Deep Reinforcement Learning for Trading

ZIHAO ZHANG, STEFAN ZOHREN, AND STEPHEN ROBERTS

ZIHAO ZHANG
is a D.Phil. student with the Oxford-Man Institute of Quantitative Finance and the Machine Learning Research Group at the University of Oxford in Oxford, UK.
zihao.zhang@worc.ox.ac.uk

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

STEPHEN ROBERTS
is the director of the Oxford-Man Institute of Quantitative Finance, the founding director of the Oxford Centre for Doctoral Training in Autonomous Intelligent Machines and Systems, and the Royal Academy of Engineering/Man Group Professor in the Machine Learning Research Group at the University of Oxford in Oxford, UK.
sjrob@robots.ox.ac.uk

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

KEY FINDINGS

- In this article, the authors introduce reinforcement learning algorithms to design trading strategies for futures contracts. They investigate both discrete and continuous action spaces and improve reward functions by using volatility scaling to scale trade positions based on market volatility.
- The authors discuss the connection between modern portfolio theory and the reinforcement learning reward hypothesis and show that they are equivalent if a linear utility function is used.
- The authors back test their methods on 50 very liquid futures contracts from 2011 to 2019, and their algorithms deliver positive profits despite heavy transaction costs.

ABSTRACT: *In this article, the authors adopt deep reinforcement learning algorithms to design trading strategies for continuous futures contracts. Both discrete and continuous action spaces are considered, and volatility scaling is incorporated to create reward functions that scale trade positions based on market volatility. They test their algorithms on 50 very liquid futures contracts from 2011 to 2019 and investigate how performance varies across different asset classes, including commodities, equity indexes, fixed income, and foreign exchange markets. They compare their algorithms against classical time-series momentum strategies and show that their method outperforms such baseline models, delivering positive profits despite heavy transaction costs. The experiments show that the proposed algorithms can follow large market trends without changing positions and can also scale down, or hold, through consolidation periods.*

TOPICS: ***Futures and forward contracts, exchanges/markets/clearinghouses, statistical methods, simulations****

Financial trading has been a widely researched topic, and a variety of methods have been proposed to trade markets over the last few decades. These include fundamental analysis (Graham and Dodd 1934), technical analysis (Murphy 1999), and algorithmic trading (Chan 2009). Indeed, many practitioners use a hybrid of these techniques to make trades (Schwager 2017). Algorithmic trading has arguably gained most recent interest and accounts for about 75% of trading volume in the US stock exchanges (Chan 2009). The advantages of algorithmic trading are widespread, ranging from strong computing foundations to faster execution and risk diversification. One key

component of such trading systems is a predictive signal that can lead to alpha (excess return); to this end, mathematical and statistical methods are widely applied. However, because of the low signal-to-noise ratio of financial data and the dynamic nature of markets, the design of these methods is nontrivial, and the effectiveness of commonly derived signals varies through time.

In recent years, machine learning algorithms have gained much popularity in many areas, with notable successes in diverse application domains, including image classification (Krizhevsky, Sutskever, and Hinton 2012) and natural language processing (Collobert et al. 2011). Similar techniques have also been applied to financial markets in an attempt to find higher alpha (for a few examples using such techniques in the context of high-frequency data, see Tsantekidis et al. 2017a; Zhang, Zohren, and Roberts 2018, 2019a, 2019b; Sirignano and Cont 2019). Most research focuses on regression and classification pipelines in which excess returns, or market movements, are predicted over some (fixed) horizons. However, there is little discussion related to transforming these predictive signals into actual trade positions (see Lim, Zohren, and Roberts 2019 for an attempt to train a deep learning model to learn positions directly). Indeed, such a mapping is nontrivial. As an example, predictive horizons are often relatively short (one day or a few days ahead if using daily data); however, large trends can persist for weeks or months with some moderate consolidation periods. We therefore need a signal that not only has good predictive power but also can consistently produce good directional calls.

In this article, we report on reinforcement learning (RL) (Sutton and Barto 1998) algorithms to tackle the aforementioned problems. Instead of making predictions—followed by a trade decision based on predictions—we train our models to directly output trade positions, bypassing the explicit forecasting step. Modern portfolio theory (Arrow 1971; Pratt 1978; Ingersoll 1987) implies that, given a finite time horizon, an investor chooses actions to maximize some expected utility of final wealth:

$$\mathbb{E}[U(W_T)] = \mathbb{E}\left[U\left(W_0 + \sum_{t=1}^{T} \delta W_t\right)\right] \tag{1}$$

where U is the utility function, W_T is the final wealth over a finite horizon T, and δW_t represents the change in wealth. The performance of the final wealth measure depends upon sequences of interdependent actions in which optimal trading decisions do not just decide immediate trade returns but also affect subsequent future returns. As mentioned by Merton (1969) and Ritter (2017), this falls under the framework of optimal control theory (Kirk 2012) and forms a classical sequential decision-making process. If the investor is risk neutral, the utility function becomes linear, and we only need to maximize the expected cumulative trades returns, $\mathbb{E}(\Sigma_{t=1}^{T}\delta W_t)$; we observe that the problem fits exactly with the framework of RL, the goal of which is to maximize some expected cumulative rewards via an agent interacting with an uncertain environment. Under the RL framework, we can directly map different market situations to trade positions and conveniently include market frictions, such as commissions, in our reward functions, allowing trading performance to be directly optimized.

In this article, we adopt state-of-the-art RL algorithms to the aforementioned problem setting, including deep Q-learning networks (DQN) (Mnih et al. 2013; Van Hasselt, Guez, and Silver 2016), policy gradients (PG) (Williams 1992), and advantage actor–critic (A2C) (Mnih et al. 2016). Both discrete and continuous action spaces are explored in our work, and we improve reward functions with volatility scaling (Moskowitz, Ooi, and Pedersen 2012; Harvey et al. 2018) to scale up trade positions when volatility is low, and vice versa. We show the robustness and consistency of our algorithms by testing them on 50 very liquid futures contracts (CLC Database 2019) between 2011 and 2019. Our dataset consists of different asset classes, including commodities, equity indexes, fixed income, and foreign exchange (FX) markets. We compare our method with classical time-series momentum strategies and find that our method outperforms baseline models and generates positive returns in all sectors despite heavy transaction costs. Time-series strategies work well in trending markets, such as fixed-income markets, but suffer losses in FX markets in which directional moves are less usual. Our experiments show that our algorithms can monetize on large moves without changing positions and deal with markets that are more mean reverting.

The remainder of the article is structured as follows. We first introduce the current literature and present our methodology. We then compare our methods with baseline algorithms and show how trading performance

varies among asset classes; a description of the dataset and training methodology is also included. At the end, we offer our conclusions and discuss extensions of our work.

LITERATURE REVIEW

In this section, we review some classical trading strategies and discuss how RL has been applied to this field. Fundamental analysis aims to measure the intrinsic value of a security by examining economic data so investors can compare the security's current price with estimates to determine whether the security is undervalued or overvalued. One of the established strategies is called CAN-SLIM (O'Neil 2009), and it is based on a major study of market winners from 1880 to 2009. A common criticism of fundamental analysis is that the entrance and exit timing of trades is not specified. Even when markets move toward the estimated price, bad timing in entering trades could lead to huge drawdowns, and such moves in account values are often not bearable to investors, shaking them out of the markets. Technical analysis is in contrast to fundamental analysis, in which a security's historical price data are used to study price patterns. Technicians place trades based on a combination of indicators such as the relative strength index (RSI) and Bollinger bands. However, owing to the lack of analysis on economic or market conditions, the predictability of these signals is not strong, often leading to false breakouts.

Algorithmic trading is a more systematic approach that involves mathematical modeling and automated execution. Examples include trend following (Schwager 2017), mean-reversion (Chan 2013), statistical arbitrage (Chan 2009), and delta-neutral trading strategies (Michaud 1989). We mainly review time-series momentum strategies by Moskowitz, Ooi, and Pedersen (2012) because we benchmark our models against their algorithms. Their work developed a very robust trading strategy by simply taking the sign of returns over the last year as a signal; they demonstrated profitability for every contract considered within 58 liquid instruments over 25 years. Thenceforth, a number of methods (Baz et al. 2015; Baltas and Kosowski 2017; Rohrbach, Suremann, and Osterrieder 2017; Lim, Zohren, and Roberts 2019) have been proposed to enhance this strategy by estimating the trend and mapping to actual trade positions. However, these strategies are designed to profit from large directional moves but can suffer huge losses if markets move sideways as the predictability of these signals deteriorates and excess turnover erodes profitability. Here, we adopt time-series momentum features along with technical indicators to represent state space and obtain trade positions directly using RL. The idea of our representation is simple: extracting information across different momentum features and outputting positions based on the aggregated information.

The current literature on RL in trading can be categorized into three main methods: critic-only, actor-only, and actor–critic approaches (Fischer 2018). The critic approach, mainly DQN, features most often in publications in this field (Tan, Quek, and Cheng 2011; Bertoluzzo and Corazza 2012; Jin and El-Saawy 2016; Ritter 2017; Huang 2018). A state-action value function, Q, is constructed to represent how good a particular action is in a state. Discrete action spaces are adopted in these works, and an agent is trained to fully go long or short a position. However, a fully invested position is risky during high-volatility periods, exposing one to severe risk when opposite moves occur. Ideally, one would like to scale positions up or down according to current market conditions. Doing this requires one to have large action spaces; however, the critic approach suffers from large action spaces because a score must be assigned for each possible action.

The second most common approach is the actor-only approach (Moody et al. 1998; Moody and Saffell 2001; Deng et al. 2016; Lim, Zohren, and Roberts 2019), in which the agent directly optimizes the objective function without computing the expected outcomes of each action in a state. Because a policy is directly learned, actor-only approaches can be generalized to continuous action spaces. In the works of Moody and Saffell (2001) and Lim, Zohren, and Roberts (2019), offline batch gradient ascent methods can be used to optimize the objective function, such as profits or Sharpe ratio, because the approach is differentiable end to end. However, this is different from standard RL actor-only approaches in which a distribution needs to be learned for the policy. To study the distribution of a policy, the policy gradient theorem (Sutton and Barto 1998) and Monte Carlo methods (Metropolis and Ulam 1949) are adopted in the training, and models are updated until the end of each episode. We often experience slow learning and need many samples to obtain an optimal policy because individual bad actions will be considered

good as long as the total rewards are good; thus, a long time is needed to adjust these actions.

The actor–critic approach forms the third category and aims to solve the aforementioned learning problems by updating the policy in real time. The key idea is to alternatively update two models in which one (the actor) controls how an agent performs given the current state, and the other (the critic) measures how good the chosen action is. However, this approach is the least studied method in financial applications (Li et al. 2007; Bekiros 2010; Xiong et al. 2018). We aim to supplement the literature and study the effectiveness of this approach in trading. For a more detailed discussion on state, action spaces, and reward functions, interested readers are pointed to the survey (Fischer 2018). Other important financial applications such as portfolio optimization and trade execution are also included in this work.

METHODOLOGY

In this section, we introduce our setups, including state, action spaces, and reward functions. We describe three RL algorithms used in our work, DQN, PG, and A2C methods.

Markov Decision Process Formalization

We can formulate the trading problem as a Markov decision process in which an agent interacts with the environment at discrete time steps. At each time step t, the agent receives some representations of the environment denoted as a state S_t. Given this state, an agent chooses an action A_t, and based on this action, a numerical reward R_{t+1} is given to the agent at the next time step, and the agent finds itself in a new state S_{t+1}. The interaction between the agent and the environment produces a trajectory $\tau = [S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, \ldots]$. At any time step t, the goal of RL is to maximize the expected return (essentially the expected discounted cumulative rewards) denoted as G_t at time t:

$$G_t = \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k \tag{2}$$

where γ is the discounting factor. If the utility function in Equation 1 has a linear form and we use R_t to represent trade returns, we can see that optimizing $\mathbb{E}(G)$ is equivalent to optimizing our expected wealth.

State space. In the literature, many different features have been used to represent state spaces. Among these features, a security's past price is always included, and the related technical indicators are often used (Fischer 2018). In our work, we take past price, returns (r_t) over different horizons, and technical indicators including moving average convergence divergence (MACD) (Baz et al. 2015) and the RSI (Wilder 1978) to represent states. At a given time step, we take the past 60 observations of each feature to form a single state. The following is a list of our features:

- Normalized close price series
- Returns over the past month, two months, three months, and one year are used. Following Lim, Zohren, and Roberts (2019), we normalize them by daily volatility adjusted to a reasonable time scale. As an example, we normalize annual returns as $r_{t-252,t}/(\sigma_t\sqrt{252})$, where σ_t is computed using an exponentially weighted moving standard deviation of r_t with a 60-day span,
- MACD indicators are proposed by Baz et al. (2015) where

$$\begin{aligned} \text{MACD}_t &= \frac{q_t}{\text{std}(q_{t-252:t})} \\ q_t &= (m(S) - m(L))/\text{std}(p_{t-63:t}) \end{aligned} \tag{3}$$

  where $\text{std}(p_{t-63:t})$ is the 63-day rolling standard deviation of prices p_t, and $m(S)$ is the exponentially weighted moving average of prices with a time scale S.
- The RSI is an oscillating indicator moving between 0 and 100. It indicates the oversold (a reading below 20) or overbought (above 80) condition of an asset by measuring the magnitude of recent price changes. We include this indicator with a look-back window of 30 days in our state representations.

Action space. We study both discrete and continuous action spaces. For discrete action spaces, a simple action set of {−1, 0, 1} is used, and each value represents the position directly (i.e., −1 corresponds to a maximally short position, 0 to no holdings, and 1 to a maximally long position). This representation of action space is also known as target orders (Huang 2018), in which a trade position is the output instead of the trading decision.

Note that if the current action and next action are the same, no transaction cost will occur and we just maintain our positions. If we move from a fully long position to a short position, transaction cost will be doubled. The design of continuous action spaces is very similar to the discrete case in which we still output trade positions but allow actions to be any value between −1 and 1 ($A_t \in [-1, 1]$).

Reward function. The design of the reward function depends on the utility function in Equation 1. Here, we let the utility function be profits representing a risk-insensitive trader, and the reward R_t at time t is

$$R_t = \mu\left[A_{t-1}\frac{\sigma_{tgt}}{\sigma_{t-1}} r_t - \text{bp}\ p_{t-1}\left|\frac{\sigma_{tgt}}{\sigma_{t-1}} A_{t-1} - \frac{\sigma_{tgt}}{\sigma_{t-2}} A_{t-2}\right|\right] \quad (4)$$

where σ_{tgt} is the volatility target and σ_{t-1} is an ex ante volatility estimate calculated using an exponentially weighted moving standard deviation with a 60-day window on r_t. This expression forms the volatility scaling of Moskowitz, Ooi, and Pedersen (2012), Harvey et al. (2018), and Lim, Zohren, and Roberts (2019), in which our position is scaled up when market volatility is low and vice versa. In addition, given a volatility target, our reward R_t is mostly driven by our actions instead of being heavily affected by market volatility. We can also consider the volatility scaling as normalizing rewards from different contracts to the same scale. Because our dataset consists of 50 futures contracts with different price ranges, we need to normalize different rewards to the same scale for training and for portfolio construction. μ is a fixed number per contract at each trade, and we set it to 1. Note that our transaction cost term also includes a price term, p_{t-1}. This is necessary because again we work with many contracts, each of which has a different cost, so we represent transaction cost as a fraction of traded value. The constant, basis point (bp), is the cost rate and 1 bp = 0.0001. As an example, if the cost rate is 1 bp, we need to pay \$0.10 to buy one unit of a contract priced at \$1,000.

We define $r_t = p_t - p_{t-1}$, and this expression represents additive profits. Additive profits are often used if a fixed number of shares or contracts is traded at each time. If we want to trade a fraction of our accumulated wealth at each time, multiplicative profits should be used, and $r_t = p_t/p_{t-1} - 1$. In this case, we also need to change the expression of R_t because R_t represents the percentage of our wealth. The exact form is given by Moody et al. (1998). We stick to additive profits in our work because logarithmic transformation needs to be done for multiplicative profits to have the cumulative rewards required by the RL setup, but logarithmic transformation penalizes large wealth growth.

RL Algorithms

DQN. By adopting a neural network, a DQN approximates the state-action value function (Q function) to estimate how good it is for the agent to perform a given action in a given state. Suppose our Q function is parameterized by some θ. We minimize the mean squared error between the current and target Q to derive the optimal state-action value function:

$$\begin{aligned} L(\theta) &= \mathbb{E}[(Q_\theta(S, A) - Q'_\theta(S, A))^2] \\ Q'_\theta(S_t, A_t) &= r + \gamma\ \text{argmax}_{A_{t+1}} Q_\theta(S_{t+1}, A_{t+1}) \end{aligned} \quad (5)$$

where $L(\theta)$ is the objective function. A problem is that the training of a vanilla DQN is not stable and suffers from variability. Many improvements have been made to stabilize the training process, and we adopt the following three strategies to improve the training in our work: fixed Q-targets (Van Hasselt, Guez, and Silver 2016), double DQN (Hasselt 2010), and dueling DQN (Wang et al. 2016). Fixed Q-targets and double DQN are used to reduce policy variances and to solve the problem of *chasing tails* by using a separate network to produce target values. Dueling DQNs separate the Q-value into state value and the advantage of each action. The benefit of doing this is that the value stream has more updates and we receive a better representation of the state values.

PG. The PG aims to maximize the expected cumulative rewards by optimizing the policy directly. If we use a neural network with parameters θ to represent the policy, $\pi_\theta(A|S)$, we can generate a trajectory $\tau = [S_0, A_0, R_1, S_1, \ldots, S_t, A_t, \ldots]$ from the environment and obtain a sequence of rewards. We maximize the expected cumulative rewards $J(\theta)$ by using gradient ascent to adjust θ:

$$\begin{aligned} J(\theta) &= \mathbb{E}\left[\sum_{t=0}^{T-1} R_{t+1}|\pi_\theta\right] \\ \nabla_\theta J(\theta) &= \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(A_t|S_t) G_t \end{aligned} \quad (6)$$

where G_t is defined in Equation 2. Compared with DQN, PG learns a policy directly and can output a probability distribution over actions. This is useful if we want to design stochastic policies or work with continuous action spaces. However, the training of PG uses Monte Carlo methods to sample trajectories from the environment, and the update is done only when the episode finishes. This often results in slow training, and it can get stuck at a (suboptimal) local maximum.

A2C. The A2C is proposed to solve the training problem of PG by updating the policy in real time. It consists of two models: One is an actor network that outputs the policy, and the other is a critic network that measures how good the chosen action is in the given state. We can update the policy network $\pi(A|S,\theta)$ by maximizing the objective function:

$$J(\theta) = \mathbb{E}[\log \pi(A|S,\theta) A_{adv}(S,A)] \tag{7}$$

where $A_{adv}(S, A)$ is the advantage function defined as

$$A_{adv}(S_t, A_t) = R_t + \gamma V(S_{t+1}|w) - V(S_t|w) \tag{8}$$

To calculate advantages, we use another network, the critic network, with parameters w to model the state value function $V(s|w)$, and we can update the critic network using gradient descent to minimize the temporal difference error:

$$J(w) = (R_t + \gamma V(S_{t+1}|w) - V(S_t|w))^2 \tag{9}$$

The A2C is most useful if we are interested in continuous action spaces because we reduce the policy variance by using the advantage function and update the policy in real time. The training of A2C can be done synchronously or asynchronously (A3C). In this article, we adopt the synchronous approach and execute agents in parallel on multiple environments.

EXPERIMENTS

Description of Dataset

We use data on 50 ratio-adjusted continuous futures contracts from the CLC Database (2019). Our dataset ranges from 2005 to 2019 and consists of a variety of asset classes, including commodity, equity index, fixed income, and FX. A full breakdown of the dataset can be found in Appendix A. We retrain our model at every five years, using all data available up to that point to optimize the parameters. Model parameters are then fixed for the next five years to produce out-of-sample results. In total, our testing period is from 2011 to 2019.

Baseline Algorithms

We compare our methods to the following baseline models including classical time-series momentum strategies:

- Long only
- Sign(R) (Moskowitz, Ooi, and Pedersen 2012; Lim, Zohren, and Roberts 2019):

$$A_t = \text{sign}(r_{t-252:t}) \tag{10}$$

- MACD signal (Baz et al. 2015)

$$A_t = \phi(\text{MACD}_t)$$
$$\phi(\text{MACD}) = \frac{\text{MACD}\exp(-\text{MACD}^2/4)}{0.89} \tag{11}$$

where MACD_t is defined in Equation 3. We can also take multiple signals with different time scales and average them to give a final indicator:

$$\widetilde{\text{MACD}}_t = \sum_k \text{MACD}_t(S_k, L_k) \tag{12}$$

where S_k and L_k define short and long time scales, namely $S_k \in \{8, 16, 32\}$ and $L_k \in \{24, 48, 96\}$, as done by Lim, Zohren, and Roberts (2019).

We compare the aforementioned baseline models with our RL algorithms, DQN, PG, and A2C. DQN and PG have discrete action spaces $\{-1, 0, 1\}$, and A2C has a continuous action space $[-1, 1]$.

Training Schemes for RL

In our work, we use long short-term memory (LSTM) (Hochreiter and Schmidhuber 1997) neural networks to model both actor and critic networks. LSTMs are traditionally used in natural language processing, but many recent works have applied them to financial time series (Di Persio and Honchar 2016; Bao, Yue,

EXHIBIT 1
Values of Hyperparameters for Different RL Algorithms

Model	α_{critic}	α_{actor}	Optimizer	Batch Size	γ	bp	Memory Size	τ
DQN	0.0001	–	Adam	64	0.3	0.0020	5,000	1,000
PG	–	0.0001	Adam	–	0.3	0.0020	–	–
A2C	0.001	0.0001	Adam	128	0.3	0.0020	–	–

and Rao 2017; Tsantekidis et al. 2017b; Fischer and Krauss 2018). In particular, the work of Lim, Zohren, and Roberts (2019) showed that LSTMs deliver superior performance on modeling daily financial data. We use two-layer LSTM networks with 64 and 32 units in all models, and leaky rectifying linear units (Leaky-ReLU) (Maas, Hannun, and Ng 2013) are used as activation functions. Because our dataset consists of different asset classes, we train a separate model for each asset class. It is a common practice to train models by grouping contracts within the same asset class, and we find it also improves our performance.

We list the value of hyperparameters for different RL algorithms in Exhibit 1. We denote the learning rates for critic and actor networks as α_{critic} and α_{actor}. The Adam optimizer (Kingma and Ba 2015) is used for training all networks, and batch size means the size of the minibatch used is in a gradient descent. As previously introduced, γ is the discounting factor and bp is the cost rate used in training. We can treat bp as a regularizing term; a large bp penalizes turnovers and lets agents maintain current trade positions. The memory size shows the size of the buffer for experience replay, and we update the parameters of our target network in DQN every τ steps.

Procedures for Controlling Overfitting

Because overfitting to backtest data can be problematic in many application domains, especially in finance, we adopt the following procedures to overcome it. The most common cause of overfitting is a poor ratio of training samples to model parameters. Although there is no universal guarantee that increasing the number of samples will resolve the overfitting issue, we find that expanding the dataset to allow training on a cross section of futures contracts, not just on a single contract, greatly helps to reduce the problem in our case. Furthermore, we intentionally design our networks to have a smaller number of free parameters whenever possible. Our models consist of two hidden layers, and each layer has a relatively small number of neurons. The resulting network has fewer parameters, which makes it less likely to overfit.

In addition, we use dropout (Hinton et al. 2016), a regularization method commonly employed in deep learning. Because our networks are small, we did not notice significant improvements, but we still recommend the usage of dropout following the general advice of selecting the least flexible model that achieves comparable cross-validation performance. This is also in line with other works (Zhang, Zohren, and Roberts 2018; Lim, Zohren, and Roberts 2019) that show improved results using dropout in financial applications, especially when handling complex network architectures.

Finally, we use 10% of any given training data as a separate cross-validation set to optimize hyperparameters. We monitor the validation performance and use early stopping with 20 epochs to help optimal model selection. Any hyperparameter optimization is done on the cross-validation set, leaving the test set for the final test performance evaluation. This reduces the leakage of the test data, another cause of overfitting.

Experimental Results

We test both baseline models and our methods between 2011 and 2019, and we calculate the trade returns net of transaction costs as in Equation 4 for each contract. We then form a simple portfolio by giving equal weights to each contract, and the trade return of a portfolio is

$$R_t^{\text{port}} = \frac{1}{N}\sum_{i=1}^{N} R_t^i \tag{13}$$

where N represents the number of contracts considered, and R_t^i is the trade return for contract i at time t. We evaluate the performance of this portfolio using the

following metrics, as suggested by Lim, Zohren, and Roberts (2019):

1. $E(R)$: annualized expected trade returns
2. std(R): annualized standard deviation of trade returns
3. Downside deviation (DD): annualized standard deviation of trade returns that are negative, also known as downside risk
4. Sharpe: annualized Sharpe ratio ($E(R)$/std(R))
5. Sortino: a variant of the Sharpe ratio that uses downside deviation as risk measures ($E(R)$/DD)
6. MDD: maximum drawdown shows the maximum observed loss from any peak of a portfolio to the trough
7. Calmar: the Calmar ratio compares the expected annual rate of return with MDD; in general, the higher the ratio, the better the performance of the portfolio
8. % +ve Returns: percentage of positive trade returns
9. $\frac{\text{Ave. P}}{\text{Ave. L}}$: the ratio between positive and negative trade returns.

We present our results in Exhibit 2, in which an additional layer of portfolio-level volatility scaling is applied for each model. This brings the volatility of different methods to the same target, so we can directly compare metrics such as expected and cumulative trade returns. We also include the results without this volatility scaling for reference in Exhibit B1 in Appendix B. Exhibit 2 is split into five parts based on different asset classes. The results show the performance of a portfolio by only using contracts from that specific asset class. An exception is the last part of the exhibit, in which we form a portfolio using all contracts from our dataset. The cumulative trade returns for different models and asset classes are presented in Exhibit 3.

We can see that RL algorithms deliver better performance for most asset classes except for the equity index, for which a long-only strategy is better. This can be explained by the fact that most equity indexes were dominated by large upward trends in the testing period. Similarly, fixed incomes had a growing trend until 2016 and then entered currently ongoing consolidation periods. Arguably, the most reasonable strategy in

EXHIBIT 2
Experiment Results for Portfolio-Level Volatility Targeting

	E(R)	Std(R)	DD	Sharpe	Sortino	MDD	Calmar	% of + Ret	$\frac{\text{Ave. P}}{\text{Ave. L}}$
Commodity									
Long	–0.710	0.979	0.604	–0.726	–1.177	0.350	–0.140	0.473	0.989
Sign(R)	0.347	0.980	0.572	0.354	0.606	0.116	0.119	0.494	1.084
MACD	–0.171	0.978	0.584	–0.175	–0.293	0.190	–0.060	0.486	1.026
DQN	**0.703**	0.973	**0.552**	**0.723**	**1.275**	**0.066**	**0.501**	**0.498**	**1.135**
PG	0.062	0.982	0.585	0.063	0.106	0.039	0.023	0.495	1.029
A2C	0.223	0.955	0.559	0.234	0.399	0.141	0.091	0.487	1.093
Equity Index									
Long	**0.668**	0.970	0.606	**0.688**	**1.102**	0.132	**0.509**	**0.542**	0.948
Sign(R)	0.228	0.966	0.610	0.236	0.374	0.344	0.077	0.528	0.930
MACD	0.016	0.962	0.618	0.017	0.027	0.311	0.006	0.519	0.927
DQN	0.629	0.970	0.606	0.648	1.038	0.161	0.381	0.541	0.944
PG	0.432	0.967	0.605	0.447	0.714	0.242	0.185	0.529	0.960
A2C	0.473	0.929	**0.593**	0.510	0.798	**0.124**	0.328	0.533	**0.962**
Fixed Income									
Long	0.680	0.975	0.576	0.698	1.180	**0.061**	0.444	0.515	1.054
Sign(R)	0.214	0.972	0.592	0.221	0.363	0.080	0.083	0.504	1.019
MACD	0.219	0.967	0.579	0.228	0.380	0.065	0.123	0.486	**1.101**
DQN	**0.908**	0.972	**0.562**	**0.935**	**1.617**	0.062	**0.543**	0.515	1.098
PG	0.705	0.974	0.576	0.724	1.225	**0.061**	0.436	**0.517**	1.052
A2C	0.699	0.979	0.582	0.714	1.203	0.067	0.408	**0.517**	1.048

(continued)

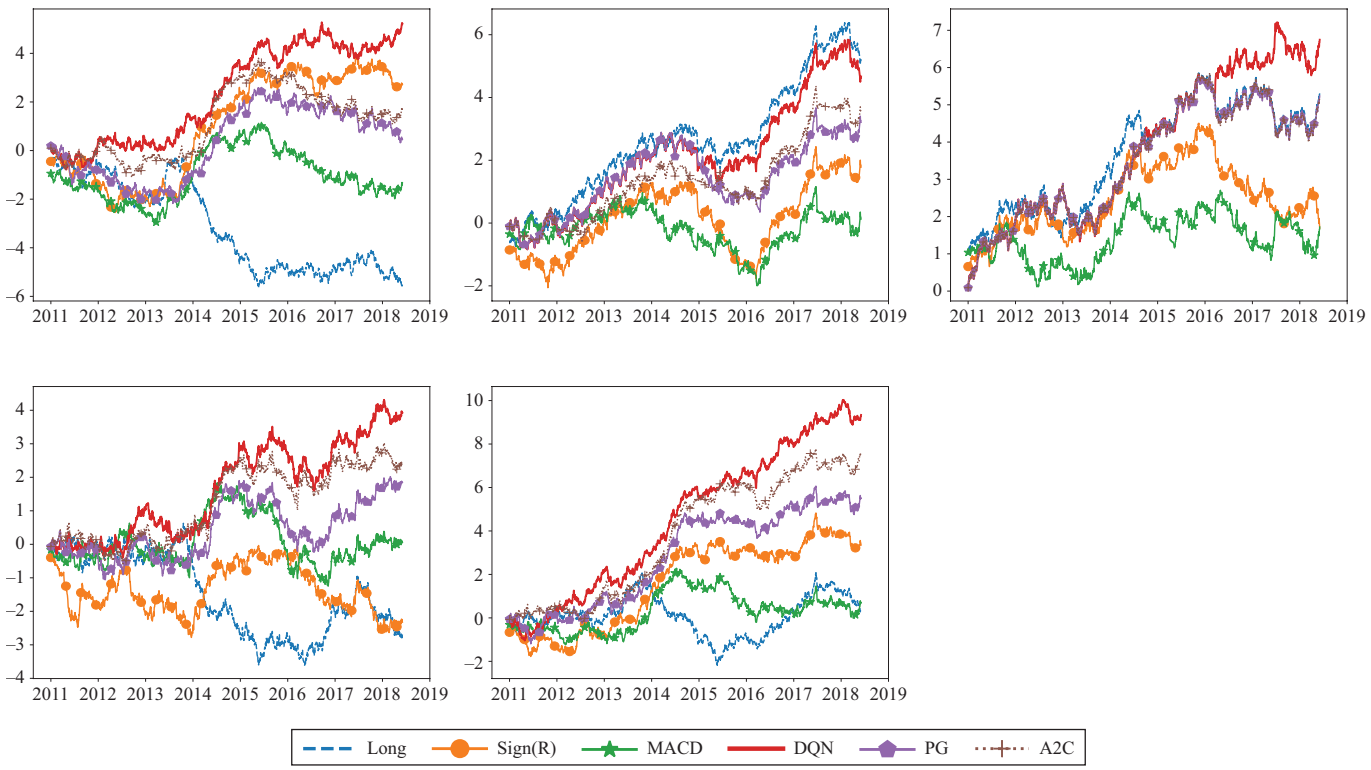
EXHIBIT 2 *(continued)*

Experiment Results for Portfolio-Level Volatility Targeting

	E(R)	Std(R)	DD	Sharpe	Sortino	MDD	Calmar	% of + Ret	Ave. P / Ave. L
FX									
Long	–0.344	0.973	0.583	–0.353	–0.590	0.423	–0.097	0.491	0.979
Sign(R)	–0.297	0.973	0.592	–0.306	–0.502	0.434	–0.111	0.499	0.954
MACD	0.006	0.970	0.582	0.007	0.011	0.329	0.002	0.493	1.029
DQN	**0.528**	0.967	**0.553**	**0.546**	**0.955**	0.183	**0.313**	**0.510**	**1.051**
PG	0.248	0.967	0.566	0.257	0.438	0.240	0.124	0.506	1.021
A2C	0.316	0.963	0.563	0.328	0.561	**0.165**	0.201	0.507	1.026
All									
Long	0.055	0.975	0.598	0.058	0.092	0.071	0.013	0.520	0.933
Sign(R)	0.429	0.972	0.582	0.441	0.737	0.038	0.201	0.510	1.031
MACD	0.089	0.978	0.582	0.091	0.153	0.008	0.035	0.493	1.043
DQN	**1.258**	0.976	**0.567**	**1.288**	**2.220**	**0.002**	**1.025**	**0.543**	**1.043**
PG	0.740	0.980	0.593	0.754	1.247	0.012	0.480	0.533	0.990
A2C	1.024	0.975	0.573	1.050	1.785	0.007	0.685	0.538	1.021

EXHIBIT 3

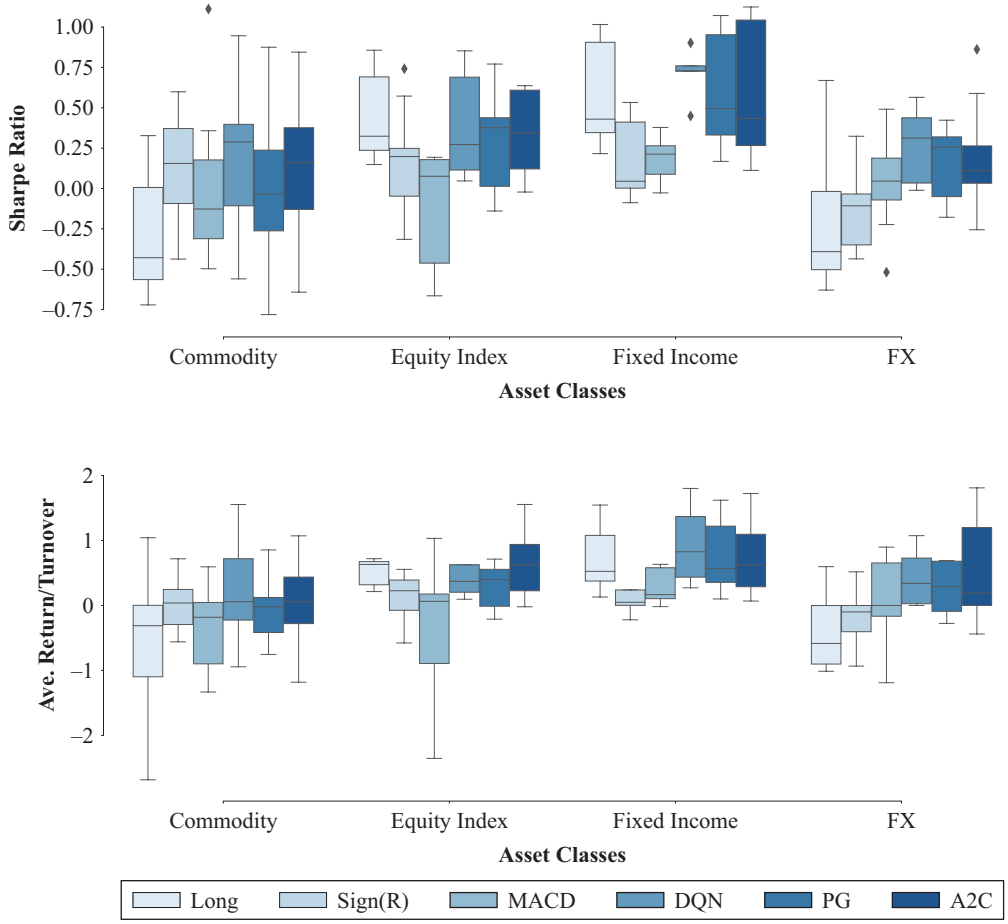
Cumulative Trade Returns



Note: First row: commodity, equity index, and fixed income; second row: FX and the portfolio of using all contracts.

EXHIBIT 4

Sharpe Ratio (top) and Average Trade Return per Turnover (bottom) for Individual Contracts



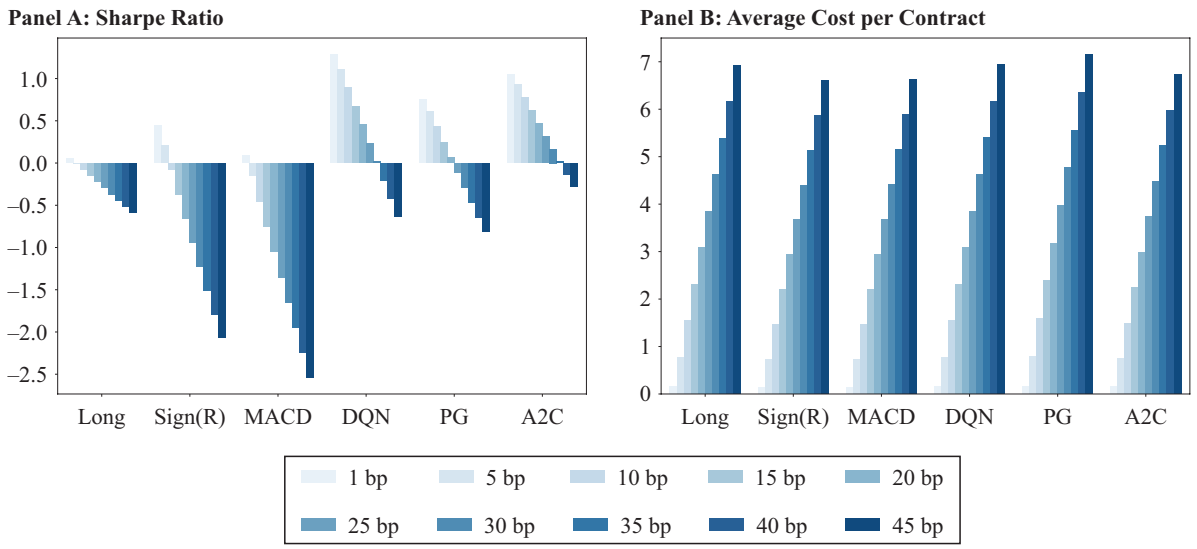
these cases might be to hold our positions if large trends persist. However, the long-only strategy performs the worst in commodity and FX markets, in which prices are more volatile. RL algorithms perform better in these markets by being able to go long or short at reasonable time points. Overall, DQN obtains the best performance among all models, and the second best is the A2C approach. We investigate the cause of this observation and find that A2C generates larger turnovers, leading to smaller average returns per turnover, as shown in Exhibit 4.

We also investigate the performance of our methods under different transaction costs. In Panel A of Exhibit 5, we plot the annualized Sharpe ratio for the portfolio using all contracts at different cost rates. We can see that RL algorithms can tolerate larger cost rates; in particular, DQN and A2C can still generate positive profits with the cost rate at 25 bp. To understand how cost rates (bp) translate into monetary values, we plot the average cost per contract in Panel B of Exhibit 5, and we see that 25 bp represents roughly $3.50 per contract. This is a realistic cost for a retail trader to pay, but institutional traders have a different fee structure based on trading volumes and often have cheaper cost. In any case, this shows the validity of our methods in a realistic setup.

The performance of a portfolio is generally better than the performance of an individual contract because risks are diversified across a range of assets, so the return per risk is higher. To investigate the raw quality of our

EXHIBIT 5

Sharpe Ratio and Average Cost per Contract under Different Cost Rates



methods, we investigate the performance of individual contracts. We use the boxplots in Exhibit 4 to present the annualized Sharpe ratio and average trade return per turnover for each futures contract. Overall, these results reinforce our previous findings that RL algorithms generally work better, and the performance of our method is not driven by a single contract that shows superior performance, reassuring us about the consistency of our model.

CONCLUSION

We adopt RL algorithms to learn trading strategies for continuous futures contracts. We discuss the connection between modern portfolio theory and the RL reward hypothesis and show that they are equivalent if a linear utility function is used. Our analysis focuses on three RL algorithms, namely DQN, PG, and A2C, and investigates both discrete and continuous action spaces. We use features from time-series momentum and technical indicators to form state representations. In addition, volatility scaling is introduced to improve reward functions. We test our methods on 50 liquid futures contracts from 2011 to 2019, and our results show that RL algorithms outperform baseline models and deliver profits even under heavy transaction costs.

In continuations of this work, we would like to investigate different forms of utility functions. In practice, an investor is often risk averse, and the objective is to maximize a risk-adjusted performance function such as the Sharpe ratio, leading to a concave utility function. As suggested by Huang (2018), we can resort to distributional RL (Bellemare, Dabney, and Munos 2017) to obtain the entire distribution over $Q(s, a)$ instead of the expected Q-value. Once the distribution is learned, we can choose actions with the highest expected Q-value and the lowest standard deviation of it, maximizing the Sharpe ratio. We can also extend our methods to portfolio optimization by modifying the action spaces to give weights of individual contracts in a portfolio. We can incorporate the reward functions with mean–variance portfolio theory (Markowitz 1952) to deliver a reasonable expected trade return with minimal volatility.

APPENDIX A

Our dataset consists of 50 futures contracts, and there are 25 commodity contracts, 11 equity index contracts, 5 fixed income contracts, and 9 forex contracts. A detailed description of each contract is given in Exhibit A1.

EXHIBIT A1

Contract Descriptions

Ticker	Contract Details	Ticker	Contract Details
Commodities		**Equity Indexes**	
CC	COCOA	CA	CAC40 INDEX
DA	MILK III, Comp	ER	RUSSELL 2000, MINI
GI	GOLDMAN SAKS C. I.	ES	S&P 500, MINI
JO	ORANGE JUICE	LX	FTSE 100 INDEX
KC	COFFEE	MD	S&P 400 (Mini Electronic)
KW	WHEAT, KC	SC	S&P 500, Composite
LB	LUMBER	SP	S&P 500, Day Session
NR	ROUGH RICE	XU	DOW JONES EUROSTOXX50
SB	SUGAR #11	XX	DOW JONES STOXX 50
ZA	PALLADIUM, Electronic	YM	Mini Dow Jones ($5.00)
ZC	CORN, Electronic	**Fixed Incomes**	
ZF	FEEDER CATTLE, Electronic	DT	EURO BOND (BUND)
ZG	GOLD, Electronic	FB	T-NOTE, 5-Year Composite
ZH	HEATING OIL, Electronic	TY	T-NOTE, 10-Year Composite
ZI	SILVER, Electronic	UB	EURO BOBL
ZK	COPPER, Electronic	US	T-BONDS, Composite
ZL	SOYBEAN OIL, Electronic	**Foreign Exchange**	
ZN	NATURAL GAS, Electronic	AN	AUSTRALIAN, Day Session
ZO	OATS, Electronic	BN	BRITISH POUND, Composite
ZP	PLATINUM, Electronic	CN	CANADIAN, Composite
ZR	ROUGH RICE, Electronic	DX	US DOLLAR INDEX
ZT	LIVE CATTLE, Electronic	FN	EURO, Composite
ZU	CRUDE OIL, Electronic	JN	JAPANESE YEN, Composite
ZW	WHEAT, Electronic	MP	MEXICAN PESO
ZZ	LEAN HOGS, Electronic	NK	NIKKEI INDEX
		SN	SWISS FRANC, Composite

APPENDIX B

Exhibit B1 presents the performance metrics for portfolios without the additional layer of volatility scaling.

EXHIBIT B1

Experiment Results for the Raw Signal

	E(R)	Std(R)	DD	Sharpe	Sortino	MDD	Calmar	% of + Ret	Ave. P / Ave. L
Commodity									
Long	–0.298	0.412	0.258	–0.723	–1.152	0.248	–0.130	0.473	0.987
Sign(R)	0.101	0.312	0.185	0.325	0.548	0.082	0.115	0.494	1.081
MACD	–0.039	0.227	0.136	–0.174	–0.290	0.132	–0.059	0.486	1.024
DQN	0.187	0.301	0.173	0.623	1.085	0.039	0.413	0.498	1.119
PG	0.013	0.287	0.172	0.047	0.078	0.072	0.017	0.495	1.026
A2C	0.072	0.163	0.098	0.440	0.729	0.099	0.161	0.487	1.151

(continued)

EXHIBIT B1 *(continued)*

Experiment Results for the Raw Signal

	E(R)	Std(R)	DD	Sharpe	Sortino	MDD	Calmar	% of + Ret	Ave. P / Ave. L
Equity Indexes									
Long	0.504	0.928	0.606	0.543	0.831	0.127	0.466	0.541	0.928
Sign(R)	0.168	0.799	0.526	0.211	0.319	0.299	0.075	0.528	0.928
MACD	–0.068	0.586	0.385	–0.117	–0.178	0.351	–0.041	0.519	0.904
DQN	0.461	0.933	0.611	0.494	0.754	0.170	0.308	0.541	0.922
PG	0.320	0.875	0.574	0.366	0.558	0.211	0.183	0.529	0.949
A2C	0.293	0.629	0.427	0.466	0.686	0.193	0.214	0.533	0.965
Fixed Income									
Long	0.605	0.939	0.561	0.645	1.081	0.108	0.455	0.515	1.048
Sign(R)	0.189	0.795	0.496	0.237	0.381	0.165	0.103	0.504	1.024
MACD	0.136	0.609	0.367	0.224	0.371	0.124	0.131	0.485	1.102
DQN	0.734	0.862	0.508	0.851	1.445	0.118	0.469	0.515	1.086
PG	0.624	0.938	0.561	0.665	1.113	0.109	0.443	0.517	1.043
A2C	0.852	1.345	0.806	0.633	1.057	0.128	0.397	0.517	1.039
Foreign Exchange									
Long	–0.198	0.472	0.285	–0.420	–0.696	0.219	–0.101	0.491	0.966
Sign(R)	–0.113	0.551	0.341	–0.207	–0.332	0.170	–0.071	0.499	0.968
MACD	0.016	0.424	0.259	0.037	0.061	0.156	0.016	0.493	1.034
DQN	0.272	0.487	0.280	0.560	0.972	0.085	0.326	0.510	1.058
PG	0.157	0.533	0.312	0.295	0.503	0.098	0.148	0.506	1.029
A2C	0.159	0.455	0.267	0.349	0.592	0.081	0.193	0.507	1.034
All									
Long	–0.013	0.363	0.230	–0.036	–0.057	0.037	–0.009	0.519	0.919
Sign(R)	0.086	0.296	0.186	0.291	0.461	0.016	0.142	0.510	1.008
MACD	–0.018	0.230	0.143	–0.080	–0.129	0.026	–0.029	0.493	1.013
DQN	0.318	0.252	0.150	1.258	2.111	0.008	1.023	0.543	1.041
PG	0.168	0.279	0.174	0.602	0.968	0.011	0.373	0.533	0.968
A2C	0.214	0.221	0.134	0.969	1.601	0.009	0.672	0.538	1.014

ACKNOWLEDGMENT

The authors would like to thank Bryan Lim, Vu Nguyen, Anthony Ledford, and members of Machine Learning Research Group at the University of Oxford for their helpful comments. We are most grateful to the Oxford-Man Institute of Quantitative Finance, which provided the Pinnacle dataset and computing facilities.

REFERENCES

Arrow, K. J. "The Theory of Risk Aversion." In *Essays in the Theory of Risk-Bearing*, pp. 90–120. Chicago: Markham, 1971.

Baltas, N., and R. Kosowski. "Demystifying Time-Series Momentum Strategies: Volatility Estimators, Trading Rules and Pairwise Correlations." *Trading Rules and Pairwise Correlations*, May 8, 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): e0180944.

Baz, J., N. Granger, C. R. Harvey, N. Le Roux, and S. Rattray. "Dissecting Investment Strategies in the Cross Section and Time Series." SSRN 2695101, 2015.

Bekiros, S. D. 2010. "Heterogeneous Trading Strategies with Adaptive Fuzzy Actor–Critic Reinforcement Learning: A Behavioral Approach." *Journal of Economic Dynamics and Control* 34 (6): 1153–1170.

Bellemare, M. G., W. Dabney, and R. Munos. 2017. "A Distributional Perspective on Reinforcement Learning." *Proceedings of the 34th International Conference on Machine Learning* 70: 449–458.

Bertoluzzo, F., and M. Corazza. 2012. "Testing Different Reinforcement Learning Configurations for Financial Trading: Introduction and Applications." *Procedia Economics and Finance* 3: 68–77.

Chan, E. *Quantitative Trading: How to Build Your Own Algorithmic Trading Business*, vol. 430. Hoboken: John Wiley & Sons, 2009.

——. *Algorithmic Trading: Winning Strategies and Their Rationale*, vol. 625. Hoboken: John Wiley & Sons, 2013.

CLC Database. Pinnacle Data Corp, 2019, https://pinnacle-data2.com/clc.html.

Collobert, R., J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. 2011. "Natural Language Processing (Almost) from Scratch." *Journal of Machine Learning Research* 12 (Aug): 2493–2537.

Deng, Y., F. Bao, Y. Kong, Z. Ren, and Q. Dai. 2016. "Deep Direct Reinforcement Learning for Financial Signal Representation and Trading." *IEEE Transactions on Neural Networks and Learning Systems* 28 (3): 653–664.

Di Persio, L., and O. Honchar. 2016. "Artificial Neural Networks Architectures for Stock Price Prediction: Comparisons and Applications." *International Journal of Circuits, Systems and Signal Processing* 10: 403–413.

Fischer, T., and C. Krauss. 2018. "Deep Learning with Long Short-Term Memory Networks for Financial Market Predictions." *European Journal of Operational Research* 270 (2): 654–669.

Fischer, T. G. "Reinforcement Learning in Financial Markets—A Survey." Technical report, FAU Discussion Papers in Economics, 2018.

Graham, B., and D. L. F. Dodd. *Security Analysis*. New York: McGraw-Hill, 1934.

Harvey, C. R., E. Hoyle, R. Korgaonkar, S. Rattray, M. Sargaison, and O. Van Hemert. 2018. "The Impact of Volatility Targeting." *The Journal of Portfolio Management* 45 (1): 14–33.

Hasselt, H. V. "Double Q-Learning." In *Advances in Neural Information Processing Systems*, pp. 2613–2621. Cambridge, MA: MIT Press, 2010.

Hinton, G. E., A. Krizhevsky, I. Sutskever, and N. Srivastva. "System and Method for Addressing Overfitting in a Neural Network." US Patent 9,406,017, August 2, 2016.

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

Huang, C. Y. 2018. "Financial Trading as a Game: A Deep Reinforcement Learning Approach." *arXiv* 1807.02787.

Ingersoll, J. E. *Theory of Financial Decision Making,* vol. 3. Lanham, MD; Rowman & Littlefield, 1987.

Jin, O., and H. El-Saawy. "Portfolio Management Using Reinforcement Learning." Technical report working paper, Stanford University, 2016.

Kingma, D. P., and J. Ba. "Adam: A Method for Stochastic Optimization." *Proceedings of the International Conference on Learning Representations*, 2015.

Kirk, D. E. *Optimal Control Theory: An Introduction*. North Chelmsford: MA; Courier Corporation, 2012.

Krizhevsky, A., I. Sutskever, and G. E. Hinton. "Imagenet Classification with Deep Convolutional Neural Networks." In *Advances in Neural Information Processing Systems*, pp. 1097–1105. Cambridge, MA: MIT Press, 2012.

Li, H., C. H. Dagli, and D. Enke. 2007. "Short-Term Stock Market Timing Prediction under Reinforcement Learning Schemes." *2007 IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*, pp. 233–240.

Lim, B., S. Zohren, and S. Roberts. 2019. "Enhancing Time-Series Momentum Strategies Using Deep Neural Networks." *The Journal of Financial Data Science* 1 (4): 19–38.

Maas, A. L., A. Y. Hannun, and A. Y. Ng. 2013. "Rectifier Nonlinearities Improve Neural Network Acoustic Models." *ICML Workshop on Deep Learning for Audio, Speech and Language Processing* 30: 3.

Markowitz, H. 1952. "Portfolio Selection." *The Journal of Finance* 7 (1): 77–91.

Merton, R. C. 1969. "Lifetime Portfolio Selection under Uncertainty: The Continuous-Time Case." *The Review of Economics and Statistics* 51 (3): 247–257.

Metropolis, N., and S. Ulam. 1949. "The Monte Carlo Method." *Journal of the American Statistical Association* 44 (247): 335–341.

Michaud, R. O. 1989. "The Markowitz Optimization Enigma: Is 'Optimized' Optimal?" *Financial Analysts Journal* 45 (1): 31–42.

Mnih, V., A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. 2016. "Asynchronous Methods for Deep Reinforcement Learning." In *International Conference on Machine Learning* 48: 1928–1937.

Mnih, V., K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. "Playing Atari with Deep Reinforcement Learning." NIPS Deep Learning Workshop, 2013.

Moody, J., and M. Saffell. 2001. "Learning to Trade via Direct Reinforcement. *IEEE Transactions on Neural Networks* 12 (4): 875–889.

Moody, J., L. Wu, Y. Liao, and M. Saffell. 1998. "Performance Functions and Reinforcement Learning for Trading Systems and Portfolios." *Journal of Forecasting* 17 (5–6): 441–470.

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

Murphy, J. J. *Technical Analysis of the Financial Markets: A Comprehensive Guide to Trading Methos and Applications*. Paramus, NJ: New York Institute of Finance, 1999.

O'Neil, W. J. *How to Make Money in Stocks: A Winning System in Good Times and Bad*, 4th ed. New York: McGraw-Hill, 2009.

Pratt, J. W. "Risk Aversion in the Small and in the Large." In *Uncertainty in Economics*, pp. 59–79. Elsevier, 1978.

Ritter, G. 2017. "Machine Learning for Trading." August 8, 2017. Available at SSRN: https://ssrn.com/abstract=3015609 or http://dx.doi.org/10.2139/ssrn.3015609.

Rohrbach, J., S. Suremann, and J. Osterrieder. 2017. "Momentum and Trend Following Trading Strategies for Currencies Revisited—Combining Academia and Industry." SSRN 2949379.

Schwager, J. D. *A Complete Guide to the Futures Market: Technical Analysis, Trading Systems, Fundamental Analysis, Options, Spreads, and Trading Principles*. Hoboken: John Wiley & Sons, 2017.

Sirignano, J., and R. Cont. 2019. "Universal Features of Price Formation in Financial Markets: Perspectives from Deep Learning." *Quantitative Finance* 19 (9): 1449–1459.

Sutton, R. S., and A. G. Barto. *Introduction to Reinforcement Learning*, vol. 2. Cambridge, MA: MIT Press, 1998.

Tan, Z., C. Quek, and P. Y. Cheng. 2011. "Stock Trading with Cycles: A Financial Application of Anfis and Reinforcement Learning." *Expert Systems with Applications* 38 (5): 4741–4755.

Tsantekidis, A., N. Passalis, A. Tefas, J. Kanniainen, M. Gabbouj, and A. Iosifidis. 2017a. "Forecasting Stock Prices from the Limit Order Book Using Convolutional Neural Networks." *2017 IEEE 19th Conference on Business Informatics (CBI*) 1: 7–12.

——. "Using Deep Learning to Detect Price Change Indications in Financial Markets." 2017b. *2017 25th European Signal Processing Conference (EUSIPCO)*, pp. 2511–2515.

Van Hasselt, H., A. Guez, and D. Silver. "Deep Reinforcement Learning with Double Q-Learning." *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

Wang, Z., T. Schaul, M. Hessel, H. Van Hasselt, M. Lanctot, and N. De Freitas. 2016. "Dueling Network Architectures for Deep Reinforcement Learning." In *Proceedings of the 33rd International Conference on International Conference on Machine Learning* 48: 1995–2003.

Wilder, J. W. "New Concepts in Technical Trading Systems." Trend Research, 1978.

Williams, R. J. 1992. "Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning." *Machine Learning* 8 (3–4): 229–256.

Xiong, Z., X. Y. Liu, S. Zhong, and A. Walid. 2018. "Practical Deep Reinforcement Learning Approach for Stock Trading." *arXiv* 1811.07522.

Zhang, Z., S. Zohren, and S. Roberts. "Bdlob: Bayesian Deep Convolutional Neural Networks for Limit Order Books." Presented at Third Workshop on Bayesian Deep Learning (NeurIPS 2018), Montréal, Canada, 2018.

——. 2019a. "Deeplob: Deep Convolutional Neural Networks for Limit Order Books." *IEEE Transactions on Signal Processing* 67 (11): 3001–3012.

——. "Extending Deep Learning Models for Limit Order Books to Quantile Regression." *Proceedings of Time Series Workshop of the 36th International Conference on Machine Learning*, 2019b.

To order reprints of this article, please contact David Rowe at d.rowe@pageantmedia.com or 646-891-2157.

ADDITIONAL READING

The Impact of Volatility Targeting

Campbell R. Harvey, Edward Hoyle, Russell Korgaonkar, Sandy Rattray, Matthew Sargaison, and Otto Van Hemert

The Journal of Portfolio Management
https://jpm.pm-research.com/content/45/1/14

ABSTRACT: *Recent studies show that volatility-managed equity portfolios realize higher Sharpe ratios than portfolios with a constant notional exposure. The authors show that this result only holds for risk assets, such as equity and credit, and they link this finding to the so-called leverage effect for those assets. In contrast, for bonds, currencies, and commodities, the impact of volatility targeting on the Sharpe ratio is negligible. However, the impact of volatility targeting goes beyond the Sharpe ratio: It reduces the likelihood of extreme returns across all asset classes. Particularly relevant for investors, left-tail events tend to be less severe because they typically occur at times of elevated volatility, when a target-volatility portfolio has a relatively small notional exposure. We also consider the popular 60–40 equity–bond balanced portfolio and an equity–bond–credit–commodity risk parity portfolio. Volatility scaling at both the asset and portfolio level improves Sharpe ratios and reduces the likelihood of tail events.*

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.*

Enhancing Time-Series Momentum Strategies Using Deep Neural Networks

Bryan Lim, Stefan Zohren, and Stephen Roberts

The Journal of Financial Data Science
https://jfds.pm-research.com/content/1/4/19

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.*