The Journal of FINANCE
The Journal of THE AMERICAN FINANCE ASSOCIATION

THE JOURNAL OF FINANCE • VOL. LXXVIII, NO. 6 • DECEMBER 2023

(Re-)Imag(in)ing Price Trends

JINGWEN JIANG, BRYAN KELLY, and DACHENG XIU*

ABSTRACT

We reconsider trend-based predictability by employing flexible learning methods to identify price patterns that are highly predictive of returns, as opposed to testing predefined patterns like momentum or reversal. Our predictor data are stock-level price charts, allowing us to extract the most predictive price patterns using machine learning image analysis techniques. These patterns differ significantly from commonly analyzed trend signals, yield more accurate return predictions, enable more profitable investment strategies, and demonstrate robustness across specifications. Remarkably, they exhibit context independence, as short-term patterns perform well on longer time scales, and patterns learned from U.S. stocks prove effective in international markets.

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, Mamaysky, and Wang (2000)

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

*Jingwen Jiang is with University of Chicago. Bryan Kelly is with Yale University, AQR Capital Management, and NBER. Dacheng Xiu is with University of Chicago. 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. We have read *The Journal of Finance*'s disclosure policy and have no conflicts of interest to disclose.

Correspondence: Bryan Kelly, School of Management, Yale University, 165 Whitney Ave., New Haven, CT 06511; e-mail: bryan.kelly@yale.edu

DOI: 10.1111/jofi.13268

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 the 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 that we can interpret those patterns to inform future theory.[2] To negotiate this delicate trade-off, we make two research design choices that together result in successful inference of new return-predictive patterns in past prices: We represent historical prices as an image, and we model the predictive association between images and future returns using a convolutional neural network (CNN). These choices 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 nonlinear transformations of the original numeric pixel values. At a high level, this is similar to more traditional kernel-based data filters used in regression analysis. However, the CNN does not simply fix the set of smoothing filters: Instead, it 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, that is, 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

[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.

time-series methods. This is the same basic rationale 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 doing so as well. Third, the process of imaging price and volume data converts all assets' data histories into a comparable scale. We show that this particular rescaling 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 U.S. stock returns from 1993 to 2019. We train a panel CNN model to predict the direction (up or down) of future stock returns. Specifically, 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 (five-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 High-Low 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), one-month short-term reversal (STR), one-week short-term reversal (WSTR), and the Han, Zhou, and Zhu (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 (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 next 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 to 5 after rebalance versus returns over days 6 to 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 to 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 who 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 nonlinear 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, 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 the main data set of daily U.S. 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 U.S. stock data transfer well to international markets and to other time scales. International markets have fewer stocks and shorter time-series compared to the United States. We transfer CNN model estimates from U.S. data to construct return forecasts in 26 foreign markets. We find that international image-based trading strategies earn a *higher* Sharpe ratio than if we train a CNN from scratch using local market data. In addition, given the scarcity

of low-frequency data in financial markets, it would be 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 five-day returns from images of five-day prior price data. We then apply 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.

The theoretical sensibility and empirical reliability of technical analysis have long been a subject of debate. Lo, Mamaysky, and Wang (2000) and Lo and Hasanhodzic (2009) 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, Easley, and O'Hara (1994), Barberis, Shleifer, and Vishny (1998), and Han, Zhou, and Zhu (2016). Several papers find strong empirical support for technical trading rules, with prominent examples including Brock, Lakonishok, and LeBaron (1992), Lo, Mamaysky, and Wang (2000), Zhu and Zhou (2009), Neely et al. (2014), Han, Zhou, and Zhu (2016), Detzel et al. (2020), and Murray, Xiao, and Xia (2021). Sullivan, Timmermann, and White (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 postpublication 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 differentiating feature of our approach is that we do not require the researcher to prespecify a set of technical patterns. Instead, we 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, Mamaysky, and Wang (2000), but with a retooled research design benefitting from 20 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 MOM and STR.

An emerging literature in computer science 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

[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 which also predicts the emergence of long-range dependence in prices (Cont (2005)).

performs time-series prediction of aggregate stock indices (examples include Chen et al. (2016), Hoseinzade and Haratizadeh (2019), Kim and Kim (2019), Lee, Kim, Koh, and Kang (2019)). Hu et al. (2018) use price plot CNNs to cluster individual stocks, while Cohen, Balch, and Veloso (2020) classify images with specific technical patterns (crossings of Bollinger Bands, MACD, Relative Strength Index). These authors do not predict returns. To our knowledge, no prior paper to date 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. In Section I, we describe the process of "imaging" market data. Section II presents the intuition and detailed mechanics of our CNN framework. We report our main empirical analysis in Section III. Section IV explores interpretations of the CNN model. In Section V, we analyze the benefits of CNN transfer learning across geographies and time scales. Section VI concludes. In the Internet Appendix, we report a variety of extensions and robustness tests of our main empirical analysis, as well as simulation evidence to illustrate the model's finite-sample properties.[4]

I. "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 by providing 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 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.

A. *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). 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,

[4] The Internet Appendix is available in the online version of the article on *The Journal of Finance* website.

Figure 1. Tesla OHLC chart from Yahoo! Finance. This figure displays an OHLC chart for Tesla stock with 20-day moving average price line and daily volume bars. Data are daily from January 1, 2020, to August 18, 2020. (Color figure can be viewed at wileyonlinelibrary.com)

and quarterly price trajectories, respectively). The width of an n-day image is therefore $3n$ 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 IA.2, Panel A.

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 nonlinear process. This type of nonlinearity would be difficult for traditional kernel methods to discern from time-series data.

We exclude images for stocks that either undergo an Initial Public Offering or are delisted during the data window, but we allow missing data if they occur in the middle of the stock's history. In the case of 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).[5]

We use black as the background color and white 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 is common practice by Bloomberg and others, is redundant because the direction of the price change is implied from the opening and closing price marks. Omitting such redundancy allows us to focus on two-dimensional pixel matrices, rather than having to track a third dimension for RGB pixel intensities.

B. 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.

[5] 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.

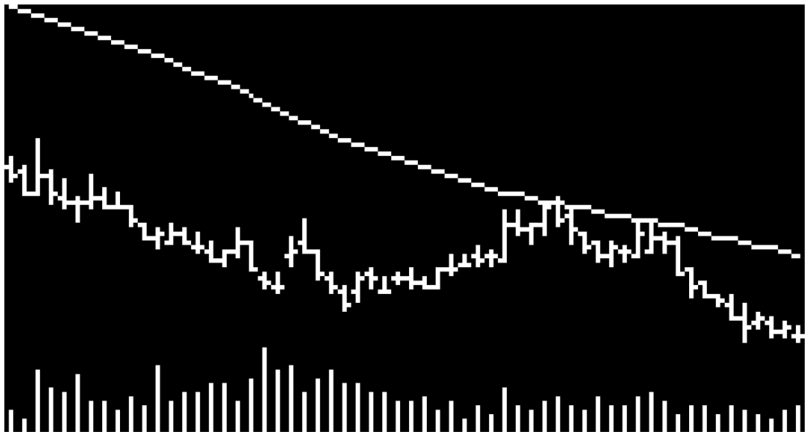

Figure 2. Generated OHLC images with volume bar and moving average line. This figure displays market data image for 60 days of data.

II. The CNN Model

In this section, we briefly describe the architecture of our CNN model specification and training approach.[6] 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.

A. A Brief Description of the CNN Architecture

Each image in our data set 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, 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. The CNN imposes 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

[6] Chapter 9 in Goodfellow, Bengio, and Courville (2016) provides a textbook introduction to CNN.

comparatively small data sets, and it embeds a number of tools that make the model resilient to deformation and repositioning of important objects in the image.

The CNN modeling scheme stacks together a sequence of operations to transform a raw image into a set of predictive features and, ultimately, a prediction. Such schemes 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 nonlinear 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 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 of each model is shown in Figure 3. The Appendix provides further details about this CNN structure along with the intuition behind each of its components.

B. Image Representation versus Time-Series Representation

To 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 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 rows corresponding to time and columns to variables. Convolutional filters in this setting have the same width as the data matrix, and the filter moves only 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 nonlinear spatial associations among various price curves. Figure 4 illustrates 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

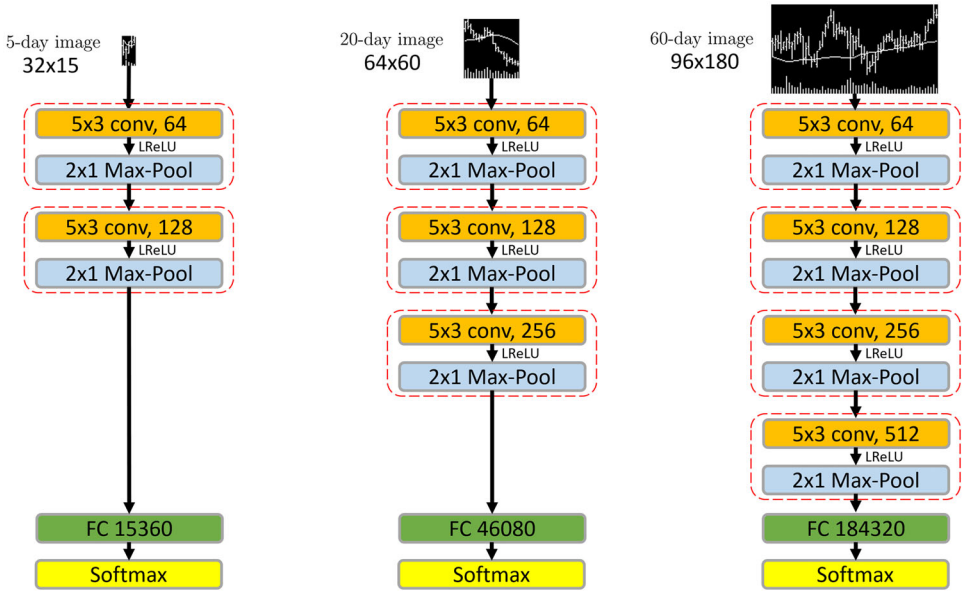

Figure 3. Diagram of CNN models. This figure displays diagrams of a five-day (left), 20-day (middle), and 60-day (right) CNN model. The 5/20/60-day CNN models are built with 2/3/4 CNN building blocks as described in Figure A2. The notation $H \times W$ shows the output size of the CNN building block, where H is the height and W the width. The number following "conv" or "Max-Pool" is the depth (number of channels, for example, 64, 128, 256, and 512). The output of the last CNN block is flattened to a vector and fed to a fully connected (FC) layer, 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." (Color figure can be viewed at wileyonlinelibrary.com)

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 nonlinear activation function. If a price increase of two units is more than twice as predictive as a price increase of one unit, the 1D kernel is unable to account for it. A 2D CNN applied to the price plot image, however, can, with 3×2 filters that treat "no change," "small increase," and "large increase" as separate features, 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 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. Moreover, incorporating volatility information would inevitably require nonlinear 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

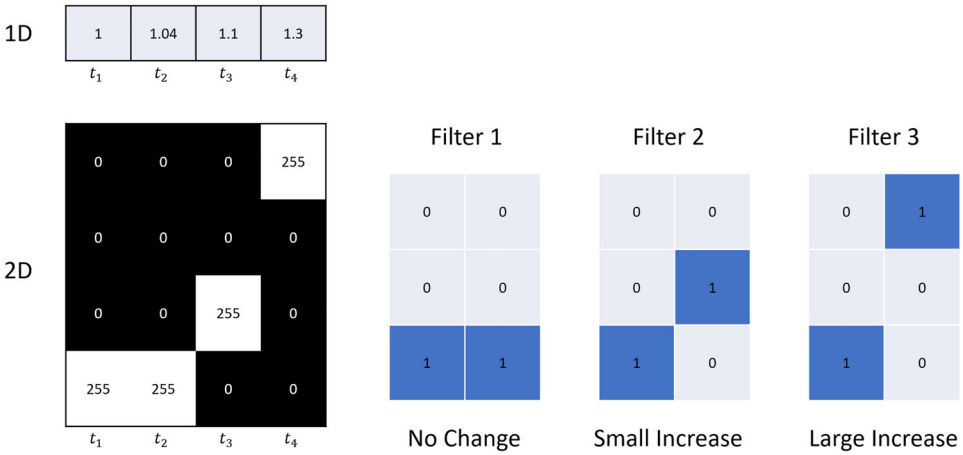

Figure 4. Convolutional filters that detect the next day's price change. The "1D" figure on the left displays a snippet of time-series prices. The "2D" figure on the left translates this time-series into a black and white image where "255" represents a white pixel (corresponding to a price entry) and "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 displays three 3×2 convolutional filters that detect no change, a small increase, and a large increase in price, respectively. (Color figure can be viewed at wileyonlinelibrary.com)

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.[7]

C. Training the CNN

Our workflow from training, to model tuning, and finally to prediction follows the basic procedure outlined by Gu, Kelly, and Xiu (2020). First, we divide the entire sample into training, validation, and testing samples. In our main U.S. data sample, we estimate and validate the model using a single eight-year sample (1993 to 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

[7] 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, Long Short-term Memory networks (LSTM), may outperform the CNN. Also, our objective here is not to find the best return prediction model, an unrealistic task in any case. Rather, our empirical analysis provides, at best, a lower bound on the extent of predictability that machine learning models can achieve.

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 19 years (2001 to 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 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, Kelly, and Xiu (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 (Kingma and Ba (2014)) 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 nonlinear activation within each building block to reduce covariate shift.[8] 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, Kelly, and Xiu (2020) outline the intuition behind these choices, so for the sake of brevity, we omit this discussion and refer interested readers there.

In Section IV of the Internet Appendix, we report results of simulation experiments that 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 insights into the effectiveness of a CNN in this new environment. The general conclusion from the simulations is that the CNN successfully detects complicated technical patterns in realistically low signal-to-noise data sets.

[8] 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, although skipping this step has negligible impact on our results.

III. CNN Prediction for U.S. Stock Returns

A. Data

We use daily stock data from CRSP for all firms listed on NYSE, AMEX, and NASDAQ. Our sample runs from 1993 to 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 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, and 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 nonpositive 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 retrain the CNN five times and average the forecasts (following Gu, Kelly, and Xiu (2020)).

Two important considerations when reading our empirical results are that we do not recursively retrain the model and 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 is 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 due primarily 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 the 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 (5, 20, or 60 days following the last date in an image). Throughout we use the notation "Ix/Ry" to indicate that the model uses x-day images to predict subsequent y-day holding-period returns.

B. 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. We therefore begin our discussion focusing on one weekly return prediction.

Table I reports the predictive strength of the CNN model when it targets five-day-ahead returns. We couch this predictive strength in economic terms

Table I

Short-Horizon (One-Week) Portfolio Performance

This table reports results on the performance of equal-weight (top panel) and value-weight (bottom panel) decile portfolios sorted on out-of-sample predicted "up" probability. Each panel reports the average holding-period return and annualized Sharpe ratio. ***,**, and * indicate significant at the 1%, 5%, and 10% significance level, respectively. We also report the monthly turnover of each strategy.

	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%	

(*Continued*)

Table I—*Continued*

	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%	

by reporting performance of portfolios sorted on CNN forecasts. The table focuses on annualized average returns and Sharpe ratios for five-day holding-period returns. The top panel reports equal-weight decile portfolios. Decile 1, which corresponds to stocks having the lowest probability of a positive future return, realizes 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-, 20-, 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.[9] In value-weight portfolios (bottom panel), CNN strategies earn Sharpe ratios ranging from 1.4 to 1.7, double the 0.8 Sharpe ratio for the value-weight WSTR strategy (which is the best value-weight performer of the benchmarks).[10] CNN 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 insights 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 five-day images, denoted "I5/R5"), averaged first within deciles in each period and then over time (i.e., average returns of equal-weight decile portfolios). To illustrate the precision of the 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, although with a somewhat flatter slope. An interesting difference between CNN forecasts and other benchmark price trend signals is in their variability. All CNN decile portfolios have annualized volatility below 20%, while 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 I, we report the fraction of the strategy that turns over on average scaled in monthly terms. Following Gu, Kelly, and Xiu (2020), we

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

[10] Internet Appendix Table IA.II studies the possibility that CNN models trained with one supervising return horizon are helpful for predicting different return horizons out of sample. Indeed, we see that no single image length, and no single supervision window, 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.

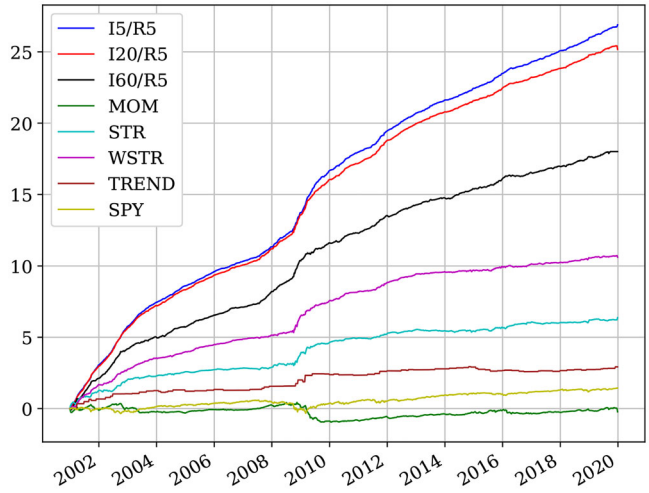

Figure 5. Cumulative volatility-adjusted returns of equal-weight portfolios. This figure displays cumulative volatility-adjusted log returns of equal-weight long-short portfolios of I5/R5, I20/R5, I60/R5, MOM, STR, WSTR, and TREND. All strategies are rescaled to have the same volatility as that of SPY over the test sample. (Color figure can be viewed at wileyonlinelibrary.com)

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 one-third or scaling up weekly strategy turnover by a factor of four. 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%.

While Table I 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.

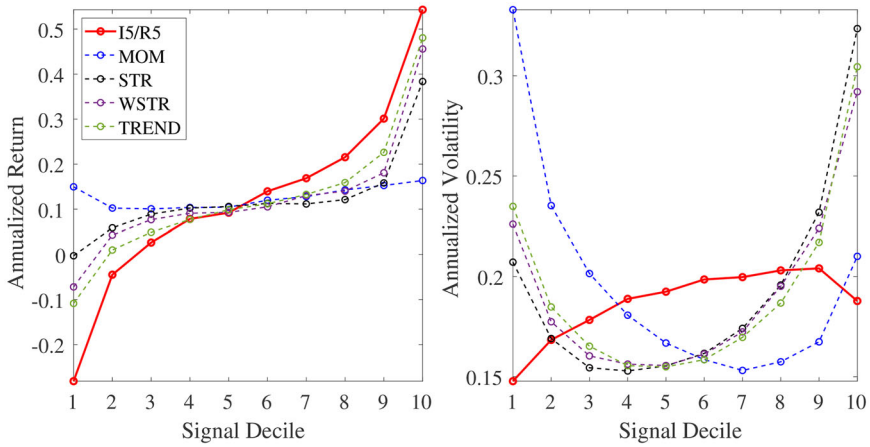

Figure 6. Prediction accuracy by decile. The left panel plots average realized returns in each decile of I5/R5 CNN model forecasts and for each decile of the benchmark signals. Return averages are based on cross-sectional decile averages in each period that are then averaged over time. The right panel plots the time-series volatility of decile returns. (Color figure can be viewed at wileyonlinelibrary.com)

C. 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 II 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 II 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 an 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

Table II

Performance of Monthly and Quarterly Strategies

This table reports results on the performance of equal-weight decile portfolios sorted on out-of-sample predicted "up" probability. Each panel reports the average annualized holding-period return and Sharpe ratio. ***,**, and * indicate significant at the 1%, 5%, and 10% significance level, respectively. We also report monthly turnover of each strategy.

	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%	

(*Continued*)

Table II—*Continued*

	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%	

(Sharpe ratio of 0.1).[11] 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 whether images can help predict returns beyond the first week. Table III decomposes the performance of the monthly rebalanced CNN strategy (and its competitors) into the return from days 1 to 5 after rebalancing (top panel) and the return from days 6 to 20 (bottom panel), focusing on CNN models supervised with 20-day returns (I*x*/R20). While the bulk of the image-based strategy performance does indeed come from the first week, a significant portion also obtains *after* the first week of trading. The annualized Sharpe ratios over days 6 to 20 are 0.4, 1.2, and 0.8 for 5-, 20-, and 60-day image models, respectively. CNN strategy mean returns over days 6 to 20 are all significant at the 10% level or better.

D. The Effect of Stock Size and Trading Costs

Table I shows that Sharpe ratios of image-based strategies more than double when using equal-weight deciles versus value-weights. However, as Jensen, Kelly, and Pedersen (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 all weight to small stocks).

In Table IV, 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 IA.IV, we show that the trading profits in Table I survive standard trading cost adjustments (10 basis points for stocks exceeding the 80th size percentile of the NYSE, and 20 basis points otherwise: see Frazzini, Israel, and Moskowitz (2018), Ke, Kelly, and Xiu (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.

[11] Internet Appendix Table IA.III reports monthly and quarterly results for value-weight decile portfolios.

Table III

R20 Equal-Weight Portfolio Performance Breakdown

This table provides breakdown of performance of equal-weight decile portfolios sorted on out-of-sample predicted "up" probability. Each panel reports the average holding-period return and annualized Sharpe ratios. ***,**, and * indicate significant at the 1%, 5%, and 10% significance level, respectively. We also report monthly turnover of each strategy.

	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%	

(*Continued*)

Table III—*Continued*

	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%	

Table IV

Portfolio Performance Restricted to 500 Largest Stocks

This table reports results on the performance of equal-weight (top panel) and value-weight (bottom panel) decile portfolios sorted on out-of-sample predicted "up" probability. Each panel reports the average holding-period return and annualized Sharpe ratios. ***,**, and * indicate significant at the 1%, 5%, and 10% significance level, respectively. We also report monthly turnover of each strategy.

	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%	

(*Continued*)

Table IV—*Continued*

	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%	

E. Robustness

Section I of the Internet Appendix 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 therefore perform sensitivity 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., using different dropout or batch normalization schemes). Finally, we compare CNN models to alternative computer vision models including "HOG" and "HAAR" (descriptions of these models are also included in the Internet Appendix).

IV. What Does the CNN Learn?

Interpreting a CNN is difficult due to its recursive nonlinear 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 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). Notwithstanding, they achieve partial success by offering some insight into the complex inner workings of the CNN model.

A. Association with Other Predictors

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).[12] Table V 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 a correlation of −26% to −34% for models supervised by future five-day returns, consistent with the CNN picking up on a weekly STR pattern. The WSTR correlation drops close to zero when CNN models are supervised by 60-day returns. Similarly, 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 nonprice firm characteristics. Other strong

[12] For variable definitions and references, see table A.6 of Gu, Kelly, and Xiu (2020).

Table V

Correlation between CNN Predictions and Stock Characteristics

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

CNN Model	MOM	STR	WSTR	TREND	Beta	Volat.	Bid-52WH	Dollar Ask	Zero Volume	Price Trade	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

Table VI
CNN Predictions and Standard Stock Characteristics

This table reports slope coefficients and R^2s 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. ***,**, and * indicate significant at the 1%, 5%, and 10% significance level, respectively, using Newey-West standard errors.

	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

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 are 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 VI shows the joint explanatory power for image-based predictions from all characteristics simultaneously (this table focuses on our main specification using five-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 VII 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

Table VII

CNN, Future Returns, and Standard Stock Characteristics

This 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. ***,**, and * indicate 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 the stock-level in-sample mean as the predictor (To ensure high-quality estimates of in-sample mean for each stock, we require a stock to have at least three years of data (12 observations for quarterly returns). For stocks that fail to meet this criterion, we instead use the overall in-sample mean across all stocks. The same adjustment applies to Table VIII.).

	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

indicator for a positive five-day realized return. We estimate 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 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 specification. 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.

B. Logistic Approximation

What do the CNN models detect in images to produce accurate forecasts that are distinct 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 therefore 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 intraday high and low prices, open prices, price moving averages, and volumes to make them comparable in the cross section.

To make logistic regressions comparable to 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 five-day images. Five-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 five-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 VIII pursue a logistic approximation to the CNN forecast itself, as in Table VI. The dependent variable is the out-of-sample forecast generated by the CNN model for five-day images (one if the CNN predicted probability is greater than 0.5, and zero otherwise), and the independent variables are data underlying five-day images rescaled 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 a consistent sign and magnitude for all return horizons. Compared to the first lag, lags 2 through 5 have a small role in the CNN forecast, although some of these are statistically significant. The regression-based approximation to the true nonlinear CNN construction explains 22% to 35% of the variation in image-based predictions. But CNN predictions are built only from the raw market data. Therefore, more than half of the variation in image-based predictions is attributable to nonlinear functions of the underlying market data.

Next, we use the logistic specification to directly forecast returns with the approach used in Table VII. Regressions are grouped by supervising the 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. Taken 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 as those isolated by the logistic model. Again, however, the approximating pattern explains less than half of the content in CNN predictions.

Columns (6), (9), and (12) compare the CNN's nonlinear predictions with those 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

Table VIII

Logistic Regressions Using Market Data with Image Scaling

This table reports 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. * indicates significant at the 1% significance level using Newey-West standard errors. McFadden R^2s are associated with regressions using test sample data. To facilitate comparison of predictive performance, we report out-of-sample McFadden R^2s, which compare the cross-entropy loss of a model (estimated in the training sample) against that of a benchmark model using the stock-level in-sample mean as the predictor (As ma lag 1 equals the simple average of close lag 1 to 5 by definition, we remove close lag 5 in the regression to avoid multicollinearity.).

	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		1.99	2.30		2.59
open lag 1	−0.58*	−0.00	0.28*		−0.25	−0.16		−0.13	−0.11		0.12	0.09
open lag 2	0.27*	0.54*	0.22*		0.08	0.05		0.13	0.09		−0.10	−0.13
open lag 3	0.15*	−0.29*	−0.09*		−0.08	−0.10		0.00	0.02		0.14	0.15
open lag 4	0.16*	0.01	0.04*		−0.08	−0.09		−0.06	−0.04		0.14	0.14
open lag 5	0.00	0.10*	0.37*		0.03	0.01		−0.00	−0.01		−0.25	−0.26
high lag 1	2.65*	1.00*	1.08*		0.39	0.09		0.17	0.08		0.07	−0.08
high lag 2	−0.26*	0.47*	0.77*		0.05	0.08		0.05	0.01		−0.12	−0.21
high lag 3	−0.42*	−0.18*	−0.20*		0.12	0.16		−0.00	0.02		0.06	0.09
high lag 4	−0.45*	−0.12*	−0.20*		0.06	0.10		0.23	0.23		0.01	0.05
high lag 5	−0.04*	0.41*	−0.31*		0.01	0.03		−0.22	−0.24		0.02	0.02
low lag 1	1.90*	0.94*	−0.06*		0.48	0.20		0.40	0.30		0.21	0.20
low lag 2	−0.16*	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	0.30*	0.43*	−0.12*		0.03	0.01		0.15	0.12		−0.17	−0.17
low lag 5	−0.04*	−0.04*	−0.31*		0.09	0.12		−0.21	−0.20		−0.17	−0.11
close lag 1	−5.50*	−4.06*	−2.30*		−0.71	−0.16		−1.01	−0.73		−0.63	−0.38
close lag 2	0.54*	0.13*	0.06*		−0.06	−0.08		−0.49	−0.49		−0.04	−0.07
close lag 3	0.46*	0.62*	0.14*		−0.12	−0.12		−0.36	−0.38		−0.05	−0.07
close lag 4	0.55*	0.09*	0.06*		−0.19	−0.19		−0.50	−0.49		−0.05	−0.05

(*Continued*)

Table VIII—*Continued*

	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)
ma lag 1	−0.95*	−0.66*	−0.28*		0.34	0.28		2.50	2.50		1.47	1.69
ma lag 2	0.44*	0.50*	0.11*		−0.28	−0.29		−0.34	−0.38		−0.65	−0.80
ma lag 3	0.03*	−0.69*	0.24*		−0.07	−0.10		−0.56	−0.55		−0.07	−0.14
ma lag 4	−0.03	0.33*	0.63*		0.49	0.44		0.42	0.46		−0.35	−0.39
ma lag 5	0.23*	−0.46*	−0.61*		−0.22	−0.19		−0.00	0.01		0.60	0.65
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
McFadden R^2	35.16	33.15	21.86									
OOS McFadden R^2				0.73	0.55	0.73	0.47	0.47	0.46	1.33	1.39	1.24

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. These results indicate that the nonlinear 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 IX compares the performance of long-short decile spread portfolios formed on CNN forecasts versus the linear approximation via logistic regression. These results are reported in the rows labeled "Logistic (image scale)." We find that the linear approximation to the CNN delivers a successful trading strategy, but is generally inferior to the full nonlinear 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 following 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 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 5, 20, or 60 days ago) and likewise for volume. We refer to this scaling 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 labeled "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 scaling 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 five-day price histories) dominate the best-performing logistic model with devolatized return scale by a large margin.

Table IX

Portfolio Performance of CNN versus Logistic Model and CNN1D

This 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.

	Five-Day Images		20-Day Images		60-Day Images	
	Equal-Weight	Value-Weight	Equal-Weight	Value-Weight	Equal-Weight	Value-Weight
Five-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	4.38	1.59
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	1.73	0.71
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	0.38	0.13
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

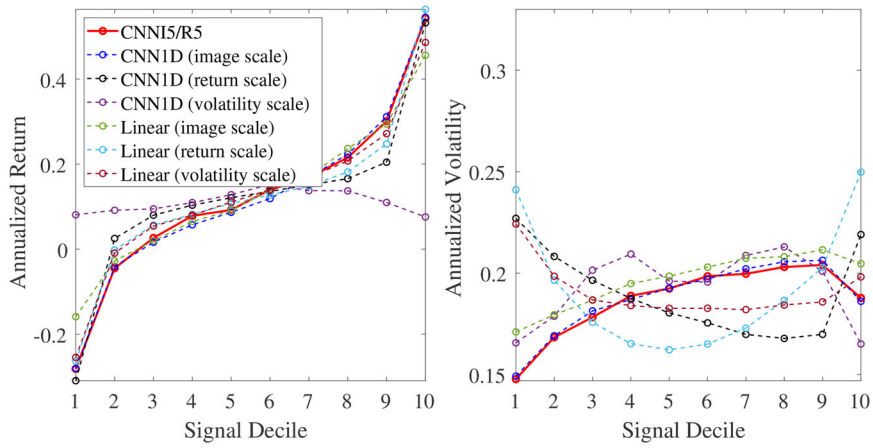

Figure 7. Prediction accuracy of CNN and logistic models by decile. The left panel plots average realized returns in each decile of I5/R5 CNN model forecasts, as well as for each decile of forecasts from 1D CNN models and logistic models. Return averages are based on cross-sectional decile averages each period that are then averaged over time. The right panel plots the time-series volatility of decile returns. (Color figure can be viewed at wileyonlinelibrary.com)

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)" compared to alternative data formulations. Two interesting observations 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. However, this is not because the 1D CNN is generally superior to the 2D CNN for return prediction. We see that "CNN1D (cum. ret. scale)" and "CNN1D (devol. ret. scale)" broadly underperform the baseline 2D CNN, as well as 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 are 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 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 at the same time, they tend to produce U shapes in volatility. Instead, it appears that the CNN decile volatility pattern is due primarily to image scaling of the raw data.

C. Traditional Technical Analysis

Brock, Lakonishok, and LeBaron (1992) find that 26 predefined (and commonly used) technical trading rules generate significantly positive investment performance. Lo, Mamaysky, and Wang (2000) introduce a kernel regression approach to identify technical indicators and find further support for the positive performance of investment strategies that rely on technical analysis.

Sullivan, Timmermann, and White (1999) assess a universe of 7,846 predefined technical trading rules while carefully controlling for multiple testing to control the likelihood of false positives. While they find that none of the Brock, Lakonishok, and LeBaron (1992) strategies delivers significant positive performance in the postpublication 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 is significant after controlling the false discovery rate.

The Sullivan, Timmermann, and White (1999) universe of 7,846 technical rules offers an excellent opportunity to calibrate the significance of our results. These rules constitute a null distribution for benchmarking other technical analysis strategies. To use these benchmarks, we apply each of the 7,846 trading rules stock-by-stock to produce a stock-level signal[13] and then construct decile spread long-short strategies in the same way 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 performance distribution for 7,846 different strategies against which we compare the CNN strategy.

Figure 8 reports histograms 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 technical signals outperform the CNN strategy (regardless of weighting scheme), with the closest technical indicator outperformed by a large margin. At the monthly horizon, again none of the technical rules exceeds 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 technical strategy universe, so in this sense the exceedance rate is more conservative than a p-value. Second,

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

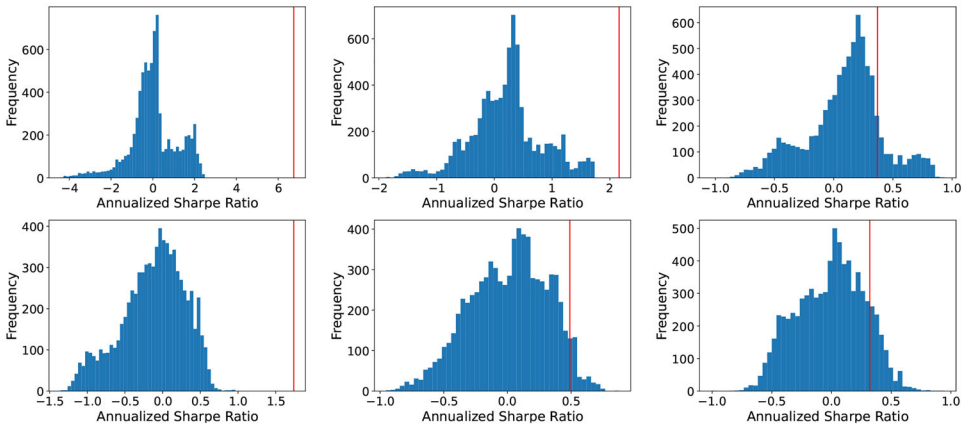

Figure 8. Comparison with traditional technical indicators. The figure shows histograms of Sharpe ratios for 7,846 technical analysis portfolios that rebalance by week, month, and quarter, respectively. The top panels show equal-weight portfolios and the bottom panels show value-weight portfolios. Red vertical bars are the Sharpe ratios for the I20/R5, I20/R20, and I20/R60 CNN strategies, respectively. (Color figure can be viewed at wileyonlinelibrary.com)

the technical indicator literature focuses only on very short holding-periods, typically a single day. The fact that a nontrivial proportion of the Sullivan, Timmermann, and White (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 as follows. 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- and 60-day images as described in Section III of the Internet Appendix. 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 IA.XIX. 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, eight go in the opposite direction of the folk wisdom prediction.

[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).

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 III. Second, we have a reassuring placebo result in Table IA.XIX: 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 nonrandom. All told, this evidence tilts us toward the conclusion that some price patterns are likely to be predictive, but those typical directional patterns from technical analysis books do not seem profitable in our context.

V. Transfer Learning

Our analysis thus far has focused on daily data for the U.S. stock market. Training the CNN in this setting confers two advantages from data size. First, because the U.S. stock 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 retraining the CNN model is infeasible. For example, 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 U.S. 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 U.S. data to construct profitable portfolios in other settings that preclude direct training of the CNN. At the same time, the analysis provides an additional out-of-sample demonstration of reliable predictive power of image-based CNN forecasts.

[15] See, for example, Pan and Yang (2009) for a survey of the computer science literature on transfer learning. In asset pricing, existing literature typically retrains the model discovered in the U.S. 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, Zhou, and Zhu (2020), which applies their genetic programming model trained with U.S. data directly to G7 international markets.

A. International Transfer

International markets have fewer stocks and shorter time-series compared to the United States. If estimates from U.S. data are applicable in international contexts, this would help 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 U.S. data to construct return forecasts in foreign markets.

We use daily stock market data from Datastream for Hong Kong, the United Kingdom, Canada, Japan, Australia, Austria, Belgium, Denmark, Finland, France, Germany, Greece, Ireland, Italy, the 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 U.S. sample, 1993 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 U.S. sample to price images in foreign markets, with no retraining. 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 X 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 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 U.S. data, and for the difference. The last row shows the simple average of each column.

In 21 out of 26 countries, transfer learning produces a Sharpe ratio gain over the locally retrained model (for equal-weight portfolios), with the gain statistically significant in 20 of these 26 countries. In the 20 significant cases, transfer from the U.S. model produces a Sharpe ratio gain of more than 0.5 (except for 0.34 for Greece). On average, transfer from the U.S. model produces a Sharpe ratio of 3.6, more than a 50% increase over the average Sharpe ratio of 2.3 for locally retrained models.

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

Table X

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

This 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 that retrains the I5/R5 CNN using local data, and the image-based strategy that directly transfers the I5/R5 model estimated in U.S. data without retraining. ***,**, and * indicate significant at the 1%, 5%, and 10% significance level, respectively.

		Equal-Weight			Value-Weight		
	Stock Count	Retrain	Direct Transfer	Transfer−Retrain	Retrain	Direct Transfer	Transfer−Retrain
U.S.	7,298	7.15		1.49			
Global	17,206	0.18	5.73	5.56***	0.46	−2.68	−3.14
Japan	3,056	3.56	5.84	2.28***	0.96	1.30	0.34*
Canada	2,924	9.01	13.02	4.01***	2.98	4.93	1.95***
India	1,861	2.52	−1.30	−3.82	0.67	−1.42	−2.09
United Kingdom	1,783	0.03	−0.10	−0.13	1.04	1.01	−0.03
France	955	2.47	4.23	1.77***	1.12	2.39	1.27***
South Korea	911	3.64	1.41	−2.23	1.74	2.07	0.33*
Australia	886	8.28	11.79	3.52***	2.78	4.09	1.31***
Germany	868	−0.29	1.21	1.50***	−0.01	2.91	2.91***
China	662	2.26	−2.30	−4.55	0.66	−1.05	−1.72
Hong Kong	543	1.97	6.05	4.08***	0.72	2.09	1.37***
Singapore	284	6.98	7.51	0.53**	2.48	4.21	1.73***
Sweden	260	5.43	7.15	1.72***	0.83	1.93	1.10***
Italy	241	2.14	3.57	1.43***	0.76	1.56	0.79***
Switzerland	240	0.48	0.70	0.21	1.30	2.61	1.31***
Denmark	223	1.94	3.76	1.82***	1.18	1.96	0.79***
Netherlands	212	−0.30	3.68	3.98***	0.11	1.56	1.45***
Greece	201	2.74	3.08	0.34*	0.98	1.97	0.99***
Belgium	171	0.73	4.55	3.82***	0.73	2.91	2.17***
Spain	170	1.62	0.28	−1.34	0.68	0.99	0.31*
Norway	169	0.79	3.64	2.85***	1.11	2.83	1.72***
Portugal	121	0.30	2.43	2.13***	0.93	1.31	0.38**
New Zealand	114	0.50	2.38	1.88***	0.65	1.30	0.65***
Finland	113	2.66	5.33	2.66***	0.95	2.28	1.33***
Austria	110	0.14	0.80	0.66***	0.66	0.94	0.27
Ireland	75	0.47	2.07	1.60***	0.31	2.63	2.32***
Russia	53	−0.72	1.97	2.69***	−0.13	0.53	0.66***
Average	1,274	2.21	3.65	1.44	0.99	1.75	0.76
Average (excluding Global)	661	2.28	3.57	1.28	1.01	1.92	0.91

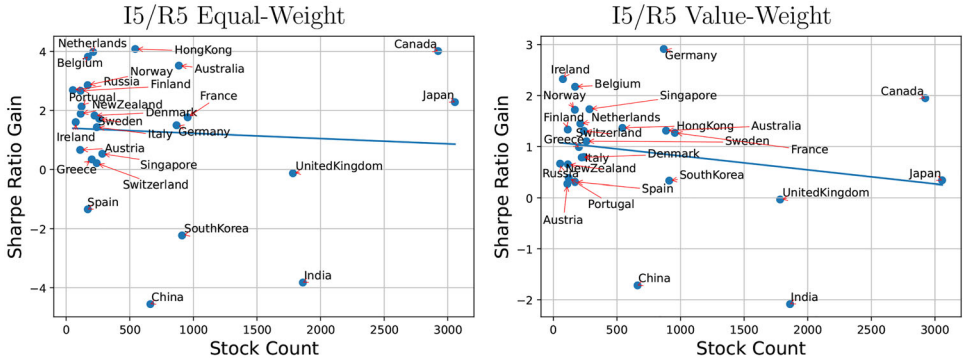

Figure 9. Sharpe ratio gains from international transfer. The panels show annualized out-of-sample Sharpe ratio gains from I5/R5 transfer learning (both equal-weight and value-weight strategies) versus the average number of stocks in each country. (Color figure can be viewed at wileyonlinelibrary.com)

The comparative gains from transfer versus local training are similar in value-weight strategies, shown in the three right 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 U.S. models nearly doubles the Sharpe ratio of value-weight strategies relative to locally retrained 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 IA.VI for I20/R20 CNN models. In Tables IA.VII and IA.VIII, we also decompose the monthly strategy performance into returns on days 1 to 5 versus days 6 to 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 X 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 benefit especially from U.S. 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. These results suggest there are benefits to local retraining 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 IA.1 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.

B. 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 five-period OHLC chart by redefining a "period" to be a four-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 shows open, high, low, and close over the four-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 XI 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, that is, it exactly repeats the strategy of the I20/R20 model in Table II. The columns labeled "Retrain, No Transfer" reestimate a CNN from scratch using 20 days of data collapsed into five-period images and supervised by future 20-day returns. This provides a benchmark for how predictability is affected by simply down-sampling the price history and retraining 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 reestimation 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 retrained 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 it picks up 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 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 retrained CNN. At 0.7, the H-L transfer

Table XI

Time-Scale Transfer I5/R5 to I20/R20

This table reports results on 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 II and IA.III, and a CNN model that is trained from scratch using images of 20 days of data sampled once every four days (under "Retrain"). Finally, "Baseline+Transfer" shows the performance of an equal-weight average of the baseline and transfer strategies. ***, **, and indicate significant at the 1%, 5%, and 10% significance level, respectively.

Panel A: Equal-Weight Portfolios

	Baseline		Retrain		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%	

(*Continued*)

Table XI—*Continued*

Panel B: Value-Weight Portfolios

	Baseline		Retrain		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%	

strategy more than doubles the Sharpe ratios from retraining (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 five-period images, that is, 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 retraining the CNN on collapsed images or directly transferring the estimated I5/R5 model to the I60/R60 problem at the 1:12 time scale. The results are reported in Table XII 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 retraining 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 retraining. 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 represents an interesting problem for future research (and unfortunately beyond the scope of this paper).

VI. 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 U.S. 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 that technical analysis comprises a major part of their investment process. As Lo, Mamaysky, and Wang (2000) at the start of this paper note, technical analysis is a

[17] In Section IV.B of the Internet Appendix, 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.

Table XII

Time-Scale Transfer I5/R5 to I60/R60

This table reports results on 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 II and IA.III, and a CNN model that is trained from scratch using images of 60 days of data sampled once every 12 days (under "Retrain"). Finally, "Baseline+Transfer" shows the performance of an equal-weight average of the baseline and transfer strategies. ***, **, and indicate significant at the 1%, 5%, and 10% significance level, respectively.

Panel A: Equal-Weight Portfolios

	Baseline		Retrain		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%	

(*Continued*)

Table XII—*Continued*

Panel B: Value-Weight Portfolios

	Baseline		Retrain		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%	

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 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.

Initial submission: August 16, 2021; Accepted: July 1, 2022
Editors: Stefan Nagel, Philip Bond, Amit Seru, and Wei Xiong

Appendix: Architecture Details of the CNN

A CNN is a modeling scheme that stacks 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 A1, Panel A, illustrates 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 that takes 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]

[18] For elements at the image's border, we fill the absent neighbor elements with zeros in order to compute the convolution. This is a common CNN practice known as "padding" and ensures that

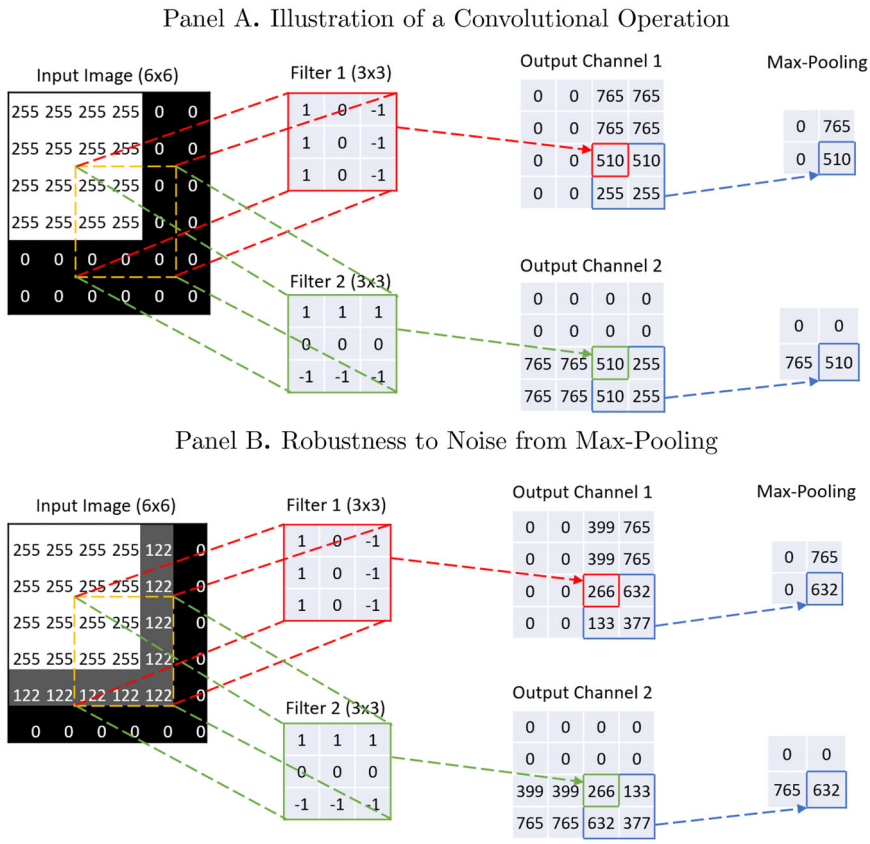

Figure A1. Illustration of convolution and max-pooling operations. Panel A: The input tensor has size 6×6 (with only one channel of gray-scale). There are two 3×3 filters that detect edges. By construction, Filter 1 detects vertical edges and Filter 2 detects horizontal edges. When Filter 1 is applied 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 outputs 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, 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, extracting the 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 fashion (green dotted line and green dotted arrow), extracting the information that there are horizontal edges at the bottom of the image and the bottom-left (765) has a sharper change than the bottom-right (510). Panel B: This example shows that the introduction of the max-pooling layer adds robustness to noise. We take the same 6×6 image from the example above and blur the borderline with gray color (pixel value 122). Take Filter 1, which detects vertical edges, for example. Nonzero values from Output Channel 1 are significantly changed. However, after max-pooling is applied, the original upper-right value of 765 remains intact while the original bottom-right value of 510 is replaced with 632, a slightly larger number, indicating a slightly stronger signal of the vertical line at the bottom than before. We observe a similar effect for Filter 2, which detects horizontal edges. (Color figure can be viewed at wileyonlinelibrary.com)

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 signals 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. It is also common for a CNN 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 dilation rate k along the vertical or horizontal dimension (or both) expands the size of the original filter by filling $(k - 1)$ 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

the convolution output has the same dimension as the image itself (see Simonyan and Zisserman (2015)).

[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.

sharing" and "sparse interactions," are critical to the tractability and small-sample robustness of the CNN. Further, 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 nonlinear 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]

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 of the elements entering the filter at each location in the image. The role of max-pooling is twofold. 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 downsampling 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 A1, Panel B). In this sense, max-pooling is a denoising 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. Building blocks in the interior layers of our CNN therefore take three-dimensional tensors as inputs. If the input layer has multiple channels,

[20] This activation function, introduced by Maas, Hannun, and Ng (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 maintaining 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.

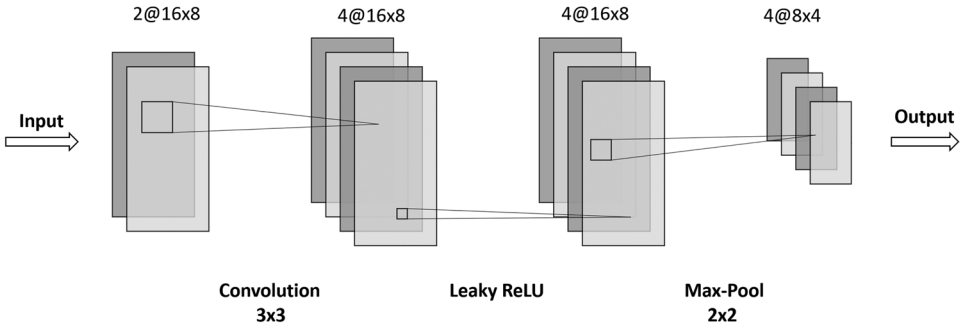

Figure A2. Diagram of a building block. This figure illustrates that a building block of the CNN model consists of a convolutional layer with 3×3 filter, a leaky ReLU layer, and a 2×2 max-pooling layer. The notation D@HxW shows the output size of the CNN building block where H is the height, W is the width, and D is the number of channels. In this example, the input has size 16×8 with two channels. To double the depth of the input, four filters are applied, which generates the output with four 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 gray-scale image as input: The input has depth of one and the number of CNN filters is 32, boosting the depth of the output to 32.

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, and then combine this local information across all channels. In Figure A2, the input has height of 16 and width of eight 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 is 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-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 and 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-, 20-, 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 two blocks to model five-day images, three blocks for 20-day images, and four 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-, 20-, 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 five-day model, 708,866 for the 20-day model, and 2,952,962 for the 60-day model, 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.

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 B. Douglas Bernheim, Stefano DellaVigna and David Laibson, eds., *Handbook of Behavioral Economics: Applications and Foundations 1*, 79–175 (Elsevier, North-Holland).

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, *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, *Journal of Finance* 47, 1731–1764.

Brown, David P., and Robert H. Jennings, 1989, On technical analysis, *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 Jacques Lévy-Véhel and Evelyne Lutton, eds., *Fractals in Engineering*, 159–179 (Springer, London).

[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 three 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 two corresponds to the two classification labels. We ignore the number of parameters from the batch normalization as it is negligible.

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* 50, 107–137.

Dobrev, Dobrislav, 2007, Capturing volatility from large price moves: Generalized range theory and applications, Working paper, 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, Working paper, Yale University.

Glorot, Xavier, and Yoshua Bengio, 2010, Understanding the difficulty of training deep feedforward neural networks, in Teh, Yee Whye and Titterington, Mike, eds., *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 249–256 (JMLR Workshop and Conference Proceedings).

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville, 2016, *Deep Learning* (MIT Press, Boston, MA).

Grundy, Bruce D., and Maureen McNichols, 1989, Trade and the revelation of information through prices and direct disclosure, *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, and Qiangwei Miemie, 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, in *Proceedings of the 32nd International Conference on International Conference on Machine Learning* 37, 448–456.

Jegadeesh, Narasimhan, and Sheridan Titman, 1993, Returns to buying winners and selling losers: Implications for stock market efficiency, *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* (forthcoming).

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, *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.

Kingma, Diederik P., and Jimmy L. Ba, 2014, Adam: A method for stochastic optimization, in *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*.

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, Working paper, Tsinghua University.

Lo, Andrew W., and Jasmina Hasanhodzic, 2009, *The Heretics of Finance: Conversations with Leading Practitioners of Technical Analysis* (Bloomberg Press, New York, NY).

Lo, Andrew W., Harry Mamaysky, and Jiang Wang, 2000, Foundations of technical analysis: Computational algorithms, statistical inference, and empirical implementation, *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 *Proceedings of the 30th International Conference on Machine Learning 28*, 3.

Mandelbrot, Benoit B., 2013, *Fractals and Scaling in Finance: Discontinuity, Concentration, Risk* (Springer Science & Business Media, Springer New York, NY).

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. New York).

Murray, Scott, Houping Xiao, and Yusen Xia, 2021, Charting by machines, Working paper, 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.

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

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

Simonyan, Karen, and Andrew Zisserman, 2015, Very Deep Convolutional Networks for Large-Scale Image Recognition, in *Proceedings of the International Conference on Learning Representations (ICLR)*.

Srivastava, Nitish, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, 2014, Dropout: A simple way to prevent neural networks from overfitting, *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, *Journal of Finance* 54, 1647–1691.

Zeiler, Matthew D., and Rob Fergus, 2014, Visualizing and understanding convolutional networks, in David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, eds., *European Conference on Computer Vision*, 818–833 (Springer, Cham).

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.

Supporting Information

Additional Supporting Information may be found in the online version of this article at the publisher's website:

Appendix S1: Internet Appendix.
Replication Code.