Enhancing Time-Series Momentum Strategies Using Deep Neural Networks

Bryan Lim, Stefan Zohren, and Stephen Roberts

Bryan Lim
is a DPhil student with the Machine Learning Research Group and the Oxford-Man Institute of Quantitative Finance at the University of Oxford in Oxford, United Kingdom.
bryan.lim@eng.ox.ac.uk

Stefan Zohren
is an associate professor (research) with the Machine Learning Research Group and the Oxford-Man Institute of Quantitative Finance at the University of Oxford in Oxford, United Kingdom.
zohren@robots.ox.ac.uk

Stephen Roberts
is the RAEng/Man Professor of Machine Learning at the University of Oxford and the Director of the Oxford-Man Institute of Quantitative Finance in Oxford, United Kingdom.
sjrob@robots.ox.ac.uk

*All articles are now categorized by topics and subtopics. **View at PM-Research.com.**

KEY FINDINGS

- While time-series momentum strategies have been extensively studied in finance, common strategies require the explicit specification of a trend estimator and position sizing rule.
- In this article, the authors introduce deep momentum networks —a hybrid approach that injects deep learning–based trading rules into the volatility scaling framework of time-series momentum.
- Backtesting on a portfolio of continuous futures contracts, Deep Momentum Networks were shown to outperform traditional methods for transaction costs of up to 2–3 bps, with a turnover regularisation term proposed for more illiquid assets.

ABSTRACT: *Although time-series momentum is a well-studied phenomenon in finance, common strategies require the explicit definition of both a trend estimator and a position sizing rule. In this article, the authors introduce deep momentum networks—a hybrid approach that injects deep learning–based trading rules into the volatility scaling framework of time-series momentum. The model also simultaneously learns both trend estimation and position sizing in a data-driven manner, with networks directly trained by optimizing the Sharpe ratio of the signal. Backtesting on a portfolio of 88 continuous futures contracts, the authors demonstrate that the Sharpe-optimized long short-term memory improved traditional methods by more than two times in the absence of transactions costs and continued outperforming when considering transaction costs up to 2–3 bps. To account for more illiquid assets, the authors also propose a turnover regularization term that trains the network to factor in costs at run-time.*

TOPICS: ***Statistical methods, simulations, big data/machine learning****

Momentum as a risk premium in finance has been extensively documented in the academic literature, with evidence of persistent abnormal returns demonstrated across a range of asset classes, prediction horizons, and time periods (Lempérière et al. 2014; Baz et al. 2015; Hurst, Ooi, and Pedersen 2017). Based on the philosophy that strong price trends have a tendency to persist, time-series momentum strategies are typically designed to increase position sizes with large directional moves and reduce positions at other times. Although the intuition underpinning the strategy is clear, specific implementation details can vary widely between signals; a plethora of methods are available to estimate the magnitude of price trends (Bruder et al. 2013; Baz et al. 2015; Levine and Pedersen 2016) and map them to actual traded positions

(Kim, Tse, and Wald 2016; Baltas and Kosowski 2017; Harvey et al. 2018).

In recent times, deep neural networks have been increasingly used for time-series prediction and have outperformed traditional benchmarks in applications such as demand forecasting (Laptev et al. 2017), medicine (Lim and van der Schaar 2018), and finance (Zhang, Zohren, and Roberts 2019). With the development of modern architectures such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs) (Goodfellow, Bengio, and Courville 2016), deep learning models have been favored for their ability to build representations of a given dataset (Bengio, Courville, and Vincent 2013), capturing temporal dynamics and cross-sectional relationships in a purely data-driven manner. The adoption of deep neural networks has also been facilitated by powerful open-source frameworks such as TensorFlow (Abadi et al. 2015) and PyTorch (Paszke et al. 2017), which use automatic differentiation to compute gradients for backpropagation without having to explicitly derive them in advance. In turn, this flexibility has allowed deep neural networks to go beyond standard classification and regression models. For instance, hybrid methods that combine traditional time-series models with neural network components have been observed to outperform pure methods in either category (Makridakis, Spiliotis, and Assimakopoulos 2018)—for example, the exponential smoothing RNN (Smyl, Ranganathan, and Pasqua 2018), autoregressive CNNs (Binkowski, Mikolaj, and Donnat 2018), and Kalman filter variants (Fraccaro et al. 2017; Rangapuram et al. 2018)—while also making outputs easier to interpret by practitioners. Furthermore, these frameworks have enabled the development of new loss functions for training neural networks, such as adversarial loss functions in generative adversarial networks (GANs) (Goodfellow et al. 2014).

Although numerous papers have investigated the use of machine learning for financial time-series prediction, they typically focus on casting the underlying prediction problem as a standard regression or classification task (Bao, Yue, and Rao 2017; Gu, Kelly, and Xiu 2017; Binkowski, Marti, and Donnat 2018; Ghoshal and Roberts 2018; Sirignano and Cont 2018; Kim 2019; Zhang, Zohren, and Roberts 2019) with regression models forecasting expected returns and classification models predicting the direction of future price movements. This approach, however, could lead to suboptimal performance in the context of time-series momentum for several reasons. First, sizing positions based on expected returns alone does not take risk characteristics (e.g., the volatility or skew of the predictive returns distribution) into account, which could inadvertently expose signals to large downside moves. This is particularly relevant because raw momentum strategies without adequate risk adjustments, such as volatility scaling (Kim, Tse, and Wald 2016), are susceptible to large crashes during periods of market panic (Barroso and Santa-Clara 2015; Daniel and Moskowitz 2016). Furthermore, even with volatility scaling—which leads to positively skewed returns distributions and long option–like behavior (Martins and Zou 2012; Jusselin et al. 2017)—trend-following strategies can place more losing trades than winning ones and still be profitable on the whole because they size up only into large but infrequent directional moves. As such, Potters and Bouchaud (2016) argued that the fraction of winning trades is a meaningless metric of performance, given that it cannot be evaluated independently from the trading style of the strategy. Similarly, high classification accuracies may not necessarily translate into positive strategy performance because profitability also depends on the magnitude of returns in each class. This is also echoed in betting strategies such as the Kelly criterion (Rotando and Thorp 1992), which requires both win/loss probabilities and betting odds for optimal sizing in binomial games. In light of the deficiencies of standard supervised learning techniques, new loss functions and training methods would need to be explored for position sizing—accounting for trade-offs between risk and reward.

In this article, we introduce a novel class of hybrid models that combines deep learning–based trading signals with the volatility scaling framework used in time-series momentum strategies (Moskowitz, Ooi, and Pedersen 2012; Baltas and Kosowski 2017), which we refer to as *deep momentum networks* (DMNs). This improves existing methods from several angles. First, by using deep neural networks to directly generate trading signals, we remove the need to manually specify both the trend estimator and position sizing methodology—allowing them to be learned directly using modern time-series prediction architectures. Second, by using automatic differentiation in existing backpropagation frameworks, we explicitly optimize networks for risk-adjusted performance metrics—that is, the Sharpe ratio (Sharpe 1994)—improving the risk profile of the signal on the whole. Lastly, retaining a consistent framework

with other momentum strategies also allows us to retain desirable attributes from previous works—specifically volatility scaling, which plays a critical role in the positive performance of time-series momentum strategies (Harvey et al. 2018). This consistency also helps when making comparisons to existing methods and facilitates the interpretation of different components of the overall signal by practitioners.

RELATED WORK

Classical Momentum Strategies

Momentum strategies are traditionally divided into two categories: (multivariate) cross-sectional momentum (Jegadeesh and Titman 1993; Kim 2019) and (univariate) time-series momentum (Moskowitz, Ooi, and Pedersen 2012; Baltas and Kosowski 2017). Cross-sectional momentum strategies focus on the relative performance of securities against each other, buying relative winners and selling relative losers. By ranking a universe of stocks based on their past return and trading the top decile against the bottom decile, Jegadeesh and Titman (1993) found that securities that recently outperformed their peers over the past 3 to 12 months continue to outperform on average over the next month. The performance of cross-sectional momentum has also been shown to be stable across time (Jegadeesh and Titman 2001) and across a variety of markets and asset classes (Baz et al. 2015).

Time-series momentum extends the idea to focus on an asset's own past returns, building portfolios comprising all securities under consideration. This was initially proposed by Moskowitz, Ooi, and Pedersen (2012), who described a concrete strategy that uses volatility scaling and trades positions based on the sign of returns over the past year; they demonstrated profitability across 58 liquid instruments individually over 25 years of data. Since then, numerous trading rules have been proposed, with various trend estimation techniques and methods mapping them to traded positions. For instance, Bruder et al. (2013) documented a wide range of linear and nonlinear filters to measure trends and a statistic to test for its significance, although they did not directly discuss methods to size positions with these estimates. Baltas and Kosowski (2017) adopted an approach similar to that of Moskowitz, Ooi, and Pedersen (2012), regressing the log price over the past 12 months against time and using the regression coefficient *t*-statistics to determine the direction of the traded position. Although Sharpe ratios were comparable between the two, *t*-statistic–based trend estimation led to a 66% reduction in portfolio turnover and consequently trading costs. More sophisticated trading rules were proposed by Baz et al. (2015) and Rohrbach, Suremann, and Osterrieder (2017), taking volatility-normalized moving average convergence divergence (MACD) indicators as inputs. Despite the diversity of options, few comparisons have been made of the trading rules themselves, offering little clear evidence or intuitive reasoning to favor one rule over the next. We hence propose the use of deep neural networks to generate these rules directly, avoiding the need for explicit specification. Training them based on risk-adjusted performance metrics, the networks hence learn optimal training rules directly from the data itself.

Deep Learning in Finance

Machine learning has long been used for financial time-series prediction, with recent deep learning applications studying mid-price prediction using daily data (Ghoshal and Roberts 2018) or using limit order book data in a high-frequency trading setting (Sirignano and Cont 2018; Zhang, Zohren, and Roberts 2018, 2019). Although a variety of CNN and RNN models have been proposed, they typically frame the forecasting task as a classification problem, demonstrating the improved accuracy of their method in predicting the direction of the next price movement. Trading rules are then manually defined in relation to class probabilities—either by using thresholds on classification probabilities to determine when to initiate positions (Ghoshal and Roberts 2018) or incorporating these thresholds into the classification problem itself by dividing price movements into buy, hold, and sell classes depending on magnitude (Zhang, Zohren, and Roberts 2018, 2019). In addition to restricting the universe of strategies to those that rely on high accuracy, further gains might be made by learning trading rules directly from the data and removing the need for manual specification, both of which are addressed in our proposed method.

Deep learning regression methods have also been considered in cross-sectional strategies (Gu, Kelly, and Xiu 2017; Kim 2019), ranking assets on the basis of expected returns over the next time period. Using a variety of linear, tree-based, and neural network

models, Gu, Kelly, and Xiu (2017) demonstrated the outperformance of nonlinear methods, with deep neural networks—specifically three-layer multilayer perceptrons (MLPs)—having the best out-of-sample predictive R^2. Machine learning portfolios were then built by ranking stocks on a monthly basis using model predictions, with the best strategy coming from a four-layer MLP that trades the top decile against the decile of predictions. In other works, Kim (2019) adopted a similar approach using autoencoder and denoising autoencoder architectures, incorporating volatility scaling into this model as well. Although the results with basic deep neural networks are promising, they do not consider more modern architectures for time-series prediction, such as long short-term memory (LSTM) (Hochreiter and Schmidhuber 1997) and WaveNet (van den Oord et al. 2016), architectures that we evaluate for the DMN. Moreover, to the best of our knowledge, our article is the first to consider the use of deep learning within the context of time-series momentum strategies, thus opening up possibilities in an alternate class of signals.

Popularized by success of DeepMind's AlphaGo Zero (Silver et al. 2017), deep reinforcement learning (RL) has also gained much attention in recent times. Prized for its ability to recommend path-dependent actions in dynamic environments, RL is particularly of interest within the context of optimal execution and automated hedging (Bühler et al. 2018; Kolm and Ritter 2019) for example, where actions taken can have an impact on future states of the world (e.g., market impact). However, deep RL methods generally require a realistic simulation environment (for Q-learning or policy gradient methods) or model of the world (for model-based RL) to provide feedback to agents during training—both of which are difficult to obtain in practice.

STRATEGY DEFINITION

Adopting the framework of Baltas and Kosowski (2017), the combined returns of a time-series momentum (*TSMOM*) strategy can be expressed as follows and is characterized by a trading rule or signal $X_t \in [-1, 1]$:

$$r_{t,t+1}^{TSMOM} = \frac{1}{N_t} \sum_{i=1}^{N_t} X_t^{(i)} \frac{\sigma_{tgt}}{\sigma_t^{(i)}} r_{t,t+1}^{(i)} \tag{1}$$

Here $r_{t,t+1}^{TSMOM}$ is the realized return of the strategy from day t to $t+1$, N_t is the number of included assets at t, and $r_{t,t+1}^{(i)}$ is the one-day return of asset i. This allows us to retain consistency with previous work on time-series momentum strategies, putting signals together in an equal-volatility-weighted portfolio to cap the returns volatility of each individual asset at a target level. We set the annualized volatility target σ_{tgt} to be 15% and scale asset returns with an ex ante volatility estimate $\sigma_t^{(i)}$—computed using an exponentially weighted moving standard deviation with a 60-day span on $r_{t,t+1}^{(i)}$.

Standard Trading Rules

In traditional financial time-series momentum strategies, the construction of a trading signal X_t is typically divided into two steps: (1) estimating future trends based on past information and (2) computing the actual positions to hold. We illustrate this in this section using two examples from the academic literature (Moskowitz, Ooi, and Pedersen 2012; Baz et al. 2015), which we also include as benchmarks for our tests.

Moskowitz, Ooi, and Pedersen. In their original paper on time-series momentum, a simple trading rule was adopted as follows:

$$\text{Trend estimation: } Y_t^{(i)} = r_{t-252,t}^{(i)} \tag{2}$$

$$\text{Position sizing: } X_t^{(i)} = \text{sgn}(Y_t^{(i)}) \tag{3}$$

This broadly uses the past year's returns as a trend estimate for the next time step, taking a maximum long position when the expected trend is positive (i.e., $\text{sgn}(r_{t-252,t}^{(i)})$) and a maximum short position when negative.

Baz et al. In practice, more sophisticated methods can be used to compute $Y_t^{(i)}$ and $X_t^{(i)}$, such as the model of Baz et al. (2015) described in the following:

$$\text{Trend estimation: } Y_t^{(i)} = \frac{q_t^{(i)}}{\text{std}(z_{t-252:t}^{(i)})} \tag{4}$$

$$q_t^{(i)} = \text{MACD}(i,t,S,L)/\text{std}(p_{t-63:t}^{(i)}) \tag{5}$$

$$\text{MACD}(i,t,S,L) = m(i,S) - m(i,L) \tag{6}$$

Here $\text{std}(p^{(i)}_{t-63:t})$ is the 63-day rolling standard deviation of asset i prices $p^{(i)}_{t-63:t} = [p^{(i)}_{t-63}, \ldots, p^{(i)}_{t}]$, and $m(i, S)$ is the exponentially weighted moving average of asset i prices with a time scale S that translates into a half-life of $HL = \log(0.5)/\log(1-\frac{1}{S})$. The MACD signal is defined in relation to a short and a long time scale S and L, respectively.

The volatility-normalized MACD signal hence measures the strength of the trend, which is then translated in to a position size as follows:

$$\text{Position sizing: } X^{(i)}_t = \phi(Y^{(i)}_t) \tag{7}$$

where $\phi(y) = \frac{y \exp\left(\frac{-y^2}{4}\right)}{0.89}$. Plotting $\phi(y)$ in Exhibit 1, we can see that positions are increased until $|Y^{(i)}_t| = \sqrt{2} \approx 1.41$ before decreasing back to zero for larger moves. This allows the signal to reduce positions in instances in which assets are overbought or oversold—defined as when $|q^{(i)}_t|$ is observed to be larger than 1.41 times its past year's standard deviation.

Increasing the complexity even further, multiple signals with different time scales can also be averaged to give a final position:

$$\tilde{Y}^{(i)}_t = \sum_{k=1}^{3} Y^{(i)}_t(S_k, L_k) \tag{8}$$

where $Y^{(i)}_t(S_k, L_k)$ is as per Equation 4 with explicitly defined short and long time scales, using $S_k \in \{8, 16, 32\}$ and $L_k \in \{24, 48, 96\}$ as defined by Baz et al. (2015).

Machine Learning Extensions

As described, many arbitrary design decisions are required to define a sophisticated time-series momentum strategy. We hence start by considering how machine learning methods can be used to learn these relationships directly from data—alleviating the need for manual specification.

Standard supervised learning. In line with numerous previous investigations (see "Related Work" section), we can cast trend estimation as a standard regression or binary classifications problem, with outputs:

$$\text{Trend estimation: } Y^{(i)}_t = f(\mathbf{u}^{(i)}_t; \theta) \tag{9}$$

where $f(\cdot)$ is the output of the machine learning model, which takes in a vector of input features $\mathbf{u}^{(i)}_t$ and model parameters θ to generate predictions. Taking volatility-normalized returns as targets, the following mean-squared error and binary cross-entropy losses can be used for training:

$$\mathcal{L}_{\text{reg}}(\theta) = \frac{1}{M} \sum_{\Omega} \left(Y^{(i)}_t - \frac{r^{(i)}_{t,t+1}}{\sigma^{(i)}_t} \right)^2 \tag{10}$$

$$\mathcal{L}_{\text{binary}}(\theta) = -\frac{1}{M} \sum_{\Omega} \{ \mathbb{I} \log(Y^{(i)}_t) + (1 - \mathbb{I}) \log(1 - Y^{(i)}_t) \} \tag{11}$$

where $\Omega = \{(Y^{(1)}_1, r^{(1)}_{1,2}/\sigma^{(1)}_t), \ldots, (Y^{(N)}_{T-1}, r^{(N)}_{T-1,T}/\sigma^{(N)}_{T-1})\}$ is the set of all $M = NT$ possible prediction and target tuples across all N assets and T time steps. For the binary classification case, $\mathbb{I}$ is the indicator function $\mathbb{I}(r^{(i)}_{t,t+1} > 0)$—making $Y^{(i)}_t$ the estimated probability of a positive return.

This still leaves us to specify how trend estimates map to positions, and we do so using a similar form to Equation 3 as follows:

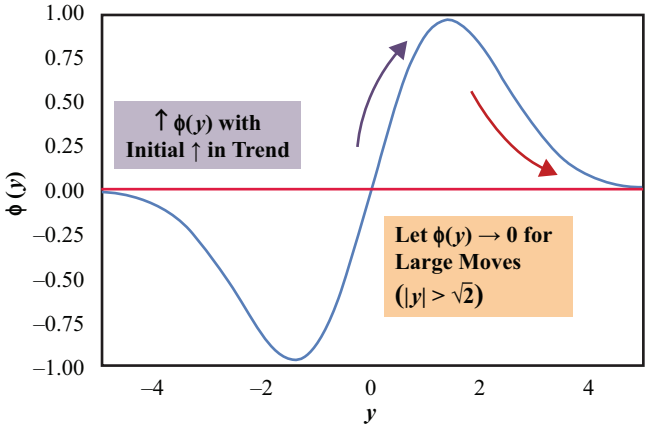
$$\text{Position sizing} \tag{12}$$

$$\text{a) Regression } X^{(i)}_t = \text{sgn}(Y^{(i)}_t) \tag{13}$$

$$\text{b) Classification } X^{(i)}_t = \text{sgn}(Y^{(i)}_t - 0.5) \tag{14}$$

As such, we take a maximum long position when the expected returns are positive in the regression case or

EXHIBIT 1
Position Sizing Function φ(*y*)



when the probability of a positive return is greater than 0.5 in the classification case. This formulation maintains consistency with past work on time-series momentum and volatility scaling, allowing us to make direct comparisons with previous methods and to evaluate the performance of sophisticated trend estimators as opposed to simply using the previous year's return.

Direct outputs. An alternative approach is to use machine learning models to generate positions directly—simultaneously learning both trend estimation and position sizing in the same function:

$$\text{Direct outputs: } X_t^{(i)} = f(\mathbf{u}_t^{(i)}; \theta) \tag{15}$$

Given the lack of direct information on the optimal positions to hold at each step—which is required to produce labels for standard regression and classification models—calibration would hence need to be performed by directly optimizing performance metrics. Specifically, we focus on optimizing the average return and the Sharpe ratio via the loss functions as follows:

$$\mathcal{L}_{\text{returns}}(\theta) = -\frac{1}{M}\sum_{\Omega} R(i,t) = -\frac{1}{M}\sum_{\Omega} X_t^{(i)} \frac{\sigma_{\text{tgt}}}{\sigma_t^{(i)}} r_{t,t+1}^{(i)} \tag{16}$$

$$\mathcal{L}_{\text{sharpe}}(\theta) = -\frac{\sum_{\Omega} R(i,t)}{\sum_{\Omega} R(i,t)^2 - \left(\sum_{\Omega} R(i,t)\right)^2} \tag{17}$$

where $R(i,t)$ is the return captured by the trading rule for asset i at time t.

DEEP MOMENTUM NETWORKS

In this section, we examine a variety of architectures that can be used in DMNs, all of which can be easily reconfigured to generate the predictions described in the previous section. This is achieved by implementing the models using the Keras API in Tensorflow (Abadi et al. 2015), in which output activation functions can be flexibly interchanged to generate predictions of different types (e.g., expected returns, binary probabilities, or direct positions). Arbitrary loss functions can also be defined for direct outputs, with gradients for backpropagation being easily computed using the built-in libraries for automatic differentiation.

Network Architectures

Lasso regression. In the simplest case, a standard linear model could be used to generate predictions as follows:

$$Z_t^{(i)} = g(\mathbf{w}^T \mathbf{u}_{t-\tau:t}^{(i)} + b) \tag{18}$$

where $Z_t^{(i)} \in \{X_t^{(i)}, Y_t^{(i)}\}$ depending on the prediction task, $\mathbf{w}$ is a weight vector for the linear model, and b is a bias term. Here $g(\cdot)$ is a activation function that depends on the specific prediction type—linear for standard regression, sigmoid for binary classification, and tanh-function for direct outputs.

Additional regularization is also provided during training by augmenting the various loss functions to include an additional L_1 regularizer as follows:

$$\widetilde{\mathcal{L}}(\theta) = \mathcal{L}(\theta) + \alpha \|\mathbf{w}\|_1 \tag{19}$$

where $\mathcal{L}(\theta)$ corresponds to one of the loss functions described in the previous section, $||\mathbf{w}||_1$ is the L_1 norm of w, and α is a constant term that we treat as an additional hyperparameter. To incorporate recent history into predictions as well, we concatenate inputs over the past τ days into a single input vector—that is, $\mathbf{u}_{t-\tau:t}^{(i)} = [\mathbf{u}_{t-\tau}^{(i)\,T}, \ldots, \mathbf{u}_t^{(i)\,T}]^T$. This was fixed to be $\tau = 5$ days in our experiments.

Multilayer perceptron. Increasing the degree of model complexity slightly, a two-layer neural network can be used to incorporate nonlinear effects:

$$\mathbf{h}_t^{(i)} = \tanh(\mathbf{W}_h \mathbf{u}_{t-\tau:t}^{(i)} + \mathbf{b}_h) \tag{20}$$

$$Z_t^{(i)} = g(\mathbf{W}_z \mathbf{h}_t^{(i)} + \mathbf{b}_z) \tag{21}$$

where $\mathbf{h}_t^{(i)}$ is the hidden state of the MLP using an internal tanh activation function, $\tanh(\cdot)$, and $\mathbf{W}_{\cdot}$ and $\mathbf{b}_{\cdot}$ are layer weight matrixes and biases, respectively.

WaveNet. More modern techniques such as CNNs have been used in the domain of time-series prediction, particularly in the form of autoregressive architectures (e.g., Binkowski, Marti, and Donnat 2018). These typically take the form of one-dimensional causal convolutions, sliding convolutional filters across time to extract useful representations that are then aggregated in higher layers of the network. To increase the size of the receptive field—or the length of history fed into the

CNN—dilated CNNs such as WaveNet (van den Oord et al. 2016) have been proposed; these skip over inputs at intermediate levels with a predetermined dilation rate and thus can effectively increase the amount of historical information used by the CNN without a large increase in computational cost.

Let us consider a dilated convolutional layer with residual connections that takes the following form:

$$\psi(\mathbf{u}) = \underbrace{\tanh(\mathbf{W}\mathbf{u}) \odot \sigma(\mathbf{V}\mathbf{u})}_{\text{Gated activation}} + \underbrace{\mathbf{A}\mathbf{u} + \mathbf{b}}_{\text{Skip connection}} \tag{22}$$

Here $\mathbf{W}$ and $\mathbf{V}$ are weight matrixes associated with the gated activation function, and $\mathbf{A}$ and $\mathbf{b}$ are the weights and biases used to transform the $\mathbf{u}$ to match the dimensionality of the layer outputs for the skip connection. The equations for WaveNet architecture used in our investigations can then be expressed as

$$\mathbf{s}^{(i)}_{\text{weekly}}(t) = \psi(\mathbf{u}^{(i)}_{t-5:t}) \tag{23}$$

$$\mathbf{s}^{(i)}_{\text{monthly}}(t) = \psi\left(\begin{bmatrix} \mathbf{s}^{(i)}_{\text{weekly}}(t) \\ \mathbf{s}^{(i)}_{\text{weekly}}(t-5) \\ \mathbf{s}^{(i)}_{\text{weekly}}(t-10) \\ \mathbf{s}^{(i)}_{\text{weekly}}(t-15) \end{bmatrix}\right) \tag{24}$$

$$\mathbf{s}^{(i)}_{\text{quarterly}}(t) = \psi\left(\begin{bmatrix} \mathbf{s}^{(i)}_{\text{monthly}}(t) \\ \mathbf{s}^{(i)}_{\text{monthly}}(t-21) \\ \mathbf{s}^{(i)}_{\text{monthly}}(t-42) \end{bmatrix}\right) \tag{25}$$

Here each intermediate layer $\mathbf{s}^{(i)}(t)$ aggregates representations at weekly, monthly, and quarterly frequencies, respectively. Intermediate layers are then concatenated at each layer before passing through a two-layer MLP to generate outputs:

$$\mathbf{s}^{(i)}_t = \begin{bmatrix} \mathbf{s}^{(i)}_{\text{weekly}}(t) \\ \mathbf{s}^{(i)}_{\text{monthly}}(t) \\ \mathbf{s}^{(i)}_{\text{quarterly}}(t) \end{bmatrix} \tag{26}$$

$$\mathbf{h}^{(i)}_t = \tanh(\mathbf{W}_h \mathbf{s}^{(i)}_t + \mathbf{b}_h) \tag{27}$$

$$Z^{(i)}_t = g(\mathbf{W}_z \mathbf{h}^{(i)}_t + \mathbf{b}_z) \tag{28}$$

State sizes for each intermediate layer $\mathbf{s}^{(i)}_{\text{weekly}}(t)$, $\mathbf{s}^{(i)}_{\text{monthly}}(t), \mathbf{s}^{(i)}_{\text{quarterly}}(t)$ and the MLP hidden state $\mathbf{h}^{(i)}_t$ are fixed to be the same, allowing us to use a single hyperparameter to define the architecture. To independently evaluate the performance of CNN and RNN architectures, the preceding also excludes the LSTM block (i.e., the context stack) described by van den Oord et al. (2016) and focuses purely on the merits of the dilated CNN model.

Long short-term memory. Traditionally used in sequence prediction for natural language processing, RNNs—specifically LSTM architectures (Hochreiter and Schmidhuber 1997)—have been increasingly used in time-series prediction tasks. The equations for the LSTM in our model are as follows:

$$\mathbf{f}^{(i)}_t = \sigma(\mathbf{W}_f \mathbf{u}^{(i)}_t + \mathbf{V}_f \mathbf{h}^{(i)}_{t-1} + \mathbf{b}_f) \tag{29}$$

$$\mathbf{i}^{(i)}_t = \sigma(\mathbf{W}_i \mathbf{u}^{(i)}_t + \mathbf{V}_i \mathbf{h}^{(i)}_{t-1} + \mathbf{b}_i) \tag{30}$$

$$\mathbf{o}^{(i)}_t = \sigma(\mathbf{W}_o \mathbf{u}^{(i)}_t + \mathbf{V}_o \mathbf{h}^{(i)}_{t-1} + \mathbf{b}_o) \tag{31}$$

$$\mathbf{c}^{(i)}_t = \mathbf{f}^{(i)}_t \odot \mathbf{c}^{(i)}_{t-1} + \mathbf{i}^{(i)}_t \odot \tanh(\mathbf{W}_c \mathbf{u}^{(i)}_t + \mathbf{V}_c \mathbf{h}^{(i)}_{t-1} + \mathbf{b}_c) \tag{32}$$

$$\mathbf{h}^{(i)}_t = \mathbf{o}^{(i)}_t \odot \tanh(\mathbf{c}^{(i)}_t) \tag{33}$$

$$Z^{(i)}_t = g(\mathbf{W}_z \mathbf{h}^{(i)}_t + \mathbf{b}_z) \tag{34}$$

where $\odot$ is the Hadamard (elementwise) product; $\sigma(.)$ is the sigmoid activation function; $\mathbf{W}$ and $\mathbf{V}$ are weight matrixes for the different layers; $\mathbf{f}^{(i)}_t$, $\mathbf{i}^{(i)}_t$, $\mathbf{o}^{(i)}_t$ correspond to the forget, input, and output gates, respectively; $\mathbf{c}^{(i)}_t$ is the cell state; and $\mathbf{h}^{(i)}_t$ is the hidden state of the LSTM. From these equations, we can see that the LSTM uses the cell state as a compact summary of past information, controlling memory retention with the forget gate and incorporating new information via the input gate. As such, the LSTM is able to learn representations of long-term relationships relevant to the prediction task,

sequentially updating its internal memory states with new observations at each step.

Training Details

Model calibration was undertaken using minibatch stochastic gradient descent with the Adam optimizer (Kingma and Ba 2015), based on the previously defined loss functions. Backpropagation was performed up to a maximum of 100 training epochs using 90% of a given block of training data with the most recent 10% retained as a validation dataset. Validation data are then used to determine convergence—with early stopping triggered when the validation loss has not improved for 25 epochs—and to identify the optimal model across hyperparameter settings. Hyperparameter optimization was conducted using 50 iterations of random search (full details are provided in the Appendix). For additional information on the deep neural network calibration, please refer to Goodfellow, Bengio, and Courville (2016).

Dropout regularization (Srivastava et al. 2014) was a key feature to avoid overfitting in the neural network models, with dropout rates included as hyperparameters during training. This was applied to the inputs and hidden state for the MLP, as well as the inputs (Equation 23) and outputs (Equation 27) of the convolutional layers in the WaveNet architecture. For the LSTM, we adopted the same dropout masks used by Gal and Ghahramani (2016)—applying dropout to the RNN inputs, recurrent states, and outputs.

PERFORMANCE EVALUATION

Overview of Dataset

The predictive performance of the different architectures was evaluated via a backtest using 88 ratio-adjusted continuous futures contracts downloaded from the Pinnacle Data Corp. CLC Database.[1] These contracts spanned a variety of asset classes—including commodities, fixed income, and currency futures—and contained prices from 1990 to 2015. A full breakdown of the dataset can be found in the Appendix.

[1] Pinnacle Data Corp. CLC Database: https//pinnacledata2.com/clc.html.

In contrast to simple panama-stitching, which permits negative prices, the positivity ensured by ratio adjustments would allow us to adopt the geometric returns formulation used by traditional definitions of time-series momentum, maintaining consistency with previous works.

Backtest Description

Throughout our backtest, the models were recalibrated from scratch every five years using an expanding window of data, rerunning the entire hyperparameter optimization procedure using all data available up to the recalibration point. Model weights were then fixed for signals generated over the next five-year period, ensuring that tests were performed out of sample (i.e., from 1995–2015, with 1990–1995 used for training only). In addition, we also report the average expected return per five-year out-of-sample block in the online supplement, detailing means and standard deviations across all blocks.

For the DMNs, we incorporate a series of useful features adopted by standard time-series momentum strategies (see "Standard Trading Rules") in to generate predictions at each step:

1. *Normalized returns*—Returns over the past day, one-month, three-month, six-month, and one-year periods are used, normalized by a measure of daily volatility scaled to an appropriate time scale. For instance, normalized annual returns were taken to be $r^{(i)}_{t-252,t}/(\sigma^{(i)}_t\sqrt{252})$.
2. *MACD indicators*—We also include the MACD indicators (i.e., trend estimates $Y^{(i)}_t$), as in Equation 4, using the same short time scales $S_k \in \{8, 16, 32\}$ and long time scales $L_k \in \{24, 48, 96\}$.

For comparisons against traditional time-series momentum strategies, we also incorporate the following reference benchmarks:

1. Long only with volatility scaling ($X^{(i)}_t = 1$)
2. Sgn(returns) (Moskowitz, Ooi, and Pedersen 2012)
3. MACD signal (Baz et al. 2015)

Finally, performance was judged based on the following metrics:

EXHIBIT 2
Performance Metrics—Raw Signal Outputs

	E[Return]	Vol.	Downside Deviation	MDD	Sharpe	Sortino	Calmar	% of Positive Returns	Ave. P / Ave. L
Reference									
Long Only	0.039	0.052	0.035	0.167	0.738	1.086	0.230	53.8%	0.970
Sgn(Returns)	0.054	0.046	0.032	0.083	1.192	1.708	0.653	54.8%	1.011
MACD	0.030	0.031	0.022	0.081	0.976	1.356	0.371	53.9%	1.015
Linear									
Sharpe	0.041	0.038	0.028	0.119	1.094	1.462	0.348	54.9%	0.997
Ave. Returns	0.047	0.045	0.031	0.164	1.048	1.500	0.287	53.9%	1.022
MSE	0.049	0.047	0.032	0.164	1.038	1.522	0.298	54.3%	1.000
Binary	0.013	0.044	0.030	0.167	0.295	0.433	0.078	50.6%	1.028
MLP									
Sharpe	0.044	0.031	0.025	0.154	1.383	1.731	0.283	56.0%	1.024
Ave. Returns	**0.064***	0.043	0.030	0.161	1.492	2.123	0.399	55.6%	1.031
MSE	0.039	0.046	0.032	0.166	0.844	1.224	0.232	52.7%	1.035
Binary	0.003	0.042	0.028	0.233	0.080	0.120	0.014	50.8%	0.981
WaveNet									
Sharpe	0.030	0.035	0.026	0.101	0.854	1.167	0.299	53.5%	1.008
Ave. Returns	0.032	0.040	0.028	0.113	0.788	1.145	0.281	53.8%	0.980
MSE	0.022	0.042	0.028	0.134	0.536	0.786	0.166	52.4%	0.994
Binary	0.000	0.043	0.029	0.313	0.011	0.016	0.001	50.2%	0.995
LSTM									
Sharpe	0.045	**0.016***	**0.011***	**0.021***	**2.804***	**3.993***	**2.177***	**59.6%***	**1.102***
Ave. Returns	0.054	0.046	0.033	0.164	1.165	1.645	0.326	54.8%	1.003
MSE	0.031	0.046	0.032	0.163	0.669	0.959	0.189	52.8%	1.003
Binary	0.012	0.039	0.026	0.255	0.300	0.454	0.046	51.0%	1.012

1. *Profitability*—expected returns (E[Returns]) and the percentage of positive returns observed across the test period
2. *Risk*—daily volatility (Vol.), downside deviation, and the maximum drawdown (MDD) of the overall portfolio
3. *Performance ratios*—Risk-adjusted performance was measured by the Sharpe ratio $\left(\frac{\mathbb{E}[\text{Returns}]}{\text{Vol.}}\right)$, Sortino ratio $\left(\frac{\mathbb{E}[\text{Returns}]}{\text{Downside deviation}}\right)$, and Calmar ratio $\left(\frac{\mathbb{E}[\text{Returns}]}{\text{MDD}}\right)$, as well as the average profit over the average loss $\left(\frac{\text{Ave. P}}{\text{Ave. L}}\right)$.

Results and Discussion

Aggregating the out-of-sample predictions from 1995 to 2015 (1990–1995 was used for training only), we compute performance metrics for both the strategy returns based on Equation 1 (Exhibit 2) and for portfolios with an additional layer of volatility scaling, which brings overall strategy returns to match the 15% volatility target (Exhibit 3). Given the large differences in returns volatility seen in Exhibit 2, this rescaling also helps to facilitate comparisons between the cumulative returns of different strategies, which are plotted for various loss functions in Exhibit 4. We note that strategy returns in this section are computed in the absence of transaction costs, allowing us to focus on the raw predictive ability of the models themselves. The impact of transaction costs is explored further in the next section, where we undertake a deeper analysis of signal turnover.

Focusing on the raw signal outputs, the Sharpe ratio–optimized LSTM outperforms all benchmarks as expected, improving the best neural network model (Sharpe-optimized MLP) by 44% and the best reference benchmark (Sgn(Returns)) by more than two times. In conjunction with Sharpe ratio improvements to both the linear and MLP models, this highlights the benefits

EXHIBIT 3
Performance Metrics—Rescaled to Target Volatility

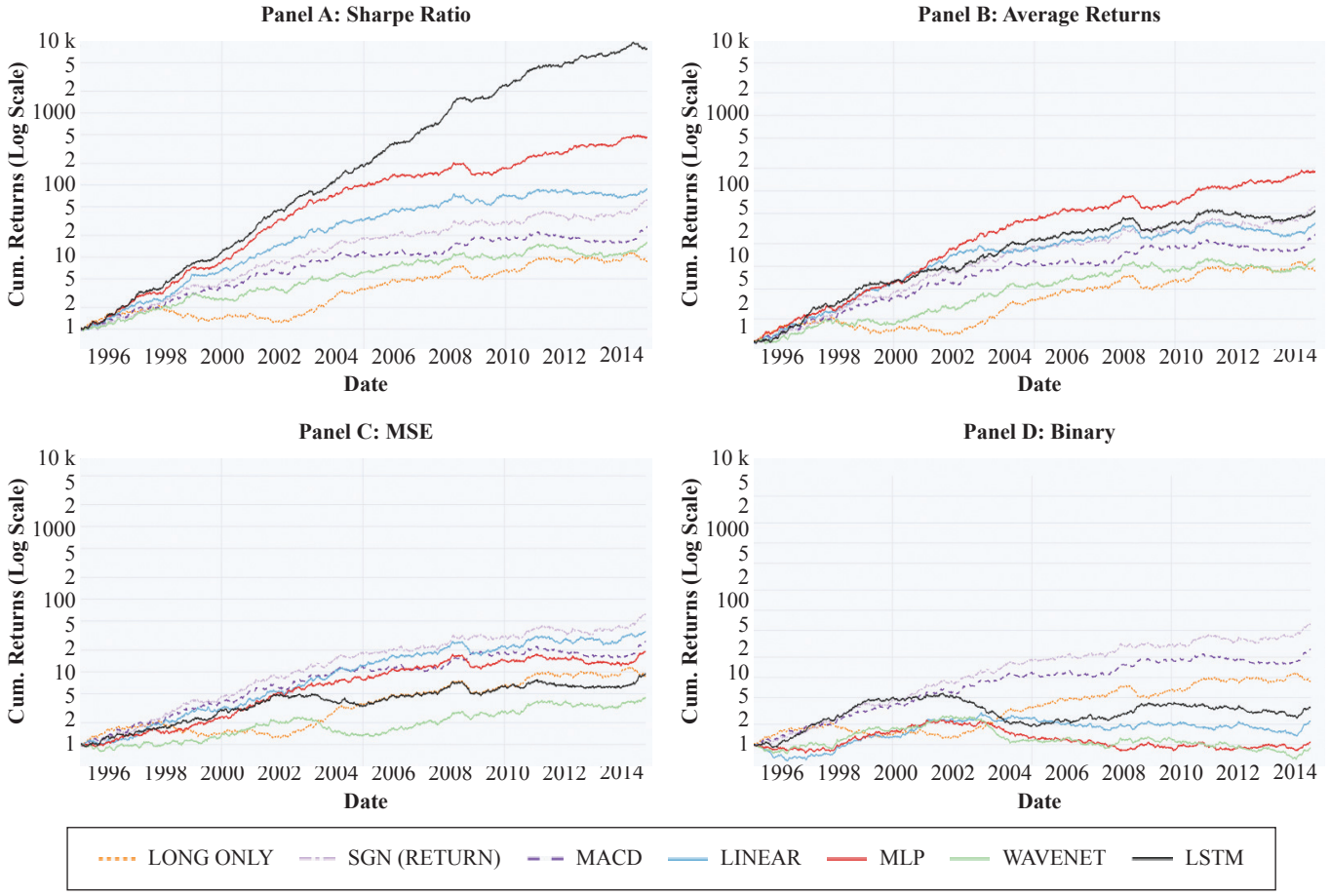
	E[Return]	Vol.	Downside Deviation	MDD	Sharpe	Sortino	Calmar	% of Positive Returns	Ave. P / Ave. L
Reference									
Long Only	0.117	0.154	0.102	0.431	0.759	1.141	0.271	53.8%	0.973
Sgn(Returns)	0.215	0.154	0.102	0.264	1.392	2.108	0.815	54.8%	1.041
MACD	0.172	0.155	0.106	0.317	1.111	1.622	0.543	53.9%	1.031
Linear									
Sharpe	0.232	0.155	0.103	0.303	1.496	2.254	0.765	54.9%	1.056
Ave. Returns	0.189	0.154	0.100	0.372	1.225	1.893	0.507	53.9%	1.047
MSE	0.186	0.154	**0.099***	0.365	1.211	1.889	0.509	54.3%	1.025
Binary	0.051	0.155	0.103	0.558	0.332	0.496	0.092	50.6%	1.033
MLP									
Sharpe	0.312	0.154	0.102	0.335	2.017	3.042	0.930	56.0%	1.104
Ave. Returns	0.266	0.154	**0.099***	0.354	1.731	2.674	0.752	55.6%	1.065
MSE	0.156	0.154	**0.099***	0.371	1.017	1.582	0.422	52.7%	1.062
Binary	0.017	0.154	0.102	0.661	0.108	0.162	0.025	50.8%	0.986
WaveNet									
Sharpe	0.148	0.155	0.103	0.349	0.956	1.429	0.424	53.5%	1.018
Ave. Returns	0.136	0.154	0.101	0.356	0.881	1.346	0.381	53.8%	0.993
MSE	0.084	**0.153***	0.101	0.459	0.550	0.837	0.184	52.4%	0.995
Binary	0.007	0.155	0.103	0.779	0.045	0.068	0.009	50.2%	1.001
LSTM									
Sharpe	**0.451***	0.155	0.105	**0.209***	**2.907***	**4.290***	**2.159***	**59.6%***	**1.113***
Ave. Returns	0.208	0.154	0.102	0.365	1.349	2.045	0.568	54.8%	1.028
MSE	0.121	0.154	0.100	0.362	0.791	1.211	0.335	52.8%	1.020
Binary	0.075	0.155	**0.099***	0.682	0.486	0.762	0.110	51.0%	1.043

of using models that capture nonlinear relationships and have access to more time history via an internal memory state. Additional model complexity, however, does not necessarily lead to better predictive performance, as demonstrated by the underperformance of WaveNet compared to both the reference benchmarks and simple linear models. Part of this can be attributed to the difficulties in tuning models with multiple design parameters—for instance, better results could possibly achieved by using alternative dilation rates, number of convolutional layers, and hidden state sizes in Equations 23 to 25 for the WaveNet. In contrast, only a single design parameter is sufficient to specify the hidden state size in both the MLP and LSTM models. Analyzing the relative performance within each model class, we can see that models that directly generate positions perform the best, demonstrating the benefits of simultaneous learning both trend estimation and position sizing functions. In addition, with the exception of a slight decrease in the MLP, Sharpe-optimized models outperform returns-optimized ones, with standard regression and classification benchmarks taking third and fourth place, respectively.

From Exhibit 3, although the addition of volatility scaling at the portfolio level improved performance ratios on the whole, it had a larger beneficial effect on machine learning models compared to the reference benchmarks, propelling Sharpe-optimized MLPs to outperform returns-optimized ones and even leading to Sharpe-optimized linear models beating reference benchmarks. From a risk perspective, we can see that both volatility and downside deviation also become a lot more comparable, with the former hovering close to 15.5% and the latter around 10%. However, Sharpe-optimized LSTMs still retained the lowest MDD across

EXHIBIT 4

Cumulative Returns—Rescaled to Target Volatility



all models, with superior risk-adjusted performance ratios across the board.

Referring to the cumulative returns plots for the rescaled portfolios in Exhibit 4, the benefits of direct outputs with Sharpe ratio optimization can also be observed, with larger cumulative returns observed for linear, MLP, and LSTM models compared to the reference benchmarks. Furthermore, we note the general underperformance of models that use standard regression and classification methods for trend estimation, hinting at the difficulties faced in selecting an appropriate position sizing function and in optimizing models to generate positions without accounting for risk. This is particularly relevant for binary classification methods, which produce relatively flat equity lines and underperform reference benchmarks in general. Some of these poor results can be explained by the implicit decision threshold adopted. From the percentage of positive returns captured in Exhibit 3, most binary classification models have about a 50% accuracy, which, although expected of a classifier with a 0.5 probability threshold, is far below the accuracy seen in other benchmarks. Furthermore, performance is made worse by the fact that the model's magnitude of gains versus losses $\left(\frac{\text{Ave. P}}{\text{Ave. L}}\right)$ is much smaller than that of competing methods, with average loss magnitudes even outweighing profits for the MLP classifier $\left(\frac{\text{Ave. P}}{\text{Ave. L}} = 0.986\right)$. As such, these observations lend support to the direct generation of position sizes with machine learning methods, given the multiple considerations (e.g., decision thresholds and profit/loss magnitudes) that would be required to incorporate standard supervising learning methods into a profitable trading strategy.

Strategy performance could also be aided by diversification across a range of assets, particularly when the correlation between signals is low. Hence, to evaluate the raw quality of the underlying signal, we investigate the performance constituents of the time-series momentum portfolios using box plots for a variety of performance metrics, plotting the minimum, lower quartile, median, upper quartile, and maximum values across individual futures contracts. We present in Exhibit 5 plots of key performance metrics, with similar results observed in other performance ratios and documented in the online supplement. In general, the Sharpe ratio plots in Exhibit 5, Panel A, echo previous findings, with direct output methods performing better than indirect trend estimation models. However, as seen in Exhibit 5, Panel C, this is mainly attributable to a significant reduction in signal volatility for the Sharpe-optimized methods, despite a comparable range of average returns in Exhibit 5, Panel B. The benefits of retaining the volatility scaling can also be observed, with individual signal volatility capped near the target across all methods—even with a naive sgn(.) position sizer. As such, the combination of volatility scaling, direct outputs, and Sharpe ratio optimization was key to performance gains in DMNs.

Feature importance. Although multiple methods for model interpretability have been proposed (Ribeiro, Singh, and Guestrin 2016; Shrikumar, Greenside, and Kundaje 2017; Lundberg and Lee 2017), they are typically formulated to independently assess the impact of exogenous inputs on model predictions at each time point. However, RNN predictions are also driven by an internal state vector, representing a compact summary of all preceding inputs. As such, feature importance at each time slice can differ depending on the history of inputs leading up to that point—even if the corresponding inputs at that time point are identical—making it difficult to apply standard techniques for interpretability. As such, to assess the importance of a given input feature across the entire length of history, we rerun the backtest, setting that feature to zero. This can be viewed as treating the feature as missing during test time, and any consequent performance degradation is reflective of its importance in generating quality predictions. From Exhibit 6—which shows the Sharpe ratio decay when a given feature is removed—we can see that the removal of daily returns results in the largest performance reduction (>2 Sharpe) across all features. This indicates that the daily returns are the most important feature driving predictions, demonstrating that the LSTM is able to build good representations directly from the raw returns. Furthermore, we also observe a Sharp ratio reduction with other features of around 1–1.5. This shows that the model is using all inputs and indicates its ability to learn meaningful relationships from the entire input data.

TURNOVER ANALYSIS

To investigate how transaction costs affect strategy performance, we first analyze the daily position changes of the signal, characterized for asset i by daily turnover $\zeta_t^{(i)}$ as defined by Baltas and Kosowski (2017):

$$\zeta_t^{(i)} = \sigma_{\text{tgt}} \left| \frac{X_t^{(i)}}{\sigma_t^{(i)}} - \frac{X_{t-1}^{(i)}}{\sigma_{t-1}^{(i)}} \right| \tag{35}$$

which is broadly proportional to the volume of asset i traded on day t with reference to the updated portfolio weights.

Exhibit 7, Panel A, shows the average strategy turnover across all assets from 1995 to 2015, focusing on positions generated by the raw signal outputs. As the box plots are charted on a logarithm scale, we note that although the machine learning–based models have a similar turnover, they also trade significantly more than the reference benchmarks—approximately 10 times more compared to the long-only benchmark. This is also reflected in Exhibit 7, Panel A, which compares the average daily returns against the average daily turnover, with ratios from machine learning models lying close to the x-axis.

To concretely quantify the impact of transaction costs on performance, we also compute the ex-cost Sharpe ratios using the rebalancing costs defined by Baltas and Kosowski (2017) to adjust our returns for a variety of transaction cost assumptions. For the results in Exhibit 8, the top of each bar chart marks the maximum cost-free Sharpe ratio of the strategy, with each colored block denoting the Sharpe ratio reduction for the corresponding cost assumption. In line with the turnover analysis, the reference benchmarks demonstrate the most resilience to high transaction costs (up to 5 bps), with the profitability across most machine learning models persisting only up to 4 bps. However, we still obtain higher

EXHIBIT 5

Performance across Individual Assets

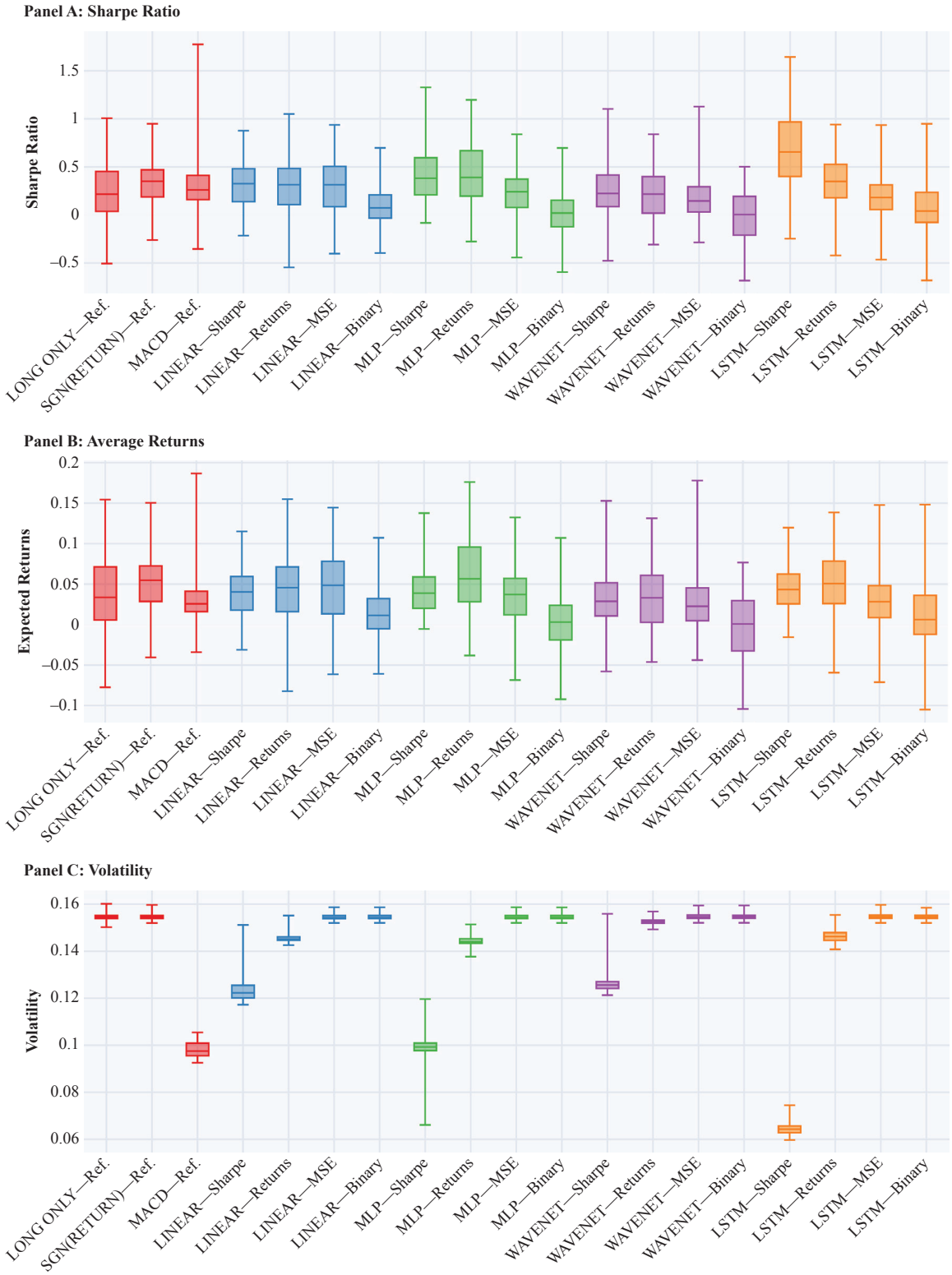
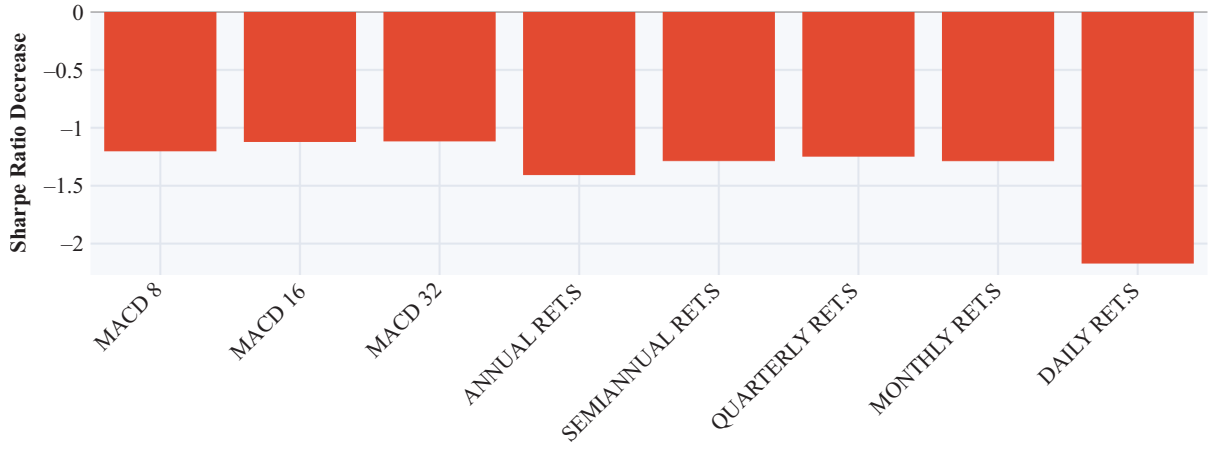


EXHIBIT 6
Feature Importance Plot



cost-adjusted Sharpe ratios with the Sharpe-optimized LSTM for up to 2–3 bps, demonstrating its suitability for trading more liquid instruments.

Turnover Regularization

One simple way to account for transaction costs is to use cost-adjusted returns $\tilde{r}_{t,t+1}^{TSMOM}$ directly during training, augmenting the strategy returns defined in Equation 1 as follows:

$$\tilde{r}_{t,t+1}^{TSMOM} = \frac{\sigma_{\text{tgt}}}{N_t} \sum_{i=1}^{N_t} \left(\frac{X_t^{(i)}}{\sigma_t^{(i)}} r_{t,t+1}^{(i)} - c \left| \frac{X_t^{(i)}}{\sigma_t^{(i)}} - \frac{X_{t-1}^{(i)}}{\sigma_{t-1}^{(i)}} \right| \right) \tag{36}$$

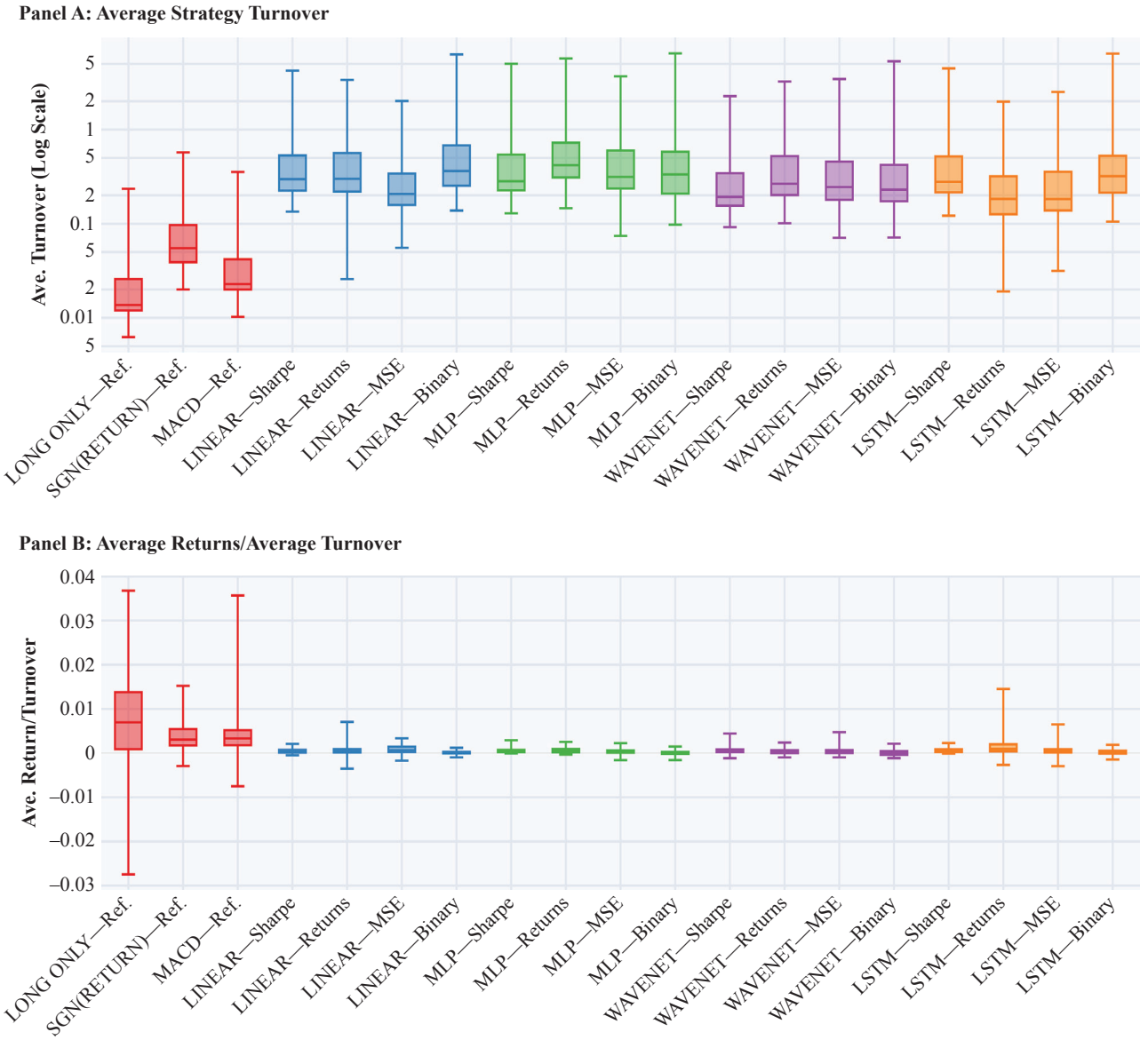
where c is a constant reflecting transaction cost assumptions. As such, using $\tilde{r}_{t,t+1}^{TSMOM}$ in Sharpe ratio loss functions during training corresponds to optimizing the ex-cost risk-adjusted returns, and $c\left|\frac{X_t^{(i)}}{\sigma_t^{(i)}} - \frac{X_{t-1}^{(i)}}{\sigma_{t-1}^{(i)}}\right|$ can also be interpreted as a regularization term for turnover.

Given that the Sharpe-optimized LSTM is still profitable in the presence of small transactions costs, we seek to quantify the effectiveness of turnover regularization when costs are prohibitively high, considering the extreme case in which $c = 10$ bps in our investigation. Tests were focused on the Sharpe-optimized LSTM with and without the turnover regularizer (LSTM + Reg. for the former), including the additional portfolio level volatility scaling to bring signal volatilities to the same level. Based on the results in Exhibit 9, we can see that the turnover regularization does help improve the LSTM in the presence of large costs, leading to slightly better performance ratios when compared to the reference benchmarks.

CONCLUSIONS

We introduce DMNs, a hybrid class of deep learning models that retains the volatility scaling framework of time-series momentum strategies while using deep neural networks to output position targeting trading signals. Two approaches to position generation were evaluated here. First, we cast trend estimation as a standard supervised learning problem—using machine learning models to forecast the expected asset returns or probability of a positive return at the next time step—and apply a simple maximum long–short trading rule based on the direction of the next return. Second, trading rules were directly generated as outputs from the model, which we calibrate by maximizing the Sharpe ratio or average strategy return. Testing this on a universe of continuous futures contracts, we demonstrate clear improvements in risk-adjusted performance by calibrating models with the Sharpe ratio, with the LSTM achieving the best results. Furthermore, we note the general underperformance of models that use standard regression and classification methods for trend estimation, hinting at the difficulties faced in selecting

EXHIBIT 7
Turnover Analysis



an appropriate position sizing function and optimizing models to generate positions without accounting for risk.

Incorporating transaction costs, the Sharpe-optimized LSTM outperforms benchmarks up to 2–3 bps of costs, demonstrating its suitability for trading more liquid assets. To accommodate high costs settings, we introduce a turnover regularizer to use during training, which was shown to be effective even in extreme scenarios (i.e., $c = 10$ bps).

Future work includes extensions of the framework presented here to incorporate ways to deal better with nonstationarity in the data, such as using the recently introduced recurrent neural filters (Lim, Zohren, and Roberts 2019). Another direction of future work focuses on the study of time-series momentum at the microstructure level.

EXHIBIT 8
Impact of Transaction Costs on Sharpe Ratio

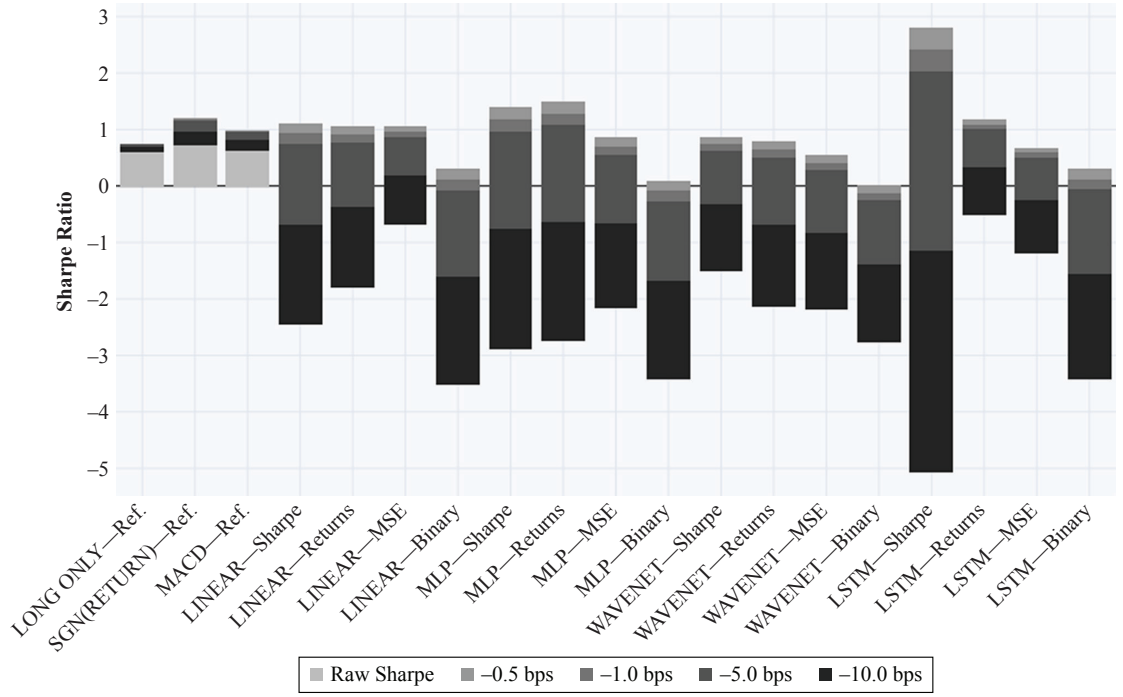


EXHIBIT 9
Performance Metrics with Transaction Costs (c = 10 bps)

	E[Return]	Vol.	Downside Deviation	MDD	Sharpe	Sortino	Calmar	% of Positive Returns	Ave. P / Ave. L
Long Only	0.097	**0.154***	0.103	0.482	0.628	0.942	0.201	53.3%	0.970
Sgn(Returns)	0.133	**0.154***	**0.102***	0.373	0.861	1.296	0.356	53.3%	1.011
MACD	0.111	0.155	0.106	0.472	0.719	1.047	0.236	52.5%	**1.020***
LSTM	–0.833	0.157	0.114	1.000	–5.313	–7.310	–0.833	33.9%	0.793
LSTM + Reg.	**+0.141***	**0.154***	**0.102***	**0.371***	**0.912***	**1.379***	**0.379***	**53.4%***	1.014

APPENDIX

DATASET DETAILS

From the full 98 ratio-adjusted continuous futures contracts in the Pinnacle Data Corp. CLC Database, we extract 88 that have <10% of data missing; a breakdown by asset class described in the following.

To reduce the impact of outliers, we also winsorize the data by capping/flooring it to be within five times its exponentially weighted moving (EWM) standard deviations from its EWM average, computed using a 252-day half-life.

HYPERPARAMETER OPTIMIZATION

Hyperparameter optimization was applied using 50 iterations of random search, with the full search grid documented in Exhibit A1, with the models fully recalibrated every five years using all available data up to that point. For LSTM-based models, time series were subdivided into trajectories of 63 time steps (≈ three months), with the LSTM unrolled across the length of the trajectory during backpropagation.

EXHIBIT A1
Hyperparameter Search Range

Hyperparameters	Random Search Grid	Notes
Dropout Rate	0.1, 0.2, 0.3, 0.4, 0.5	Neural networks only
Hidden Layer Size	5, 10, 20, 40, 80	Neural networks only
Minibatch Size	256, 512, 1024, 2048	
Learning Rate	10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}	
Max Gradient Norm	10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}	
L1 Regularization Weight (α)	10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}	Lasso regression only

Identifier	Description	Identifier	Description
Commodities		**Commodities**	
BC	BRENT CRUDE OIL, comp.	ZS	SOYBEANS, Electronic
BG	BRENT GASOIL, comp.	ZT	LIVE CATTLE, Electronic
BO	SOYBEAN OIL	ZU	CRUDE OIL, Electronic
CC	COCOA	ZW	WHEAT, Electronic
CL	CRUDE OIL	ZZ	LEAN HOGS, Electronic
CT	COTTON #2	**Equities**	
C_	CORN	AX	GERMAN DAX INDEX
DA	MILK III, Comp.	CA	CAC40 INDEX
FC	FEEDER CATTLE	EN	NASDAQ, MINI
GC	GOLD (COMMEX)	ER	RUSSELL 2000, MINI
GI	GOLDMAN SAKS C. I.	ES	S&P 500, MINI
HG	COPPER	HS	HANG SENG
HO	HEATING OIL #2	LX	FTSE 100 INDEX
JO	ORANGE JUICE	MD	S&P 400 (Mini electronic)
KC	COFFEE	SC	S&P 500, composite
KW	WHEAT, KC	SP	S&P 500, day session
LB	LUMBER	XU	DOW JONES EUROSTOXX50
LC	LIVE CATTLE	XX	DOW JONES STOXX 50
LH	LIVE HOGS	YM	Mini Dow Jones ($5.00)
MW	WHEAT, MINN	**Fixed Income**	
NG	NATURAL GAS	AP	AUSTRALIAN PRICE INDEX
NR	ROUGH RICE	DT	EURO BOND (BUND)
O_	OATS	FA	T-NOTE, 5-yr day session
PA	PALLADIUM	FB	T-NOTE, 5-yr composite
PL	PLATINUM	GS	GILT, LONG BOND
RB	RBOB GASOLINE	TA	T-NOTE, 10-yr day session
SB	SUGAR #11	TD	T-NOTES, 2-yr day session
SI	SILVER (COMMEX)	TU	T-NOTES, 2-yr composite
SM	SOYBEAN MEAL	TY	T-NOTE, 10-yr composite
S_	SOYBEANS	UA	T-BONDS, day session
W_	WHEAT, CBOT	UB	EURO BOBL
ZA	PALLADIUM, electronic	US	T-BONDS, composite
ZB	RBOB, Electronic	**FX**	
ZC	CORN, Electronic	AD	AUSTRALIAN $$, day session
ZF	FEEDER CATTLE, Electronic	AN	AUSTRALIAN $$, composite
ZG	GOLD, Electronic	BN	BRITISH POUND, composite
ZH	HEATING OIL, electronic	CB	CANADIAN 10YR BOND
ZI	SILVER, Electronic	CN	CANADIAN $$, composite
ZK	COPPER, electronic	DX	US DOLLAR INDEX
ZL	SOYBEAN OIL, Electronic	FN	EURO, composite
ZM	SOYBEAN MEAL, Electronic	FX	EURO, day session
ZN	NATURAL GAS, electronic	JN	JAPANESE YEN, composite
ZO	OATS, Electronic	MP	MEXICAN PESO
ZP	PLATINUM, electronic	NK	NIKKEI INDEX
ZR	ROUGH RICE, Electronic	SF	SWISS FRANC, day session
		SN	SWISS FRANC, composite

ACKNOWLEDGMENTS

We would like to thank Anthony Ledford, James Powrie, and Thomas Flury for their interesting comments, as well the Oxford-Man Institute of Quantitative Finance for financial support.

REFERENCES

Abadi, M., A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Chemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. *TensorFlow: Large-scale Machine Learning on Heterogeneous Systems*. 2015, tensorflow.org.

Baltas, N., and R. Kosowski. "Demystifying Time-Series Momentum Strategies: Volatility Estimators, Trading Rules and Pairwise Correlations." *SSRN*, 2017.

Bao, W., J. Yue, and Y. Rao. 2017. "A Deep Learning Framework for Financial Time Series Using Stacked Autoencoders and Long–Short Term Memory." *PLOS ONE* 12 (7): 1–24.

Barroso, P., and P. Santa-Clara. 2015. "Momentum Has Its moments." *Journal of Financial Economics* 116 (1): 111–120.

Baz, J., N. Granger, H. Nicolas, R. Campbell, N. Le Roux, and S. Rattray. "Dissecting Investment Strategies in the Cross Section and Time Series." *SSRN*, 2015.

Bengio, Y., A. Courville, and P. Vincent. 2013. "Representation Learning: A Review and New Perspectives." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35 (8): 1798–1828.

Binkowski, M., G. Marti, and P. Donnat. 2018. "Autoregressive Convolutional Neural Networks for Asynchronous Time Series." *Proceedings of the 35th International Conference on Machine Learning* 80: 580–589.

Bruder, B., T. L. Dao, J. C. Richard, and T. Roncalli. "Trend Filtering Methods for Momentum Strategies." *SSRN*, 2013.

Bühler, H., L. Gonon, J. Teichmann, and B. Wood. 2018. "Deep Hedging." *arXiv e-prints* arXiv:1802.03042.

Daniel, K., and T. J. Moskowitz. 2016. "Momentum Crashes." *Journal of Financial Economics* 122 (2): 221–247.

Fraccaro, M., S. Kamronn, U. Paquet, and O. Winther. "A Disentangled Recognition and Nonlinear Dynamics Model for Unsupervised Learning." *Advances in Neural Information Processing Systems 30 (NIPS)*, 2017.

Gal, Y., and Z. Ghahramani. "A Theoretically Grounded Application of Dropout in Recurrent Neural Networks." *Advances in Neural Information Processing Systems 29 (NIPS)*, 2016.

Ghoshal, S., and S. Roberts. "Thresholded ConvNet Ensembles: Neural Networks for Technical Forecasting." *Data Science in Fintech Workshop—Conference on Knowledge Discover and Data Mining (KDD)*, 2018.

Goodfellow, I., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. "Generative Adversarial Nets." *Advances in Neural Information Processing Systems 27 (NIPS)*, 2014.

Goodfellow, I., Y. Bengio, and A. Courville. 2016. *Deep Learning*. Cambridge, MA: MIT Press, 2016.

Gu, S., B. T. Kelly, and D. Xiu. "Empirical Asset Pricing via Machine Learning." Chicago Booth research paper no. 18-04, 31st Australasian Finance and Banking Conference, 2017.

Harvey, C. R., E. Hoyle, R. Korgaonkar, S. Rattray, M. Sargaison, and O. van Hemert. "The Impact of Volatility Targeting." *SSRN*, 2018.

Hochreiter, S., and J. Schmidhuber. 1997. "Long Short-Term Memory." *Neural Computation* 9 (8): 1735–1780.

Hurst, B., Y. H. Ooi, and L. H. Pedersen. 2017. "A Century of Evidence on Trend-Following Investing." *The Journal of Portfolio Management* 44 (1): 15–29.

Jegadeesh, N., and S. Titman. 1993. "Returns to Buying Winners and Selling Losers: Implications for Stock Market Efficiency." *The Journal of Finance* 48 (1): 65–91.

———. 2001. "Profitability of Momentum Strategies: An Evaluation of Alternative Explanations." *The Journal of Finance* 56 (2): 699–720.

Jusselin, P., E. Lezmi, H. Malongo, C. Masselin, T. Roncalli, and T. L. Dao. "Understanding the Momentum Risk

Premium: An In-Depth Journey through Trend-following Strategies." *SSRN*, 2017.

Kim, A. Y., Y. Tse, and J. K. Wald. 2016. "Time Series Momentum and Volatility Scaling." *Journal of Financial Markets* 30: 103–124.

Kim, S. 2019. "Enhancing the Momentum Strategy through Deep Regression." *Quantitative Finance* 19 (7): 1121–1133.

Kingma, D., and J. Ba. "Adam: A Method for Stochastic Optimization." *International Conference on Learning Representations (ICLR)*, 2015.

Kolm, P. N., and G. Ritter. 2019. "Dynamic Replication and Hedging: A Reinforcement Learning Approach." *The Journal of Financial Data Science* 1 (1): 159–171.

Laptev, N., J. Yosinski, L. E. Li, and S. Smyl. "Time-Series Extreme Event Forecasting with Neural Networks at Uber." Presented at Time Series Workshop—International Conference on Machine Learning (ICML), 2017.

Lempérière, Y., C. Deremble, P. Seager, M. Potters, and J.-P. Bouchaud. 2014. "Two Centuries of Trend Following." *Journal of Investment Strategies* 3 (3): 41–61.

Levine, A., and L. H. Pedersen. 2016. "Which Trend Is Your Friend." *Financial Analysts Journal* 72 (3).

Lim, B., and M. van der Schaar. 2018. "Disease-Atlas: Navigating Disease Trajectories Using Deep Learning." *Proceedings of the 3rd Machine Learning for Healthcare Conference (MLHC)* 85: 137–160.

Lim, B., S. Zohren, and S. Roberts. 2019. "Recurrent Neural Filters: Learning Independent Bayesian Filtering Steps for Time Series Prediction." *arXiv e-prints*, arXiv:1901.08096.

Lundberg, S., and S.-I. Lee. "A Unified Approach to Interpreting Model Predictions." *Advances in Neural Information Processing Systems 30 (NIPS)*, 2017.

Makridakis, S., E. Spiliotis, and V. Assimakopoulos. 2018. "The M4 Competition: Results, Findings, Conclusion and Way Forward." *International Journal of Forecasting* 34 (4): 802–808.

Martins, R., and D. Zou. "Momentum Strategies Offer a Positive Point of Skew." *Risk Magazine*, 2012.

Moskowitz, T. J., Y. H. Ooi, and L. H. Pedersen. 2012. "Time Series Momentum." *Journal of Financial Economics* 104 (2): 228–250.

Paszke, A., S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. "Automatic Differentiation in PyTorch." *Autodiff Workshop—Conference on Neural Information Processing (NIPS)*, 2017.

Potters, M., and J.-P. Bouchaud. "Trend Followers Lose More Than They Gain." *Wilmott Magazine*, 2016.

Rangapuram, S. S., M. W. Seeger, J. Gasthaus, L. Stella, Y. Wang, and T. Januschowski. "Deep State Space Models for Time Series Forecasting." *Advances in Neural Information Processing Systems 31 (NeurIPS)*, 2018.

Ribeiro, M. T., S. Singh, and C. Guestrin. " 'Why Should I Trust You?': Explaining the Predictions of Any Classifier." *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

Rohrbach, J., S. Suremann, and J. Osterrieder. "Momentum and Trend Following Trading Strategies for Currencies Revisited—Combining Academia and Industry." *SSRN*, 2017.

Rotando, L. M., and E. O. Thorp. 1992. "The Kelly Criterion and the Stock Market." *The American Mathematical Monthly* 99 (10): 922–931.

Sharpe, W. F. 1994. "The Sharpe Ratio." *The Journal of Portfolio Management* 21 (1): 49–58.

Shrikumar, A., P. Greenside, and A. Kundaje. "Learning Important Features through Propagating Activation Differences." *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017.

Silver, D., J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis. 2017. "Mastering the Game of Go without Human Knowledge." *Nature* 550: 354–359.

Sirignano, J., and R. Cont. "Universal Features of Price Formation in Financial Markets: Perspectives from Deep Learning." *SSRN*, 2018.

Smyl, S., J. Ranganathan, and A. Pasqua. *M4 Forecasting Competition: Introducing a New Hybrid ES-RNN Model*, 2018.

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

van den Oord, A., S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu. 2016. "WaveNet: A Generative Model for Raw Audio." arXiv:1609.03499.

Zhang, Z., S. Zohren, and S. Roberts. 2018. "BDLOB: Bayesian Deep Convolutional Neural Networks for Limit Order Books." *Bayesian Deep Learning Workshop—Conference on Neural Information Processing (NeurIPS)*, 2018.

——. 2019. "DeepLOB: Deep Convolutional Neural Networks for Limit Order Books." *IEEE Transactions on Signal Processing*, 2019.

ADDITIONAL READING

A Century of Evidence on Trend-Following Investing

BRIAN HURST, YAO HUA OOI, AND LASSE HEJE PEDERSEN

The Journal of Portfolio Management
https://jpm.pm-research.com/content/44/1/15

ABSTRACT: *In this article, the authors study the performance of trend-following investing across global markets since 1880, extending the existing evidence by more than 100 years using a novel data set. They find that in each decade since 1880, time-series momentum has delivered positive average returns with low correlations to traditional asset classes. Further, time-series momentum has performed well in 8 out of 10 of the largest crisis periods over the century, defined as the largest drawdowns for a 60/40 stock/bond portfolio. Lastly, the authors find that time-series momentum has performed well across different macro environments, including recessions and booms, war and peace, high- and low-interest-rate regimes, and high- and low-inflation periods.*

The Deflated Sharpe Ratio: *Correcting for Selection Bias, Backtest Overfitting, and Non-Normality*

DAVID H. BAILEY AND MARCOS LÓPEZ DE PRADO

The Journal of Portfolio Management
https://jpm.pm-research.com/content/40/5/94

ABSTRACT: *With the advent in recent years of large financial data sets, machine learning, and high-performance computing, analysts can back test millions (if not billions) of alternative investment strategies. Backtest optimizers search for combinations of parameters that maximize the simulated historical performance of a strategy, leading to back test overfitting. The problem of performance inflation extends beyond back testing. More generally, researchers and investment managers tend to report only positive outcomes, a phenomenon known as selection bias. Not controlling for the number of trials involved in a particular discovery leads to overly optimistic performance expectations. The deflated Sharpe ratio (DSR) corrects for two leading sources of performance inflation: Selection bias under multiple testing and non-normally distributed returns. In doing so, DSR helps separate legitimate empirical findings from statistical flukes.*

On Default Correlation

A Copula Function Approach

DAVID X. LI

The Journal of Fixed Income
https://jfi.pm-research.com/content/9/4/43

ABSTRACT: *This article studies the problem of default correlation. It introduces a random variable called "time-until-default" to denote the survival time of each defaultable entity or financial instrument, and defines the default correlation between two credit risks as the correlation coefficient between their survival times. The author explains why a copula function approach should be used to specify the joint distribution of survival times after marginal distributions of survival times are derived from market information, such as risky bond prices or asset swap spreads. He shows that the current approach to default correlation through asset correlation is equivalent to using a normal copula function. Numerical examples illustrate the use of copula functions in the valuation of some credit derivatives, such as credit default swaps and first-to-default contracts.*