1607.06450

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., Gulcehre, C., Song, F., Ballard, A., Gilmer, J., Dahl, G., Vaswani, A., Allen, K., Nash, C., Langston, V., et al. (2018). Relational Inductive Biases, Deep Learning, and Graph Networks. https://doi.org/10.48550/arXiv.1806.01261

Brock, W., Lakonishok, J., & LeBaron, B. (1992). Simple Technical Trading Rules and the Stochastic Properties of Stock Returns. *The Journal of Finance*, *47*, 1731–1764.

Brown, M. S., Pelosi, M. J., & Dirska, H. (2013). Dynamic-Radius Species-Conserving Genetic Algorithm for the Financial Forecasting of Dow Jones Index Stocks. In P. Perner (Ed.), *Machine Learning and Data Mining in Pattern Recognition* (pp. 27–41, Vol. 7988). Springer. https://doi.org/10.1007/978-3-642-39712-7_3

Cao, L. J., & Tay, F. H. (2003). Support Vector Machine with Adaptive Parameters in Financial Time Series Forecasting. *IEEE Transactions on Neural Networks*, *14*(6), 1506–1518. https://doi.org/10.1109/TNN.2003.820556

Chen, J.-F., Chen, W.-L., Huang, C.-P., Huang, S.-H., & Chen, A.-P. (2016). Financial Time-Series Data Analysis Using Deep Convolutional Neural Networks, 87–92. https://doi.org/10.1109/CCBD.2016.027

Cohen, N., Balch, T., & Veloso, M. (2020). Trading via Image Classification. https://doi.org/10.48550/arXiv.1907.10046

Courant, R., Edberg, M., Dufour, N., & Kalogeiton, V. (2023). Transformers and Visual Transformers. In O. Colliot (Ed.), *Machine Learning for Brain* (pp. 193–233). Humana Press.

Cowles, A. (1933). Can Stock Market Forecasters Forecast? *Econometrica*, *1*(3), 309. https://doi.org/10.2307/1907042

Darji, M. (2021). Implementing Bilinear Interpolation for Image Resizing. *Medium*. https://meghal-darji.medium.com/implementing-bilinear-interpolation-for-image-resizing-357cbb2c2722

Detzel, A., Liu, H., Strauss, J., Zhou, G., & Zhu, Y. (2021). Learning and Predictability via Technical Analysis: Evidence from Bitcoin and Stocks with Hard-to-Value Fundamentals. *Financial Management*, *50*(1), 107–137. https://doi.org/10.1111/fima.12310

Di Persio, L., & Honchar, O. (2016). Artificial Neural Networks Approach to the Forecast of Stock Market Price Movements. *International Journal of Economics and Management Systems*, *1*, 158–162.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2020). An Image Is Worth 16x16 Words: Transformers for Image Recognition at Scale. https://doi.org/10.48550/arXiv.2010.11929

Fama, E. F. (1970). Efficient Capital Markets: A Review of Theory and Empirical Work. *The Journal of Finance*, *25*(2), 383–417. https://doi.org/10.2307/2325486

Fama, E. F., & Blume, M. E. (1966). Filter Rules and Stock-Market Trading. *The Journal of Business*, *39*(1), 226–241.

Frazzini, A., Israel, R., & Moskowitz, T. J. (2018). Trading Costs. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.3229719

Gani, H., Naseer, M., & Yaqub, M. (2022). How to Train Vision Transformer on Small-Scale Datasets? https://doi.org/10.48550/arXiv.2210.07240

Glorot, X., & Bengio, Y. (2010). Understanding the Difficulty of Training Deep Feedforward Neural Networks.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.

Gu, S., Kelly, B., & Xiu, D. (2020). Empirical Asset Pricing via Machine Learning. *The Review of Financial Studies*, *33*(5), 2223–2273. https://doi.org/10.1093/rfs/hhaa009

Gunduz, H., Yaslan, Y., & Cataltepe, Z. (2017). Intraday Prediction of Borsa Istanbul Using Convolutional Neural Networks and Feature Correlations. *Knowledge-Based Systems*, *137*, 138–148. https://doi.org/10.1016/j.knosys.2017.09.023

Guresen, E., Kayakutlu, G., & Daim, T. U. (2011). Using Artificial Neural Network Models in Stock Market Index Prediction. *Expert Systems with Applications*, *38*(8), 10389–10397. https://doi.org/10.1016/j.eswa.2011.02.068

Han, Y., Liu, Y., Zhou, G., & Zhu, Y. (2021). Technical Analysis in the Stock Market: A Review. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.3850494

Han, Y., Zhou, G., & Zhu, Y. (2016). A Trend Factor: Any Economic Gains from Using Information over Investment Horizons? *Journal of Financial Economics*, *122*(2), 352–375. https://doi.org/10.1016/j.jfineco.2016.01.029

Haussler, D. (1988). Quantifying Inductive Bias: AI Learning Algorithms and Valiant's Learning Framework. *Artificial Intelligence*, *36*(2), 177–221. https://doi.org/10.1016/0004-3702(88)90002-1

He, K., Zhang, X. [, Ren, S., & Sun, J. (2016a). Deep Residual Learning for Image Recognition. https://doi.org/10.48550/arXiv.1512.03385

He, K., Zhang, X. [, Ren, S., & Sun, J. (2016b). Identity Mappings in Deep Residual Networks. https://doi.org/10.48550/arXiv.1603.05027

Hendrycks, D., & Gimpel, K. (2016). Gaussian Error Linear Units (GELUs). https://doi.org/10.48550/arXiv.1606.08415

Heo, B., Yun, S., Han, D., Chun, S., Choe, J., & Oh, S. J. (2021). Rethinking Spatial Dimensions of Vision Transformers. https://doi.org/10.48550/arXiv.2103.16302

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. The MIT Press.

Hoseinzade, E., & Haratizadeh, S. (2019). CNNpred: CNN-Based Stock Market Prediction Using a Diverse Set of Variables. *Expert Systems with Applications*, *129*, 273–285. https://doi.org/10.1016/j.eswa.2019.03.029

Hu, G., Hu, Y. [, Yang, K., Yu, Z., Sung, F., Zhang, Z., Xie, F., Liu, J., Robertson, N., Hospedales, T., & Miemie, Q. (2018). Deep Stock Representation Learning: From Candlestick Charts to Investment Decisions, 2706–2710. https://doi.org/10.1109/ICASSP.2018.8462215

Hu, Y., Feng, B., Zhang, X. [, Ngai, E., & Liu, M. (2015). Stock Trading Rule Discovery with an Evolutionary Trend Following Model. *Expert Systems with Applications*, *42*(1), 212–222. https://doi.org/10.1016/j.eswa.2014.07.059

Huang, G., Sun, Y., Liu, Z., Sedra, D., & Weinberger, K. Q. (2016). Deep Networks with Stochastic Depth. *9908*, 646–661. https://doi.org/10.1007/978-3-319-46493-0_39

Ioffe, S., & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. https://doi.org/10.48550/arXiv.1502.03167

Jegadeesh, N., & Titman, S. (1993). Returns to Buying Winners and Selling Losers: Implications for Stock Market Efficiency. *The Journal of Finance*, *48*(1), 65–91. https://doi.org/10.1111/j.1540-6261.1993.tb04702.x

Jegadeesh, N., & Titman, S. (2001). Profitability of Momentum Strategies: An Evaluation of Alternative Explanations. *The Journal of Finance*, *56*(2), 699–720.

Jensen, M. C., & Benington, G. A. (1970). Random Walks and Technical Theories: Some Additional Evidence. *The Journal of Finance*, *25*(2), 469–482. https://doi.org/10.2307/2325495

Jiang, J., Kelly, B., & Xiu, D. (2023). (Re-)Imag(in)ing Price Trends. *The Journal of Finance*, *78*(6), 3193–3249. https://doi.org/10.1111/jofi.13268

Kamath, U., Graham, K., & Emara, W. (2022). *Transformers for Machine Learning: A Deep Dive (1st ed.)* Chapman & Hall.

Ke, Z., Kelly, B. T., & Xiu, D. (2021). Predicting Returns with Text Data. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.3389884

Kercheval, A. N., & Zhang, Y. (2015). Modelling High-Frequency Limit Order Book Dynamics with Support Vector Machines. *Quantitative Finance*, *15*(8), 1315–1329. https://doi.org/10.1080/14697688.2015.1032546

Kim, T., & Kim, H. Y. (2019). Forecasting Stock Prices with a Feature Fusion LSTM-CNN Model Using Different Representations of the Same Data. *PLOS ONE*, *14*(2), e0212320. https://doi.org/10.1371/journal.pone.0212320

Kingma, D. P., & Ba, J. (2017). Adam: A Method for Stochastic Optimization. https://doi.org/10.48550/arXiv.1412.6980

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems*, *25*. https://doi.org/10.1145/3065386

Lecun, Y., & Bengio, Y. (1995). Convolutional Networks for Images, Speech, and Time-Series. In M. A. Arbib (Ed.), *The Handbook of Brain Theory and Neural Networks*. The MIT Press.

Lee, J., Kim, R., Koh, Y., & Kang, J. (2019). Global Stock Market Prediction Based on Stock Chart Images Using Deep Q-Network. *IEEE Access*, *7*, 167260–167277. https://doi.org/10.1109/ACCESS.2019.2953542

Lee, M. (2009). Using Support Vector Machine with a Hybrid Feature Selection Method for Stock Trend Prediction. *Expert Systems with Applications*, *36*(8), 10896–10904. https://doi.org/10.1016/j.eswa.2009.02.038

Lee, S. H., Lee, S., & Song, B. C. (2021). Vision transformer for small-size datasets. *arXiv preprint arXiv:2112.13492*. https://doi.org/10.48550/arXiv.2112.13492

Li, Y., Wang, J. [, Dai, X., Wang, L., Yeh, C. C. M., Zheng, Y., Zhang, W., & Ma, K. L. (2023). How Does Attention Work in Vision Transformers? A Visual Analytics Attempt. https://doi.org/10.48550/arXiv.2303.13731

Lin, T., Wang, Y. [, Liu, X., & Qiu, X. (2022). A Survey of Transformers. *AI Open*, *3*, 111–132. https://doi.org/10.1016/j.aiopen.2022.10.001

Lo, A. W., & Hasanhodzic, J. (2010). *The Heretics of Finance: Conversations with Leading Practitioners of Technical Analysis* (Vol. 16). John Wiley and Sons.

Lo, A. W., Mamaysky, H., & Wang, J. [(2000). Foundations of Technical Analysis: Computational Algorithms, Statistical Inference, and Empirical Implementation. *The Journal of Finance*, *55*(4), 1705–1765. https://doi.org/10.1111/0022-1082.00265

Loshchilov, I., & Hutter, F. (2019). Decoupled Weight Decay Regularization. https://doi.org/10.48550/arXiv.1711.05101

Maas, A. L., Hannun, A. Y., & Ng, A. Y. (2013). Rectifier Nonlinearities Improve Neural Network Acoustic Models. *30th International Conference on Machine Learning*.

Maurício, J., Domingues, I., & Bernardino, J. (2023). Comparing Vision Transformers and Convolutional Neural Networks for Image Classification: A Literature Review. *Applied Sciences*, *13*(9), Article 5521. https://doi.org/10.3390/app13095521

Menkhoff, L., & Taylor, M. P. (2007). The Obstinate Passion of Foreign Exchange Professionals: Technical Analysis. *Journal of Economic Literature*, *45*(4), 936–972.

Murray, S., Xia, Y., & Xiao, H. (2024). Charting by Machines. *Journal of Financial Economics*, *153*, Article 103791. https://doi.org/10.1016/j.jfineco.2024.103791

Neely, C. J., Rapach, D. E., Tu, J., & Zhou, G. (2014). Forecasting the Equity Risk Premium: The Role of Technical Indicators. *Management Science*, *60*(7), 1772–1791.

Nelson, D. M. Q., Pereira, A. C. M., & Oliveira, R. A. d. (2017). Stock Market's Price Movement Prediction with LSTM Neural Networks. *2017 International Joint Conference on Neural Networks (IJCNN), Anchorage, AK, USA*.

Noble, W. S. (2006). What Is a Support Vector Machine? *Nature Biotechnology*, *24*(12), 1565–1567. https://doi.org/10.1038/nbt1206-1565

Peng, Z., Guo, Z., Huang, W., Wang, Y. [, Xie, L., Jiao, J., Tian, Q., & Ye, Q. (2023). Conformer: Local Features Coupling Global Representations for Recognition and Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *45*(8), 9454–9468. https://doi.org/10.1109/TPAMI.2023.3243048

Raschka, S. (2024). *Build a Large Language Model (From Scratch)*. Manning Publications.

Shao, R., & Bi, X.-J. (2022). Transformers Meet Small Datasets. *IEEE Access*, *10*, 118454–118464. https://doi.org/10.1109/ACCESS.2022.3221138

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *The Journal of Machine Learning Research*, *15*(56), 1929–1958.

Sullivan, R., Timmermann, A., & White, H. (1999). Data-Snooping, Technical Trading Rule Performance, and the Bootstrap. *The Journal of Finance*, *54*(5), 1647–1691.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2014). Going Deeper with Convolutions. https://doi.org/10.48550/arXiv.1409.4842

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the Inception Architecture for Computer Vision, 2818–2826. https://doi.org/10.1109/CVPR.2016.308

Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., & Jégou, H. (2020). Training Data-Efficient Image Transformers & Distillation Through Attention. https://doi.org/10.48550/arXiv.2012.12877

Touvron, H., Cord, M., Sablayrolles, A., Synnaeve, G., & Jégou, H. (2021). Going Deeper with Image Transformers. https://doi.org/10.48550/arXiv.2103.17239

Tsang, P. M., Kwok, P., Choy, S. O., Kwan, R., Ng, S. C., Mak, J., Tsang, J., Koong, K., & Wong, T.-L. (2007). Design and Implementation of NN5 for Hong Kong Stock Price Forecasting. *Engineering Applications of Artificial Intelligence*, *20*(4), 453–461. https://doi.org/10.1016/j.engappai.2006.10.002

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention Is All You Need. https://doi.org/10.48550/arXiv.1706.03762

Zhu, Y., & Zhou, G. (2009). Technical Analysis: An Asset Allocation Perspective on the Use of Moving Averages. *Journal of Financial Economics*, *92*(3), 519–544. https://doi.org/10.1016/j.jfineco.2008.07.002

List of Abbreviations

CaiT Class-Attention in Image Transformers

CNN Convolutional Neural Network

CRSP Center for Research in Security Prices

DeiT Data-efficient Image Transformers

EMH Efficient Market Hypothesis

FNN Feedforward Neural Network

GA Genetic Algorithm

LSA Locality Self-Attention

LSTM Long Short-Term Memory

MA Moving Average

ML Machine Learning

MLP Multi-Layer Perceptron

MOM Momentum

NLP Natural Language Processing

ReLU Rectified Linear Unit

RNN Recurrent Neural Network

RSI Relative Strength Index

SPT Shifted Patch Tokenization

STR Short-term Reversal

SVM Support Vector Machine

ViT Vision Transformer

WSTR Weekly Short-term Reversal