(Re-)Imag(in)ing Price Trends*

Jingwen Jiang
Department of Computer Science
University of Chicago

Bryan Kelly
Yale University, AQR Capital Management, and NBER

Dacheng Xiu
Booth School of Business
University of Chicago

Abstract

We reconsider the idea of trend-based predictability using methods that flexibly *learn* price patterns that are most predictive of future returns, rather than testing hypothesized or pre-specified patterns (e.g., momentum and reversal). Our raw predictor data are images—stock-level price charts—from which we elicit the price patterns that best predict returns using machine learning image analysis methods. The predictive patterns we identify are largely distinct from trend signals commonly analyzed in the literature, give more accurate return predictions, translate into more profitable investment strategies, and are robust to a battery of specification variations. They also appear context-independent: Predictive patterns estimated at short time scales (e.g., daily data) give similarly strong predictions when applied at longer time scales (e.g., monthly), and patterns learned from US stocks predict equally well in international markets.

Keywords: convolutional neural network (CNN), image classification, transfer learning, machine learning, technical analysis, return prediction

*We are grateful for comments from Ronen Israel, Serhiy Kozak (discussant), Ari Levine, Chris Neely (discussant), and seminar and conference participants at Washington University in St. Louis, University of Oxford, University of Rochester, Rutgers Business School, Boston University, Chinese University of Hong Kong, ITAM Business School, Singapore Management University, National University of Singapore, Cheung Kong Graduate School of Business, University of Science and Technology of China, Nanjing Audit University, University of Iowa, University of Houston, Renmin University, Hong Kong University of Science and Technology, AEA/ASSA North American Meetings, SFS Cavalcade, Society of Financial Econometrics, China International Conference in Finance, Society of Quantitative Analysts, and INQUIRE UK. AQR Capital Management is a global investment management firm, which may or may not apply similar investment techniques or methods of analysis as described herein. The views expressed here are those of the authors and not necessarily those of AQR.

Nevertheless, technical analysis has survived through the years, perhaps because its visual mode of analysis is more conducive to human cognition, and because pattern recognition is one of the few repetitive activities for which computers do not have an absolute advantage (yet). *Lo et al. (2000)*

1 Introduction

A large literature investigates the ability of past prices to forecast future returns, producing a handful of famous and robust predictors including price momentum and reversal. Given recent strides in understanding how price dynamics are influenced by human behavior and psychology,[1] it is plausible that prices contain subtle and complex patterns that can be difficult to detect with the standard methods of empirical finance. In this paper, we reconsider price trend patterns from a machine learning perspective.

Perhaps the most daunting challenge to a machine learning perspective on price-based return prediction is settling on a methodology to achieve a balance between two countervailing concerns. On one hand, we prefer a method that is flexible enough to find potentially complex predictive patterns. On the other hand, we prefer a method that is tractable and constrained enough so that we can interpret those patterns in order to inform future theory.[2] To negotiate this delicate tradeoff, we make two research design choices that together result in successful inference of new return-predictive patterns in past prices. First we represent historical prices as an image, and second we model the predictive association between images and future returns using a convolutional neural network (CNN). These are conjugate choices that work together and enhance each other. We describe the logic behind each choice in turn, beginning with the CNN.

The input to a CNN is typically an image, and in our setting the image is a plot of past market information (open, high, low, and close prices and trading volume) represented as a matrix of black and white pixel values. A CNN is designed to produce forecasts from an image without requiring a researcher to manually engineer predictive features. Instead, the CNN automates the feature extraction process. In a given "layer" the CNN spatially smooths image contents to reduce noise and accentuate shape configurations that correlate with future returns. This smoothing operation is applied recursively by stacking multiple layers together, which gives the CNN flexibility to capture potentially complex predictive patterns (and earns it the synonym "deep learning"). The final predictor set consists of smoothed non-linear transformations of the original numeric pixel values. At a high level, this is similar to more

[1] See Barberis (2018) for a survey.

[2] In addition to interpretability, we also favor more tractable and constrained methods to ensure that they can be reliably estimated from available data with manageable computational cost.

traditional kernel-based data filters used in regression analysis. But the CNN does not simply fix the set of smoothing filters. It instead estimates the filters that best detect shapes and other attributes of the images that are most predictive of the target variable. In short, we use a CNN due to its ability to automatically extract predictive signals from raw input data, which makes it ideally suited to elicit patterns that underly financial markets but may be too complex for a human to hypothesize.

Why is it beneficial to encode market data as an image rather than in the more standard time series numerical format? The first reason is that the leading CNN architectures are custom crafted for image analysis. Therefore, to enjoy the CNN's benefits of automated signal generation, it is useful to represent price data in the format naturally ingested by the CNN, i.e. as an image. Second, representing time series as an image allows the model to focus on relational attributes of the data that would be difficult to tease out with time series methods. It is the same basic logic for why humans illustrate patterns graphically rather than with lists of numbers. If a human can more readily detect patterns in images by consuming an entire data matrix through a single visualization, a statistical pattern recognition algorithm may benefit from it as well. Third, the process of imaging price and volume data converts all assets' data histories to a comparable scale. We show that this particular re-scaling choice has large benefits for prediction in the panel of stocks. Fourth, technical trading hinges on the presence of geometric shapes visually defined and observed by human cognition. Technical traders have long used price charts as an information source to predict returns and make investment decisions.

Our empirical analysis revolves around a panel prediction model for US stock returns from 1993 to 2019. We train a panel CNN model to predict the direction (up or down) of future stock returns. That is, in each training sample, we estimate a single model with a fixed structure and set of parameters that produces forecasts for all individual stocks. The data input to this model are images depicting price and volume (daily open, close, high, and low prices, daily trading volume, and moving average price) over the past 5, 20, and 60 days. The output of the CNN is a set of stock-level estimates for the probability of a positive subsequent return over short (5-day), medium (20-day), and long (60-day) horizons. We use CNN-based out-of-sample predictions as signals in a number of asset pricing analyses.

Our main empirical finding is that image-based CNN predictions are powerful and robust predictors of future returns. We summarize this predictive power in economic terms with out-of-sample portfolio performance. We sort stocks into decile portfolios using image-based CNN forecasts and track the returns of a decile spread H-L portfolio. We first consider a weekly trading strategy, and use a CNN supervised by weekly returns (thus CNN training and the

strategy's rebalancing frequency align). Image-based decile spreads perform extraordinarily well, earning annualized out-of-sample gross Sharpe ratios as high as 7.2 for equal-weight portfolios and 1.7 for value-weight portfolios. We benchmark this performance against four other price trend strategies: 2-12 momentum (MOM), 1-month short-term reversal (STR), 1-week short-term reversal (WSTR), and the Han et al. (2016) trend strategy that combines short, intermediate, and long-term price trends (TREND). Of these, the closest competitors to image based forecasts are TREND and WSTR, which achieve equal-weight Sharpe ratios of 2.9 and 2.8, respectively (value-weight Sharpe ratios of 0.7 and 0.8, respectively). The performance differential between the CNN and competitors does not appear attributable to differences in limits to arbitrage such as trading costs. Image-based CNN strategies have portfolio turnover that is nearly identical to WSTR, but manage to double the annualized performance of WSTR.

We find similar qualitative results for strategies with longer holding periods of one month and one quarter (again, each CNN is trained to forecast returns over horizons that align with strategy holding period). The image-based H-L strategy reaches an annualized out-of-sample equal-weight Sharpe ratio as high as 2.4 and 1.3 for monthly and quarterly strategies, respectively. We then explore whether this longer-horizon performance is driven by especially strong predictability over the first week by decomposing the monthly rebalanced strategy into returns from days 1–5 after rebalance versus returns over days 6–20. While the bulk of the return in the monthly equal-weight strategy accrues over the first five days, we find that the annualized Sharpe ratio for days 6–20 reaches 1.2 and is highly significant. Summarizing this analysis, we find that image-based price trend forecasts are most potent in the first week after observing the image. A weekly rebalancing strategy that exploits these potent short-horizon image-based predictions incurs high turnover, suggesting it is mostly accessible to investors that behave as market makers. But we also find that profits to image-based strategies remain significantly positive after standard trading cost adjustments, when the strategy rebalances at lower frequencies that are accessible to longer-horizon investors, and when trading focuses on predictability beyond the first week following image observance.

CNN interpretation is a notoriously difficult problem in the machine learning literature because the model uses several telescoping layers of non-linear image transformations to generate its forecasts. We make progress interpreting the predictive patterns identified by the CNN using two approaches. First, we compare image-based signals from the CNN to previously studied signals in the literature. Using logistic regression we identify the traditional signals that most closely approximate the CNN's prediction; these include dollar volume, size, reversal, and Amihud illiquidity. However, previously studied signals explain only about 10%

of the cross-sectional variation in CNN forecasts.

Second, we search for simple logistic regression specifications (using price and volume data underlying our images) to best approximate the CNN model. A key component of image-based prediction is the implicit data scaling achieved by the image representation—images put all stocks' past price data on the same scale so that their recent maximum high and minimum low prices span the height of the image and all other prices (open, high, low, close, and moving average) are rescaled accordingly, and likewise for volume. Following this data transformation, a logistic model can produce a reasonable approximation to the CNN which aids in interpretation. For example, a simple approximator for one of the patterns detected by the CNN is that when a stock closes on the low end of its recent high-low range, future returns tend to be high.

An intriguing aspect of our analysis is that return-predictive patterns detected by the CNN extrapolate to contexts outside of the main data set of daily US stock prices. In particular, we consider the possibility of image-based *transfer learning*, using a model estimated in one context to forecast in a distinct context. We show that the predictive patterns identified by the CNN from daily US stock data transfer well to international markets and to other time scales. International markets have fewer stocks and shorter time series compared to the US. We transfer CNN model estimates from US data to construct return forecasts in 26 foreign markets. We find that international image-based trading strategies earn a *higher* Sharpe than if we train a CNN from scratch using local market data. Next, given the scarcity of low frequency data in financial markets, it would be greatly beneficial to know if patterns that unfold at high frequencies (which we can potentially measure well) are similar to patterns that unfold at low frequencies.[3] To illustrate this idea, we train a CNN model to predict 5-day returns from images of 5-day prior price data. We transfer this model to the problem of predicting 60-day returns using prior 60-day price data by sampling the data once every 12 days. A quarterly trading strategy based on this high-frequency transfer approach outperforms a CNN trained directly on quarterly data.

A large literature has debated the theoretical sensibility and empirical reliability of technical analysis. Lo et al. (2000) and Lo and Hasanhodzic (2010) insightfully juxtapose arguments on either side of the debate. A number of papers present theoretical arguments for the existence of equilibrium predictability with technical analysis, including Brown and Jennings (1989), Grundy and McNichols (1989), Blume et al. (1994), Barberis et al. (1998), and Han et al. (2016), among others. Several papers find strong empirical support for technical trading

[3]This is motivated by the Mandelbrot (2013) "fractal" hypothesis in financial markets, which predicts that asset prices exhibit statistically self-similar patterns when studied at different time scales, and also predicts the emergence of long-range dependence in prices (Cont, 2005).

rules, with prominent examples including Brock et al. (1992), Lo et al. (2000), Zhu and Zhou (2009), Neely et al. (2014), Han et al. (2016), Detzel et al. (2020), and Murray et al. (2021). Sullivan et al. (1999) and Bajgrowicz and Scaillet (2012) study large collections of candidate trading rules with careful corrections for multiple hypothesis testing and reach skeptical conclusions regarding the significance of technical trading profits. The debate about the viability of technical trading is to some extent moot given the widely documented, robust, and by now uncontroversial momentum effect (Jegadeesh and Titman, 1993), which Schwert (2003) concludes is the most reliable technical pattern in the post-publication sample.

We contribute to this literature by investigating price trends (and technical analysis more generally) with a machine learning analysis of price chart images. The key differentiator of our approach is that we do not require the researcher to pre-specify a set of technical patterns. Instead, we simply present our model with historical market data in the form of an image. In place of human-generated predictive signals, our CNN conducts an automated search for image patterns that are most predictive of future returns. This is a continuation of the agenda set forth by Lo et al. (2000), but with a re-tooled research design benefitting from twenty years of progress in machine learning and computer vision. Ultimately, our CNN approach extracts robust predictive patterns from images that outperform stalwart price trend patterns from the literature, including momentum and short-term reversal.

Lastly, there is an emergent literature in computer science that uses price plots and CNN-based models to forecast stock returns. These papers give a short description of methods and present small-scale empirical analyses. The large majority of this work performs time series prediction of aggregate stock indices (examples include Chen et al., 2016, Kim and Kim, 2019, Lee et al., 2019, Hoseinzade and Haratizadeh, 2019). Hu et al. (2018) use price plot CNN's to cluster individual stocks; Cohen et al. (2020) classify images with some specific technical pattens (crossings of Bollinger Bands, MACD, Relative Strength Index). They do not predict returns. To our knowledge, no other paper performs a large-scale, thorough, and methodologically transparent analysis of return prediction for individual stocks with the fine granularity that is standard in empirical asset pricing research.

The remainder of the paper proceeds as follows. First, we describe the process of "imaging" market data in Section 2. Section 3 presents the intuition and detailed mechanics of our CNN framework. We report our main empirical analysis in Section 4. Section 5 explores interpretations of the CNN model. In Section 6, we analyze the benefits of CNN transfer learning across geographies and time scales. Section 7 concludes. We report a variety of extensions and robustness of our main empirical analysis in the Internet Appendix, along with simulation evidence to illustrate the model's finite sample properties.

Figure 1: Tesla OHLC Chart from Yahoo! Finance

Note: OHLC chart for Tesla stock with 20-day moving average price line and daily volume bars. Daily data from January 1, 2020 to August 18, 2020.

2 "Imaging" Market Data

In this section, we describe the process of representing historical market data as an image input for the CNN prediction model. Many popular websites such as Bloomberg, Yahoo! Finance, and Google Finance provide historical price charts for a wide range of financial assets. Figure 1 illustrates an example of Tesla's stock price data through August 18, 2020 in a common price chart format. It includes "OHLC" bars that depict daily opening, high, low, and closing prices. It then overlays a 20-day moving average closing price. The bottom of the chart shows daily trading volume. While these charts (and many variations on them) can be captured from the Internet, we instead generate our own price charts from scratch. This allows us to conduct various experiments by controlling the amount of information that our CNN "trader" can observe.

2.1 The OHLC Chart

The images we generate follow the basic format of Figure 1. In particular, our main price plot uses OHLC bars (colored in black) in this figure. High and low prices are represented by the top and bottom of the middle vertical bar, while opening and closing prices are represented by the small horizontal lines on the left and right of the bar, respectively. In our images, one day occupies an area three pixels wide (the center bar, open mark, and closing mark are each one pixel wide).

The main component of our image is a concatenation of daily OHLC bars over consecutive 5, 20, or 60-day intervals (approximately weekly, monthly, and quarterly price trajectories, respectively). The width of a n-day image is thus $3n$ pixels. We replace prices by CRSP adjusted returns to translate the opening, closing, high and low prices into relative scales that abstract from price effects of stock splits and dividend issuance.

Once days are concatenated, we impose a constant height for all images and scale the vertical axis so that the maximum and minimum of the OHLC path coincides with the top and bottom of the image. As a result, all images for the same number of days have the same pixel dimensions. The resulting image is shown in Figure IA2a.

The vertical dimension of the image conveys two main types of information. The first and most obvious are directional price trends that are viewed as the critical content of typical technical signals such as momentum and reversal. The second and more subtle is volatility information. Parkinson (1980) shows that the daily high-low range, described by the vertical length of the OHLC bar, provides an accurate snapshot of daily stock price volatility. Generalizations by Dobrev (2007) and others show that the high-low range over intervals other than a day are likewise beneficial for volatility inference. This helps motivate the visual representation of price paths, which allows the viewer to immediately and simultaneously perceive price ranges at different frequencies, which is an essentially non-linear process. This type of non-linearity would be difficult for traditional kernel methods to discern from time series data.

We exclude images for stocks that IPO or delist during the data window, but we allow missing data if it occurs in the middle of the stock's history. If there is missing data, the columns of pixels corresponding to the missing days are left blank (or partially blank if only part of the OHLC information is available).[4]

We use black as the background color and white color for visible objects on the charts. This means that most space on the chart is in black, which eases data storage requirements because a black pixel is represented by $(0, 0, 0)$ and our resulting images are sparse. The use of different colors for "up" and "down" days, as commonly used by Bloomberg and others, is redundant because the direction of price change is implied from the opening and closing price marks. This allows us to focus on two-dimensional pixel matrices, rather than having to track a third dimension for RGB pixel intensities.

[4]The ability to obtain price-trend based forecasts in the presence of incomplete data is an example of image-based CNN robustness to noisy data. When a high or low price is missing, we leave the bar entirely black because it is impossible to draw the middle bar. If both high and low prices are available but opening and closing prices are not, we draw a vertical bar only.

2.2 Moving Average Lines and Volume Bars

As in the Yahoo! Finance chart of Figure 1, we consider supplementing the main OHLC image with two additional pieces of information. The first is a moving average price line. In traditional technical analysis, moving averages are viewed as useful for inferring potential deviations from fair value by providing a long horizon reference point for prevailing point-in-time prices. The comparison of price to its moving average may be useful as a value signal that avoids the need for balance sheet data, as recommended by Fama and French (1988) and Kelly and Pruitt (2013). We use a moving average line with a window length equal to the number of days in the image (e.g., 20-day images will have a moving average line of 20 days). The daily moving average is reported using one pixel in the middle column for each day, and a line is drawn by connecting those dots.

The second addition is a set of daily trading volume bars. When including volume data, we design images so that volume is shown in the bottom one-fifth of the image while the top four-fifths contain the main OHLC plot. Similar to our OHLC scaling, the maximum volume in a given image is set equal to the upper limit of the volume bar section and the remaining volume bars are scaled accordingly.

Figure 2 shows an example from our final image data set. This image concisely embeds a variety of information on price trends, volatility, intraday and overnight return patterns, and trading volume. The image design strikes a balance between information content and storage efficiency, delivering a rich information set as an input to the CNN while controlling the computational burden of estimation.

3 The Convolutional Neural Network Model

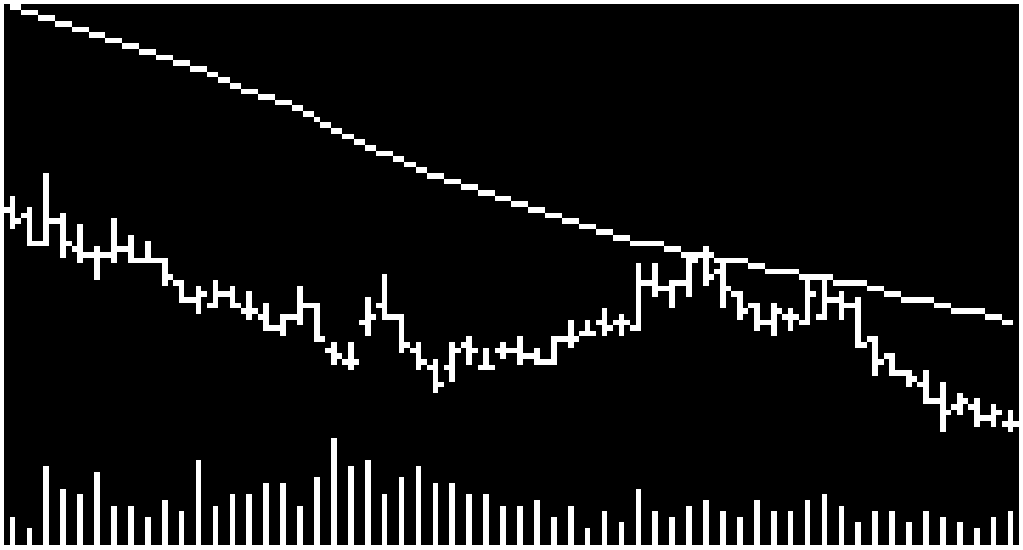
In this section, we briefly describe the architecture of our CNN model specification and training approach.[5] We also discuss the rationale and beneficial aspects of representing time series market data as an image for use in a CNN, rather than using a traditional time series prediction model.

3.1 A Brief Description of the CNN Architecture

Each image in our dataset is represented as a matrix of pixel values (0 or 255 for black or white pixels). In principal, this matrix can be vectorized and treated as the input to a standard feed-forward neural network. There are a number of problems with this approach. First, generic unconstrained neural networks tend to be massively parameterized, and this problem is exacerbated by the large number of pixels in a typical image. The amount of data required to support such a flexible parameterization is unrealistic in many research problems,

[5] Goodfellow et al. (2016) Chapter 9 provides a textbook introduction to CNN.

Figure 2: Generated OHLC Images with Volume Bar and Moving Average Line



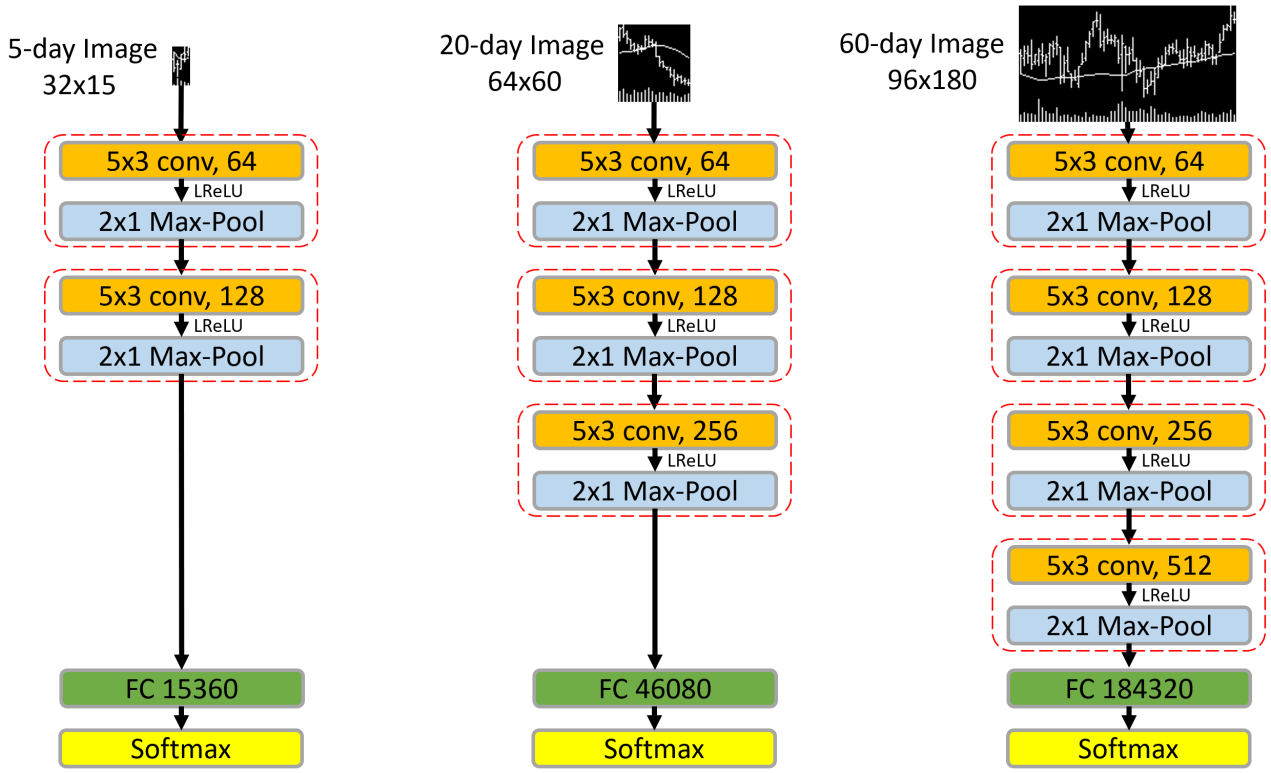
Note: Market data image for 60 days of data.

and particularly in our application. Second, such a model would be inherently sensitive to position, scale, and orientation of an object in the image. Moving the object from one corner of the image to another, or from foreground to the distance, would confuse such a model and severely limit its usefulness in forecasting.

Instead, CNN is the workhorse model for constructing predictions based on raw image data. Its popularity stems from its success in resolving the two problems above. They impose cross-parameter restrictions that dramatically reduce parameterization relative to standard feed-forward neural networks, making them more tractable to train and more effective in prediction with comparatively small data sets. And they embed a number of tools that make the model resilient to deformation and re-positioning of important objects in the image.

A CNN is a modeling scheme that stacks together a sequence of operations to transform a raw image into a set of predictive features and, ultimately, a prediction. They are inherently modular, using a single core building block that can be assembled in various configurations depending on the application at hand. A core building block consists of three operations: convolution, activation, and pooling. Convolution works similarly to kernel smoothing. It scans horizontally and vertically through the image and, for each element in the image matrix, produces a summary of image contents in the immediately surrounding area. Activation is a non-linear transformation (specifically, "leaky ReLU") applied element-wise to the output of a convolution filter. The last operation in a building block is "max-pooling," which again scans over the input matrix and returns the maximum value over surrounding areas in the

Figure 3: Diagram of CNN Models



Note: This figure shows diagrams of 5-day CNN model (left), 20-day CNN model (middle), and 60-day CNN model (right). The 5/20/60-day CNN models are built with 2/3/4 CNN building blocks described in Figure A2. The notation HxW shows the output size of the CNN building block where H is the height and W is the width. The number following "conv" or "Max-Pool" is the depth (number of channels, e.g., 64, 128, 256 and 512). The output of the last CNN block is flattened to a vector and fed to a fully connected layer (FC), where the reported input size (after FC) is calculated as the product $H \times W \times D$ from the last CNN building block. The final softmax layer yields the probabilities of up and down.

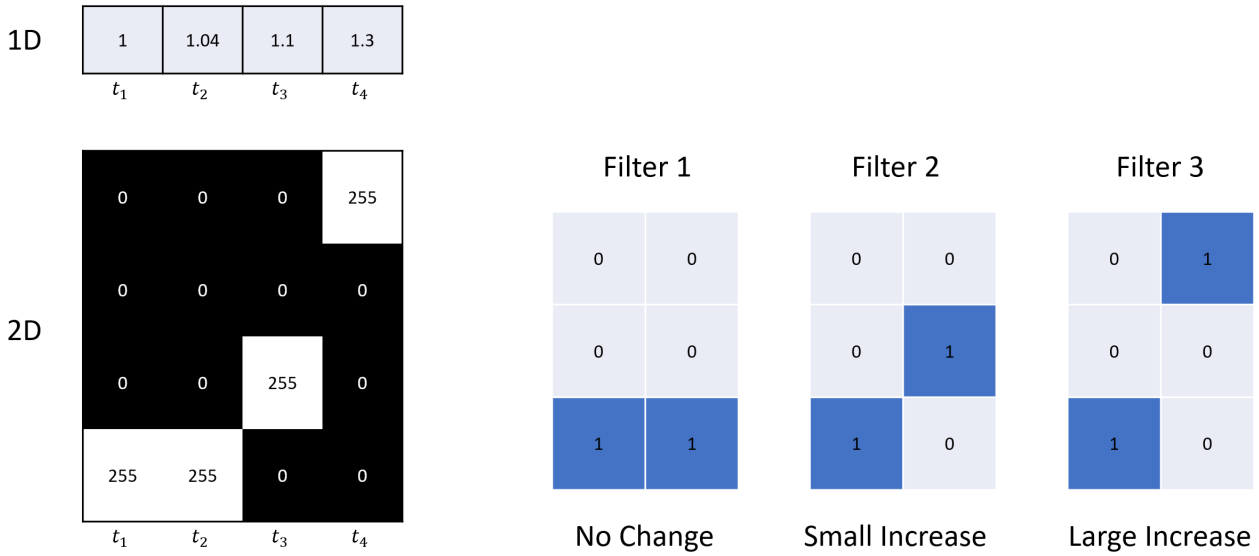
image to reduce the dimension of the data and reduce noise. The final CNN layer is fully connected and activated by a softmax function. The CNN targets a binary outcome equal to one for a positive return over the specified forward horizon and zero otherwise. As a result, the fitted value from the CNN is an estimate for the probability of a positive outcome.

We consider three main CNN specifications with varying degrees of complexity. The architecture for each model is shown in Figure 3. In the Appendix, we provide complete details of this CNN structure along with the intuition behind each of its components.

3.2 Image Representation vs. Time-series Representation

To help understand our empirical design, it is useful to understand the CNN terms "1D" and "2D." These describe the way that a CNN's convolutional filters move through the data. A

Figure 4: Convolutional Filters that Detect the Next Day's Price Change



Note: The table on top displays a snippet of time series prices. The bottom figure translates this time series into a black-and-white image where "255" represents a white pixel (corresponding to a price entry) and "0" represents empty space in the image. The image is a discretized representation of the prices. For illustration, each unit on the y-axis of the image is 0.1. Because the difference between 1.04 and 1.0 is smaller than 0.05, the discretized prices on the image are flat over the first two periods. The figure on the right demonstrates three 3×2 convolutional filters that detect no change, small increase, and large increase in price, respectively.

black-and-white image is represented as a matrix with the value of each element corresponding to the shade of gray at the corresponding position of the image. Because its rectangular filters move both horizontally and vertically through the image matrix, a standard image-processing CNN is said to be "2D." Another common machine learning model is a CNN applied to time series data. In this case, the data are again represented as a matrix with time corresponding to rows and variables to columns. Convolutional filters in this setting have the same width as the data matrix, and the filter only moves through the time dimension. Such a CNN is therefore said to be "1D."

Our research design recommends embedding time series data as an image before bringing it to predictive analysis. Why is it beneficial to work with images rather than directly modeling the time series? Representing the data as an image makes it possible for a convolutional filter to capture non-linear spatial associations among various price curves. Figure 4 is a simple example of how this works. It shows a price plot over four periods. In the first two periods the price is flat, in the third period it rises by one unit, and in the fourth period the price jumps by two units. A 1D kernel filter can only extract the linear difference between prices on two consecutive days without the further help of a non-linear activation function. If a price

rise of two units is more than twice as predictive as a price rise of one unit, the 1D kernel is unable to account for it. But a 2D CNN applied to the price plot image can. The example shows how this is achieved with 3×2 filters that treat "no change," "small increase," and "large increase" as separate features that can enter into an ultimate prediction with distinct weights. That is, when these filters are applied to images, they act as indicator functions that detect and bin price movements of different sizes.

Likewise, the image automatically combines information on both directional price movements, movements relative to moving average trend, price volatility, and trading volume in a single representation. To jointly consider price direction and volatility, for example, a traditional time series model would require restrictive choices to manually engineer features. And incorporating volatility information would inevitably require non-linear transformations of price series along the lines of stochastic volatility or GARCH models. Fortunately, 2D CNN eliminates any need to manually engineer such features and instead extracts predictive patterns from the market data series within the CNN itself.

In short, ingesting data in the form of an image allows the model to isolate data relationships that may be difficult to detect with time series methods. Just as humans find it easier to detect patterns graphically rather than with lists of numbers, a statistical pattern recognition algorithm may also benefit from visualizing an entire data matrix in a single image.[6]

3.3 Training the CNN

Our workflow from training, to model tuning, and finally to prediction follows the basic procedure outlined by Gu et al. (2020). First, we divide the entire sample into training, validation, and testing samples. In our main US data sample, we estimate and validate the model using a single eight-year sample (1993-2000) at the start of our sample. In this eight-year sample, we randomly select 70% images for training and 30% for validation. Randomly selecting the training and validation sample helps balance positive and negative labels in our classification problem, which attenuates a potential bias in classification due to extended periods of bullish or bearish market swings. The resulting training and validation images have approximately 50% up and 50% down labels in all scenarios we consider. The remaining nineteen years (2001-2019) of data comprise the out-of-sample test data set.

We treat the prediction analysis as a classification problem. In particular, the label for an image is defined as $y = 1$ if the subsequent return is positive and $y = 0$ otherwise. The training step minimizes the standard objective function for classification problems, a cross-entropy loss

[6]While we advocate the use of CNN models over time-series models, we cannot rule out the possibility that a well-crafted time-series model, say, LSTM, may outperform the CNN. Also, our objective here is not to find the best return prediction model, which is an unrealistic task anyway. Our empirical analysis provides, at best, a lower bound on the extent of predictability machine learning models can achieve.

function. It is defined as:

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

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

We adopt the same regularization procedures in Gu et al. (2020) to combat overfit and aid efficient computation. We apply the Xavier initializer for weights in each layer (Glorot and Bengio, 2010). This promotes faster convergence by generating starting values for weights to ensure that prediction variance begins on a comparable scale to that of the labels. Loss function optimization uses stochastic gradient descent and the Adam algorithm (Kinga and Adam, 2015) with initial learning rate of 1×10^{-5} and batch size of 128. We use a batch normalization (Ioffe and Szegedy, 2015) layer between the convolution and non-linear activation within each building block to reduce covariate shift.[7] We apply 50% dropout (Srivastava et al., 2014) to the fully connected layer (the relatively low parameterization in convolutional blocks avoids the need for dropout there). Finally, we use early stopping to halt training once the validation sample loss function fails to improve for two consecutive epochs. Gu et al. (2020) outline the intuition behind these choices, so for the sake of brevity, we omit this discussion and instead refer interested readers there.

In Internet Appendix IA.3, we design simulation experiments to investigate the finite sample performance of the CNN classifier. Since our images look rather different from standard CNN research data sets like ImageNet, the simulation evidence provides insight into the effectiveness of a CNN in this new environment. The general conclusion from simulations is that the CNN successfully detects complicated technical patterns in realistically low signal-to-noise data sets.

4 CNN Prediction for US Stock Returns

4.1 Data

We use daily stock data from CRSP for all firms listed on NYSE, AMEX, and NASDAQ. Our sample runs from 1993–2019 based on the fact that daily opening, high, and low prices first become available in June 1992.

Our price trend analysis focuses on returns adjusted for corporate actions by using returns

[7] We also normalize all images using the mean and standard deviation of all pixel values from images in the training data, which are then used to normalize the validation and testing images. We adopt it as a standard operation in the CNN literature, though skipping this step has negligible impact on our results.

to construct a price series. In each image, we normalize the first day closing price to one, and construct each subsequent daily close from returns (RET_t) according to

$$p_{t+1} = (1 + RET_{t+1})p_t.$$

Each day's opening/high/low price levels are scaled in proportion to that day's closing price level.

We consider three input choices that include images of market data over the past 5, 20, or 60 days. Image labels take a value of one or zero for positive or non-positive returns over the 5, 20, or 60 days subsequent to the image. Thus, our main analysis amounts to nine separately estimated models. Because the CNN optimization is stochastic, for each model configuration we independently re-train the CNN five times and average their forecasts (following Gu et al., 2020).

Two important considerations when reading our empirical results are that *we do not recursively re-train the model* and that we *randomly* select training and validation samples. Specifically, we train and validate each model only once using data from 1993 to 2000, in which 70% of the sample are randomly selected for training and the remaining 30% for validation. The trained CNN model is then held fixed for the entire 2001 to 2019 test sample. This design is primarily due to capacity in computational resources. Adopting a rolling window and repeatedly retraining is likely to further improve the predictions.

Every period in which we construct new forecasts (weekly, monthly, or quarterly, depending on the model's forecast horizon), we sort stocks into decile portfolios based on out-of-sample CNN estimates for probability of a positive subsequent return. We also construct a long-short spread portfolio ("H-L") that is long decile 10 and short decile 1. The holding period for each portfolio coincides with the forecast horizon for each model (either 5, 20, or 60 days following the last date in an image). Throughout we use the notation "Ix/Ry" to denote that the model uses x-day images to predict subsequent y-day holding period returns.

4.2 Short-horizon Portfolio Performance

Image-based return predictions, which constitute a technical price trend signal, are likely to be most potent over relatively short horizons. Thus, we begin our reportage focusing on one weekly return prediction.

Table 1 reports the predictive strength of the CNN model when it targets 5-day ahead returns. We couch this predictive strength in economic terms by reporting performance of portfolios sorted on CNN forecasts. The table focuses on annualized average returns and Sharpe ratios for 5-day holding period returns. The top panel reports equal-weight decile

Table 1: Short-horizon (One Week) Portfolio Performance

	Equal Weight													
	I5/R5		I20/R5		I60/R5		MOM/R5		STR/R5		WSTR/R5		TREND/R5	
	Ret	SR	Ret	SR	Ret	SR	Ret	SR	Ret	SR	Ret	SR	Ret	SR
Low	-0.28	-1.92	-0.32	-1.94	-0.21	-1.10	0.15	0.44	-0.01	-0.03	-0.08	-0.34	-0.11	-0.46
2	-0.04	-0.27	-0.04	-0.21	0.02	0.12	0.10	0.44	0.06	0.35	0.04	0.24	0.01	0.05
3	0.03	0.15	0.04	0.20	0.07	0.35	0.10	0.50	0.09	0.58	0.08	0.48	0.05	0.30
4	0.08	0.41	0.08	0.43	0.11	0.58	0.10	0.57	0.10	0.67	0.09	0.58	0.08	0.50
5	0.09	0.48	0.12	0.65	0.14	0.75	0.10	0.63	0.11	0.68	0.09	0.60	0.10	0.64
6	0.14	0.70	0.15	0.80	0.16	0.88	0.12	0.76	0.11	0.70	0.11	0.65	0.11	0.71
7	0.17	0.84	0.19	0.97	0.17	0.93	0.13	0.83	0.11	0.64	0.13	0.75	0.13	0.78
8	0.22	1.06	0.23	1.19	0.20	1.08	0.14	0.90	0.12	0.62	0.14	0.72	0.16	0.85
9	0.30	1.48	0.27	1.40	0.22	1.23	0.15	0.91	0.16	0.68	0.18	0.81	0.23	1.04
High	0.54	2.89	0.52	2.76	0.33	1.85	0.16	0.78	0.38	1.19	0.46	1.56	0.48	1.58
H-L	0.83***	7.15	0.84***	6.75	0.54***	4.89	0.02	0.07	0.39***	1.76	0.53***	2.84	0.59***	2.92
Turnover	690%		667%		619%		123%		341%		660%		499%	
	Value Weight													
	I5/R5		I20/R5		I60/R5		MOM/R5		STR/R5		WSTR/R5		TREND/R5	
	Ret	SR	Ret	SR	Ret	SR	Ret	SR	Ret	SR	Ret	SR		
Low	-0.03	-0.19	-0.03	-0.16	-0.02	-0.12	0.03	0.06	0.04	0.16	-0.02	-0.09	0.02	0.08
2	0.00	0.00	0.02	0.12	0.01	0.06	0.02	0.05	0.04	0.22	0.03	0.14	0.02	0.10
3	0.04	0.24	0.04	0.24	0.04	0.24	0.07	0.27	0.06	0.34	0.06	0.34	0.06	0.33
4	0.06	0.35	0.06	0.32	0.06	0.31	0.08	0.36	0.09	0.54	0.06	0.36	0.07	0.41
5	0.06	0.34	0.06	0.34	0.06	0.34	0.08	0.41	0.09	0.57	0.08	0.48	0.08	0.50
6	0.09	0.50	0.09	0.51	0.06	0.32	0.08	0.46	0.10	0.57	0.10	0.60	0.09	0.55
7	0.11	0.58	0.09	0.49	0.08	0.45	0.08	0.51	0.10	0.51	0.11	0.63	0.12	0.66
8	0.12	0.62	0.11	0.61	0.09	0.49	0.10	0.62	0.13	0.63	0.15	0.73	0.14	0.71
9	0.16	0.81	0.11	0.59	0.11	0.61	0.11	0.62	0.13	0.51	0.18	0.72	0.14	0.57
High	0.20	0.91	0.20	0.99	0.14	0.77	0.14	0.63	0.16	0.46	0.18	0.59	0.18	0.55
H-L	0.23***	1.49	0.22***	1.74	0.16***	1.44	0.12	0.33	0.13*	0.44	0.21***	0.77	0.16***	0.66
Turnover	758%		728%		671%		163%		433%		733%		575%	

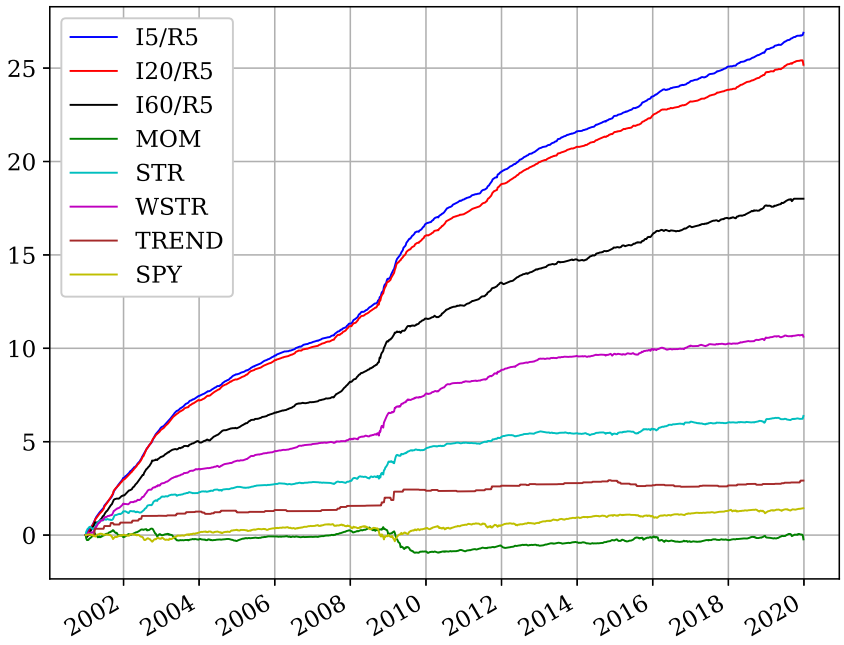
Note: Performance of equal-weighted (top panel) and value-weighted (bottom panel) decile portfolios sorted on out-of-sample predicted up probability. Each panel reports the average holding period return and annualized Sharpe ratios. Average returns accompanied by ***,**,* are significant at the 1%, 5% and 10% significance level, respectively. We also report monthly turnover of each strategy.

portfolios. Decile 1, which corresponds to stocks having the lowest probability of a positive future return indeed realize large negative Sharpe ratios in excess of −1.0 for all image sizes. Sharpe ratios increase monotonically across predicted "up" probability deciles. A long-only strategy based on stocks in decile 10 alone earns a Sharpe ratio in excess of 1.8 across CNN models. Long-short H–L strategies earn annualized Sharpe ratios of 7.2, 6.8, and 4.9 for CNN models based on 5-day, 20-day, and 60-day images, respectively. To benchmark these results, we also report performance of one week holding period strategies based on MOM, STR, WSTR, and TREND, whose decile spreads earn annualized Sharpe ratios of 0.1, 1.8, 2.8, and 2.9, respectively.[8] In value weight portfolios (bottom panel), CNN strategies earn Sharpe ratios ranging from 1.4 to 1.7 Sharpe ratio, double the 0.8 Sharpe ratio for the value weight WSTR strategy (which is the best value-weight performer of the benchmarks).[9] CNN

[8]Internet Appendix Table IA5 reports correlations between the long and short legs of CNN strategies versus those for other technical indicator strategies.

[9]Internet Appendix Table IA2 studies the possibility that CNN models trained with one supervising return

Figure 5: Cumulative Volatility Adjusted Returns of Equal-Weighted Portfolios



Note: Cumulative volatility adjusted log returns of equal-weight long-short portfolios of I5/R5, I20/R5, I60/R5, MOM, STR, WSTR, and TREND. All strategies are re-scaled to have the same volatility as that of SPY over the test sample.

strategies outperform competing models in both the long and short legs of the H–L portfolio. Figure 5 plots the cumulative volatility adjusted returns to the weekly H–L strategies for the CNN model and competing price trend signals.

Decile portfolio analysis provides detailed insight into CNN forecast accuracy. By studying realized returns at different quantiles of model predictions, we can understand accuracy across the full distribution of CNN forecasts, and how this compares to traditional trend-based benchmark strategies. For example, the left panel in Figure 6 shows the average weekly realized return at each decile of the CNN forecast distribution (based on 5-day images, denoted "I5/R5" in the figure), first averaged within deciles each period, then averaged over time (i.e., average returns of equal weight decile portfolios). To illustrate the precision of forecasts, the right panel shows the time series standard deviation of average decile returns. More positive CNN forecasts translate monotonically into higher returns on average. This is also true for MOM, STR, WSTR, and TREND, though with a somewhat flatter slope. An interesting difference

horizon are helpful for predicting different return horizons out-of-sample. Indeed, we see that there is no single image length, and no single supervision window, that is dominant in terms of portfolio performance across investment horizons. For example, it is often the case that quarterly trading strategies benefit when the prediction model is supervised with shorter horizon returns.

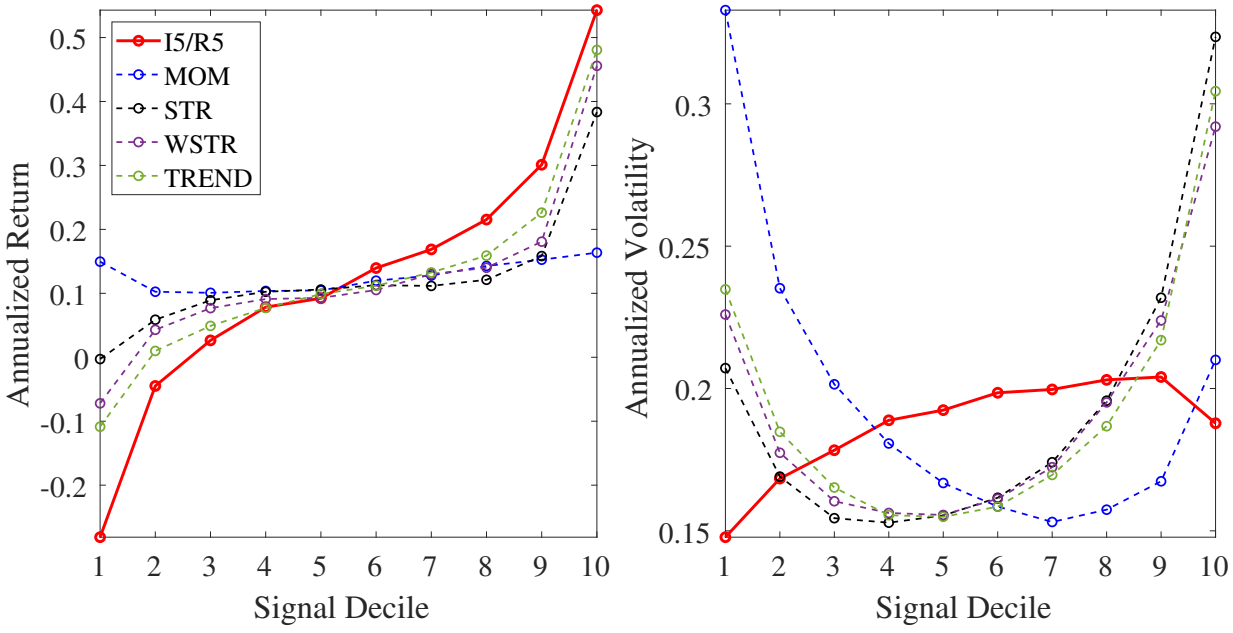
between CNN forecasts and other benchmark price trend signals is in their variability. All CNN decile portfolios have annualized volatility below 20%. Volatilities of MOM, STR, WSTR, and TREND realizations reach around 30% in extreme deciles.

Trend strategies, particularly those used for short horizons, tend to have high turnover. In Table 1 we report the fraction of the strategy that turns over on average scaled in *monthly* terms. Following Gu et al. (2020), we calculate monthly turnover as:

$$\text{Turnover} = \frac{1}{M}\frac{1}{T}\sum_{t=1}^{T}\left(\sum_{i}\left|w_{i,t+1} - \frac{w_{i,t}(1+r_{i,t+1})}{1+\sum_j w_{j,t}r_{j,t+1}}\right|\right),$$

where M is the number of months in the holding period, T is the number of trading periods, $r_{i,t+1}$ is the return of stock i at time $t+1$, and $w_{i,t}$ is the portfolio weight of stock i at time t. Dividing by the number of months makes the turnover measure comparable across different holding periods that we investigate, for example scaling down quarterly strategy turnover by a factor of 1/3 or scaling up weekly strategy turnover by a factor of 4. A strategy that completely reconstitutes its holdings with no overlap from one holding period to the next will have maximum turnover of $200\%/M$ while a buy-and-hold strategy that never rebalances achieves minimum turnover of 0%.

Figure 6: Prediction Accuracy By Decile



Note: The left figure reports average realized returns in each decile of I5/R5 CNN model forecasts and for each decile of the benchmark signals. Return averages are based on cross-sectional decile averages each period that are then averaged over time. The right figure shows the time series volatility of decile returns.

Table 2: Performance of Monthly and Quarterly Strategies

	I5/R20		I20/R20		I60/R20		MOM/R20		STR/R20		WSTR/R20		TREND/R20	
	Ret	SR	Ret	SR	Ret	SR	Ret	SR	Ret	SR	Ret	SR	Ret	SR
Low	-0.00	-0.03	-0.02	-0.12	-0.02	-0.07	0.07	0.20	0.05	0.23	0.01	0.03	-0.01	-0.05
2	0.05	0.28	0.05	0.28	0.06	0.28	0.07	0.27	0.08	0.44	0.06	0.31	0.05	0.25
3	0.07	0.41	0.07	0.39	0.09	0.44	0.09	0.43	0.10	0.63	0.09	0.56	0.07	0.40
4	0.07	0.40	0.09	0.49	0.10	0.52	0.09	0.50	0.10	0.70	0.10	0.65	0.09	0.56
5	0.10	0.53	0.11	0.59	0.11	0.60	0.09	0.56	0.11	0.72	0.11	0.74	0.09	0.65
6	0.11	0.59	0.11	0.59	0.13	0.73	0.11	0.76	0.10	0.66	0.11	0.74	0.10	0.70
7	0.11	0.60	0.13	0.74	0.13	0.73	0.12	0.88	0.12	0.74	0.11	0.71	0.12	0.78
8	0.14	0.73	0.14	0.81	0.13	0.79	0.14	0.96	0.10	0.57	0.11	0.64	0.13	0.81
9	0.16	0.85	0.15	0.87	0.14	0.85	0.14	0.90	0.09	0.42	0.13	0.62	0.17	0.88
High	0.21	1.09	0.18	1.04	0.14	0.99	0.14	0.74	0.16	0.51	0.19	0.66	0.21	0.76
H-L	0.22***	2.35	0.21***	2.16	0.16***	1.29	0.07	0.25	0.11**	0.55	0.18***	1.23	0.22***	1.39
Turnover	175%		173%		155%		63%		168%		167%		140%	
	I5/R60		I20/R60		I60/R60		MOM/R60		STR/R60		WSTR/R60		TREND/R60	
	Ret	SR	Ret	SR	Ret	SR	Ret	SR	Ret	SR	Ret	SR	Ret	SR
Low	0.07	0.31	0.08	0.32	0.08	0.27	0.11	0.25	0.07	0.27	0.06	0.22	0.09	0.23
2	0.10	0.43	0.10	0.41	0.09	0.37	0.10	0.33	0.11	0.50	0.11	0.46	0.09	0.34
3	0.10	0.48	0.11	0.48	0.11	0.46	0.12	0.50	0.12	0.65	0.11	0.55	0.10	0.45
4	0.11	0.49	0.12	0.53	0.12	0.52	0.12	0.57	0.11	0.66	0.11	0.63	0.11	0.58
5	0.11	0.53	0.12	0.54	0.12	0.55	0.11	0.63	0.12	0.73	0.12	0.73	0.11	0.64
6	0.13	0.58	0.12	0.57	0.12	0.57	0.12	0.70	0.12	0.65	0.12	0.69	0.12	0.74
7	0.13	0.60	0.12	0.60	0.14	0.68	0.12	0.71	0.13	0.68	0.12	0.67	0.12	0.78
8	0.13	0.62	0.12	0.61	0.14	0.72	0.13	0.81	0.12	0.54	0.12	0.58	0.12	0.70
9	0.13	0.62	0.13	0.70	0.14	0.75	0.13	0.78	0.12	0.45	0.13	0.52	0.13	0.68
High	0.16	0.77	0.13	0.74	0.15	0.88	0.13	0.57	0.14	0.40	0.16	0.48	0.16	0.55
H-L	0.09***	1.30	0.05	0.37	0.07*	0.43	0.02	0.06	0.07	0.34	0.10***	0.65	0.07*	0.38
Turnover	59%		59%		58%		37%		56%		56%		51%	

Note: Performance of equal-weighted decile portfolios sorted on out-of-sample predicted up probability. Each panel reports the average annualized holding period return and Sharpe ratio. Average returns accompanied by ***,**,* are significant at the 1%, 5% and 10% significance level, respectively. We also report monthly turnover of each strategy.

While Table 1 demonstrates that image-based strategies are highly profitable in gross terms, it also shows that they require significant trading. Image-based strategies turn over roughly as frequently as the weekly return reversal strategy. Other strategies, such as monthly reversal and momentum, have mechanically lower turnover because their signals are moving averages of weekly returns. Below we investigate image-based strategies that trade less frequently.

4.3 Portfolio Performance Over Longer Horizons

MOM and STR strategies are useful turnover benchmarks from the asset pricing literature, with MOM typically viewed as a strategy that survives trading costs while STR does not. To make more direct comparisons with these strategies, we study the performance of image-based strategies that rebalance at the monthly frequency.

The top panel of Table 2 reports portfolio performance for one month holding period equal-weight strategies based on each image size (5, 20, or 60 days). To align the model

with the strategy's rebalance frequency, each CNN is trained to forecast returns over a 20-day horizon. The H–L Sharpe ratios for the I5/R20, I20/R20, and I60/R20 CNN models are 2.4, 2.2, and 1.3, respectively. One-month holding period CNN strategies have essentially the same turnover as STR (and WSTR), but with more than three times the Sharpe ratio of STR and nearly double the Sharpe ratio of WSTR. Momentum has substantially lower turnover than the monthly CNN strategy, but its Sharpe ratio is an order of magnitude smaller.

The bottom panel of Table 2 reports quarterly strategies with CNN models trained on 60-day returns. In this case, the image-based strategies have monthly turnover of about 60%, roughly equal to the turnover of monthly-rebalanced MOM. In this case, the I5/R60 model produces a H–L Sharpe ratio of 1.3, roughly double the next best benchmark (quarterly-rebalanced WSTR with a Sharpe ratio of 0.7) and well in excess of quarterly-rebalanced MOM (Sharpe ratio of 0.1).[10] Over longer horizons, the relative outperformance of CNN strategies is concentrated in the long leg of the H–L portfolio.

The weekly results above illustrate that image predictions are especially valuable at short horizons. The results for monthly and quarterly trading strategies demonstrate that the benefits of image-based predictions continue to be sizable and outperform competitors even when traded at levels of turnover that are much lower and on par with the turnover of MOM. In other words, image-based strategies appear to be profitable in net terms since they can be traded with the same turnover as momentum while earning higher gross Sharpe ratios.

We next investigate whether this longer-horizon performance is due solely to predictability of the first week of returns, or if images can help predict returns beyond the first week. Table 3 decomposes the performance of the monthly-rebalanced CNN strategy (and its competitors) into the return from days 1–5 after rebalance (top panel) and the return from days 6–20 (bottom panel), focusing on CNN models supervised with 20-day returns (Ix/R20). While indeed the bulk of the image-based strategy performance comes from the first week, a significant portion also realizes *after* the first week of trading. The annualized Sharpe ratios over days 6–20 are 0.4, 1.2, and 0.8 for 5-day, 20-day, and 60-day image models, respectively. CNN strategy mean returns over days 6–20 are all significant at the 10% level or better.

4.4 The Effect of Stock Size and Trading Costs

Table 1 shows that Sharpe ratios of image-based strategies more than double when using equal-weight deciles versus value weights. However, as Jensen et al. (2022) point out, pure value weighting does not necessarily give a more economically representative description of empirical return patterns. Nor are strict value weights necessary for constructing portfolios with manageable trading costs (even Fama and French, 1993, value-weight factors give half of

[10]Internet Appendix Table IA3 reports monthly and quarterly results for value-weight decile portfolios.

Table 3: R20 Equal-Weight Portfolio Performance Breakdown

	Day 1 to Day 5													
	I5/R20		I20/R20		I60/R20		MOM/R20		STR/R20		WSTR/R20		TREND/R20	
	Ret	SR	Ret	SR	Ret	SR	Ret	SR	Ret	SR	Ret	SR	Ret	SR
Low	-0.08	-0.84	-0.05	-0.48	-0.02	-0.21	0.07	0.33	-0.00	-0.03	-0.05	-0.37	-0.03	-0.20
2	-0.03	-0.31	-0.01	-0.09	0.00	0.01	0.02	0.12	0.01	0.08	-0.01	-0.12	-0.02	-0.18
3	-0.01	-0.10	0.00	0.02	0.02	0.15	0.01	0.10	0.02	0.18	0.00	0.03	-0.00	-0.05
4	0.00	0.02	0.01	0.12	0.02	0.18	0.01	0.11	0.02	0.18	0.01	0.06	0.01	0.11
5	0.01	0.14	0.02	0.18	0.02	0.22	0.01	0.09	0.02	0.22	0.02	0.18	0.01	0.16
6	0.02	0.23	0.03	0.25	0.03	0.28	0.01	0.17	0.01	0.14	0.02	0.23	0.02	0.22
7	0.04	0.33	0.04	0.35	0.03	0.29	0.02	0.23	0.02	0.18	0.02	0.24	0.02	0.25
8	0.05	0.49	0.05	0.45	0.03	0.31	0.02	0.28	0.01	0.10	0.03	0.30	0.03	0.33
9	0.07	0.69	0.05	0.50	0.04	0.40	0.02	0.20	0.02	0.12	0.04	0.35	0.05	0.48
High	0.12	1.15	0.08	0.74	0.04	0.46	0.02	0.18	0.10	0.55	0.14	0.83	0.12	0.83
H-L	0.20***	3.58	0.13***	2.50	0.07***	1.07	-0.04	-0.28	0.11***	0.91	0.19***	2.11	0.16***	1.54
Turnover	173%		173%		154%		57%		164%		165%		137%	
	Day 6 to Day 20													
	I5/R20		I20/R20		I60/R20		MOM/R20		STR/R20		WSTR/R20		TREND/R20	
	Ret	SR	Ret	SR	Ret	SR	Ret	SR	Ret	SR	Ret	SR		
Low	0.05	0.34	0.01	0.07	0.00	0.01	0.00	0.01	0.03	0.14	0.02	0.12	0.01	0.06
2	0.06	0.36	0.04	0.27	0.04	0.23	0.03	0.15	0.05	0.32	0.05	0.30	0.05	0.26
3	0.06	0.40	0.05	0.31	0.05	0.29	0.06	0.30	0.06	0.41	0.06	0.41	0.05	0.31
4	0.05	0.30	0.06	0.35	0.06	0.36	0.05	0.34	0.07	0.50	0.07	0.52	0.06	0.42
5	0.06	0.37	0.06	0.39	0.07	0.40	0.06	0.42	0.07	0.52	0.07	0.53	0.06	0.45
6	0.07	0.39	0.06	0.36	0.08	0.48	0.07	0.56	0.07	0.49	0.08	0.54	0.06	0.46
7	0.06	0.34	0.08	0.48	0.08	0.48	0.09	0.69	0.08	0.54	0.07	0.46	0.07	0.52
8	0.07	0.39	0.08	0.48	0.08	0.50	0.09	0.69	0.08	0.45	0.07	0.45	0.08	0.54
9	0.07	0.42	0.09	0.52	0.08	0.51	0.09	0.68	0.07	0.32	0.08	0.42	0.10	0.58
High	0.08	0.44	0.09	0.54	0.08	0.60	0.08	0.49	0.06	0.23	0.06	0.24	0.07	0.33
H-L	0.03*	0.42	0.08***	1.21	0.08***	0.83	0.08	0.35	0.04	0.22	0.04	0.33	0.06**	0.58
Turnover	175%		174%		155%		61%		167%		167%		140%	

Note: Breakdown of performance of equal-weighted decile portfolios sorted on out-of-sample predicted up probability. Each panel reports the average holding period return and annualized Sharpe ratios. Average returns accompanied by ***,**,* are significant at the 1%, 5% and 10% significance level, respectively. We also report monthly turnover of each strategy.

all weight to small stocks).

In Table 4, we analyze the effect of stock size by restricting the strategy to only the largest 500 stocks by market capitalization each month. Even with this severe sample restriction to the most liquid stocks, we continue to find significantly positive Sharpe ratios in excess of 1.0 in both equal and value weight strategies. Furthermore, in Internet Appendix Table IA4, we show that the trading profits in Table 1 survive standard trading cost adjustments (10 basis points for stocks exceeding the 80^{th} size percentile of the NYSE, and 20 basis points otherwise, see Frazzini et al., 2018, Ke et al., 2021). We find net-of-cost Sharpe ratios as high as 4.0, 1.5, and 0.9 for weekly, monthly, and quarterly equal-weight strategies, respectively.

4.5 Robustness

Internet Appendix IA.1 reports a number of robustness analyses that support the main empirical findings presented above. We show that image-based strategy returns are not explained by exposure to the market or to well known price trend strategies. We perform sensitivity

Table 4: Portfolio Performance Restricted to 500 Largest Stocks

	Equal Weight													
	I5/R5		I20/R5		I60/R5		MOM/R5		STR/R5		WSTR/R5		TREND/R5	
	Ret	SR	Ret	SR	Ret	SR	Ret	SR	Ret	SR	Ret	SR	Ret	SR
Low	0.05	0.28	0.06	0.35	0.03	0.16	0.07	0.23	0.09	0.37	0.10	0.42	0.07	0.26
2	0.06	0.34	0.06	0.36	0.04	0.24	0.09	0.39	0.12	0.60	0.11	0.61	0.07	0.35
3	0.07	0.39	0.07	0.41	0.06	0.33	0.10	0.54	0.11	0.64	0.12	0.65	0.09	0.47
4	0.09	0.50	0.10	0.52	0.10	0.51	0.10	0.63	0.11	0.68	0.12	0.78	0.11	0.66
5	0.07	0.41	0.08	0.45	0.08	0.45	0.11	0.72	0.10	0.66	0.09	0.57	0.11	0.67
6	0.10	0.54	0.09	0.47	0.09	0.51	0.10	0.72	0.10	0.65	0.11	0.68	0.11	0.74
7	0.08	0.42	0.08	0.45	0.11	0.57	0.11	0.72	0.10	0.60	0.10	0.65	0.10	0.67
8	0.10	0.53	0.11	0.56	0.11	0.60	0.09	0.60	0.09	0.52	0.09	0.53	0.10	0.68
9	0.13	0.68	0.12	0.61	0.13	0.69	0.09	0.50	0.08	0.47	0.08	0.46	0.10	0.63
High	0.17	0.82	0.15	0.75	0.14	0.79	0.11	0.53	0.07	0.32	0.05	0.21	0.11	0.51
H-L	0.12***	1.02	0.09***	0.78	0.11***	1.08	0.05	0.19	-0.02	-0.11	-0.05	-0.30	0.04	0.22
Turnover	712%		676%		638%		30%		47%		47%		42%	
	Value Weight													
	I5/R5		I20/R5		I60/R5		MOM/R5		STR/R5		WSTR/R5		TREND/R5	
	Ret	SR	Ret	SR	Ret	SR	Ret	SR	Ret	SR	Ret	SR		
Low	-0.01	-0.05	0.03	0.18	0.02	0.13	0.06	0.20	0.08	0.33	0.09	0.39	0.08	0.34
2	0.04	0.21	0.04	0.23	0.03	0.18	0.08	0.38	0.10	0.53	0.08	0.46	0.07	0.32
3	0.06	0.33	0.06	0.31	0.05	0.26	0.10	0.57	0.10	0.58	0.10	0.64	0.09	0.52
4	0.09	0.48	0.06	0.33	0.08	0.43	0.12	0.72	0.11	0.70	0.11	0.73	0.09	0.57
5	0.06	0.32	0.07	0.40	0.05	0.26	0.08	0.52	0.11	0.67	0.07	0.47	0.11	0.70
6	0.09	0.52	0.07	0.41	0.08	0.45	0.09	0.67	0.08	0.59	0.10	0.66	0.10	0.69
7	0.07	0.37	0.09	0.53	0.09	0.49	0.08	0.58	0.09	0.55	0.10	0.69	0.08	0.49
8	0.11	0.61	0.11	0.56	0.08	0.47	0.08	0.56	0.08	0.50	0.08	0.49	0.09	0.59
9	0.13	0.69	0.12	0.63	0.10	0.57	0.08	0.46	0.08	0.42	0.07	0.38	0.08	0.51
High	0.16	0.80	0.15	0.77	0.15	0.83	0.11	0.53	0.06	0.25	0.05	0.23	0.08	0.40
H-L	0.17***	1.29	0.12***	0.96	0.13***	1.03	0.06	0.23	-0.02	-0.10	-0.04	-0.19	-0.00	-0.02
Turnover	722%		693%		657%		29%		48%		734%		575%	

Note: Performance of equal-weighted (top panel) and value-weighted (bottom panel) decile portfolios sorted on out-of-sample predicted up probability. Each panel reports the average holding period return and annualized Sharpe ratios. Average returns accompanied by ***,**,* are significant at the 1%, 5% and 10% significance level, respectively. We also report monthly turnover of each strategy.

analysis of the CNN prediction model to alternative choices in image representation (e.g., excluding volume data or moving average price line or using minimal white pixels to represent the data), model architecture (e.g., varying the number of filters in each layer or varying the number of layers), and estimation (e.g., different dropout or batch normalization schemes). Finally, we compare CNN models to alternative computer vision models including "HOG" and "HAAR" (whose model descriptions are also included in the Internet Appendix).

5 What Does the CNN Learn?

Interpreting a CNN is difficult due to its recursive non-linear structure. We attempt to interpret the predictive patterns identified by the CNN using two approaches. First, we relate CNN fits to a collection of conceptually related (price, risk, and liquidity) signals widely studied in the literature. Second, we investigate regression-based logistic approximations to

Table 5: Correlation Between CNN Predictions and Stock Characteristics

CNN Model	MOM	STR	WSTR	TREND	Beta	Volat.	52WH	Bid-Ask	Dollar Volume	Zero Trade	Price Delay	Size	Illiq.
I5/R5	0.00	-0.16	-0.34	0.26	0.07	-0.01	-0.02	0.01	0.13	-0.10	-0.03	0.12	-0.12
I5/R20	0.01	-0.12	-0.28	0.33	0.06	-0.03	-0.04	-0.01	0.15	-0.12	-0.04	0.14	-0.15
I5/R60	0.03	-0.05	-0.11	0.22	0.03	-0.08	-0.09	-0.07	0.15	-0.09	-0.03	0.15	-0.16
I20/R5	0.05	-0.09	-0.34	0.26	0.01	-0.11	-0.10	-0.10	0.14	-0.06	-0.03	0.16	-0.15
I20/R20	0.08	0.02	-0.23	0.22	0.01	-0.18	-0.18	-0.15	0.24	-0.11	-0.04	0.27	-0.26
I20/R60	0.10	0.21	-0.02	0.09	-0.00	-0.19	-0.19	-0.16	0.24	-0.10	-0.04	0.27	-0.27
I60/R5	0.18	-0.05	-0.26	0.20	-0.06	-0.24	-0.22	-0.23	0.18	-0.01	-0.02	0.24	-0.23
I60/R20	0.20	0.11	-0.07	0.12	-0.05	-0.31	-0.30	-0.27	0.29	-0.08	-0.04	0.37	-0.35
I60/R60	0.21	0.11	-0.01	0.09	-0.06	-0.26	-0.26	-0.23	0.23	-0.03	-0.02	0.31	-0.28

Note: The table reports average cross-sectional correlations among model forecasts averaged over all period in the test sample. All correlations are based on predictions made at the end of each month.

the CNN using the data underlying the CNN's image input. Our attempts at interpretation are admittedly incomplete (as in the CNN literature more broadly). Yet they achieve partial success by offering some visibility into the complex inner workings of the CNN model.

5.1 Association with Other Predictors

Table 6: CNN Predictions and Standard Stock Characteristics

	5D5P	20D5P	60D5P
MOM	-0.10***	0.01**	0.40***
STR	-0.09***	0.27***	0.24***
Lag Weekly Return	-0.85***	-1.00***	-0.89***
TREND	0.51***	0.46***	0.23***
Beta	0.11***	0.15***	0.22***
Volatility	-0.09***	-0.20***	-0.24***
52WH	-0.05***	-0.03***	-0.09***
Bid-Ask	0.10***	-0.11***	-0.08***
Dollar Volume	0.20***	0.16***	-1.22***
Zero Trade	-0.09***	0.00	0.32***
Price Delay	-0.01***	-0.01**	0.00
Size	0.21***	0.40***	0.44***
Illiquidity	0.08***	0.19***	-1.43***
McFadden R^2	8.20	8.56	9.78

Note: The table reports slope coefficients and R^2 from panel logistic regressions of CNN model forecasts on stock characteristics. Panel regressions are estimated during the test sample using CNN models estimated in the training sample. Coefficients accompanied by ***,**,* are significant at the 1%, 5% and 10% significance level, respectively, using Newey-West standard errors.

How unique are image-based forecasts compared to standard characteristics in the literature? We focus our comparison on measures of recent price trends (MOM, STR, WSTR, and distance from 52-week high), risk (beta and volatility), and liquidity (bid-ask spread, dollar

volume, number of no-trade days, price delay, size, and Amihud illiquidity).[11] Table 5 reports univariate correlations between each of these characteristics and each image-based forecast. We estimate period-by-period cross-sectional correlations among cross-sectional ranks of each variable, then report the time series average correlation during the test sample. Among the largest associations is WSTR, which has correlation between −26% and −34% for models supervised by future 5-day returns, consistent with the CNN picking up in part on a weekly short-term reversal pattern. The WSTR correlation drops close to zero when CNN models are supervised by 60-day returns. In a similar vein, MOM has its highest correlation (21%) with long-horizon forecasts from large images (I60/R60), but essentially zero correlation with short-horizon forecasts from small images (I5/R5). Image-based predictions from the CNN also resemble non-price firm characteristics. Other strong correlates are size, volatility, bid-ask spread, illiquidity, and dollar volume, all of which reach correlations near 30% and associate most strongly with forecasts from 60-day images.

These correlations are an impressive feat for the CNN. It shows that the CNN model has the ability to discern meaningful predictive information from data that is represented abstractly in the form of an image. MOM, STR, and other common price trend variations are predictive features that have been manually curated by human researchers over a decades long research process. But the CNN is oblivious to human-engineered features; instead, feature engineering is fully automated and integrated into the CNN model itself. Without requiring hand-crafted trend signals, the CNN still manages to identify trend-like features and liquidity features in the raw images.

Table 6 shows the joint explanatory power for image-based predictions from all characteristics simultaneously (this table focuses on our main specification using 5-day CNN predictions). We estimate a panel logistic regression using cross-sectional ranks of all regressors. In multiple regression, the most important explanatory variables across image size are TREND and WSTR (for 60-day images, illiquidity and volume become especially important). The McFadden logistic R^2 ranges from 8.2% to 9.8%. So, while the CNN is able to detect signals within images that are significantly correlated with other well known predictors, the variation in image-based predictions is by and large unique.

Next, Table 7 reports logistic regressions of future stock returns on image-based forecasts while simultaneously controlling for the other characteristics. To remain comparable with the CNN model, the dependent variable is an indicator for a positive 5-day realized return. We es-

[11] For variable definitions and references, see Table A.6 of Gu et al. (2020).

[12] To ensure high-quality estimates of in-sample mean for each stock, we require a stock to have at least 3 years of data (12 observations for quarterly returns). For stocks that fail to meet this criterion, we use the overall in-sample mean across all stocks instead. The same adjustment applies to Table 8.

Table 7: CNN, Future Returns, and Standard Stock Characteristics

	I5/R5			I20/R5			I60/R5		
CNN	0.28***		0.24***	0.30***		0.23***	0.23***		0.13***
MOM		0.04***	0.04***		0.04***	0.04***		0.04***	0.02***
STR		0.02***	0.02***		0.02***	0.00		0.02***	0.01***
Lag Weekly Return		-0.14***	-0.08***		-0.14***	-0.07***		-0.14***	-0.11***
TREND		0.11***	0.07***		0.11***	0.07***		0.11***	0.10***
Beta		0.04***	0.03***		0.04***	0.03***		0.04***	0.03***
Volatility		-0.07***	-0.07***		-0.07***	-0.06***		-0.07***	-0.06***
52WH		-0.01*	-0.01		-0.01*	-0.01		-0.01*	-0.01
Bid-Ask		-0.13***	-0.14***		-0.13***	-0.12***		-0.13***	-0.13***
Dollar Volume		-0.12***	-0.13***		-0.12***	-0.13***		-0.12***	-0.09***
Zero Trade		-0.01	-0.00		-0.01	-0.01		-0.01	-0.02**
Price Delay		0.01*	0.01**		0.01*	0.01**		0.01*	0.01*
Size		-0.04***	-0.05***		-0.04***	-0.06***		-0.04***	-0.06***
Illiquidity		-0.21***	-0.21***		-0.21***	-0.22***		-0.21***	-0.17***
OOS McFadden R^2	0.13	0.09	0.38	0.20	0.09	0.37	0.19	0.09	0.17

Note: The table reports slope coefficients from panel logistic regressions of future returns on image-based CNN model forecasts and standard stock characteristics. Panel regressions are estimated during the test sample using CNN models estimated in the training sample. Coefficients accompanied by ***,**,* are significant at the 1%, 5% and 10% significance level, respectively, using Newey-West standard errors. The table also reports out-of-sample McFadden R^2s, comparing the cross-entropy loss of a model (estimated in the training sample) against that of a benchmark model using stock-level in-sample mean as the predictor.[12]

timate the regressions using only the test sample data (so that CNN parameters are estimated from an entirely distinct sample). For each CNN specification, we estimate three regressions using test sample data, one using only the CNN forecast, one using stock characteristics and excluding the CNN forecast, and one joint regression using all predictors. The coefficients corresponding to the second regression are identical across three CNN specifications since the regressors (characteristics) and the dependent variable (R5) are the same. Coefficients on CNN image-based predictions are fairly close in univariate and multivariate regressions.

To evaluate the overall model performance, we report out-of-sample MacFadden R^2s. We see that, across CNN models, the image-based prediction is by far the strongest return forecaster. The out-of-sample McFadden R^2 due to image-based predictions alone ranges from 0.13% to 0.20% depending on the model configuration. The R^2 from all non-image based characteristics together is 0.09%. In addition, a large fraction of the predictive R^2 in the multivariate model is accounted for by the univariate CNN prediction, indicating that unique aspects of the CNN signal and not the previously studied characteristics drive the CNN's predictive power.

5.2 Logistic Approximation

What do the CNN models detect in images to produce accurate forecasts that are differentiated from traditional stock-level predictors? In this section, we use logistic regressions to

approximate the CNN based on the raw market data that underlie our images. Our CNN is structured as a binary classifier, so its forecast output is a probability. We thus use logistic regression for our approximating models rather than the simple linear model. The argument to the logistic function is a linear model, so logistic regression estimates can be viewed as a linear approximation to the CNN.

To estimate the approximating models, we must first decide on a representation of the price history data that is amenable for regression. The beauty of the CNN model, and the reason for its widespread usage in machine learning applications, is that it has the ability to extract predictive features with minimal data engineering on the part of the researcher. This contrasts with the standard approach from the momentum and reversal literature, which makes a number of human feature engineering choices—for example, representing historical data as returns rather than price levels to make series more comparable across assets. Our logistic approximation analysis highlights the typical reliance on human feature engineering, because now we must also decide how to transform not only close prices, but also intra-day high and low prices, open prices, price moving averages, and volumes to make them relatable in the cross section.

To make logistic regressions comparable with the image CNN model, we scale all price series such that the maximum of all prices appearing in the image (usually the maximum high price, but sometimes the maximum moving average price) is normalized to one and the minimum is normalized to zero. Daily volumes are likewise scaled by the maximum volume appearing in the image.

Our regressions focus on 5-day images. 5-day images contain relatively few data points (five observations each for open, high, low, and close price, volume, and moving average price), which makes them a convenient context for attempting to isolate image attributes that contribute to CNN success. In particular, there are few enough observations in a 5-day image that we can add each as a separate regressor in a logistic model without risk of severe overfit.

Columns (1) through (3) of Table 8 pursue a logistic approximation to the CNN forecast itself, as in Table 6. The dependent variable is the out-of-sample forecast generated by the CNN model for 5-day images (1 if the CNN predicted probability is greater than 0.5, and 0 otherwise), and the independent variables are data underlying 5-day images re-scaled to mimic the image representation. Across all return horizons (5, 20, and 60 days), the most important explanatory variables for the CNN forecast are the first lags of closing, high, and low prices, with the next most important variables being the first lag of moving average price and trading volume. These variables generally have consistent sign and magnitude for all return horizons. Compared to the first lag, lags two through five have a small role in the CNN forecast, though

Table 8: Logistic Regressions Using Market Data With Image Scaling

	CNN as Dep. Var.			Positive Return Indicator as Dep. Var.								
	I5/R5	I5/R20	I5/R60	I5/R5			I5/R20			I5/R60		
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)
CNN				2.21*		2.12	2.51*		1.98	2.30*		2.57
open lag 1	-0.57*	0.01	0.29*		-0.25	-0.16		-0.13	-0.12		0.12	0.09
open lag 2	0.27*	0.55*	0.22*		0.08*	0.05		0.13	0.09		-0.10	-0.13
open lag 3	0.16*	-0.29*	-0.09*		-0.08	-0.10		0.00	0.02		0.14	0.15
open lag 4	0.04*	-0.13*	-0.06*		-0.08	-0.09		-0.05	-0.05		0.14	0.14
open lag 5	0.10*	0.22*	0.45*		0.03*	0.02		-0.00	-0.01		-0.25	-0.26
high lag 1	2.64*	0.99*	1.07*		0.39*	0.09		0.17	0.08		0.07	-0.08
high lag 2	-0.27*	0.47*	0.77*		0.05*	0.08		0.05	0.01		-0.12	-0.21
high lag 3	-0.43*	-0.18*	-0.20*		0.12*	0.16		0.00	0.02		0.07	0.09
high lag 4	-0.47*	-0.14*	-0.22*		0.06*	0.11		0.23	0.23		0.01	0.05
high lag 5	-0.13*	0.31*	-0.38*		0.01	0.03		-0.22	-0.24		0.02	0.01
low lag 1	1.89*	0.94*	-0.06*		0.48*	0.20		0.40	0.31		0.21	0.20
low lag 2	-0.17*	0.87*	0.74*		0.12*	0.16		0.07	0.02		0.14	0.04
low lag 3	-0.19*	-0.06*	-0.18*		0.05*	0.08		-0.11	-0.10		-0.10	-0.08
low lag 4	0.28*	0.41*	-0.13*		0.03*	0.01		0.15	0.12		-0.17	-0.17
low lag 5	-0.12*	-0.14*	-0.37*		0.09*	0.12		-0.21	-0.20		-0.17	-0.12
close lag 1	-5.36*	-3.91*	-2.19*		-0.62	-0.09		-0.52	-0.24		-0.36	-0.08
close lag 2	0.72*	0.32*	0.19*		0.03	-0.02		0.01	-0.01		0.23	0.23
close lag 3	0.64*	0.83*	0.28*		-0.03	-0.06		0.14	0.11		0.22	0.24
close lag 4	0.73*	0.29*	0.20*		-0.10	-0.13		0.00	-0.00		0.23	0.25
close lag 5	0.49*	0.55*	0.39*		0.09	0.07		0.50	0.49		0.28	0.30
ma lag 1	-1.33*	-1.06*	-0.56*		-0.12	-0.05		0.03	0.07		0.12	0.19
ma lag 2	0.31*	0.35*	-0.01		-0.29	-0.30		-0.39	-0.37		-0.66	-0.80
ma lag 3	-0.03*	-0.77*	0.20*		-0.06	-0.10		-0.52	-0.57		-0.09	-0.15
ma lag 4	-0.02	0.34*	0.63*		0.48*	0.45		0.41	0.46		-0.32	-0.37
ma lag 5	0.24*	-0.45*	-0.61*		-0.22	-0.20		0.00	0.01		0.59	0.64
vol lag 1	0.81*	0.71*	0.42*		0.06*	0.00		0.12	0.08		0.16	0.12
vol lag 2	0.21*	0.52*	0.58*		0.05*	0.04		-0.01	-0.03		0.03	-0.02
vol lag 3	0.13*	0.15*	0.57*		0.04*	0.03		-0.01	-0.02		0.08	0.03
vol lag 4	-0.07*	0.12*	0.09*		0.05*	0.05		0.04	0.04		0.18	0.17
vol lag 5	0.05*	0.01	0.30*		0.02*	0.02		0.05	0.05		0.04	0.02
OOS McFadden R^2	35.29	33.32	21.96	0.73	0.55	0.73	0.47	0.47	0.45	1.33	1.38	1.24

Note: Results of logistic regressions on out-of-sample CNN forecasts and realized future return indicators on daily price and volume data. Panel regressions are estimated during the test sample using CNN models estimated in the training sample. Coefficients accompanied by * are significant at the 1% significance level using Newey-West standard errors. The reported out-of-sample McFadden R^2s compares the cross-entropy loss of a model (estimated in the training sample) against that of a benchmark model using stock-level in-sample mean as the predictor.

some of these are statistically significant. The regression-based approximation to the true non-linear CNN construction explains 22% to 35% of the variation in image-based predictions. But CNN predictions are built only and entirely from this raw market data. Therefore, more than half of the variation in image-based predictions is attributable to non-linear functions of the underlying market data.

Next, we use the logistic specification to directly forecast returns with the approach used in Table 7. Regressions are grouped by supervising return window (5, 20, or 60 days). As

a frame of reference, columns (4), (7), and (10) report return predictive regressions using only the out-of-sample CNN forecast. Next, columns (5), (8), and (11) show another approach to approximating the CNN by directly predicting returns using image-scaled market series. We find that the largest regression coefficients tend to be on the first lag of the high, low, and close prices. Together, their coefficients suggest a signal that is roughly equal to $\frac{1}{2}$(High+Low)−Close for the previous day. In other words, future returns tend to be high when the price closes at the low end of the recent high-low range. The regression also isolates features that look like deviations of the lagged prices from their recent averages. Recent rises in volume also notably predict positive future returns. These patterns are consistent with the coefficient estimates from columns (1) through (3), indicating that the CNN extracts similar patterns to those isolated by the logistic model. Though, again, the approximating pattern explains less than half of the content in CNN predictions.

Columns (6), (9), and (12) compare the CNN's non-linear predictions with that of a simple logistic model by regressing the sign of realized returns on CNN predictions while controlling for the image-scaled market data. First, we see that controlling for the underlying market data does not increase the predictive coefficient on the CNN-based forecast. The out-of-sample R^2 using only the CNN as a predictor ranges from 0.7% to 1.3% compared to the benchmark based on stock-level in-sample averages. The logistic approximation delivers a lower R^2 at the weekly horizon, yet a similar R^2 at monthly and quarterly horizons. The combined R^2 in the third column of each group of regressions is worse than that of the CNN-only regression. This indicates that the non-linear component of the CNN forecast incorporates useful additional information most evident at a shorter horizon from the underlying images relative to the logistic model.

Table 9 compares the performance of long-short decile spread portfolios formed on CNN forecasts versus the linear approximation via logistic regression. This is reported in the rows labeled "Logistic (image scale)." It shows that the linear approximation to the CNN delivers a successful trading strategy, but is generally inferior to the full non-linear CNN model (and generally superior to the benchmark models reported earlier). The logistic model is most competitive at longer horizons, where the 60-day holding period strategies in some cases outperform the full CNN.

The strength of trading strategies based on the logistic model begs the question: is the image representation necessary at all? Or would a logistic model with a traditional time series representation of past data work just as well? To demonstrate the key role of representing market data as an image, we also study logistic prediction models based on more "standard" time series representations. Of course *some* scaling choice must be made for market data to

Table 9: Portfolio Performance of CNN versus Logistic Model and CNN1D

	5-day Images		20-day Images		60-day Images	
	Equal Weight	Value Weight	Equal Weight	Value Weight	Equal Weight	Value Weight
			5-day Return Horizon			
CNN	7.15	1.49	6.75	1.74	4.89	1.44
Logistic (image scale)	5.56	1.60	4.82	1.52	4.91	1.83
Logistic (cum. ret. scale)	2.50	0.23	1.19	0.36	1.14	0.46
Logistic (devol. ret. scale)	4.70	1.14	4.73	1.32	4.68	1.27
CNN1D (image scale)	7.20	1.66	7.88	1.84	7.85	2.61
CNN1D (cum. ret. scale)	5.33	1.71	5.36	1.52	5.45	1.85
CNN1D (devol. ret. scale)	-0.13	0.07	2.03	0.34	1.25	0.16
			20-day Return Horizon			
CNN	2.35	0.45	2.16	0.49	1.29	0.17
Logistic (image scale)	2.07	0.66	1.99	0.44	1.23	0.34
Logistic (cum. ret. scale)	0.51	0.11	0.41	0.33	0.43	0.30
Logistic (devol. ret. scale)	1.42	0.90	1.44	0.78	1.12	0.36
CNN1D (image scale)	2.49	0.70	2.72	0.59	2.45	0.88
CNN1D (cum. ret. scale)	1.47	0.67	1.45	0.57	1.46	0.44
CNN1D (devol. ret. scale)	-0.46	0.01	0.69	0.19	0.97	0.12
			60-day Return Horizon			
CNN	1.30	0.47	0.37	0.32	0.43	0.23
Logistic (image scale)	1.19	0.54	0.75	0.43	0.74	0.63
Logistic (cum. ret. scale)	-0.04	-0.07	0.07	0.19	0.13	0.17
Logistic (devol. ret. scale)	0.23	0.29	0.27	0.30	0.26	0.25
CNN1D (image scale)	1.28	0.48	1.04	0.44	0.76	0.49
CNN1D (cum. ret. scale)	0.24	0.19	0.35	0.21	0.24	-0.06
CNN1D (devol. ret. scale)	0.19	0.00	0.20	-0.12	0.32	-0.06

Note: The table reports annualized Sharpe ratios of long-short decile spread portfolios (with equal weights or value weights) sorted on out-of-sample predicted up probability from each model. "CNN" is the baseline 2D CNN, "logistic" is the simple logistic regression model, and "CNN1D" is the time series CNN model. "Image scale" indicates the same high/low price normalization used to produce images. "Cum. ret. scale" indicates that market prices are normalized by the closing price at the start of each image. "Devol. ret. scale" indicates that prices each day are converted to returns then normalized by conditional volatility.

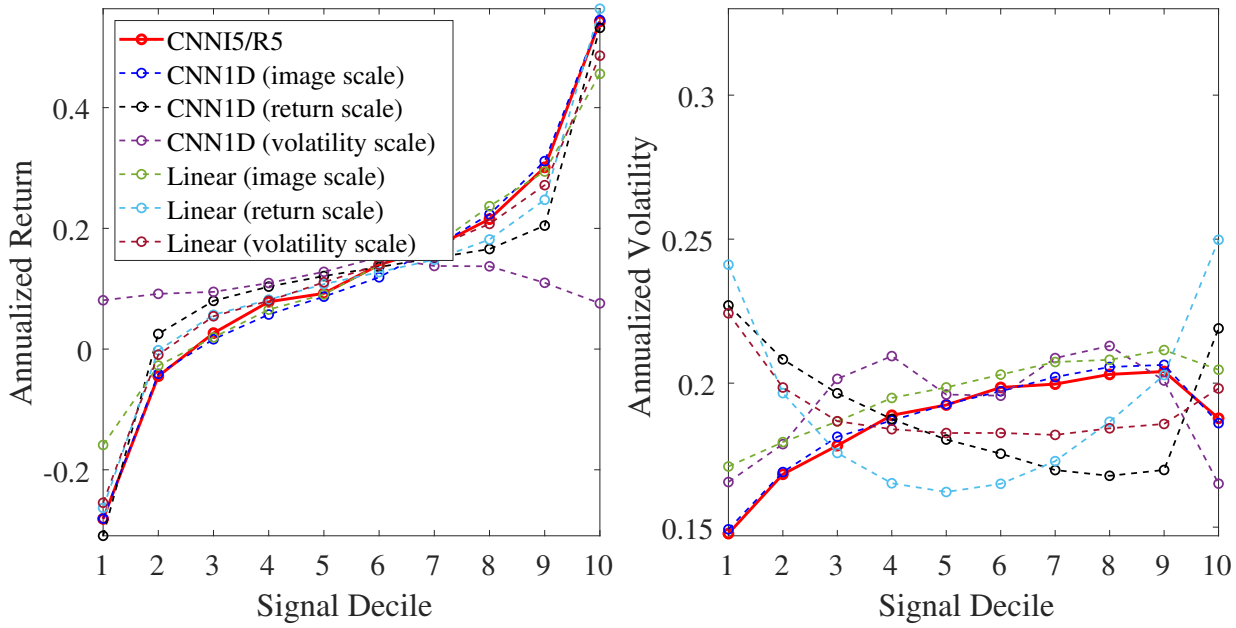
be comparable across stocks in a logistic model. One natural candidate is to scale all prices by the first closing price in the historical data window (i.e., the closing price either 5, 20, or 60 days ago) and likewise for volume. We refer to this as "cumulative return scale" because it represents prices similarly to those in a cumulative return plot. The trading strategy based on a logistic model with this data representation is shown in the rows "Logistic (cum. ret. scale)." While this model also produces positive trading performance across the board, it is substantially weaker than the logistic model with image-scaled market data. For example, with 20 days of historical data, the equal weight 20-day holding period strategy drops from an annualized Sharpe ratio of 2.0 in the "Logistic (image scale)" case to 0.4 for "Logistic (cum. ret. scale)." For value weight strategies, "Logistic (cum. ret. scale)" is again worse than "Logistic (image scale)" in all cases.

Cumulative return scaling continues to look at prices in levels. Next, we study a representation in terms of price and volume changes. In particular, we use daily returns normalized by their exponentially-weighted moving average volatility (with smoothing parameter 0.05). We refer to this as "devolatized return scale." Given the standard asset pricing approach of building price trend signals from past returns, this is a natural data representation for a logistic model. Indeed, consistent with standard price trend signals, logistic models of past returns perform reasonable well. They are even competitive with the CNN for 60-day price history specifications. However, the overall best performing CNN models (those based on 5-day price histories) dominate the best performing logistic model with devolatized return scale by a large margin.

As a last point of emphasis for the importance of the image representation, we analyze 1D CNN models. These are time series CNN models in the sense that they use a continuous numerical representation of market data time series, and perform convolution only by sliding convolutional filters along the time series dimension of the data matrix. "CNN1D (image scale)" reports a trading strategy when the 1D CNN is applied to market data time series with image scaling, while "CNN1D (cum. ret. scale)" and "CNN1D (devol. ret. scale)" show cases with alternative data formulations. Two interesting points emerge. The first is that with image scaling, the 1D CNN performs roughly as well as, and in some cases better than, the baseline 2D CNN model. This, however, is not because the 1D CNN is superior to the 2D CNN for return prediction more generally. We see that "CNN1D (cum. ret. scale)" and "CNN1D (devol. ret. scale)" broadly underperform the baseline 2D CNN, as well as underperforming the "CNN1D (image scale)" model. In other words, the key performance differentiator, be it for a 2D CNN, a 1D CNN, or a logistic model, is using an image representation. Once the data is represented as an image, either literally (as in our baseline case) or figuratively (using image scaling of time series data), a deep learning model can use historical market data to produce powerful return forecasts.

Figure 7 repeats the analysis of Figure 6 but compares 2D CNN decile portfolios to those from 1D CNN and logistic models in order to better understand why the CNN avoids of high volatility in deciles 1 and 10. One candidate explanation for the volatility shape in Figure 6 might be that the forecast output from a CNN is a probability, while other signals (like MOM and STR) may mechanically have higher volatility in extreme deciles. Figure 7 shows that this is unlikely to be the explanation because all of the logistic models also output a probability, but they also tend to produce U shapes in volatility. Instead, it appears that the CNN decile volatility pattern is primarily due to image scaling of the raw data.

Figure 7: Prediction Accuracy of CNN and Logistic Models By Decile



Note: The left figure reports average realized returns in each decile of I5/R5 CNN model forecasts, as well as for each decile of forecasts from 1D CNN models and logistic models. Return averages are based on cross-sectional decile averages each period that are then averaged over time. The right figure shows the time series volatility of decile returns.

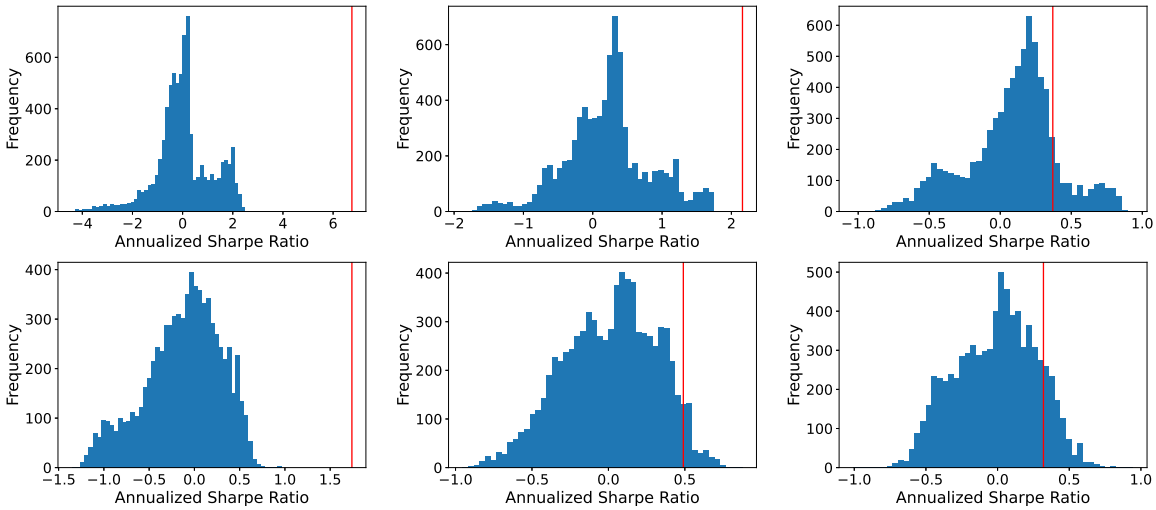
5.3 Traditional Technical Analysis

Brock et al. (1992) find that 26 pre-defined (and commonly used) technical trading rules generate significantly positive investment performance. Lo et al. (2000) introduce a kernel regression approach to recognizing technical indicators and find further support for the positive performance of investment strategies that rely on technical analysis.

Sullivan et al. (1999) assess a universe of 7,846 pre-defined technical trading rules while carefully controlling for multiple testing to control the likelihood of false positives. While they find that none of the Brock et al. (1992) strategies deliver significant positive performance in the post-publication sample, they find evidence of significant positive performance in their larger universe of trading rules. More recently, Bajgrowicz and Scaillet (2012) revisit the same universe of 7,846 rules and conclude that none are significant after controlling the false discovery rate.

The Sullivan et al. (1999) universe of 7,846 technical rules offers an excellent opportunity to calibrate the significance of our results. They constitute a null distribution for benchmarking other technical analysis strategies. To use these benchmarks, we apply each of the 7,846

Figure 8: Comparison With Traditional Technical Indicators



Note: The figure shows the histogram of Sharpe ratios for 7,846 technical analysis portfolios that rebalance by week, month, and quarter, respectively. The top panel shows equal weight portfolios and the bottom panel shows value weight portfolios. Red vertical bars are the corresponding Sharpe ratios for the I20/R5, I20/R20, and I20/R60 CNN strategies, respectively.

trading rules stock-by-stock and produce a stock-level signal,[13] then construct decile spread long-short strategies in the same way that we build our long-short CNN strategy. We use technical signals at the end of each week, month, or quarter to rebalance, depending on the portfolio holding period, and consider equal weighting and value weighting. This gives us a distribution of performances for 7,846 different strategies against which we compare the CNN strategy.

Figure 8 reports the histogram of annualized Sharpe ratios for the 7,846 technical signals and the corresponding CNN strategy Sharpe ratio (red bars). At the weekly horizon, zero of the 7,846 outperform the CNN strategy (regardless of weighting scheme), and the closest technical indicator is outperformed by a large margin. At the monthly horizon, again none of the technical rules exceed CNN performance in equal weighted terms, and only 4.4% exceed CNN when value weighted. At the quarterly horizon, the outperformance of CNN diminishes somewhat, with 13.4% and 12.9% of technical rules outperforming CNN with equal and value weighting, respectively.

There are two considerations to bear in mind when interpreting the results of Figure 8. First, the CNN exceedence rates are not the same as p-values because the technical strategy distribution is not truly a null distribution. There may in fact be some true positives in the

[13]We are grateful to Olivier Scaillet for sharing his code.

technical strategy universe, so in this sense the exceedance rate is more conservative than a p-value. Second, the technical indicator literature has focused only on very short holding periods, typically a single day. The fact that a non-trivial proportion of the Sullivan et al. (1999) technical signals achieve high Sharpe ratios at the weekly, monthly, and even quarterly frequency is a new and potentially interesting finding that warrants further investigation (though is beyond the scope of this paper).

A handful of common technical patterns have evolved into a frequently marketed folk wisdom[14] that associates each price pattern (such as "head and shoulders" or "double bottom") with a sign for future returns. We use our estimated CNN model to evaluate the reliability of such popular chartist patterns. The logic of our analysis is the following. Our model learns general price patterns that predict future returns. We can feed an image of a candidate price pattern into the estimated model to query whether the CNN expects it to be followed by a positive or a negative return.

We study 23 price patterns commonly referenced in technical analysis textbooks. We simulate each pattern in 20-day and 60-day images as described in Internet Appendix IA.2. We feed these into the CNN model estimated from real data and evaluate the model-based probability of a positive subsequent return. We repeat this for 10,000 simulated images and report the average probability in Table IA19. For 20-day images, we find that about half of popular patterns (13 of 23) have a significant association with directional return forecasts from the CNN. However, among these 13 patterns, 8 go in the opposite direction of the folk wisdom prediction.

Of course, our CNN model is not ground truth. But it has a few attributes that make it a useful proving ground for folk wisdom recommendations. First, the model is empirically successful as documented in Section 4. Second, we have a reassuring placebo result in Table IA19: When we feed the CNN images simulated from totally random Brownian motions, the average probability of positive subsequent return is statistically indistinguishable from 50%. Third, associations between certain price patterns and CNN predictions appear significantly non-random. All told, this evidence tilts us toward the conclusion that some price patterns are likely to be predictive, but that those typical directional patterns from technical analysis books do not seem profitable in our context.

6 Transfer Learning

Our analysis to this point has focused on daily data for the US stock market. Training the CNN in this setting confers two advantages from data size. First, because the US stock

[14] A brief internet search yields hundreds of books on technical analysis patterns, many of them independently published. For instance, a classical reference on technical trading is Murphy (1999).

market is the largest in the world, the CNN benefits from a wide cross section of observations. Second, our use of daily data extends the time series dimension of our data set by a factor of five compared to using weekly data and a factor of 20 compared to using monthly data. However, we may be interested in applying CNN forecasts in contexts for which re-training the CNN model is infeasible. For example, in order to isolate low frequency dynamics in returns, one may wish to use images of long past price histories and forecast several months into the future. This is likely to be prohibitive for a CNN due to the data requirements necessary to achieve a sufficiently well trained model. Likewise, we may be interested in forecasting returns in small or emerging markets where the cross section may contain only a few hundred stocks.

In this section, we explore the possibility of *transfer learning* for constructing accurate image-based forecasts in contexts with limited data. Transfer learning uses patterns identified in one context to guide analysis or prediction in a different context.[15] We show that the prediction patterns identified by the CNN from daily US stock data transfer well to international markets and to other time scales. In doing so, we show that it is possible to apply CNN models estimated in daily US data to construct profitable portfolios in other settings that preclude direct training of the CNN. At the same time, it provides an additional out-of-sample demonstration of reliable predictive power of image-based CNN forecasts.

6.1 International Transfer

International markets have fewer stocks and shorter time series compared to the US. If estimates from US data are applicable in international contexts, it helps alleviate the limitations of conducting data-intensive analysis in small markets. More generally, if predictive patterns in stock prices are to some degree global phenomena, then forecasts in data sparse markets can draw estimation strength from information in data rich markets. We explore the viability of international transfer learning by using CNN models estimated from US data to construct return forecasts in foreign markets.

We use daily stock market data from Datastream for Hong Kong, United Kingdom, Canada, Japan, Australia, Austria, Belgium, Denmark, Finland, France, Germany, Greece, Ireland, Italy, Netherlands, New Zealand, Norway, Portugal, Singapore, Spain, Sweden, Switzerland, Russia, India, and South Korea; and we use daily stock market data from CSMAR for mainland China.[16] The date range for most countries matches the US sample, 1993

[15] See, e.g., Pan and Yang (2009) for a survey of the computer science literature on transfer learning. In asset pricing, existing literature typically retrain the model discovered in US market using data from international markets. The direct-transfer approach similar to what we propose here is rare, with the exception of a concurrent paper by Liu et al. (2020), which applies their genetic programing model trained with US data directly to G7 international markets.

[16] For exposition, we refer to each of these markets as "countries," though Hong Kong is a special administrative region of China.

Table 10: International Transfer and H-L Decile Portfolio Sharpe Ratios (I5/R5)

		Equal Weight			Value Weight		
	Stock Count	Re-train	Direct Transfer	Transfer−Re-train	Re-train	Direct Transfer	Transfer−Re-train
US	7298	7.15			1.49		
Global	17206	0.18	5.20	5.03***	0.46	-3.05	-3.50
Japan	3056	3.56	5.68	2.12***	0.96	1.23	0.27
Canada	2924	9.01	12.12	3.11***	2.98	5.34	2.36***
India	1861	2.52	-1.46	-3.98	0.67	-1.08	-1.75
UnitedKingdom	1783	0.03	-0.23	-0.26	1.04	0.98	-0.06
France	955	2.47	4.09	1.63***	1.12	2.10	0.98***
SouthKorea	911	3.64	1.66	-1.97	1.74	2.39	0.65***
Australia	886	8.28	11.37	3.09***	2.78	3.48	0.70***
Germany	868	-0.29	2.43	2.72***	-0.01	2.93	2.94***
China	662	2.26	-2.19	-4.45	0.66	-0.95	-1.62
HongKong	543	1.97	5.35	3.37***	0.72	2.08	1.36***
Singapore	284	6.98	6.79	-0.20	2.48	3.94	1.46***
Sweden	260	5.43	6.99	1.56***	0.83	2.37	1.54***
Italy	241	2.14	3.55	1.40***	0.76	1.60	0.84***
Switzerland	240	0.48	0.67	0.19	1.30	2.62	1.33***
Denmark	223	1.94	3.56	1.62***	1.18	1.85	0.68***
Netherlands	212	-0.30	3.75	4.05***	0.11	1.67	1.56***
Greece	201	2.74	3.26	0.51**	0.98	1.88	0.90***
Belgium	171	0.73	4.34	3.60***	0.73	2.88	2.15***
Spain	170	1.62	0.28	-1.35	0.68	1.02	0.34*
Norway	169	0.79	3.38	2.59***	1.11	2.88	1.77***
Portugal	121	0.30	2.64	2.33***	0.93	1.40	0.47**
NewZealand	114	0.50	2.34	1.84***	0.65	1.19	0.54***
Finland	113	2.66	5.38	2.72***	0.95	2.55	1.60***
Austria	110	0.14	0.67	0.53**	0.66	1.05	0.39**
Ireland	75	0.47	1.80	1.34***	0.31	1.99	1.69***
Russia	53	-0.72	2.19	2.91***	-0.13	0.44	0.57***
Average	1274	2.21	3.54	1.34	0.99	1.73	0.75
Average (excluding Global)	661	2.28	3.48	1.19	1.01	1.92	0.91

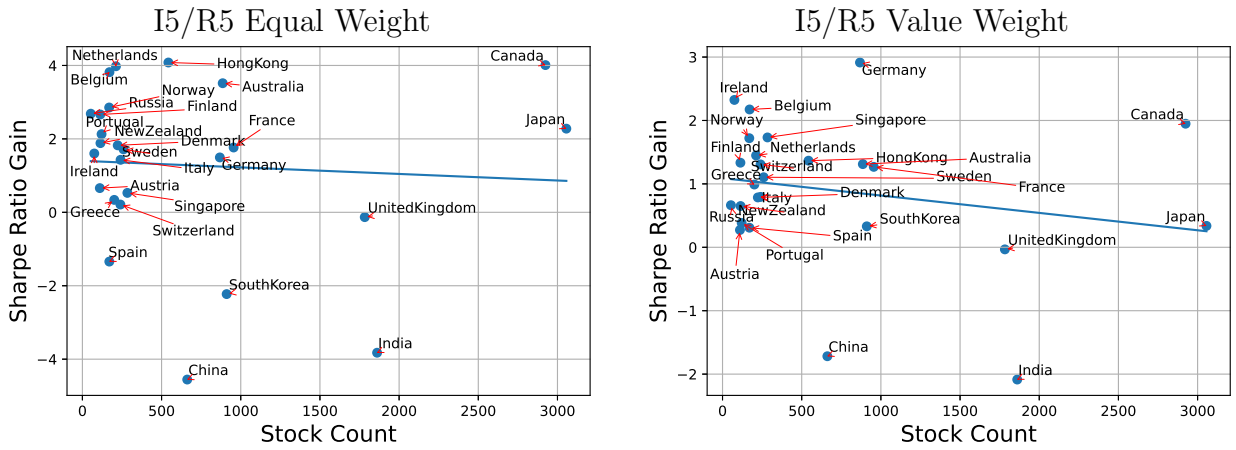
Note: The table reports annualized out-of-sample Sharpe ratios for H-L decile spread portfolios within each country. We report the average monthly stock count by country, the image-based strategy from re-training the I5/R5 CNN using local data, and the image-based strategy directly transfers the I5/R5 model estimated in US data without re-training. Sharpe ratio gains (Transfer−Re-train) accompanied by ***,**,* are significant at the 1%, 5% and 10% significance level, respectively.

to 2019, with the exception of Russia, Greece, Finland, Ireland, and Sweden which start in 1999. While some markets such as Japan and Canada have samples of around 3,000 stocks each month on average, the median country has roughly 300 stocks.

Our transfer learning implementation directly applies estimated CNN models from the US sample to price images in foreign markets, with no re-training. We then conduct portfolio sorts country-by-country using transfer-based return forecasts and calculate out-of-sample Sharpe ratios for decile spread H-L portfolios. We assess the benefits of transfer learning by comparing with otherwise identical CNN models estimated using local image data for each foreign market.

Table 10 reports I5/R5 portfolio performance for each country in our international sample, ordered by the monthly average stock count in each country. The left side of the table shows

Figure 9: Sharpe Ratio Gains From International Transfer



Note: The figures show annualized out-of-sample Sharpe ratio gains from I5/R5 transfer learning (both equal-weight and value-weight strategies) versus the average number of stocks in each country.

results for H-L strategies based on equally weighted decile portfolios. We report performance for the CNN forecasts trained from scratch on country-specific data, for forecasts based on direct application of the CNN model estimated in US data, and for the difference. The last row shows the simple average of each column.

In 20 out of 26 countries, transfer learning produces a Sharpe ratio gain over the locally re-trained model (for equal weight portfolios), and the gain is statistically significant in 19 of these. In the 19 significant cases, transfer from the US model produces a Sharpe ratio gain of more than 0.5. On average, transfer from the US model produces a Sharpe ratio of 3.5, more than a 50% increase over the average Sharpe ratio of 2.3 for locally re-trained models.

The comparative gains from transfer versus local training are similar in value weight strategies, shown in the right three columns of the table. In this case the gain in Sharpe ratio from transfer learning is significantly positive for 22 out of 26 countries. On average, transfer from US models nearly doubles the Sharpe ratio of value weight strategies relative to locally re-trained models, from 1.0 locally to 1.9 with transfer. The evidence is broadly similar, though smaller in magnitude, for lower frequency (monthly) strategies as shown in Internet Appendix Table IA6 for I20/R20 CNN models. In Tables IA7 and IA8, we also decompose the monthly strategy performance into returns on days 1–5 versus days 6–20. There we find that international return prediction is insignificant beyond the first five days.

Figure 9 summarizes the benefits of international transfer learning by plotting the Sharpe ratio gains reported in Table 10 against the number of stocks in each country. Most international markets are small and thus clustered on the left side of the plots. In these countries,

portfolio Sharpe ratios tend to especially benefit from US transfer relative to locally trained CNN models. But for larger markets on the right side of the plots, the expected gains (based on the best fit line) are small or slightly negative. This suggests there are benefits to local re-training when there is sufficient data, likely due to some degree of heterogeneity in predictive patterns across countries that transfer learning does not account for. Internet Appendix Figure IA1 shows that the same pattern holds for I20/R20 transfer. A direction for further optimization of image-based prediction models would combine a global image model to capture shared differences with a country-specific model that accommodates some degree of heterogeneity. The model weights in this combination could be dictated by the relative informativeness of global and country-specific data in a Bayesian fashion.

6.2 Time Scale Transfer

The most constraining aspect of empirical asset pricing data is in the time series dimension. The finance literature focuses most often on monthly or annual data because many economically important patterns in asset markets unfold over such frequencies. Yet we experience only one history of financial market prices, meaning that we have at most several hundred monthly time series observations and several dozen annual observations. Given the scarcity of low frequency data, it would be greatly beneficial to know if patterns that unfold at high frequencies (which we can potentially measure well thanks to the availability of large high frequency data sets) are similar to patterns that unfold at low frequencies.

Transfer learning provides a means of investigating the possibility that price patterns apply at multiple time scales. While it is difficult to effectively train a CNN using monthly or annual observations, we can apply our CNNs trained on daily data to data sampled at lower frequencies. We first consider transferring the I5/R5 CNN to the problem of using 20-day price histories to forecast future 20-day returns. To do so, we draw the 20-day price history using a 5-period OHLC chart by re-defining a "period" to be a 4-day interval. Each tick mark in an image now corresponds to four days of market data collapsed into a single observation, and a given OHLC bar show open, high, low, close over the 4-day interval. By down-sampling market data from once per day to once every four days, we can apply I5/R5 estimates to a 20-day image.

Panel A of Table 11 compares approaches to portfolio construction using 20-day price histories to make 20-day future return forecasts. The columns labeled "baseline" correspond to CNN-based forecasts using daily data; i.e., it exactly repeats the strategy of the I20/R20 model in Table 2. The columns labeled "Re-train, No Transfer" re-estimate a CNN from scratch using 20 days of data collapsed into 5-period images and supervised by future 20-day returns. This provides a benchmark for how predictability is affected by simply down-sampling

Table 11: Time Scale Transfer I5/R5 to I20/R20

	Panel A: Equal Weight Portfolios							
	Baseline		Re-train		Transfer		Baseline+Transfer	
	Ret	SR	Ret	SR	Ret	SR	Ret	SR
Low	-0.02	-0.12	-0.02	-0.10	-0.01	-0.07	-0.04	-0.19
2	0.05	0.28	0.05	0.26	0.04	0.25	0.04	0.20
3	0.07	0.39	0.07	0.37	0.07	0.40	0.06	0.35
4	0.09	0.49	0.09	0.49	0.08	0.43	0.08	0.44
5	0.11	0.59	0.10	0.56	0.09	0.50	0.09	0.52
6	0.11	0.59	0.11	0.61	0.10	0.53	0.11	0.64
7	0.13	0.74	0.12	0.65	0.11	0.59	0.13	0.70
8	0.14	0.81	0.14	0.79	0.14	0.70	0.15	0.80
9	0.15	0.87	0.15	0.82	0.17	0.84	0.17	0.89
High	0.18	1.04	0.20	1.15	0.24	1.14	0.23	1.17
H-L	0.21***	2.16	0.22***	2.21	0.25***	2.14	0.26***	2.46
Turnover	173%		177%		176%		175%	
	Panel B: Value Weight Portfolios							
	Baseline		Re-train		Transfer		Baseline+Transfer	
	Ret	SR	Ret	SR	Ret	SR	Ret	SR
Low	0.05	0.28	0.06	0.38	0.03	0.22	0.04	0.24
2	0.04	0.21	0.06	0.36	0.05	0.34	0.05	0.32
3	0.05	0.32	0.05	0.34	0.06	0.42	0.04	0.27
4	0.05	0.33	0.08	0.53	0.05	0.35	0.05	0.30
5	0.06	0.40	0.07	0.42	0.07	0.46	0.06	0.40
6	0.08	0.50	0.08	0.50	0.08	0.50	0.07	0.45
7	0.08	0.51	0.08	0.54	0.09	0.56	0.07	0.45
8	0.09	0.58	0.09	0.58	0.11	0.66	0.10	0.61
9	0.08	0.55	0.10	0.64	0.12	0.65	0.10	0.61
High	0.11	0.67	0.10	0.66	0.12	0.69	0.12	0.73
H-L	0.05**	0.49	0.04	0.33	0.09***	0.73	0.08***	0.67
Turnover	181%		187%		186%		184%	

Note: The table shows the performance of strategies using time scale transfer learning. In particular, we directly apply the I5/R5 CNN model estimated from daily data to construct forecasts from images that include 20 days of data sampled once every four days. Transfer learning portfolio performance (with a 20-day holding period) is reported under "Transfer." For comparison, we report a "Baseline" strategy that uses the I20/R20 CNN model of Tables 2 and IA3, and a CNN model that is trained from scratch using images of 20 days of data sampled once every four days (under "Re-train"). Finally, "Baseline+Transfer" shows the performance of an equal-weighted average of the baseline and transfer strategies.

the price history and re-training the CNN. The columns labeled "Transfer" use estimates from the I5/R5 CNN to construct 20-day return forecasts from the collapsed 20-day images, thus applying transfer learning at a 1:4 time scale.

Directly transferring the I5/R5 model to collapsed 20-day images with no re-estimation produces a remarkable 2.1 Sharpe ratio in equal weight portfolios. In this example, the transfer-based strategy performs nearly as well as the CNN re-trained with lower frequency data (whose Sharpe ratio is 2.2). Interestingly, the transfer learning signal is only 42% correlated with the baseline signal, indicating that they pick up in part on different predictive patterns. In the columns labeled "50/50 Baseline+Transfer" we show that differences in the information content from baseline and transfer signals can be leveraged to form an even

Table 12: Time Scale Transfer I5/R5 to I60/R60

Panel A: Equal Weight Portfolios

	Baseline		Re-train		Transfer		Baseline+Transfer	
	Ret	SR	Ret	SR	Ret	SR	Ret	SR
Low	0.08	0.27	0.08	0.30	0.06	0.31	0.06	0.22
2	0.09	0.37	0.08	0.34	0.08	0.41	0.09	0.35
3	0.11	0.46	0.10	0.43	0.10	0.50	0.11	0.46
4	0.12	0.52	0.12	0.53	0.11	0.54	0.11	0.51
5	0.12	0.55	0.12	0.55	0.11	0.54	0.12	0.55
6	0.12	0.57	0.12	0.57	0.13	0.61	0.12	0.61
7	0.14	0.68	0.13	0.64	0.12	0.55	0.14	0.67
8	0.14	0.72	0.13	0.64	0.14	0.60	0.14	0.69
9	0.14	0.75	0.13	0.72	0.14	0.59	0.15	0.78
High	0.15	0.88	0.14	0.88	0.17	0.69	0.16	0.93
H-L	0.07*	0.43	0.06	0.37	0.10***	0.90	0.10***	0.81
Turnover	58%		59%		59%		58%	

Panel B: Value Weight Portfolios

	Baseline		Re-train		Transfer		Baseline+Transfer	
	Ret	SR	Ret	SR	Ret	SR	Ret	SR
Low	0.07	0.32	0.11	0.50	0.06	0.38	0.08	0.37
2	0.10	0.50	0.08	0.41	0.07	0.34	0.08	0.40
3	0.08	0.46	0.07	0.39	0.08	0.47	0.08	0.41
4	0.09	0.47	0.06	0.35	0.09	0.58	0.08	0.46
5	0.08	0.43	0.09	0.50	0.10	0.61	0.09	0.49
6	0.09	0.49	0.08	0.48	0.09	0.55	0.08	0.45
7	0.09	0.54	0.10	0.55	0.09	0.53	0.09	0.49
8	0.09	0.53	0.10	0.58	0.10	0.56	0.10	0.57
9	0.09	0.52	0.09	0.55	0.11	0.65	0.09	0.58
High	0.11	0.75	0.11	0.77	0.10	0.51	0.11	0.81
H-L	0.03	0.23	0.00	0.01	0.04	0.27	0.03	0.31
Turnover	59%		61%		62%		59%	

Note: The table shows the performance of strategies using time scale transfer learning. In particular, we directly apply the I5/R5 CNN model estimated from daily data to construct forecasts from images that include 60 days of data sampled once every 12 days. Transfer learning portfolio performance (with a 60-day holding period) is reported under "Transfer." For comparison, we report a "Baseline" strategy that uses the I60/R60 CNN model of Tables 2 and IA3, and a CNN model that is trained from scratch using images of 60 days of data sampled once every 12 days (under "Re-train"). Finally, "Baseline+Transfer" shows the performance of an equal-weighted average of the baseline and transfer strategies.

stronger trading strategy. In particular, an equal weight combination of the two individual strategies returns a Sharpe ratio of 2.5. Panel B shows that when forming value weight decile portfolios, the transfer learning signal is in fact more powerful than the re-trained CNN. At 0.7, the H-L transfer strategy more than doubles the Sharpe ratios from re-training (0.3) and is around 50% larger than the baseline Sharpe ratio from daily data (0.5).[17]

Next, we analyze 60-day market data collapsed into 5-period images—i.e., sampling prices once every 12 days and forecasting returns over the next 60 days. We repeat the analysis of comparing the baseline I60/R60 strategy to either re-training the CNN on collapsed images, or

[17]In Internet Appendix IA.3.2 we conduct a simulation strategy that illustrates how time scale transfer, from high frequency estimates to low frequency forecasts, can produce more powerful low frequency forecasts than a model directly trained on low frequency data.

directly transferring the estimated I5/R5 model to the I60/R60 problem at the 1:12 time scale. The results are reported in Table 12 and show that the comparative success of transfer learning becomes even stronger at this lower frequency. In Panel A, we find that transfer achieves an H-L Sharpe ratio of 0.9 (for equally weighted portfolios), compared with the baseline I60/R60 Sharpe ratio of 0.4 and a re-training Sharpe ratio of 0.4. Value weight portfolios, reported in Panel B, also demonstrate success of the transfer approach with a Sharpe ratio of 0.3, versus a Sharpe ratio of 0.0 for re-training.

This evidence of time scale transfer is inconsistent with the linear autoregressive model ubiquitous in models of price dynamics and conditional expected returns. Instead, these patterns are reminiscent of the Mandelbrot (2013) hypothesis that prices are fractal processes that demonstrate self-similarity when studied at different time scales. Honing in on a model of price dynamics that is consistent with the evidence above is an interesting problem for future research (and unfortunately beyond the scope of this paper).

7 Conclusion

A fascinating research agenda is to develop models capable of translating visual data into an optimal portfolio. In this paper, we take a modest step in this direction by extracting trading signals from price chart images. We analyze these images with a return prediction CNN and find that our image-based forecasts in general outperform (and are in large part distinct from) traditional price trend signals in the asset pricing literature. We show that the predictive patterns isolated by the CNN are highly robust to variations in model specification and controlling for a wide range of alternative predictor variables. One of the most compelling aspects of CNN robustness is its transferability to international markets and other time scales. Models trained on daily data are similarly powerful when transferred to data sets sampled at lower frequencies, and models trained on US data but applied to international stock markets outperform models trained on data from local markets.

In a sprawling survey of 692 asset managers in five countries, Menkhoff (2010) finds that 87% of those surveyed rely on some form of technical analysis in their decision process, and 18% of respondents indicate it is a major part of their investment process. As the Lo et al. (2000) at the start of this article notes, technical analysis is a primarily visual mode of analysis. In light of this, our CNN model has potentially intriguing implications for economic modeling. If trading decisions implied by a statistical representation benefit from being more closely aligned with the formats in which investors intake their market perceptions for eventual trading decisions, the CNN becomes more than a pure statistical tool and moves closer to a model of investor perception. Our ideal research agenda is to develop a model that can translate visual data into an optimal portfolio in a way that mimics the human perceptions and decision

processes. In this paper, we take a first modest step in this direction by extracting trading signals from price images using a CNN model.

Market data images, when combined with a CNN, constitute a new and powerful tool for understanding market dynamics and forming efficient portfolios. Yet our analysis of images connects existing time series analysis with ad hoc chart studies primarily for the purpose of technical trading. In light of this, our findings highlight image analysis as a future research direction with great potential to improve our understanding of financial market phenomena.

References

Bajgrowicz, Pierre, and Olivier Scaillet, 2012, Technical trading revisited: False discoveries, persistence tests, and transaction costs, *Journal of Financial Economics* 106, 473–491.

Barberis, Nicholas, 2018, Psychology-based models of asset prices and trading volume, in *Handbook of behavioral economics: applications and foundations 1*, volume 1, 79–175 (Elsevier).

Barberis, Nicholas, Andrei Shleifer, and Robert Vishny, 1998, A model of investor sentiment, *Journal of financial economics* 49, 307–343.

Blume, Lawrence, David Easley, and Maureen O'hara, 1994, Market statistics and technical analysis: The role of volume, *The Journal of Finance* 49, 153–181.

Brock, William, Josef Lakonishok, and Blake LeBaron, 1992, Simple technical trading rules and the stochastic properties of stock returns, *The Journal of finance* 47, 1731–1764.

Brown, David P, and Robert H Jennings, 1989, On technical analysis, *The Review of Financial Studies* 2, 527–551.

Chen, Jou-Fan, Wei-Lun Chen, Chun-Ping Huang, Szu-Hao Huang, and An-Pin Chen, 2016, Financial time-series data analysis using deep convolutional neural networks, in *2016 7th International conference on cloud computing and big data (CCBD)*, 87–92, IEEE.

Cohen, Naftali, Tucker Balch, and Manuela Veloso, 2020, Trading via image classification, *Proceedings of the First ACM International Conference on AI in Finance* 1–6.

Cont, Rama, 2005, Long range dependence in financial markets, in *Fractals in engineering*, 159–179 (Springer).

Dalal, Navneet, and Bill Triggs, 2005, Histograms of oriented gradients for human detection, in *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, 886–893, IEEE.

Detzel, Andrew, Hong Liu, Jack Strauss, Guofu Zhou, and Yingzi Zhu, 2020, Learning and predictability via technical analysis: Evidence from bitcoin and stocks with hard-to-value fundamentals, *Financial Management* .

Dobrev, Dobrislav, 2007, Capturing volatility from large price moves: generalized range theory and applications, *Manuscript, Department of Finance, Kellogg School, Northwestern University* .

Fama, Eugene F, and Kenneth R French, 1988, Dividend yields and expected stock returns, *Journal of financial economics* 22, 3–25.

Fama, Eugene F, and Kenneth R French, 1993, Common risk factors in the returns on stocks

and bonds, *Journal of Financial Economics* 33, 3–56.

Frazzini, Andrea, Ronen Israel, and Tobias J Moskowitz, 2018, Trading costs, *Available at SSRN 3229719* .

Freund, Yoav, and Robert E Schapire, 1995, A desicion-theoretic generalization of on-line learning and an application to boosting, in *European conference on computational learning theory*, 23–37, Springer.

Glorot, Xavier, and Yoshua Bengio, 2010, Understanding the difficulty of training deep feedforward neural networks, in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 249–256.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville, 2016, *Deep learning* (MIT press).

Grundy, Bruce D, and Maureen McNichols, 1989, Trade and the revelation of information through prices and direct disclosure, *The Review of Financial Studies* 2, 495–526.

Gu, Shihao, Bryan Kelly, and Dacheng Xiu, 2020, Empirical asset pricing via machine learning, *Review of Financial Studies* 33, 2223–2273.

Han, Yufeng, Guofu Zhou, and Yingzi Zhu, 2016, A trend factor: Any economic gains from using information over investment horizons?, *Journal of Financial Economics* 122, 352–375.

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

Hu, Guosheng, Yuxin Hu, Kai Yang, Zehao Yu, Flood Sung, Zhihong Zhang, Fei Xie, Jianguo Liu, Neil Robertson, Timpathy Hospedales, et al., 2018, Deep stock representation learning: From candlestick charts to investment decisions, in *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, 2706–2710, IEEE.

Ioffe, Sergey, and Christian Szegedy, 2015, Batch normalization: Accelerating deep network training by reducing internal covariate shift, *arXiv preprint arXiv:1502.03167* .

Jegadeesh, Narasimhan, and Sheridan Titman, 1993, Returns to buying winners and selling losers: Implications for stock market efficiency, *The Journal of finance* 48, 65–91.

Jensen, Theis Ingerslev, Bryan T Kelly, and Lasse Heje Pedersen, 2022, Is there a replication crisis in finance?, *Journal of Finance* .

Ke, Zheng Tracy, Bryan T Kelly, and Dacheng Xiu, 2021, Predicting returns with text data, Technical report, National Bureau of Economic Research.

Kelly, Bryan, and Seth Pruitt, 2013, Market expectations in the cross-section of present values, *The Journal of Finance* 68, 1721–1756.

Kim, Taewook, and Ha Young Kim, 2019, Forecasting stock prices with a feature fusion

lstm-cnn model using different representations of the same data, *PloS one* 14, e0212320.

Kinga, D, and J Ba Adam, 2015, A method for stochastic optimization, in *International Conference on Learning Representations (ICLR)*, volume 5.

Lee, Jinho, Raehyun Kim, Yookyung Koh, and Jaewoo Kang, 2019, Global stock market prediction based on stock chart images using deep q-network, *IEEE Access* 7, 167260–167277.

Liu, Yang, Guofu Zhou, and Yingzi Zhu, 2020, Maximizing the sharpe ratio: A genetic programming approach, Technical report, Tsinghua University and Washington University in St. Louis.

Lo, Andrew W, and Jasmina Hasanhodzic, 2010, *The heretics of finance: Conversations with leading practitioners of technical analysis*, volume 16 (John Wiley and Sons).

Lo, Andrew W, Harry Mamaysky, and Jiang Wang, 2000, Foundations of technical analysis: Computational algorithms, statistical inference, and empirical implementation, *The journal of finance* 55, 1705–1765.

Maas, Andrew L, Awni Y Hannun, and Andrew Y Ng, 2013, Rectifier nonlinearities improve neural network acoustic models, in *Proc. icml*, volume 30, 3.

Mandelbrot, Benoit B, 2013, *Fractals and scaling in finance: Discontinuity, concentration, risk. Selecta volume E* (Springer Science & Business Media).

Menkhoff, Lukas, 2010, The use of technical analysis by fund managers: International evidence, *Journal of Banking & Finance* 34, 2573–2586.

Murphy, John J., 1999, *Technical Analysis of the Financial Markets: A Comprehensive Guide to Trading Methods and Applications* (New York Institute of Finance).

Murray, Scott, Houping Xiao, and Yusen Xia, 2021, Charting by machines, Technical report, Georgia State University.

Neely, Christopher J., David E. Rapack, Jun Tu, and Guofu Zhou, 2014, Forecasting the equity risk premia: The role of technical indicators, *Management Science* 60, 1772–1791.

Pan, Sinno Jialin, and Qiang Yang, 2009, A survey on transfer learning, *IEEE Transactions on knowledge and data engineering* 22, 1345–1359.

Papageorgiou, Constantine P, Michael Oren, and Tomaso Poggio, 1998, A general framework for object detection, in *Sixth International Conference on Computer Vision (IEEE Cat. No. 98CH36271)*, 555–562, IEEE.

Parkinson, Michael, 1980, The extreme value method for estimating the variance of the rate of return, *The Journal of Business* 53, 61–65.

Schwert, G. William, 2003, Chapter 15 anomalies and market efficiency, in *Financial Markets and Asset Pricing*, volume 1 of *Handbook of the Economics of Finance*, 939 – 974 (Elsevier).

Simonyan, Karen, and Andrew Zisserman, 2015, Very deep convolutional networks for large-scale image recognition, in *ICLR*.

Srivastava, Nitish, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, 2014, Dropout: A Simple Way to Prevent Neural Networks from Overfitting, *The Journal of Machine Learning Research* 15, 1929–1958.

Sullivan, Ryan, Allan Timmermann, and Halbert White, 1999, Data-snooping, technical trading rule performance, and the bootstrap, *The journal of Finance* 54, 1647–1691.

Viola, Paul, and Michael Jones, 2001, Rapid object detection using a boosted cascade of simple features, in *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, volume 1, I–I, IEEE.

Zeiler, Matthew D, and Rob Fergus, 2014, Visualizing and understanding convolutional networks, in *European conference on computer vision*, 818–833, Springer.

Zhu, Yingzi, and Guofu Zhou, 2009, Technical analysis: An asset allocation perspective on the use of moving averages, *Journal of Financial Economics* 92, 519–544.

Appendix: Architecture Details of the CNN

A CNN is a modeling scheme that stacks together a sequence of operations to transform a raw image into a set of predictive features and, ultimately, a prediction. They are inherently modular, using a single core building block that can be assembled in various configurations depending on the application at hand. A core building block consists of three operations: convolution, activation, and pooling.

The first operation in a CNN building block is "convolution." To understand convolution, consider the simpler operation of applying a kernel smoother to a time series. For example, a rectangular kernel with a width of three smooths the time series by scanning through observations, averaging each data point with its two neighbors. Convolution works similarly by scanning through the image and, for each element in the image matrix, producing a summary of image contents in the immediately surrounding area.

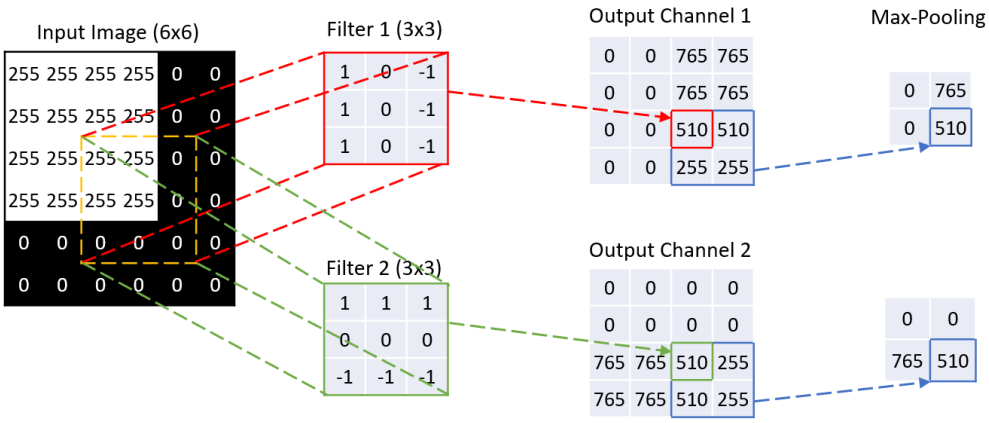
More specifically, the convolution operation uses a set of "filters." A filter is a low dimension kernel weighting matrix that, for each matrix element in the image, calculates the weighted average of the immediately adjacent matrix elements. Figure A1a is an illustrative example with a 6×6 image that is white in the 4×4 upper-left corner and black elsewhere. Filter 1 is an example of a 3×3 filter taking the values of $(1, 0, -1)$ in each row. For each interior element (i, j) of the image, the convolution operation sums the element-wise product of Filter 1 and the 3×3 image contents centered on (i, j). The result is stored in the corresponding element of the convolution output matrix.[18]

The example highlights the critical difference between convolution and 2D kernel smoothing. Smoothing calculates local averages in the image matrix, while convolution instead calculates a weighted sum of nearby image contents where the filter weights are parameters to be estimated. Through estimation, the CNN constructs filters that emphasize aspects of images that are most predictive of the outcome of interest and blurs out uninformative content. The weight configuration in Filter 1, for example, is particularly useful for detecting vertical boundaries, while Filter 2 finds horizontal boundaries. Sliding the filter over the entire image amounts to searching the image for this specific pattern. If such a pattern exists in part of the input image, the corresponding output value will be large—meaning that the output neuron is stimulated and signaling that a pattern is detected. As in our model below, it is common to use several filters in a CNN, each specializing in certain patterns and thus each outputting a different set of features for use in the next layer of the model. And it is common for a CNN

[18]For elements at the image's border, we fill in the absent neighbor elements with zeros in order to compute the convolution. This is a common CNN practice known as "padding," and ensures that the convolution output has the same dimension as the image itself (see Simonyan and Zisserman, 2015).

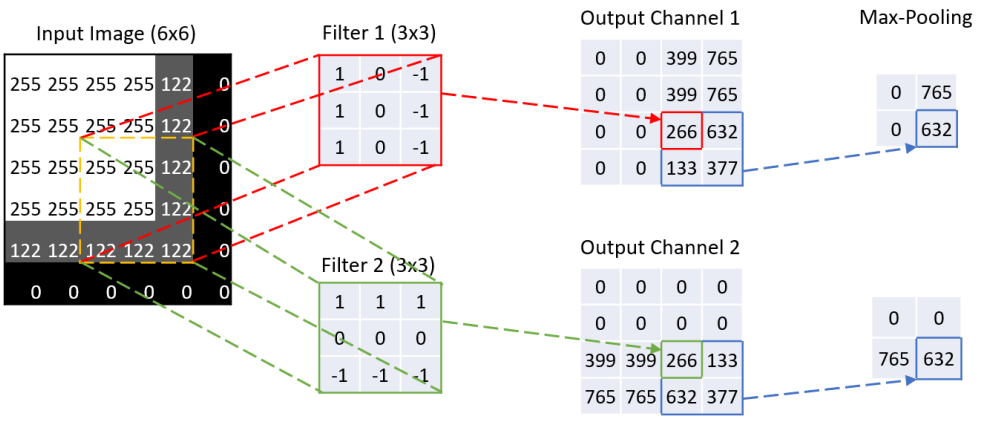
Figure A1: Illustration of Convolution and Max-pooling Operations

(a) Illustration of a Convolutional Operation



Note: The input tensor has size 6×6 (with only one channel of grey-scale). There are two 3×3 filters that detects edges. By construction, Filter 1 detects vertical edges and Filter 2 detects horizontal edge. When Filter 1 is applied to the yellow part of the input, the convolutional operation first calculates the element-wise products between the 3×3 yellow tensor and Filter 1 of the same size (solid red line) and then sums up the products and output a scalar (dotted red arrow) to the corresponding output location. As we can see from Output Channel 1, the right part of the output is activated by large values while the left part remains inactivated with all zeros, indicating that there is a vertical line on the right of the input image. Finally, the 2×2 max-pooling is applied to the output and shrinks the shape by half from 4×4 to 2×2 by taking the max out of the 2×2 rolling window from the output. The final output in channel 1 has value 765 at the upper-right corner and value 510 at the bottom-right corner, represented the extracted information that there are vertical edges on the right and the vertical edge is sharper at the top than at the bottom. Filter 2 is applied in the same sense (green dotted line and green dotted arrow) to extract the information that there are horizontal edges at the bottom of the image and the bottom-left (765) has sharper change than the bottom-right (510).

(b) Robustness to Noise from Max-Pooling



Note: This example shows that the introduction of the max-pooling layer adds robustness to noise. We take the same 6×6 image from the above example and blur the borderline with grey color (pixel value 122). Take Filter 1, which detects vertical edges, for example. Non-zero values from Output Channel 1 are significantly changed. However, after max-pooling is applied, the original upper-right value 765 remains intact while the original bottom-right value 510 is replaced with 632, a slightly larger number, indicating a slightly stronger signal of vertical line at the bottom than before. We observe a similar effect for Filter 2, which detects horizontal edges.

to use telescoping layers of filters, with the output from the filters in one layer of the CNN used as inputs for the subsequent layer to capture more complex patterns.

We use a filter size of 5×3 pixels in our baseline model. In addition, a convolution is flexible in "stride" of the filter, which defines the number of horizontal or vertical steps the filter takes as it moves through the image matrix. A larger stride corresponds to coarser convolution output. We use horizontal stride of one and vertical stride of three, meaning that the filter slides across the image and recalculates the filter output at every position in the row and jumps three rows at a time as it moves down the image vertically. We sometimes use a "dilated" filter to cover a larger neighborhood on the image than the size of the filter. A dilated filter with a dilation rate k along the vertical or the horizontal dimension (or both), expands the size of the original filter by filling in $(k-1)$ zeros in between adjacent entries along the corresponding dimension(s). Our baseline model uses a vertical dilation of two and no horizontal dilation.[19]

Convolution endows the CNN with a number of attractive properties relative to more general neural networks connecting an $N \times M$ input matrix to an $N \times M$ output. The CNN tightly constrains model parameterization by imposing two forms of (severe) cross-parameter restrictions. First, CNN applies a small filter uniformly to all locations in an image, while a general network allows separate weight parameters for each element of the image matrix. Second, the CNN maps the information from a given location in the input matrix *only* to the neighborhood of the same location in the output matrix, while a general network would allow for cross-connectivity between all elements of the input and output matrices. These restrictions, known respectively as "parameter sharing" and "sparse interactions," are critical to the tractability and small-sample robustness of the CNN. And, because the same filters are applied uniformly to all locations in the image, they detect relevant objects regardless of their position (in other words, CNN forecasts are "translation equivariant").

The second operation in a building block is "activation." This simple operation is a non-linear transformation applied element-wise to the output of a convolution filter. The activation function we use is "leaky ReLU." This, roughly speaking, takes the max of the filter output value and zero, which sharpens the resolution of the convolution filter output.[20]

[19] Unit stride is a common choice in the CNN literature, but larger strides are sometimes used to reduce the dimensionality of the model. A stride of two, for example, recalculates the convolution at every other element of the input matrix, giving coarser convolution output with half the dimensionality. While both stride and dilation reduce the computational burden and encourage parsimony, dilation preserves the resolution of the original image.

[20] This activation function, introduced by Maas et al. (2013), is defined as $\text{LeakyReLU}(x) = \begin{cases} x, \text{if } x \geq 0 \\ kx, \text{if } x < 0 \end{cases}$, where $k = 0.01$ is the coefficient that controls the angle of the negative slope. Leaky ReLU is an improved variant of the standard ReLU function that simply zeros out the unresponsive (negative) outputs while main-

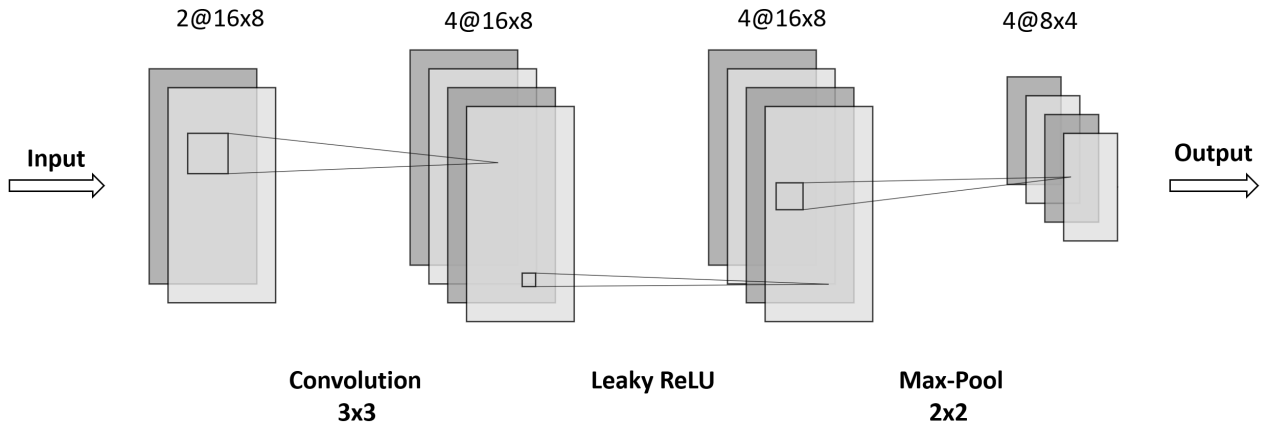
The final operation in a building block is "max-pooling." We implement this operation using a small filter that scans over the input matrix and returns the maximum value the elements entering the filter at each location in the image. The role of max-pooling is two-fold. First, it acts as a dimension reduction device. Nearby neurons output from the convolution operation often carry similar information. If any of the neurons in the filter region are stimulated, max-pooling detects it. At the same time, it discards locally redundant information. For example, a 2×2 pooling filter shrinks the height and width of the input by half, and does so without introducing new parameters to be estimated further promoting parsimony in the CNN. In this respect, max-pooling is an image counterpart to the familiar idea of down-sampling from the signal processing literature. Second, by taking local maxima throughout the image, the output is left generally unaffected by small perturbations of the input pattern (illustrated in Figure A1b). In this sense, max-pooling is a de-noising tool that enhances CNN robustness to local deformation, much like the convolution operation aids CNN robustness to variation in object position.

Figure A2 illustrates how convolution, activation, and max-pooling combine to form the basic building block for a CNN model. These blocks are then stacked together to customize predictive CNN models with varying degrees of richness. In general, the data input to a building block is a tensor with dimensions $h \times w \times d$. The lingua franca of CNN refers to image depth d as the number of "channels." In the very first block of our network, the input is a black and white image, which is a matrix. Our black and white images thus have one channel. Each filter output has dimension $h \times w \times 1$. If a building block uses multiple filters, their outputs are stacked together in the third dimension of the tensor so outputs have multiple channels, one channel corresponding to each filter. Therefore building blocks in the interior layers of our CNN take 3-dimensional tensors as inputs. If the input layer has multiple channels, each filter of size (3×3) gains depth equal to the number of input channels, d (e.g., the filter is $3 \times 3 \times d$). Filters therefore aggregate input information locally in the height and width dimensions, but then combine this local information across all channels. In Figure A2, the input has height of 16 and width of 8 and has two channels. Thus each filter is $3 \times 3 \times 2$. The example uses four filters, so the size of the convolution output is $16 \times 8 \times 4$.

Within a building block, the output from all convolutional filters are fed element-wise through the leaky ReLU activation function, which preserves the dimensions of the convolution output. In the example, the ReLU activation output is therefore $16 \times 8 \times 4$. Finally, a max-

taining the stimulated ones. A drawback of ReLU is that when inputs are all positive, the gradients are either all positive or all negative, which introduces obstacles to the gradient descent in the training step. Another drawback of ReLU is that certain neurons may never activate because all gradients flowing through these units fall into the zero region of the ReLU function. Leaky ReLU resolves both issues.

Figure A2: Diagram of a Building Block



Note: A building block of the CNN model consists of a convolutional layer with 3×3 filter, a leaky ReLU layer, and a 2×2 max-pooling layer. The notation D@HxW shows the output size of the CNN building block where H is the height, W is the width, and D is the number of channels. In this toy example, the input has size 16×8 with 2 channels. To double the depth of the input, 4 filters are applied, which generates the output with 4 channels. The max-pooling layer shrinks the first two dimensions (height and width) of the input by half and keeps the same depth. Leaky ReLU keeps the same size of the previous input. In general, with input of size $h \times w \times d$, the output has size $h/2 \times w/2 \times 2d$. One exception is the first building block of each CNN model that takes the grey-scale image as input: the input has depth 1 and the number of CNN filters is 32, boosting the depth of the output to 32.

pooling filter of size 2×2 with stride of two is applied to each channel separately, reducing their height and width by one-half each. The final output of the building block is $8 \times 4 \times 4$.

This output serves as the input to the next building block. By stacking many of these blocks together, the network first creates representations of small components of the image then gradually assembles them into representations of larger areas. The output from the last building block is flattened into a vector and each element is treated as a feature in a standard, fully connected feed-forward layer for the final prediction step. The final prediction is a linear combination of the vectorized image features, which is fed through a softmax (i.e., logistic) function to generate a probability of whether the future price will rise.

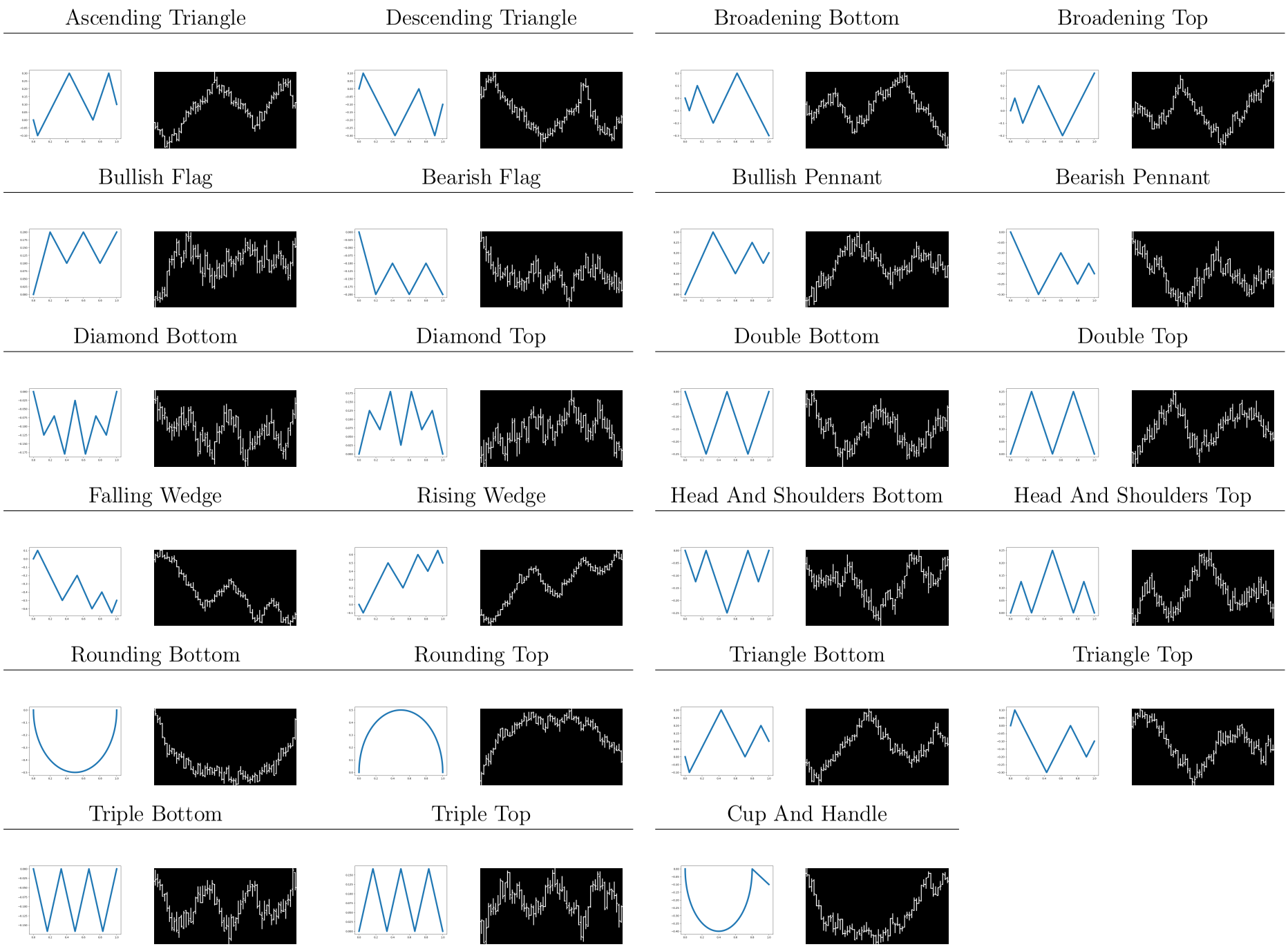
Having introduced the generic architectural components of CNN, we now discuss specific choices for our models. Since our images are largely sparse in the vertical dimension, we use 5×3 convolutional filters and 2×1 max-pooling filters. We use the same filter sizes in all layers for convenience. We use vertical strides of 1, 3, and 3, and vertical dilation rates of 1, 2, and 3 for 5-day, 20-day, and 60-day images, respectively, only on the first layer, where inputs are sparse raw images. The number of CNN building blocks in our model is based on the size of the input image. We use 2 blocks to model 5-day images, 3 blocks for 20-day images,

and 4 blocks for 60-day images. The number of filters for the first building block is 64 for all three models.[21] As pointed out by Zeiler and Fergus (2014), learned features become more complex in deeper layers, so we follow the literature and increase the number of filters after each convolutional layer by a factor of two (e.g., 64, 128, 256, and 512 filters, respectively, in the four layers of the 60-day model). In turn, the fully connected layers have $15,360$, $46,080$, and $184,320$ neurons for 5-day, 20-day, and 60-day models, respectively, which are determined by the outputs of the convolutional blocks. The total number of parameters are $155,138$ for the 5-day model, $708,866$ for 20-day, and $2,952,962$ for 60-day models, respectively.[22] Of course, the effective parameterization of these models is much smaller than this parameter count due to heavy regularization that shrinks most parameters close to zero.

[21] We follow the popular VGG model (Simonyan and Zisserman, 2015) that uses 64 filters in its first layer.

[22] The total number of parameters is calculated by summing up the number of parameters in convolutional layers and the number of parameters in the fully connected layer. The number of parameters in a convolutional layer is calculated as $(K^2 \times D + 1) \times F$, where K is the size of filters (fixed as 3 in our models), D is the depth (number of channels) of the input, and F is the number of filters. The number of parameters in a fully connected layer is calculated as $L \times 2$, where L is the length of the input vector and 2 corresponds to the two classification labels. We ignore the number of parameters from the batch normalization as it is negligible.

Figure A3: Chart Patterns



Note: This figure plots 23 diagrams of well-known chart patterns and a simulated sample path for each pattern.

Internet Appendix

IA.1 Robustness Analyses

Table IA1: Exposure to Technical Analysis Factors

	Equal Weight			Value Weight		
	5-day	20-day	60-day	5-day	20-day	60-day
Mean Ret	1.81%	1.75%	1.32%	0.51%	0.53%	0.17%
t-stat	10.3	9.5	5.5	2.3	2.7	0.7
Alpha	1.85%	1.77%	1.3%	0.81%	0.68%	0.29%
t-stat	10.2	9.9	5.3	3.5	3.3	1.1
Mkt-Rf	-0.27	-0.33	-0.27	-0.23	-0.25	-0.14
t-stat	-6.0	-7.5	-4.5	-4.0	-5.0	-2.2
Momentum	-0.05	-0.01	0.16	-0.11	-0.00	0.12
t-stat	-1.4	-0.2	3.1	-2.1	-0.0	2.1
STR	-0.11	-0.06	0.06	-0.10	-0.07	-0.01
t-stat	-1.9	-1.0	0.8	-1.3	-1.1	-0.1
WSTR	0.95	0.49	-0.04	-0.74	-0.11	-0.57
t-stat	2.5	1.3	-0.1	-1.5	-0.2	-1.0
TREND	-0.16	0.39	0.52	0.18	0.24	0.24
t-stat	-0.5	1.1	1.1	0.4	0.6	0.5
R^2	18.8%	27.7%	21.8%	12.0%	16.7%	10.2%

Note: The table shows time-series regression statistics on excess market return, momentum, and short-term reversal (from Ken French's data library), weekly short-term reversal factors, and trend factors (constructed as the $H - L$ WSTR/TREND spread from Table 2). The columns show equal-weighted portfolio (EW) and value-weighted portfolio (VW) for 5-day, 20-day, and 60-day images, respectively.

Table IA1 estimates the extent to which image-based strategy returns are explained by exposure to the market or to well known price trend strategies. For conciseness, we collapse the CNN portfolios from nine down to three by taking the equal weighted average of all models with the same supervision window. Of the five control factors, the market portfolio is most significantly associated with image-based strategies. After accounting for their negative betas, image-based alphas rise slightly above the raw average return. The primary conclusion from the table, however, is that image strategies bear little in common with well known momentum or reversal phenomena, and that these well known strategies are not responsible for the success of the CNN strategy.

In the main text, our portfolio analyses study holding periods equal to the forecast horizon in each model (e.g., rebalancing positions every five days for I5/R5 but every 60 days for I5/R60). But there is no guarantee that the best quarterly trading portfolio, for example, comes from a model that was trained with 60-day returns. Table IA2 studies the possibility that CNN models trained with one supervising return horizon are helpful for predicting different return horizons out-of-sample. To this end, we investigate portfolio performance for a range of holding periods (5, 20, or 60 days) regardless of the horizon of the supervising return.

Table IA2: Other Holding Periods

	Holding Period					
	Equal Weight			Value Weight		
Model	5 days	20 days	60 days	5 days	20 days	60 days
I5/R5	7.15	2.77	1.40	1.49	0.77	0.46
I5/R20	5.53	2.35	1.36	1.26	0.45	0.31
I5/R60	4.32	2.05	1.30	0.74	0.32	0.47
I20/R5	6.75	3.17	1.59	1.74	0.73	0.37
I20/R20	3.87	2.16	1.05	0.89	0.49	0.44
I20/R60	0.97	0.45	0.37	0.01	0.26	0.32
I60/R5	4.89	2.33	1.08	1.44	0.55	0.35
I60/R20	2.19	1.29	0.60	0.99	0.17	0.13
I60/R60	1.00	0.75	0.43	0.36	0.38	0.23
Combo	4.07	1.92	1.02	0.99	0.46	0.34

Note: The table reports out-of-sample Sharpe ratios for holding periods of 5, 20, or 60 days with equal weight and value weight portfolios. All strategies use H-L decile spread portfolios sorted by image-based return predictions from each model.

Indeed, we see that there is no single image length, and no single supervision window, that is dominant in terms of portfolio performance across investment horizons. For example, it is often the case that quarterly trading strategies benefit when the prediction model is supervised with shorter horizon returns. Furthermore, we find that image-based forecasts from different models pick up on different signals. For example, the correlation between the I5/R5 and I20/R20 models is 32%. Pairwise correlations between all nine model configurations are below 50% on average and as low as 7% (and as high as 74%) between some models. This suggests that investment strategies can benefit from combining forecasts from multiple CNN models.

In Table IA3 we conduct the same analysis as Table 2 but using value weights rather than equal weights. For one-month holding periods, CNN strategies deliver out-of-sample H-L Sharpe ratios of 0.5, 0.5, and 0.2, versus 0.4, 0.0, and 0.3 for MOM, STR, and WSTR, respectively. This demonstrates the well known robustness of MOM for value-weight strategies, and the well known *lack* of robustness of STR, which is wiped out by the use of value weights and is consistent with the fact that STR's performance is driven by micro-cap stocks. CNN is thus an important contrast with STR because it continues to excel amid value weighting, illustrating that image data is predictive across the size spectrum. Finally, at the quarterly investment horizon, the only strategy delivering significant H–L returns is the CNN using 5-day images, which achieves a Sharpe ratio of 0.5, followed by insignificant 0.3 and 0.2 Sharpe ratios using 20- and 60-day images. The next best quarterly strategy is WSTR with an insignificant Sharpe ratio of 0.2, while the Sharpe ratio of MOM drops to 0.1.

Table IA3: Performance of Value Weight Portfolios At Lower Frequency

	I5/R20		I20/R20		I60/R20		MOM/R20		STR/R20		WSTR/R20		TREND/R20	
	Ret	SR	Ret	SR	Ret	SR	Ret	SR	Ret	SR	Ret	SR	Ret	SR
Low	0.05	0.31	0.05	0.28	0.06	0.32	0.01	0.03	0.04	0.21	0.02	0.07	0.01	0.05
2	0.05	0.29	0.04	0.21	0.05	0.24	0.02	0.08	0.06	0.38	0.05	0.26	0.05	0.25
3	0.05	0.30	0.05	0.32	0.07	0.42	0.04	0.18	0.06	0.36	0.05	0.30	0.06	0.33
4	0.07	0.44	0.05	0.33	0.10	0.55	0.07	0.35	0.08	0.57	0.06	0.44	0.05	0.35
5	0.06	0.42	0.06	0.40	0.05	0.32	0.07	0.43	0.09	0.61	0.09	0.62	0.07	0.51
6	0.08	0.48	0.08	0.50	0.07	0.41	0.08	0.56	0.09	0.60	0.09	0.60	0.08	0.60
7	0.07	0.46	0.08	0.51	0.08	0.48	0.09	0.64	0.11	0.68	0.11	0.78	0.10	0.66
8	0.09	0.56	0.09	0.58	0.09	0.59	0.10	0.73	0.10	0.57	0.10	0.63	0.10	0.63
9	0.09	0.56	0.08	0.55	0.07	0.48	0.10	0.69	0.08	0.37	0.10	0.51	0.13	0.66
High	0.10	0.65	0.11	0.67	0.09	0.62	0.14	0.67	0.05	0.18	0.07	0.27	0.12	0.49
H-L	0.05*	0.45	0.05**	0.49	0.02	0.17	0.12	0.36	0.01	0.03	0.06	0.30	0.10**	0.53
Turnover	187%		181%		165%		76%		184%		185%		156%	
	I5/R60		I20/R60		I60/R60		MOM/R60		STR/R60		WSTR/R60		TREND/R60	
	Ret	SR	Ret	SR	Ret	SR	Ret	SR	Ret	SR	Ret	SR	Ret	SR
Low	0.07	0.34	0.06	0.27	0.07	0.32	0.08	0.19	0.06	0.24	0.04	0.15	0.10	0.30
2	0.11	0.55	0.07	0.33	0.10	0.50	0.04	0.12	0.09	0.46	0.07	0.35	0.07	0.28
3	0.08	0.41	0.08	0.38	0.08	0.46	0.09	0.38	0.08	0.49	0.08	0.47	0.09	0.42
4	0.08	0.45	0.09	0.47	0.09	0.47	0.09	0.45	0.09	0.60	0.10	0.63	0.11	0.65
5	0.10	0.60	0.09	0.45	0.08	0.43	0.08	0.53	0.11	0.72	0.11	0.72	0.10	0.67
6	0.08	0.48	0.08	0.48	0.09	0.49	0.11	0.67	0.10	0.62	0.10	0.59	0.11	0.74
7	0.09	0.52	0.09	0.50	0.09	0.54	0.10	0.70	0.11	0.68	0.11	0.68	0.09	0.63
8	0.09	0.56	0.10	0.57	0.09	0.53	0.09	0.61	0.10	0.54	0.10	0.56	0.08	0.55
9	0.08	0.51	0.10	0.69	0.09	0.52	0.09	0.62	0.09	0.41	0.09	0.45	0.09	0.55
High	0.11	0.76	0.09	0.65	0.11	0.75	0.12	0.57	0.08	0.26	0.08	0.28	0.10	0.42
H-L	0.05**	0.47	0.04	0.32	0.03	0.23	0.04	0.10	0.02	0.08	0.04	0.19	-0.00	-0.01
Turnover	62%		60%		59%		42%		61%		62%		56%	

Note: Performance of value-weighted decile portfolios sorted on out-of-sample predicted up probability. Each panel reports the average annualized holding period return and Sharpe ratio. Average returns accompanied by ***,**,* are significant at the 1%, 5% and 10% significance level, respectively. We also report monthly turnover of each strategy.

Table IA4: Performance of Equal Weight Portfolios with Transaction Cost

	I5/R5		I20/R5		I60/R5	
	Ret	SR	Ret	SR	Ret	SR
Low	-0.10	-0.66	-0.14	-0.82	-0.02	-0.13
High	0.36	1.94	0.35	1.85	0.17	0.97
H-L	0.46***	4.03	0.48***	3.90	0.20***	1.78
Turnover	690%		667%		619%	
	I5/R20		I20/R20		I60/R20	
	Ret	SR	Ret	SR	Ret	SR
Low	0.04	0.23	0.02	0.10	0.03	0.13
High	0.17	0.89	0.15	0.84	0.11	0.76
H-L	0.13***	1.47	0.13***	1.35	0.08***	0.66
Turnover	175%		173%		155%	
	I5/R60		I20/R60		I60/R60	
	Ret	SR	Ret	SR	Ret	SR
Low	0.09	0.37	0.10	0.37	0.09	0.32
High	0.15	0.71	0.12	0.68	0.13	0.81
H-L	0.06***	0.91	0.02	0.17	0.04	0.27
Turnover	59%		59%		58%	

Note: We add 10 bps transaction cost for large stocks with market capitalization greater than the NYSE 20% breakpoint and 20 bps for the rest.

Table IA5: Correlation Between Long/Short Legs of CNN Predictions and Technical Indicators

	Long Leg				Short Leg			
	MOM	STR	WSTR	TREND	MOM	STR	WSTR	TREND
I5/R5	0.81	0.92	0.95	0.91	0.81	0.88	0.87	0.90
I5/R20	0.82	0.92	0.94	0.91	0.84	0.89	0.90	0.91
I5/R60	0.84	0.92	0.95	0.92	0.91	0.92	0.91	0.94
I20/R5	0.81	0.90	0.94	0.88	0.86	0.89	0.93	0.91
I20/R20	0.82	0.90	0.92	0.88	0.91	0.90	0.94	0.94
I20/R60	0.86	0.85	0.88	0.87	0.96	0.92	0.96	0.97
I60/R5	0.87	0.88	0.92	0.89	0.90	0.92	0.94	0.93
I60/R20	0.88	0.88	0.89	0.89	0.94	0.91	0.94	0.95
I60/R60	0.90	0.87	0.89	0.89	0.98	0.93	0.96	0.98

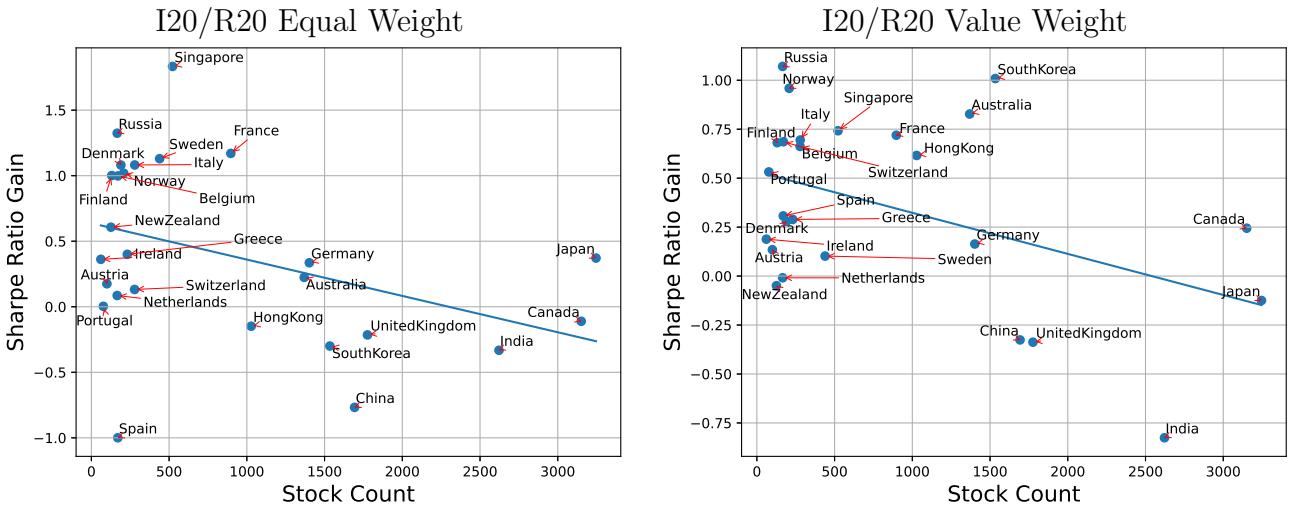
Note: The table reports the time-series correlation between the long and short leg returns of CNN strategies versus other technical indicator strategies.

Table IA6: International Transfer and H-L Decile Portfolio Sharpe Ratios (I20/R20)

		Equal Weight			Value Weight		
	Stock Count	Re-train	Direct Transfer	Transfer−Re-train	Re-train	Direct Transfer	Transfer−Re-train
US	7298	2.16			0.49		
Global	17206	-0.07	0.21	0.29	0.64	-0.25	-0.90
Japan	3056	0.61	0.99	0.37*	0.24	0.12	-0.12
Canada	2924	0.04	-0.07	-0.11	0.52	0.77	0.24
India	1861	0.99	0.66	-0.33	0.94	0.11	-0.83
UnitedKingdom	1783	0.19	-0.03	-0.21	0.66	0.32	-0.34
France	955	-0.36	0.81	1.17***	-0.42	0.30	0.72***
SouthKorea	911	0.89	0.59	-0.30	-0.26	0.75	1.01***
Australia	886	1.97	2.20	0.22	-0.22	0.61	0.83***
Germany	868	-0.26	0.08	0.34*	0.24	0.40	0.16
China	662	0.82	0.06	-0.77	0.39	0.07	-0.33
HongKong	543	1.63	1.48	-0.15	0.51	1.12	0.62***
Singapore	284	0.36	2.20	1.83***	0.24	0.98	0.74***
Sweden	260	0.30	1.43	1.13***	0.61	0.71	0.10
Italy	241	0.63	1.71	1.08***	-0.05	0.65	0.69***
Switzerland	240	-0.07	0.06	0.13	0.06	0.72	0.66***
Denmark	223	-0.17	0.92	1.08***	0.35	0.63	0.28
Netherlands	212	-0.40	-0.31	0.09	0.24	0.23	-0.01
Greece	201	-0.30	0.10	0.40**	0.07	0.36	0.29
Belgium	171	-0.19	0.81	1.00***	-0.21	0.48	0.69***
Spain	170	0.76	-0.24	-1.00	0.09	0.40	0.31*
Norway	169	-0.02	1.00	1.02***	-0.19	0.77	0.96***
Portugal	121	0.27	0.27	0.00	-0.02	0.51	0.53**
NewZealand	114	0.43	1.04	0.61***	0.56	0.51	-0.05
Finland	113	0.37	1.37	1.00***	-0.23	0.45	0.68***
Austria	110	-0.10	0.08	0.18	0.31	0.44	0.13
Ireland	75	-0.17	0.19	0.36*	0.46	0.65	0.19
Russia	53	-0.42	0.91	1.32***	-0.32	0.75	1.07***
Average	1274	0.29	0.69	0.40	0.19	0.50	0.31
Average (excluding Global)	661	0.30	0.70	0.40	0.18	0.53	0.36

Note: The table reports annualized out-of-sample Sharpe ratios for H-L decile spread portfolios within each country. We report the average monthly stock count by country, the image-based strategy from re-training the I20/R20 CNN using local data, and the image-based strategy directly transfers the I20/R20 model estimated in US data without re-training. Sharpe ratio gains (Transfer−Re-train) accompanied by ***,**,* are significant at the 1%, 5% and 10% significance level, respectively.

Figure IA1: Sharpe Ratio Gains From International Transfer



Note: The figures show annualized out-of-sample Sharpe ratio gains from I20/R20 transfer learning (both equal-weight and value-weight strategies) versus the average number of stocks in each country.

Table IA7: International Transfer and H-L Decile Equal Weighted Portfolio Performance Breakdown (I20/R20)

		Day 1 to Day 5			Day 6 to Day 20		
	Stock Count	Re-train	Direct Transfer	Transfer−Re-train	Re-train	Direct Transfer	Transfer−Re-train
US	7298	2.50			1.21		
Global	17206	1.52	0.63	-0.90	-0.42	-0.49	-0.07
Japan	3056	1.05	1.15	0.10	0.22	0.52	0.30*
Canada	2924	1.37	0.11	-1.26	-0.38	-0.38	-0.01
India	1861	1.25	1.19	-0.06	0.49	0.03	-0.46
UnitedKingdom	1783	0.27	-0.04	-0.31	-0.14	-0.31	-0.17
France	955	0.17	1.26	1.09***	-0.30	0.11	0.41**
SouthKorea	911	0.32	0.30	-0.02	0.69	0.14	-0.55
Australia	886	2.20	3.18	0.98***	0.64	0.64	0.00
Germany	868	0.03	0.46	0.44**	-0.33	-0.60	-0.27
China	662	0.71	0.06	-0.65	0.53	0.04	-0.49
HongKong	543	1.26	2.85	1.59***	1.05	0.52	-0.54
Singapore	284	0.19	3.22	3.03***	0.38	0.52	0.14
Sweden	260	0.12	2.22	2.10***	-0.16	0.26	0.42**
Italy	241	0.51	0.72	0.21	0.50	1.21	0.71***
Switzerland	240	0.11	0.90	0.79***	-0.15	-0.24	-0.09
Denmark	223	-0.10	1.14	1.24***	-0.27	0.29	0.56***
Netherlands	212	0.19	0.77	0.58***	-0.45	-0.35	0.10
Greece	201	0.05	1.18	1.13***	-0.27	-0.18	0.10
Belgium	171	-0.23	0.45	0.69***	0.02	0.13	0.11
Spain	170	0.42	0.83	0.41**	0.63	-0.25	-0.88
Norway	169	0.48	1.53	1.05***	-0.08	0.28	0.36*
Portugal	121	0.27	0.69	0.42**	-0.01	-0.10	-0.10
NewZealand	114	0.74	1.23	0.49**	-0.16	0.34	0.51**
Finland	113	0.32	1.60	1.28***	0.36	0.25	-0.11
Austria	110	-0.19	-0.02	0.17	-0.10	0.26	0.36*
Ireland	75	-0.21	0.73	0.94***	-0.18	-0.27	-0.09
Russia	53	-0.00	1.89	1.89***	-0.65	-0.08	0.56***
Average	1274	0.47	1.12	0.65	0.05	0.08	0.03
Average (excluding Global)	661	0.43	1.14	0.70	0.07	0.11	0.03

Note: The table reports breakdown of performance of out-of-sample H-L decile spread portfolios within each country. We report the average monthly stock count by country, the image-based strategy from re-training the I20/R20 CNN using local data, and the image-based strategy directly transfers the I20/R20 model estimated in US data without re-training. Sharpe ratio gains (Transfer−Re-train) accompanied by ***,**,* are significant at the 1%, 5% and 10% significance level, respectively.

Table IA8: International Transfer and H-L Decile Value Weighted Portfolio Performance Breakdown (I20/R20)

		Day 1 to Day 5			Day 6 to Day 20		
	Stock Count	Re-train	Direct Transfer	Transfer−Re-train	Re-train	Direct Transfer	Transfer−Re-train
US	7298	0.98			0.10		
Global	17206	0.30	-0.30	-0.60	0.49	-0.23	-0.72
Japan	3056	0.14	0.47	0.33*	0.08	0.00	-0.08
Canada	2924	0.73	1.07	0.35*	0.11	0.17	0.05
India	1861	0.32	-0.06	-0.38	0.89	0.15	-0.74
UnitedKingdom	1783	0.52	0.44	-0.08	0.37	0.20	-0.17
France	955	-0.32	0.05	0.37*	-0.27	0.56	0.84***
SouthKorea	911	-0.21	0.51	0.72***	-0.13	0.61	0.74***
Australia	886	0.15	1.08	0.93***	-0.31	0.13	0.44**
Germany	868	0.04	0.24	0.20	0.27	0.36	0.09
China	662	0.19	-0.14	-0.33	0.37	0.12	-0.25
HongKong	543	0.07	1.22	1.15***	0.44	0.62	0.19
Singapore	284	0.27	1.13	0.86***	0.10	0.22	0.12
Sweden	260	0.33	0.81	0.48**	0.53	0.52	-0.01
Italy	241	-0.12	0.31	0.43**	0.13	0.54	0.41**
Switzerland	240	0.42	0.70	0.28	0.03	0.51	0.47**
Denmark	223	-0.01	0.94	0.95***	0.34	0.01	-0.33
Netherlands	212	0.24	0.30	0.06	0.07	0.00	-0.07
Greece	201	0.45	1.14	0.69***	-0.21	-0.20	0.00
Belgium	171	0.10	0.50	0.40**	-0.25	0.22	0.47**
Spain	170	-0.11	0.69	0.80***	0.13	0.11	-0.02
Norway	169	0.05	1.10	1.06***	-0.08	0.37	0.45**
Portugal	121	0.21	0.78	0.57***	-0.07	0.16	0.23
NewZealand	114	0.64	0.64	0.01	0.01	0.27	0.26
Finland	113	-0.25	0.40	0.64***	-0.12	0.12	0.24
Austria	110	0.36	0.77	0.41**	0.08	-0.05	-0.13
Ireland	75	0.21	0.67	0.46**	0.25	0.25	-0.00
Russia	53	0.09	1.02	0.93***	-0.36	0.24	0.59***
Average	1274	0.18	0.61	0.43	0.11	0.22	0.11
Average (excluding Global)	661	0.17	0.64	0.47	0.09	0.24	0.15

Note: The table reports breakdown of performance of out-of-sample H-L decile spread portfolios within each country. We report the average monthly stock count by country, the image-based strategy from re-training the I20/R20 CNN using local data, and the image-based strategy directly transfers the I20/R20 model estimated in US data without re-training. Sharpe ratio gains (Transfer−Re-train) accompanied by ***,**,* are significant at the 1%, 5% and 10% significance level, respectively.

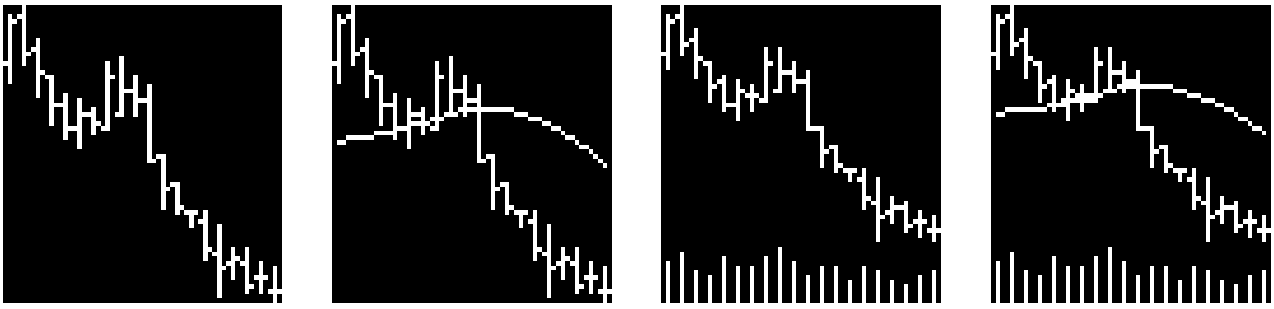
IA.1.1 Sensitivity to Data Choices, Model Structure, and Estimation Methods

In this section we conduct a number of sensitivity analyses. We report annualized Sharpe ratios for long-short decile strategies (using equal and value weights). We focus on the 20-day return horizon and use images of 5, 20, and 60 days. In all sensitivity analyses, the baseline version of each variation is listed first and shown in bold.

First, in Table IA10, we consider how model performance is affected by excluding volume bars (VB) or moving average price curve (MA) from images. In order to assess the predictive contribution of these elements, we report prediction sensitivity analyses using OHLC charts with and without moving average and volume elements according to the four combinations in Figure IA2. Table IA9 summarizes pixel dimensions for each of the design choices we consider.

We report classification accuracy and annualized Sharpe ratios for long-short decile strategies (using equal and value weights). We find that for small images (5 days), there is some gain (especially in value weight strategies) to omitting volume and moving average price. For 20-day and 60-day images, however, excluding volume and moving average information is detrimental to model performance. A possible reason for that is moving average lines create more noise than information in 5-day images, but this is not an issue for 20 and 60-day images.

Figure IA2: Examples of 20-day Image under Different Settings



(a) w/o VB, w/o MA (b) w/o VB, w/ MA (c) w/ VB, w/o MA (d) w/ VB, w/ MA

Note: From left to right are 20-day images (a) without volume bar and moving average line, (b) without volume bar but with moving average line, (c) with volume bar but without moving average line, and (d) with volume bar and moving average line.

Table IA9: Image Dimensions of the OHLC Chart with/without Volume Bars

	Sample Size (in Days)	Image Width (in Pixels)	Image Height (in Pixels)	
OHLC	5	15	32	
	20	60	64	
	60	180	96	
			Prices	Volume Bars
OHLC+V	5	15	25	6
	20	60	51	12
	60	180	76	19

Table IA10: Sensitivity to Data Augmentation

		I5/R20		I20/R20		I60/R20	
		Sharpe Ratio		Sharpe Ratio		Sharpe Ratio	
		EW	VW	EW	VW	EW	VW
Baseline		**2.35**	**0.45**	**2.16**	**0.49**	**1.29**	**0.17**
VB/MA	no VB, MA	2.42	0.55	1.03	-0.03	0.79	0.13
	VB, no MA	2.52	0.79	1.72	0.40	1.06	0.09
	no VB, no MA	2.51	0.64	0.99	0.23	0.67	0.17
Return Filter	10%	2.49	0.57	1.99	0.24	1.18	0.23
	20%	2.29	0.56	1.55	0.02	1.25	0.21
	30%	2.32	0.61	1.62	0.27	0.91	0.14
	40%	2.17	0.72	1.23	-0.02	0.77	0.26
	50%	1.94	0.46	1.25	-0.01	0.85	0.11
Largest Stocks	4000	1.67	0.52	0.85	0.35	0.21	0.16
for Training	2000	1.16	0.64	0.44	0.42	-0.10	-0.13

Note: This table reports model performance sensitivity to data variations. "VB/MA" reports the effect of excluding volume bars and/or the moving average price curve from images. "Return Filter" reports the effect of excluding x% of return with the smallest volatility-scaled returns in the training samples. "Largest Stocks for Training" reports the effect of training using only the top 4,000 or 2,000 firms in a training sample. The baseline version is listed first and shown in bold; it includes both VB and MA, uses a return filter of 0%, and uses all stocks with no size cutoff. We report annualized Sharpe ratios for long-short decile strategies (using equal and value weights). Models are based on a 20-day return horizon and use images of 5, 20, and 60 days.

In some applications, removing observations that are close to the classification boundary (i.e., returns near zero in our case) helps to de-noise the training data and improve out-of-sample performance. We consider filtering the training data to remove observations that have small returns relative to their volatility. In particular, we look at estimation variants that remove the smallest 10%, 20%, 30%, 40%, or 50% of volatility-scaled returns from the training sample. We find that this de-noising degrades model performance in general, though for 5-day images there are small benefits to using a mild filter.

Likewise, in the spirit of de-noising in training, we explore whether there are benefits to excluding the smallest stocks from the training sample, as these tend to be less liquid, more volatile, and heavier tailed. Across the board we find no benefits in model performance to limiting stocks to the top 4,000 or top 2,000 stocks during training.

In Table IA11, we explore performance sensitivity to various dimensions of model structure and estimation choices. We report sensitivity in terms of annualized Sharpe ratios for long-short decile strategies.

First, we report the effect of varying the number of filters in the first layer of the CNN from 64 down to 32 or up to 128. We see that model performance is fairly insensitive to the number of filters. Next, we increase/decrease the number of convolution layers from 3 to 2 or 4. Decreasing the number of layers meaningfully degrades the model performance. The

Table IA11: Sensitivity to Model Structure and Estimation, I20/R20

		Sharpe Ratio	
		EW	VW
Baseline		**2.16**	**0.49**
Filters (64)	32	2.00	0.28
	128	1.85	0.40
Layers (3)	2	1.77	0.33
	4	2.14	0.22
Dropout (0.50)	0.00	2.14	0.59
	0.25	2.31	0.51
	0.75	1.47	0.16
BN (yes)	no	2.33	0.51
Xavier (yes)	no	2.08	0.44
Activation (LReLU)	ReLU	1.49	0.23
Max-pool Size (2×1)	(2×2)	1.62	0.32
FilterSize (5×3)	(3×3)	1.53	0.16
	(7×3)	1.84	0.09
Dilation/Stride (2,1)/(3,1)	(2,1)/(1,1)	2.20	0.26
	(1,1)/(3,1)	2.00	0.30
	(1,1)/(1,1)	1.80	0.25

Note: This table reports model performance sensitivity to model and estimation variations including the number of filters in the first layer (equal to 64 in baseline model), number of convolutional layers (baseline 3), dropout probability (baseline 0.50), use of batch normalization (BN, baseline yes), use Xavier initialization (baseline yes), activation function (baseline leaky ReLU), size of max-pooling layers (baseline (2,1)), filter size (baseline 5×3 pixels), and combinations of dilation and stride (baseline (2,1) and (3,1)). The columns show annualized decile spread Sharpe ratio with equal and value weights.

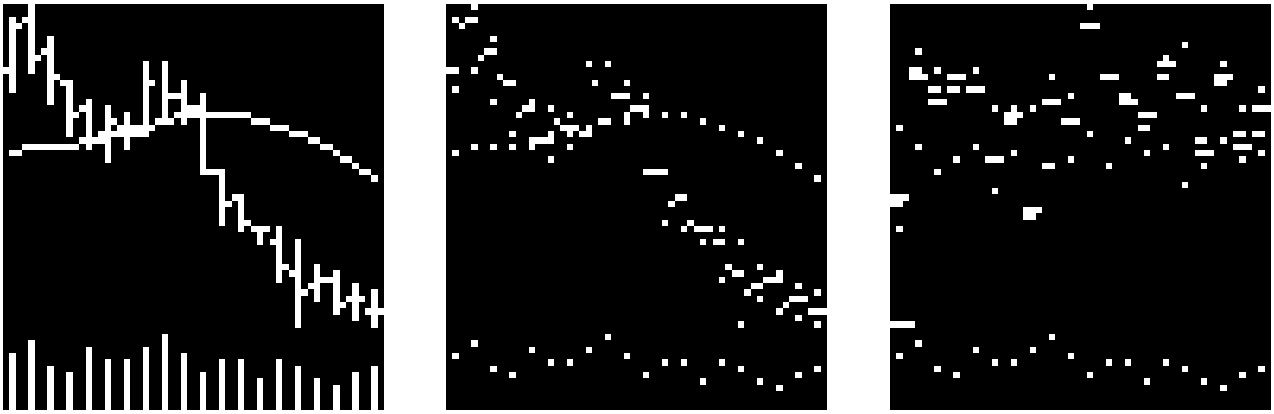
model is essentially unchanged if we lower the dropout probability, though raising it to 0.75 produces a noticeable loss in performance. Performance is insensitive to omitting the batch normalization step or Xavier initialization, but we see a loss in trading strategy performance when we transition from leaky ReLU to ReLU. Changing the max-pooling size from (2×1) to (2×2) reduces the performance significantly. Either smaller filters or larger filters hurt performance. Finally, we notice some loss in performance when we alter dilation, though the results in this dimension are mixed.

Overall, the broad conclusion of this analysis is that our main results are robust to alternative modeling choices.

Next, we study the robustness of our results to additional image representations of the data. The first, which we call the "pixel" representation, presents the data with less profligate use of the color white. In particular, it replaces price bars with just one pixel each for high, low, open, and close price. It also replaces daily volume bars with a single pixel. (the top of the volume bar). This is shown in Panel (b) of Figure IA3.

The second representation we call "centered pixel." It normalizes all prices by the closing

Figure IA3: Examples of 20-day Images With Different Representations



(a) OHLC Chart (b) Pixel Representation (c) Centered Pixel Repr.

Note: From left to right are 20-day images (a) with bar representation, (b) with pixel representation, and (c) with centered pixel representation.

price, then plots pixels for high–close, low–close, open–close, close–previous close, moving average–close. This is shown in Panel (c) of Figure IA3.

Table IA12 reports short-horizon (weekly) portfolio performance for CNNs trained on the pixel and centered pixel images with 5-day return supervision. Table IA13 reports longer-horizon (monthly and quarterly) equal-weight portfolio performance for CNNs trained on the pixel and centered pixel images with 20-day and 60-day return supervision, while Table IA14 reports the corresponding value-weight portfolios. The performance are a bit worse than the benchmark image specification but remain significantly profitable, with equal weighted portfolios achieving 6.4 and 5.2 Sharpe ratios, respectively, based on Pixel and Centered Pixel images. While these alternative image representations have the same information content as the benchmark image, they are less intuitive and more sparse. Not surprisingly, the choice of image specifications still matters, although the differences are not substantial.

Table IA15 report cross-sectional correlations for different image specifications using three different models for I20/R20. Our baseline representation is based on the bar image with 3 layers for I20/R20. Consistent with aforementioned experiments, altering the design of input images leads to lower correlations than altering the number of layers in CNN models. Finally, Table IA16 reports additional model specifications when images use pixel or centered pixel representations. Again, the performance does not appear sensitive to modeling choices.

Table IA12: Short-horizon Portfolio Performance With Different Image Representations

	Equal Weight											
	Pixel I5/R5		Pixel I20/R5		Pixel I60/R5		Centered I5/R5		Centered I20/R5		Centered I60/R5	
	Ret	SR	Ret	SR	Ret	SR	Ret	SR	Ret	SR	Ret	SR
Low	-0.24	-1.63	-0.19	-1.13	0.09	0.45	-0.13	-0.84	-0.16	-0.92	0.02	0.12
2	-0.01	-0.08	0.04	0.23	0.13	0.68	0.02	0.12	0.01	0.07	0.08	0.44
3	0.04	0.22	0.08	0.43	0.12	0.65	0.06	0.34	0.05	0.27	0.10	0.59
4	0.09	0.50	0.11	0.58	0.14	0.73	0.11	0.59	0.10	0.53	0.11	0.64
5	0.13	0.68	0.13	0.69	0.12	0.66	0.12	0.65	0.12	0.65	0.13	0.73
6	0.14	0.73	0.15	0.80	0.15	0.79	0.14	0.76	0.16	0.84	0.14	0.79
7	0.16	0.82	0.17	0.92	0.13	0.73	0.16	0.88	0.18	0.98	0.15	0.81
8	0.21	1.07	0.19	1.03	0.13	0.75	0.19	0.98	0.20	1.09	0.16	0.88
9	0.27	1.37	0.21	1.14	0.14	0.79	0.24	1.25	0.24	1.30	0.17	0.88
High	0.49	2.63	0.39	2.18	0.13	0.78	0.38	2.08	0.38	1.98	0.22	1.14
H-L	0.73***	6.40	0.58***	5.30	0.04	0.29	0.51***	5.24	0.53***	4.36	0.19***	1.82
Turnover	677%		672%		557%		700%		691%		686%	
	Value Weight											
	Pixel I5/R5		Pixel I20/R5		Pixel I60/R5		Centered I5/R5		Centered I20/R5		Centered I60/R5	
	Ret	SR	Ret	SR	Ret	SR	Ret	SR	Ret	SR	Ret	SR
Low	-0.04	-0.22	0.03	0.15	0.06	0.31	0.02	0.14	0.01	0.06	0.04	0.21
2	-0.00	-0.01	0.06	0.37	0.13	0.63	0.05	0.29	0.05	0.31	0.08	0.44
3	0.02	0.14	0.06	0.31	0.07	0.34	0.04	0.25	0.06	0.35	0.08	0.44
4	0.08	0.46	0.05	0.29	0.06	0.34	0.09	0.50	0.06	0.36	0.09	0.51
5	0.09	0.50	0.09	0.48	0.09	0.51	0.08	0.44	0.05	0.30	0.07	0.41
6	0.08	0.46	0.08	0.47	0.08	0.42	0.09	0.52	0.08	0.46	0.08	0.46
7	0.09	0.51	0.09	0.49	0.09	0.54	0.10	0.58	0.11	0.63	0.08	0.45
8	0.12	0.61	0.08	0.43	0.09	0.53	0.08	0.43	0.11	0.61	0.10	0.56
9	0.14	0.75	0.11	0.59	0.10	0.59	0.10	0.50	0.11	0.60	0.09	0.51
High	0.19	0.86	0.12	0.68	0.08	0.47	0.14	0.71	0.10	0.48	0.11	0.60
H-L	0.23***	1.38	0.10***	0.73	0.01	0.09	0.11***	0.83	0.08***	0.60	0.07***	0.63
Turnover	759%		734%		602%		768%		736%		707%	

Note: Performance of equal-weighted (top panel) and value-weighted (bottom panel) decile portfolios sorted on out-of-sample predicted up probability. Each panel reports the average holding period return and annualized Sharpe ratios. Average returns accompanied by ***,**,* are significant at the 1%, 5% and 10% significance level, respectively. We also report monthly turnover of each strategy.

Table IA13: Longer-horizon Portfolio Performance With Different Image Representations (Equal Weight)

	Pixel I5/R20		Pixel I20/R20		Pixel I60/R20		Centered I5/R20		Centered I20/R20		Centered I60/R20	
	Ret	SR	Ret	SR	Ret	SR	Ret	SR	Ret	SR	Ret	SR
Low	0.02	0.12	0.06	0.31	0.11	0.54	0.03	0.18	0.03	0.16	0.08	0.45
2	0.06	0.34	0.08	0.46	0.10	0.53	0.07	0.42	0.09	0.49	0.09	0.54
3	0.09	0.52	0.11	0.61	0.11	0.61	0.09	0.51	0.09	0.50	0.09	0.53
4	0.09	0.54	0.11	0.61	0.12	0.69	0.10	0.55	0.11	0.65	0.12	0.70
5	0.11	0.60	0.11	0.63	0.10	0.62	0.12	0.68	0.11	0.63	0.11	0.65
6	0.13	0.71	0.12	0.69	0.10	0.57	0.12	0.70	0.13	0.72	0.11	0.65
7	0.13	0.71	0.13	0.73	0.12	0.73	0.13	0.72	0.12	0.69	0.12	0.66
8	0.14	0.79	0.13	0.74	0.12	0.70	0.13	0.73	0.14	0.77	0.12	0.70
9	0.15	0.85	0.13	0.78	0.11	0.66	0.14	0.81	0.14	0.81	0.13	0.76
High	0.19	1.03	0.12	0.77	0.11	0.69	0.17	0.96	0.15	0.90	0.12	0.70
H-L	0.17***	1.89	0.07***	0.85	-0.00	-0.02	0.14***	1.56	0.12***	1.49	0.04***	0.61
Turnover	174%		175%		174%		176%		177%		176%	
	Pixel I5/R60		Pixel I20/R60		Pixel I60/R60		Centered I5/R60		Centered I20/R60		Centered I60/R60	
	Ret	SR	Ret	SR	Ret	SR	Ret	SR	Ret	SR	Ret	SR
Low	0.09	0.42	0.09	0.42	0.11	0.42	0.10	0.47	0.09	0.38	0.12	0.51
2	0.10	0.51	0.11	0.52	0.13	0.59	0.09	0.44	0.11	0.50	0.11	0.50
3	0.11	0.52	0.11	0.53	0.13	0.55	0.11	0.51	0.11	0.51	0.12	0.56
4	0.11	0.53	0.11	0.55	0.12	0.55	0.11	0.50	0.13	0.61	0.12	0.57
5	0.12	0.57	0.12	0.58	0.12	0.61	0.11	0.55	0.13	0.62	0.12	0.57
6	0.14	0.70	0.13	0.61	0.11	0.56	0.13	0.63	0.13	0.61	0.12	0.62
7	0.12	0.61	0.13	0.64	0.13	0.64	0.14	0.68	0.11	0.55	0.13	0.64
8	0.14	0.66	0.13	0.64	0.13	0.66	0.13	0.66	0.12	0.63	0.13	0.69
9	0.13	0.65	0.13	0.65	0.11	0.62	0.14	0.72	0.13	0.66	0.12	0.62
High	0.14	0.68	0.13	0.73	0.12	0.71	0.14	0.68	0.15	0.81	0.11	0.61
H-L	0.05***	0.89	0.04**	0.52	0.01	0.12	0.04**	0.54	0.06***	0.73	-0.01	-0.09
Turnover	60%		60%		60%		60%		60%		60%	

Note: Performance of equal-weighted decile portfolios sorted on out-of-sample predicted up probability. Each panel reports the average annualized holding period return and Sharpe ratio. Average returns accompanied by ***,**,* are significant at the 1%, 5% and 10% significance level, respectively. We also report monthly turnover of each strategy.

Table IA14: Longer-horizon Portfolio Performance With Different Image Representations (Value Weight)

	Pixel I5/R20		Pixel I20/R20		Pixel I60/R20		Centered I5/R20		Centered I20/R20		Centered I60/R20	
	Ret	SR	Ret	SR	Ret	SR	Ret	SR	Ret	SR	Ret	SR
Low	0.06	0.37	0.07	0.38	0.09	0.59	0.07	0.49	0.04	0.22	0.04	0.23
2	0.05	0.29	0.05	0.30	0.06	0.36	0.05	0.29	0.05	0.35	0.06	0.41
3	0.06	0.37	0.10	0.60	0.08	0.50	0.08	0.51	0.06	0.40	0.08	0.47
4	0.06	0.42	0.06	0.39	0.09	0.52	0.07	0.50	0.07	0.48	0.07	0.46
5	0.07	0.46	0.09	0.59	0.07	0.44	0.07	0.46	0.09	0.58	0.10	0.61
6	0.07	0.47	0.08	0.53	0.09	0.56	0.08	0.50	0.09	0.59	0.08	0.50
7	0.09	0.58	0.09	0.62	0.10	0.64	0.10	0.61	0.08	0.54	0.10	0.65
8	0.09	0.61	0.07	0.46	0.07	0.47	0.09	0.60	0.09	0.61	0.07	0.49
9	0.09	0.60	0.08	0.54	0.08	0.55	0.08	0.53	0.08	0.53	0.08	0.57
High	0.07	0.45	0.09	0.62	0.07	0.49	0.11	0.72	0.09	0.60	0.09	0.58
H-L	0.01	0.10	0.02	0.22	-0.02	-0.26	0.03	0.32	0.05*	0.42	0.05*	0.43
Turnover	187%		182%		179%		187%		182%		178%	
	Pixel I5/R60		Pixel I20/R60		Pixel I60/R60		Centered I5/R60		Centered I20/R60		Centered I60/R60	
	Ret	SR	Ret	SR	Ret	SR	Ret	SR	Ret	SR	Ret	SR
Low	0.06	0.31	0.07	0.34	0.10	0.55	0.06	0.35	0.08	0.42	0.09	0.50
2	0.08	0.46	0.08	0.41	0.09	0.50	0.07	0.43	0.09	0.52	0.09	0.49
3	0.11	0.61	0.10	0.55	0.07	0.37	0.09	0.51	0.08	0.43	0.11	0.66
4	0.09	0.51	0.10	0.54	0.10	0.60	0.10	0.56	0.09	0.50	0.12	0.72
5	0.10	0.61	0.07	0.42	0.09	0.59	0.09	0.50	0.08	0.54	0.09	0.59
6	0.12	0.72	0.09	0.49	0.09	0.51	0.09	0.52	0.09	0.60	0.08	0.48
7	0.08	0.48	0.09	0.54	0.10	0.67	0.09	0.55	0.10	0.55	0.10	0.58
8	0.09	0.58	0.08	0.56	0.09	0.53	0.10	0.64	0.09	0.55	0.09	0.57
9	0.08	0.47	0.11	0.77	0.09	0.55	0.09	0.59	0.11	0.65	0.09	0.55
High	0.10	0.67	0.10	0.67	0.10	0.63	0.11	0.72	0.10	0.63	0.09	0.55
H-L	0.05*	0.39	0.03	0.37	0.01	0.07	0.05**	0.47	0.02	0.21	-0.01	-0.07
Turnover	62%		60%		60%		62%		61%		60%	

Note: Performance of value-weighted decile portfolios sorted on out-of-sample predicted up probability. Each panel reports the average annualized holding period return and Sharpe ratio. Average returns accompanied by ***,**,* are significant at the 1%, 5% and 10% significance level, respectively. We also report monthly turnover of each strategy.

Table IA15: Prediction Correlation Between Different Image Representations and CNN Specifications

	Bar L2	Bar L3	Bar L4	Pixel L2	Pixel L3	Pixel L4	Centered L2	Centered L3
Bar L2								
Bar L3	0.88							
Bar L4	0.86	0.88						
Pixel L2	0.60	0.56	0.54					
Pixel L3	0.60	0.58	0.57	0.74				
Pixel L4	0.50	0.48	0.45	0.63	0.67			
Centered L2	0.45	0.46	0.46	0.40	0.42	0.33		
Centered L3	0.47	0.48	0.47	0.41	0.43	0.34	0.79	
Centered L4	0.45	0.45	0.45	0.40	0.41	0.33	0.79	0.83

Note: The table reports average cross-sectional correlations among model forecasts averaged over all periods in the test sample for I20/R20 models. Bar, Pixel, and Centered corresponds to baseline (bar) representation, pixel representation, and centered pixel representation. L2, L3, and L4 corresponds to CNN models with 2, 3, and 4 convolutional layers (Our main CNN specification has 3 layers for I20/R20).

IA.1.2 Comparison With Other Computer Vision Models

In this subsection, we compare CNN to two classical visual recognition methods, namely, Adaboost with HAAR-like Features by Viola and Jones (2001) and Histogram of Oriented Gradients (HOG) by Dalal and Triggs (2005), both of which are widely used precursors to CNN. A distinction between CNN and these methods is that CNN need little or no data preprocessing and automatically extract features from the data, while features in the precursor algorithms are hand-engineered. Indeed, neither method works directly with raw pixel images at all; instead they rely on pre-selected features extracted from the pixels, in contrast to CNN.

IA.1.2.1 Adaboost with HAAR-like Features

HAAR-like features are first introduced in Papageorgiou et al. (1998), which later gained popularity due to its widely adopted applications in real-time face detection proposed by Viola and Jones (2001). The HAAR-like features are reminiscent of HAAR basis functions in rectangle shapes, which manage to capture the differences in intensity within a region of image. For example, when used for face detection, it is shown in Viola and Jones (2001) that certain HAAR-like features learn that the region of eyes is darker than the region of the nose and cheeks.

There are five types of rectangles, as illustrated in Figure IA4, colored in red and green, respectively. A feature is constructed by taking the difference between the total sums of the pixels from two colored rectangular regions of a detection window. Because the detection window slides over the entire image and its size can vary arbitrarily within the image, the

Table IA16: Sensitivity to Model Structure and Estimation, I20/R20

Panel A: Pixel		Sharpe Ratio	
		EW	VW
Filters (64)	64 (BL)	**0.85**	**0.22**
	32	0.83	0.17
	128	1.09	0.27
Layers (3)	2	0.77	-0.03
	4	1.05	0.31
Dropout (0.50)	0.00	1.08	0.27
	0.25	0.82	0.15
	0.75	0.93	0.10
BN (yes)	no	0.80	0.12
Xavier (yes)	no	0.71	0.02
Activation (LReLU)	ReLU	0.93	0.26
Max Pool Size (2×1)	(2×2)	0.59	0.17
FilterSize (5×3)	(3×3)	0.92	0.23
	(7×3)	0.86	0.08
Dilation/Stride (2,1)/(3,1)	(2,1)/(1,1)	1.27	0.29
	(1,1)/(3,1)	0.80	0.19
	(1,1)/(1,1)	1.18	0.13
Panel B: Centered Pixel			
		Sharpe Ratio	
		EW	VW
Filters (64)	64 (BL)	**1.49**	**0.42**
	32	1.69	0.54
	128	1.53	0.15
Layers (3)	2	1.51	0.34
	4	1.39	0.55
Dropout (0.50)	0.00	1.04	0.56
	0.25	1.23	0.31
	0.75	1.52	0.40
BN (yes)	no	1.62	0.53
Xavier (yes)	no	1.54	0.32
Activation (LReLU)	ReLU	1.57	0.55
Max Pool Size (2×1)	(2×2)	1.68	0.65
FilterSize (5×3)	(3×3)	1.55	0.27
	(7×3)	1.59	0.38
Dilation/Stride (2,1)/(3,1)	(2,1)/(1,1)	1.46	0.50
	(1,1)/(3,1)	1.47	0.50
	(1,1)/(1,1)	1.63	0.55

Note: This table reports model performance sensitivity to model and estimation variations for pixel representation (Panel A) and centered pixel representation (Panel B) including the number of filters in the first layer (equal to 64 in baseline model), number of convolutional layers (baseline 3), dropout probability (baseline 0.50), use of batch normalization (BN, baseline yes), use Xavier initialization (baseline yes), activation function (baseline leaky ReLU), size of max-pooling layers (baseline (2,1)), filter size (baseline 5×3 pixels), and combinations of dilation and stride (baseline (2,1) and (3,1)). The columns show annualized decile spread Sharpe Ratio with equal and value weights.

total number of features constructed is massive. Specifically, there are over 100K features for a 5-day image of size 32×15, 7 million for a 20-day image of size 64×60, and 140 million for a 60-day image of size 96×180. Most of these features are either useless or redundant.

To combine these features, Viola and Jones (2001) adopt a boosting algorithm, Adaboost

Figure IA4: HAAR-like Features

Note: This figure shows the five types of HAAR-like features. Each type corresponds to a feature calculated by taking the differences between the sum of pixels over red and green regions.

first introduced by Freund and Schapire (1995), that incorporates an increasing number of weak learners, which in this case are individual features themselves. The final output of the algorithm is a classification probability by (weighted) voting from selected weak learners.

IA.1.2.2 Histogram of Oriented Gradients (HOG)

Similarly, the HOG method creates features from the input image by extracting the distribution (histograms) of directions of gradients (oriented gradients) at each pixel. As coordinates of pixels in an image are discreet, the horizontal gradient of that pixel is simply defined as the color difference between the adjacent pixels on the left and right (denoted as g_x) and the vertical gradient as the color difference between the adjacent pixels on the top and bottom of it (denoted as g_y).[23] Only four neighboring pixels are needed to calculate the gradient for each centered pixel whose value is not used. The magnitude of the gradient vector is given by $|g| = \sqrt{g_x^2 + g_y^2}$ and the direction by $\Theta_g = arctan(g_y/g_x)$. Intuitively, $|g_x|$ detects vertical edges, where there is substantial difference between the left and right pixels, and similarly $|g_y|$ for horizontal edges. As a result, $|g|$ detects regions of abrupt intensity changes, e.g., edges and corners, and Θ_g detects the orientations.

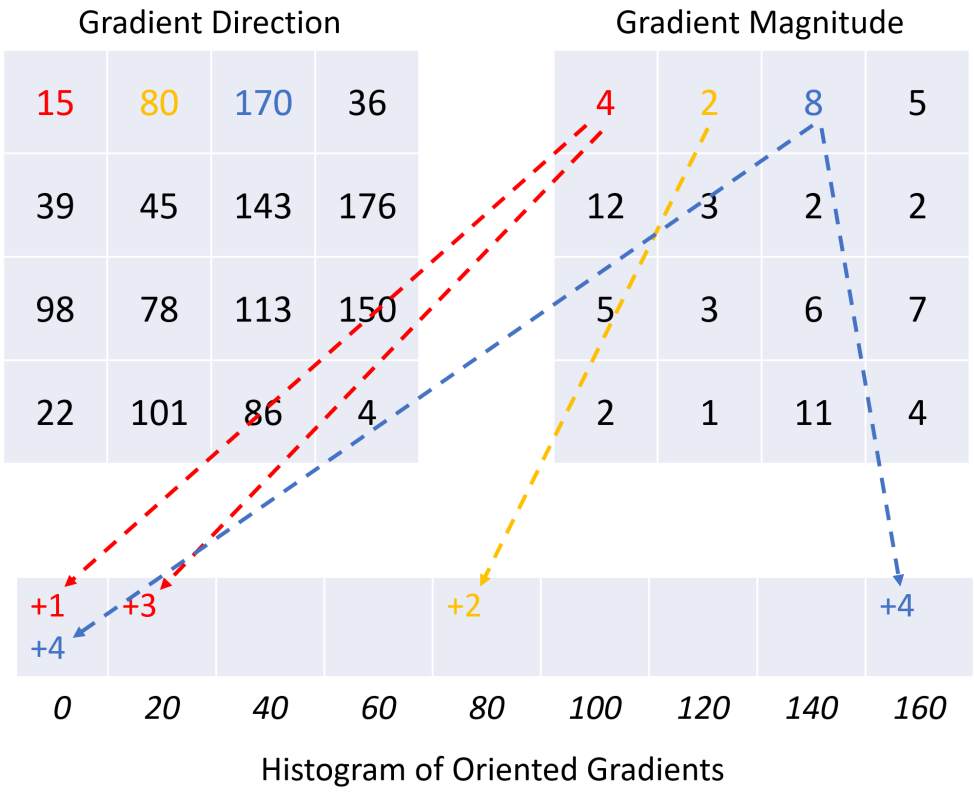
The resulting gradients are grouped into bins (like a histogram) and their summary statistics based on these bins are used as features. We follow the canonical approach by dividing the image into many cells of 8×8 pixels and creating a feature for each cell. An 8×8 cell has 64 values of gradient magnitude and 64 values of gradient direction. Using the histogram of gradients, these 128 values are summarized by a 9-dimensional vector.[24] The HOG algorithm therefore effectively reduces the dimension of input variables and potentially improves the model's out of sample performance. In total, there are 192 features for 5-day images, 2,464

[23] The HOG algorithm also supports colored images (e.g. RGB channels) but we focus our discussion on grey-scale images here.

[24] The histogram of all pixels in each cell is based on 9 bins corresponding to angles ranging from 0 to 180. The angles range from 0 to 180 instead of 360, the gradients are called "unsigned" gradients (a gradient and the other 180 degrees opposite to it are considered the same). We follow this tradition and use unsigned gradients. For each pixel's gradient g, Θ_g determines which bins get a share of the value $|g|$. Often Θ_g falls in between two bins, sharing the votes. The resulting feature is a size 9 vector.

features for 20-day images, and 12,320 features for 60-day images. If the direction is between two bins (e.g. 15 between 0 and 20), then the votes are divided proportionally to each bins (1/4 to 0 and 3/4 to 20). Finally, the histogram represented by the vector of size 9 is generated for each cell of pixel size 8×8. See Figure IA5 for a detailed example. In the original paper, histograms are also normalized within each block consisting of 2×2 cells. We follow the same setting in our feature generating process. The total number of features is not excessive, we thus directly use these features in a logistic regression to classify the images.

Figure IA5: A Detailed Procedure to Generate HOG



Note: For simplicity, this toy example shows the histogram from a cell of pixels size 4×4. The gradient magnitude of each pixel is divided proportionally to the bin(s) according to the gradient direction. Note that angle 170 lies between angle 160 and angle 0.

One thing to note here is that in the original HOG paper, SVM is used as the classifier. However, since we need a probability distribution of up and down moves to form the long-short portfolio, we replace SVM by logistic regression, a more intuitive classifier that directly outputs an "up" probability. In fact, we also implemented a HOG model with SVM and use the distance to the hyperplane as an alternative to up probability. The portfolio performance is inferior to the model with logistic regression. As a result, we only report the performance of HOG model with logistic regression.

Table IA17: Comparison of Sharpe Ratios Across Computer Vision Models

Variants	CNN	HOG	HAAR	CNN	HOG	HAAR	CNN	HOG	HAAR
	I5/R5			I5/R20			I5/R60		
EW	7.15	4.11	3.62	2.35	1.59	1.26	1.30	0.58	0.34
VW	1.49	0.73	0.37	0.45	0.60	0.13	0.47	0.31	0.30
	I20/R5			I20/R20			I20/R60		
EW	6.75	2.87	2.25	2.16	0.84	0.64	0.37	0.57	0.03
VW	1.74	0.30	0.35	0.49	0.15	-0.20	0.32	0.44	0.06
	I60/R5			I60/R20			I60/R60		
EW	4.89	1.91	0.45	1.29	1.17	0.35	0.43	0.64	0.09
VW	1.44	0.36	0.46	0.17	0.19	0.30	0.23	0.11	-0.15

Note: The table compares long-short decile spread portfolio returns with equal weights (EW) and value weights (VW) based on predictions from CNN, HOG, and HAAR models.

To ensure a fair comparison, the same setup of our CNN model is applied to these benchmark models. Specifically, all models are trained with the same training dataset of CNN model. In addition, for the HOG model, 5 models are trained independently to form an ensemble model. As for the HAAR-like model, since Adaboost is already an ensemble model (with 50 weak learners), it is not necessary to apply additional ensembling.

For the HOG model, there are 192 features for 5-day images, 2,464 features for 20-day images, and 12,320 features for 60-day images. Due to the large size of the training data, the model is trained with stochastic gradient descent that minimize the log loss, which gives logistic regression. L2 norm is used as regularizer with optimal coefficient selected from 0.001, 0.0001, and 0.00001. Finally, 2/7 of the training data are set aside to be used for early stopping that stop the training process after 2 epochs without loss decreasing.

As for the model on HAAR-like features, recall that there are $142,654,368$ HAAR-like features in a 60-day image of size 96×180. It is impossible to enumerate all those features and use them for training. Due to limitation in time and computing power, we have to use a heuristic method to generate most important HAAR-like features. One popular approach to select top HAAR-like features is to train an Adaboost model on only part of training data and select features with the highest feature importance. Since the pixel images are very sparse, to further speed up the process, we also resize the images to smaller sizes before extracting features. We keep the size 32×15 of 5-day images but shrinks the size of 20-day images from 64×60 to 22×22, and the size of 60-day images from 96×180 to 16×30. After the resizing, the total number of HAAR-like features are 112392, 114433, and 112606 for 5/20/60-day images respectively. For each model, we randomly sample 10k images and train an Adaboost model on all features and select the top 100 features with the highest feature importance. Then we generate the above 100 top features for all images and train an Adaboost model with 50 weak learners (decision stumps).

Table IA17 compares the performance of decile H-L portfolios from CNN, HAAR, and HOG models. In general, CNN outperforms both HOG and HAAR with few exceptions of strategies for longer horizons.

IA.2 Evaluating Popular Technical Analysis Patterns

Table IA18 describes the functional forms used to simulate 23 popular technical analysis patterns. Further details of the simulation coincide with the Monte Carlo design in Sections IA.3. Table IA19 reports the average probability of an up move estimated by the CNN across 10,000 simulations for each pattern. When the average probability is greater than 50% to a statistically significant degree, it is shown in green. If it is significantly less than 50% it is shown in red. The column labeled "sign" states the sign of the return forecast corresponding to each pattern based on folk wisdom (e.g., described in a variety of technical analysis books and websites).

Table IA18: Functional Forms For Simulating Patterns

	Functional Form
	Non-Linear Patterns
Cup And Handle	$Y = \begin{cases} -[(2/5)^2 - (X - 2/5)^2]^{1/2}, & \text{if } 0 \leq X \leq 4/5 \\ -X/2 + 2/5, & \text{o/w} \end{cases}$
Rounding Top	$Y = [1/4 - (X - 1/2)^2]^{1/2}$
Rounding Bottom	Negated Y-axis of the Above
	Linear Patterns from Interpolation of Local Extrema
Head And Shoulders Top	(0, 0), (1/8, 1/8), (1/4, 0), (1/2, 1/4), (3/4, 0), (7/8, 1/8), (1, 0)
Head And Shoulders Bottom	Negated Y-axis of the Above
Broadening Top	(0.00, 0.00), (0.05, 0.10), (0.15, -0.10), (0.33, 0.20), (0.62, -0.20), (1.00, 0.30)
Broadening Bottom	Negated Y-axis of the Above
Triangle Top	(0.00, 0.00), (0.05, 0.10), (0.43, -0.30), (0.71, 0.00), (0.91, -0.20), (1.00, -0.10)
Triangle Bottom	Negated Y-axis of the Above
Double Top	(0, 0), (1/4, 1/4), (1/2, 0), (3/4, 1/4), (1, 0)
Double Bottom	Negated Y-axis of the Above
Falling Wedge	(0.00, 0.00), (0.05, 0.10), (0.35, -0.50), (0.53, -0.20), (0.70, -0.60), (0.82, -0.40), (0.94, -0.65), (1.00, -0.50)
Rising Wedge	Negated Y-axis of the Above
Descending Triangle	(0.00, 0.00), (0.05, 0.10), (0.43, -0.30), (0.71, 0.00), (0.91, -0.30), (1.00, -0.10)
Ascending Triangle	Negated Y-axis of the Above
Triple Top	(0.00, 0.00), (0.17, 0.17), (0.33, 0.00), (0.50, 0.17), (0.67, 0.00), (0.83, 0.17), (1.00, 0.00)
Triple Bottom	Negated Y-axis of the Above
Bullish Flag	(0.00, 0.00), (0.20, 0.20), (0.40, 0.10), (0.60, 0.20), (0.80, 0.10), (1.00, 0.20)
Bearish Flag	Negated Y-axis of the Above
Bullish Pennant	(0.00, 0.00), (0.33, 0.30), (0.60, 0.10), (0.80, 0.25), (0.93, 0.15), (1.00, 0.20)
Bearish Pennant	Negated Y-axis of the Above
Diamond Top	(0.00, 0.00), (0.12, 0.12), (0.25, 0.07), (0.38, 0.18), (0.50, 0.03), (0.62, 0.18), (0.75, 0.07), (0.88, 0.12), (1.00, 0.00)
Diamond Bottom	Negated Y-axis of the Above

Note: The X coordinates range from 0 to 1 for all patterns.

Table IA19: CNN Predictions on Simulated Charts

	Sign	20-Day Images		60-Day Images			Sign	20-Day Images		60-Day Images	
		Mean	Std	Mean	Std			Mean	Std	Mean	Std
Noise (Brownian motion)		48.6	(4.0)	50.7	(4.2)	Descending Triangle	−	42.1	(2.3)	46.5	(3.3)
Cup And Handle	+	46.7	(2.5)	51.3	(2.8)	Ascending Triangle	+	54.9	(2.0)	57.5	(2.8)
Head And Shoulders Top	−	56.3	(2.3)	55.5	(3.6)	Rounding Top	−	54.6	(1.9)	51.4	(2.5)
Head And Shoulders Bottom	+	49.3	(2.0)	54.8	(3.6)	Rounding Bottom	+	36.8	(1.8)	43.4	(2.8)
Broadening Top	−	45.7	(2.1)	52.4	(2.8)	Triple Top	−	62.0	(2.1)	60.3	(3.5)
Broadening Bottom	+	52.0	(2.6)	47.1	(3.0)	Triple Bottom	+	53.9	(2.0)	55.9	(3.7)
Triangle Top	−	44.6	(2.3)	48.7	(3.4)	Bearish Flag	−	55.8	(2.4)	56.4	(3.8)
Triangle Bottom	+	51.4	(2.2)	55.3	(3.0)	Bullish Flag	+	53.6	(2.1)	58.7	(3.4)
Double Top	−	55.6	(2.1)	56.0	(3.4)	Bearish Pennant	−	51.2	(2.5)	53.5	(3.7)
Double Bottom	+	50.1	(2.1)	54.3	(3.7)	Bullish Pennant	+	50.3	(2.1)	54.4	(3.2)
Rising Wedge	−	49.1	(1.9)	48.2	(1.9)	Diamond Top	−	59.3	(2.1)	62.4	(3.5)
Falling Wedge	+	40.0	(2.1)	41.6	(2.2)	Diamond Bottom	+	56.2	(2.0)	57.5	(3.6)

Note: This table reports average predicted probabilities of realizing a positive 20-day return on 10K simulated images with various patterns shown in Figure A3 and random noise (Brownian motion). The CNN models are I20/R20 and I60/R20 trained using images without volume bar or moving average lines. Average probabilities two standard error above or below 50% (random guess) are colored in green or red.

IA.3 Monte Carlo Simulations

This section conducts simulation experiments to investigate the finite sample performance of the CNN classifier. Return prediction is a rather challenging problem because the signal-to-noise ratio is notorious low. The first objective of our simulations is to demonstrate how CNN models recognize patterns and make correct predictions in environments with various signal-to-noise ratios. The second simulation exercise we conduct below demonstrates how transfer learning helps improve longer-horizon predictions by exploiting the self-similarity in the sample path of prices.

IA.3.1 Classification Accuracy vs Signal-to-Noise Ratio

To create images we simulate price trajectories at a 5-minute frequency, from which we obtain daily open, close, high, and low prices.

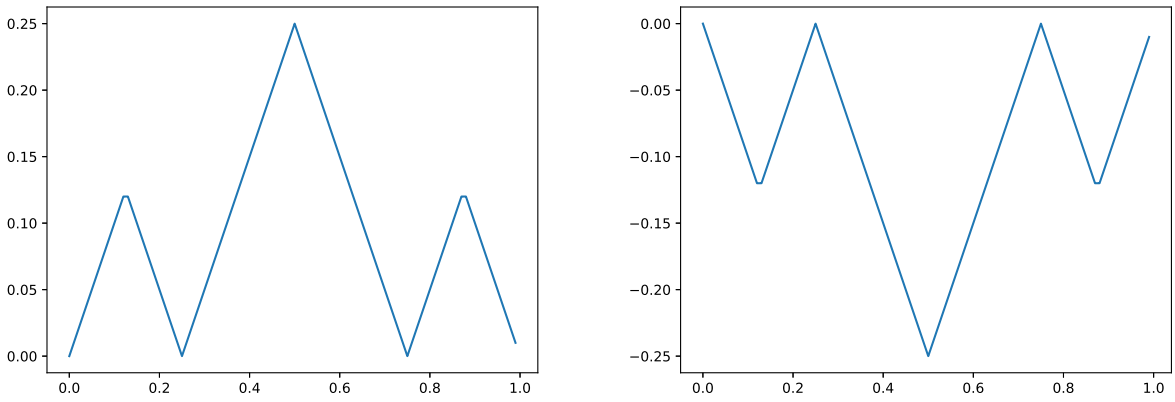
We start with two specific patterns, Head and Shoulders Top (HNST) and Head and Shoulders Bottom (HNSB). In technical analysis, HNST is believed to indicate short-term reversal from an upward trend into a downward trend, i.e. prices are likely to drop following HNST. In the other direction, HNSB is believed to predict a future rise in prices.

The functional form for HNST is:

$$f_{hnst}(x) = \begin{cases} x, \text{ if } 0 \leq x < 1/8 \\ -x + 1/4, \text{ if } 1/8 \leq x < 1/4 \\ x - 1/4, \text{ if } 1/4 \leq x < 1/2 \\ -x + 3/4, \text{ if } 1/2 \leq x < 3/4 \\ x - 3/4, \text{ if } 3/4 \leq x < 7/8 \\ -x + 1, \text{ o/w} \end{cases}, \tag{1}$$

where $x \in [0, 1]$. The functional form of HNSB negates the sign on the right hand side of equation 1. These patterns are shown in Figure IA6.

Figure IA6: Function Forms of HNST and HNSB Patterns



(a) HNST

(b) HNSB

To generate price trajectories with these patterns, we simulate the logarithm of the price as a mean-reverting process over $[0, T]$:

$$r^{pattern}_{t+k\Delta} := p^{pattern}_{t+k\Delta} - p^{pattern}_{t+(k-1)\Delta} = \kappa(\bar{r}_{t+k\Delta} - p^{pattern}_{t+(k-1)\Delta})\Delta + \sigma\sqrt{\Delta}\varepsilon_{t+k\Delta}, \quad k = 0, 1, 2, \ldots, 78, \tag{2}$$

where $p^{pattern}_0 = 0$, σ is drawn from $\mathcal{U}(0.2, 0.6)$ and fixed for each image but differ across images, Δ is the sampling frequency (fixed at every 5 minutes), $\varepsilon_{t+k\Delta} \sim \mathcal{N}(0, 1)$, $\kappa = 50/T$, $T = 20/252$ for 20-day samples (or $60/252$ for 60 days),[25] and $\bar{r}_{t+k\Delta}$ is the mean level on

[25]Because the HNST and HNSB patterns cannot be shown on a 5-day image, our first experiment focuses

timestamp $t + k\Delta$ that satisfies

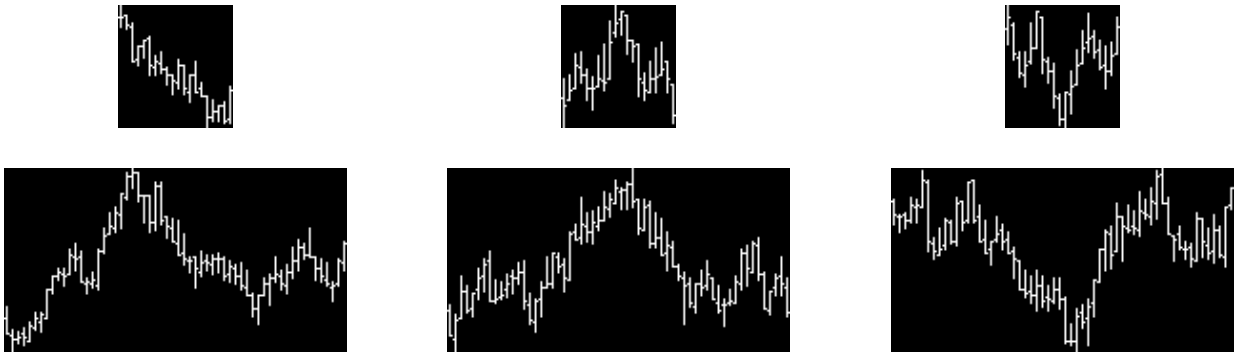
$$\bar{r}_{t+k\Delta} = 3\sigma\sqrt{T} f_{pattern}((t + k\Delta)/T).$$

For price trajectories without any patterns, we simply simulate a geometric Brownian motion without any drift:

$$r^{noise}_{t+k\Delta} := p^{noise}_{t+k\Delta} - p^{noise}_{t+(k-1)\Delta} = \sigma\sqrt{\Delta}\varepsilon_{t+k\Delta}, \quad k = 0, 1, 2, \ldots, 78. \tag{3}$$

Since p_t is the logarithm of price, $1 + p^{noise}_{t+k\Delta} = \sum_{s=0}^{t+k\Delta} r_s$ produces cumulative returns that appear in the image. See Figure IA7 for examples of 20-day and 60-day images with and without HNS patterns.

Figure IA7: Examples of Simulated Images



(a) Pure Noise Image (b) An HNST Image (c) An HNSB Image

Note: For each subfigure, at the top is the simulated image for 20 days and at the bottom is for 60 days.

We first conduct experiments in which only one pattern (HNST or HNSB) exists in the data. We then consider experiments with both patterns included so as to check whether a CNN model can recognize multiple co-existing patterns.

We conduct 100 independent experiments. In each experiment, we generate 100K images in total. To control the signal-to-noise ratio in our prediction exercise, we select a small percentage, say 1%, of images with either one or both patterns, and the rest purely noise images. We assign "up" labels for HNST images, "down" for HNSB ones, and random "up" or "down" labels to noise images (with 50% probability). For example, if there are 99% noise images and 1% HNST patterned images, the actual probability of an "up" move is $(99/2 + 1)\% = 50.5\%$. In the case both HNSB and HNST are included in the data, each patten accounts for half of the patterned images, so the (unconditional) probability of an

on 20-day and 60-day images.

"up" move in the data is 50%. As the portion of noise images increases, the signal-to-noise ratio diminishes.

We maintain the same architecture design of the CNN models as in our empirical study, including 3 layers for 20-day images and 4 for 60-day ones, doubling the number of channels from bottom to top, except that we decrease the number of initial channels from 64 to 8 (and so on for subsequent layers), because our simulation setting is more stylized. The training algorithm is identical to what we use empirically, except that we increase the initial learning rate to 10^{-4} to speed up the process. Throughout all experiments, we divide all images proportionally to form the training, validation, and testing datasets, which are split according to $6:2:2$ ratios.

Figure IA8 plots the average prediction accuracy on patterned images and noise images on the testing sets, respectively, against the portion of noise images in the data. We find that for both 20-day and 60-day models, the prediction accuracy is always indistinguishable from a random guess (0.5) for noise images and reaches nearly 100% on HNST and HNSB images as long as the portion of noise images drops below 99.0%, regardless of whether only a single pattern (top panel) or both patterns (bottom panel) are included in the datasets.

This suggests that the CNN model achieves nearly perfect recognition power on patterned images while remaining indifferent to noise ones, even though the whole dataset consists only 1% images with patterns. When only a single pattern HNST is included in the training set, with a 99.8% portion of noise images, the predictive accuracy is still as high as 0.88 for the 20-day model and 0.81 for the 60-day model. In comparison, models trained with both HNST and HNSB images (Figure IA8b) perform slightly worse than models trained with HNST images alone (Figure IA8a). This is not surprising because the number of patterned images is halved for either group in the former case.

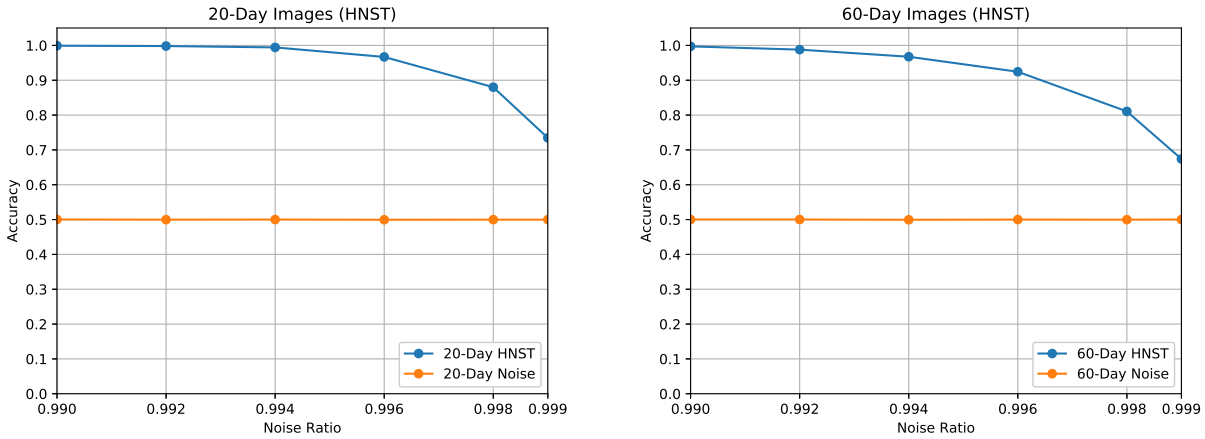
IA.3.2 Self-Similarity and Transfer Learning

We now design the second experiment to demonstrate the use of transfer learning for long-horizon prediction. We simulate a rounding bottom pattern, shown in Figure IA9 for both 5-day and 60-day images. We use this pattern because it is relatively simple and visible even from five OHLC units.

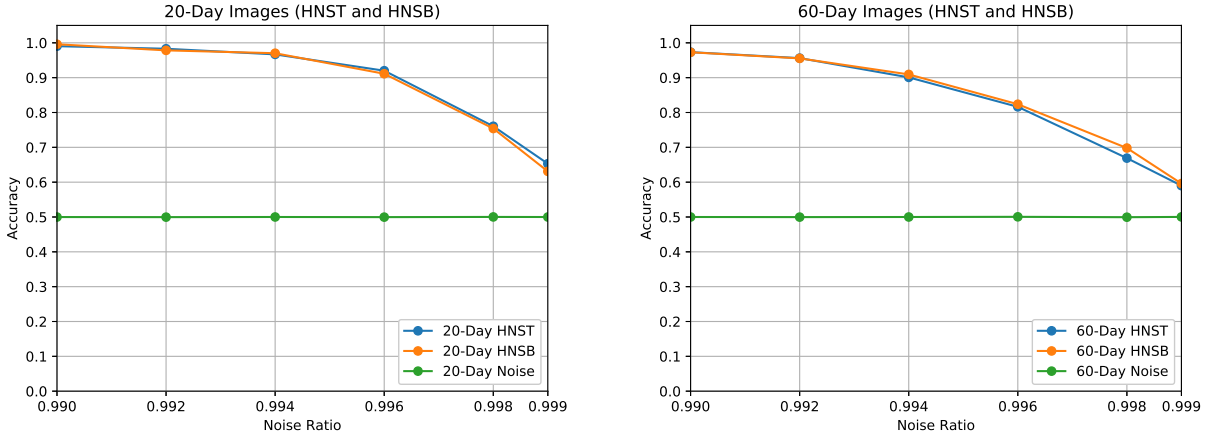
Similar to the previous experiment, the portion of patterned images controls the signal-to-noise ratio. In total, we generate 10K 60-day images and 120K 5-day images, to reflect their relative sample sizes in practice. We train our CNN model using only 5-day images, and then directly apply this model for prediction based on transformed 60-day images.

The transformed 60-day images comprise of five OHLC units, each representing the open, high, low, close prices within a 12-day subwindow. Figure IA10 provides an example of 5-

Figure IA8: Accuracy vs Noise Ratio in Simulation



(a) Predictive accuracy for models trained with dataset that includes only HNST



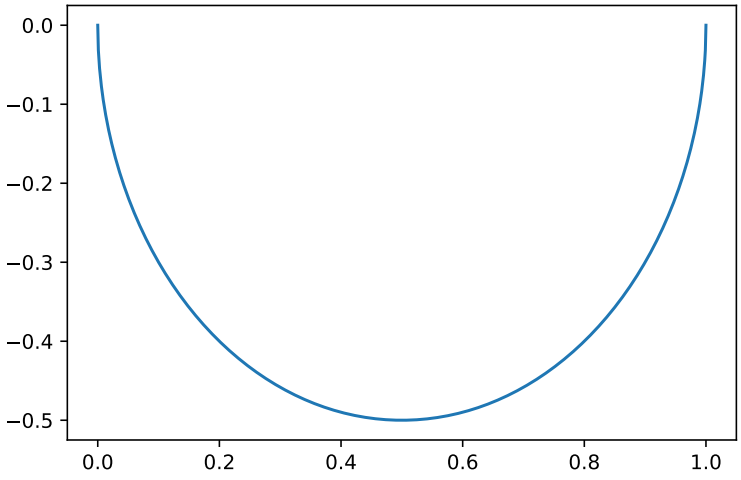
(b) Predictive accuracy for models trained with dataset that includes both HNST and HNSB

Note: For each subfigure, the left plot shows the simulation result for the 20-day model and the right plot shows that for the 60-day model. The x-axis is the noise ratio in the training set. The accuracy obtained from the OOS testing set that consists of all noise images, all HNST images, or all HNSB images. The number of Monte Carlo repetitions is 100.

day image, a 60-day image and its transformed version, showing that the transformed 60-day image resembles the original 5-day image closely.

Figure IA11 compares the prediction accuracy based on 1K testing sample for four scenarios. The first case is testing result of our 5-day CNN trained with 5-day images; the second is result of our 60-day CNN trained with 60-day images (benchmark); the third is the result of a second 5-unit CNN model retrained with transformed 60-day images (retrain); and the last

Figure IA9: Function Form of the Rounding Bottom Pattern



Note: The mathematical formula that describes the rounding bottom pattern is given by $f(x) = -\sqrt{1/4 - (x - 1/2)^2}$, for $x \in [0, 1]$.

Figure IA10: Examples of Simulated Images with the Rounding Bottom Pattern

(a) 60-day Image

(b) 5-day Image

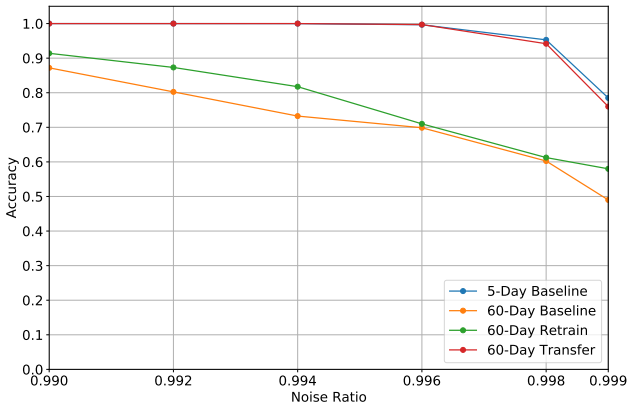
(c) Transformed 60-day Image

Note: Subfigures (a) and (b) are 60-day and 5-day images a with rounding bottom pattern, respectively. Subfigure (c) is the transformed 60-day image (a), where each OHLC bar report open, high, low, and close prices over a 12-day subwindow.

is based on the 5-day model directly applied to transformed 60-day images (transfer).

Figure IA11 shows that with a successfully trained 5-day CNN model (blue), transfer learning (red) inherits its accuracy and outperforms the benchmark (orange) and the retrained result (green). The advantage of transfer learning stems from the fact that the transferred model is learned from a sample 12-times larger which also features similar patterns. This experiment demonstrates that transfer learning holds promise for long-horizon prediction for which the sample size may be restrictive.

Figure IA11: Accuracy vs Noise Ratio in Transfer Learning Simulation



Note: "5-day Baseline" corresponds to the 5-day model trained with 120K 5-day images; "60-day Baseline" corresponds to the 60-day model trained with 10K 60-day images; "60-day Retrain" shows the prediction performance of a 5-unit CNN model trained from scratch with 10K transformed 60-day images; "60-day Transfer" corresponds to directly applying the 5-day baseline model on the transformed 60-day images. The accuracy is measured by 1K out-of-sample images simulated with a rounding bottom pattern.