Enhancing Time Series Momentum Strategies Using Deep Neural Networks

Bryan Lim, Stefan Zohren, Stephen Roberts

***Abstract*—While time series momentum [1] 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 paper, we introduce Deep Momentum Networks – a hybrid approach which 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 optimising the Sharpe ratio of the signal. Back-testing on a portfolio of 88 continuous futures contracts, we demonstrate that the Sharpe-optimised LSTM improved traditional methods by more than two times in the absence of transactions costs, and continue outperforming when considering transaction costs up to 2-3 basis points. To account for more illiquid assets, we also propose a turnover regularisation term which trains the network to factor in costs at run-time.**

I. Introduction

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 [2, 3, 4]. 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 with a plethora of methods available to estimate the magnitude of price trends [5, 6, 4] and map them to actual traded positions [7, 8, 9].

In recent times, deep neural networks have been increasingly used for time series prediction, outperforming traditional benchmarks in applications such as demand forecasting [10], medicine [11] and finance [12]. With the development of modern architectures such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs) [13], deep learning models have been favoured for their ability to build representations of a given dataset [14] – 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` [15] and `PyTorch` [16] – 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, the creation of hybrid methods that combine traditional time-series models with neural network components have been observed to outperform pure methods in either category [17] e.g. the exponential smoothing RNN [18], autoregressive CNNs [19] and Kalman filter variants [20, 21] – while also making outputs easier to interpret by practitioners. Furthermore, these frameworks have also enabled the development of new loss functions for training neural networks, such as adversarial loss functions in generative adversarial networks (GANs) [22].

While 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 [23, 24, 25, 12, 26, 19, 27] – 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 time-series momentum for several reasons. Firstly, sizing positions based on expected returns alone does not take risk characteristics into account such as the volatility or skew of the predictive

B. Lim, S. Zohren and S. Roberts are with the Department of Engineering Science and the Oxford-Man Institute of Quantitative Finance, University of Oxford, Oxford, United Kingdom (email: bryan.lim@eng.ox.ac.uk, zohren@robots.ox.ac.uk, sjrob@robots.ox.ac.uk).

returns distribution - which could inadvertently expose signals to large downside moves. This is particularly relevant as raw momentum strategies without adequate risk adjustments, such as volatility scaling [7], are susceptible to large crashes during periods of market panic [28, 29]. Furthermore, even with volatility scaling which leads to positively skewed returns distributions and long-option-like behaviour [30, 31] – trend following strategies can place more losing trades than winning ones and still be profitable on the whole – as they size up only into large but infrequent directional moves. As such, [32] argue 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, as profitability also depends on the magnitude of returns in each class. This is also echoed in betting strategies such as the Kelly criterion [33], 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 paper, 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 [8, 1] – which we refer to as the *Deep Momentum Networks* (DMNs). This improves existing methods from several angles. Firstly, 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 learnt directly using modern time series prediction architectures. Secondly, by utilising automatic differentiation in existing backpropagation frameworks, we explicitly optimise networks for risk-adjusted performance metrics, i.e. the Sharpe ratio [34], 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 [9]. This consistency also helps when making comparisons to existing methods, and facilitates the interpretation of different components of the overall signal by practitioners.

II. Related Works

A. Classical Momentum Strategies

Momentum strategies are traditionally divided into two categories – namely (multivariate) cross sectional momentum [35, 24] and (univariate) time series momentum [1, 8]. 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, [35] find 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 [36], and across a variety of markets and asset classes [4].

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 [1], who describe a concrete strategy which uses volatility scaling and trades positions based on the sign of returns over the past year – demonstrating profitability across 58 different liquid instruments *individually* over 25 years of data. Since then, numerous trading rules have been proposed – with various trend estimation techniques and methods map them to traded positions. For instance, [6] documents a wide range of linear and non-linear filters to measure trends and a statistic to test for its significance – although methods to size positions with these estimates are not directly discussed. [8] adopt a similar approach to [1], 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. While 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 are proposed in [4] and [37], taking volatility-normalised moving average convergence divergence (MACD) indicators

as inputs. Despite the diversity of options, few comparisons have been made between the trading rules themselves, offering little clear evidence or intuitive reasoning to favour 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.

B. 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 [26], or using limit order book data in a high frequency trading setting [25, 12, 38]. While 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 [26], or incorporating these thresholds into the classification problem itself by dividing price movements into buy, hold and sell classes depending on magnitude [12, 38]. In addition to restricting the universe of strategies to those which 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 [23, 24], ranking assets on the basis of expected returns over the next time period. Using a variety of linear, tree-based and neural network models [23] demonstrate the outperformance of non-linear methods, with deep neural networks – specifically 3-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 4-layer MLP that trades the top decile against the decile of predictions. In other works, [24] adopt a similar approach using autoencoder and denoising autoencoder architectures, incorporating volatility scaling into their model as well. While the results with basic deep neural are promising, they do not consider more modern architectures for time series prediction, such as the LSTM [39] and WaveNet [40] architectures which we evaluate for the DMN. Moreover, to the best of our knowledge, our paper is the first to consider the use of deep learning within the context of time series momentum strategies – opening up possibilities in an alternate class of signals.

Popularised by success of DeepMind's AlphaGo Zero [41], 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 [42, 43] 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.

III. Strategy Definition

Adopting the terminology of [8], the combined returns of a time series momentum (TSMOM) strategy can be expressed as below – characterised by a trading rule or signal $X_t \in [-1, 1]$:

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

Here $r_{t,t+1}^{TSMOM}$ is the realised 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. We set the annualised 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 $r_{t,t+1}^{(i)}$.

A. 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 [1, 4], which we also include as benchmarks into our tests.

Moskowitz et al. 2012 [1]: In their original paper on time series momentum, a simple trading rule is adopted as below:

$$\textbf{Trend Estimation:} \quad Y_t^{(i)} = r_{t-252,t}^{(i)} \tag{2}$$

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

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. 2015 [4]: In practice, more sophisticated methods can be used to compute $Y_t^{(i)}$ and $X_t^{(i)}$ – such as the model of [4] described below:

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

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

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

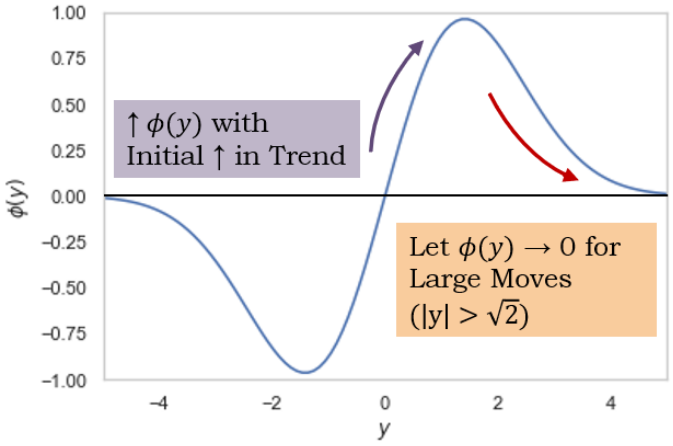
Here $\text{std}(p_{t-63:t}^{(i)})$ is the 63-day rolling standard deviation of asset i prices $p_{t-63:t}^{(i)} = [p_{t-63}^{(i)}, \ldots, p_t^{(i)}]$, $m(i, S)$ is the exponentially weighted moving average of asset i prices with a time-scale S that translates into a half-life of $HL = \log(0.5)/\log(1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

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

$$\textbf{Position Sizing:} \quad X_t^{(i)} = \phi(Y_t^{(i)}), \tag{7}$$

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

Exhibit 1: Position Sizing Function $\phi(y)$



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

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

where $Y_t^{(i)}(S_k, L_k)$ 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 in [4].

B. Machine Learning Extensions

As can be seen from Section III-A, many explicit 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 works (see Section II-B), we can cast trend estimation as a standard regression or binary classification problem, with outputs:

$$\textbf{Trend Estimation:} \quad Y_t^{(i)} = f\left(\mathbf{u}_t^{(i)}; \boldsymbol{\theta}\right), \tag{9}$$

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

$$\mathcal{L}_{\text{reg}}(\boldsymbol{\theta}) = \frac{1}{M}\sum_{\Omega}\left(Y_t^{(i)} - \frac{r_{t,t+1}^{(i)}}{\sigma_t^{(i)}}\right)^2 \quad (10)$$

$$\mathcal{L}_{\text{binary}}(\boldsymbol{\theta}) = -\frac{1}{M}\sum_{\Omega}\left\{\mathbb{I}\ \log\left(Y_t^{(i)}\right) + (1-\mathbb{I})\log\left(1-Y_t^{(i)}\right)\right\}, \quad (11)$$

where $\Omega = \left\{\left(Y_1^{(1)}, r_{1,2}^{(1)}/\sigma_t^{(1)}\right), \ \dots\ , \left(Y_{T-1}^{(N)}, r_{T-1,T}^{(N)}/\sigma_{T-1}^{(N)}\right)\right\}$ is the set of all M possible prediction and target tuples across all N assets and T time steps. For the binary classification case, $\mathbb{I}$ is the indicator function $\mathbb{I}\left(r_{t,t+1}^{(i)}/\sigma_t^{(i)} > 0\right)$ – making $Y_t^{(i)}$ 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:

Position Sizing:

$$\text{Regression} \quad X_t^{(i)} = \text{sgn}(Y_t^{(i)}) \quad (12)$$

$$\text{Classification} \quad X_t^{(i)} = \text{sgn}(Y_t^{(i)} - 0.5) \quad (13)$$

As such, we take a maximum long position when the expected returns are positive in the regression case, or when the probability of a positive return is greater than 0.5 in the classification case.

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, i.e.:

$$\textbf{Direct Outputs:} \quad X_t^{(i)} = f\left(\mathbf{u}_t^{(i)}; \boldsymbol{\theta}\right). \quad (14)$$

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 optimising performance metrics. Specifically, we focus on optimising the average return and the Sharpe ratio via the loss functions below:

$$\mathcal{L}_{\text{returns}}(\boldsymbol{\theta}) = -\frac{1}{M}\sum_{\Omega} R(i,t) - \frac{1}{M}\sum_{\Omega} X_t^{(i)}\ \frac{\sigma_{\text{tgt}}}{\sigma_t^{(i)}}\ r_{t,t+1}^{(i)} \quad (15)$$

$$\mathcal{L}_{\text{sharpe}}(\boldsymbol{\theta}) = -\frac{\sum_{\Omega} R(i,t)}{\sum_{\Omega} R(i,t)^2 - \left(\sum_{\Omega} R(i,t)\right)^2} \quad (16)$$

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

IV. Deep Momentum Networks

In this section, we examine a variety of architectures that can be used in Deep Momentum Networks – all of which can be easily reconfigured to generate the predictions described in Section III-B. This is achieved by implementing the models using the `Keras` API in `Tensorflow` [15], where output activation functions can be flexibly interchanged to generate the 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.

A. Network Architectures

Lasso Regression: In the simplest case, a standard linear model could be used to generate predictions as below:

$$Z_t^{(i)} = g\left(\mathbf{w}^T \mathbf{u}_{t-\tau:t}^{(i)} + b\right), \quad (17)$$

where $Z_t^{(i)} \in \left\{X_t^{(i)}, Y_t^{(i)}\right\}$ 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 which depends on the specific prediction type – linear for standard regression, sigmoid for binary classification, and tanh-function for direct outputs.

Additional regularisation is also provided during training by augmenting the various loss functions to include an additional L_1 regulariser as below:

$$\tilde{\mathcal{L}}(\boldsymbol{\theta}) = \mathcal{L}(\boldsymbol{\theta}) + \alpha||\mathbf{w}||_1, \quad (18)$$

where $\mathcal{L}(\boldsymbol{\theta})$ corresponds to one of the loss functions described in Section III-B, $||\mathbf{w}||_1$ is the L_1 norm of $\mathbf{w}$, and α is a constant term which 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 – i.e. $\mathbf{u}_{t-\tau:t}^{(i)} = [\mathbf{u}_{t-\tau}^{(i)\ T}, \ldots, \mathbf{u}_{t}^{(i)\ T}]^T$. This was fixed to be $\tau = 5$ days for tests in Section V.

Multilayer Perceptron (MLP): Increasing the degree of model complexity slightly, a 2-layer neural network can be used to incorporated non-linear effects:

$$\mathbf{h}_t^{(i)} = \tanh\left(\mathbf{W}_h \mathbf{u}_{t-\tau:t}^{(i)} + \mathbf{b}_h\right) \tag{19}$$

$$Z_t^{(i)} = g\left(\mathbf{W}_z \mathbf{h}_t^{(i)} + \mathbf{b}_z\right), \tag{20}$$

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 matrices and biases respectively.

WaveNet: More modern techniques such as convolutional neural networks (CNNs) have been used in the domain of time series prediction – particularly in the form of autoregressive architectures e.g. [19]. These typically take the form of 1D causal convolutions, sliding convolutional filters across time to extract useful representations which 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 [40] have been proposed, which skip over inputs at intermediate levels with a predetermined dilation rate. This allows it to 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 take the form below:

$$\psi(\mathbf{u}) = \underbrace{\tanh(\mathbf{W}\mathbf{u}) \odot \sigma(\mathbf{V}\mathbf{u})}_{\text{Gated Activation}} + \underbrace{\mathbf{A}\mathbf{u} + \mathbf{b}}_{\text{Skip Connection}} . \tag{21}$$

Here $\mathbf{W}$ and $\mathbf{V}$ are weight matrices 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 dimensionality of the layer outputs for the skip connection. The equations for WaveNet architecture used in our investigations can then be expressed as:

$$\mathbf{s}_{\text{weekly}}^{(i)}(t) = \psi(\mathbf{u}_{t-5:t}^{(i)}) \tag{22}$$

$$\mathbf{s}_{\text{monthly}}^{(i)}(t) = \psi\left(\begin{bmatrix} \mathbf{s}_{\text{weekly}}^{(i)}(t) \\ \mathbf{s}_{\text{weekly}}^{(i)}(t-5) \\ \mathbf{s}_{\text{weekly}}^{(i)}(t-10) \\ \mathbf{s}_{\text{weekly}}^{(i)}(t-15) \end{bmatrix}\right) \tag{23}$$

$$\mathbf{s}_{\text{quarterly}}^{(i)}(t) = \psi\left(\begin{bmatrix} \mathbf{s}_{\text{monthly}}^{(i)}(t) \\ \mathbf{s}_{\text{monthly}}^{(i)}(t-21) \\ \mathbf{s}_{\text{monthly}}^{(i)}(t-42) \end{bmatrix}\right) . \tag{24}$$

Here each intermediate layer $\mathbf{s}_\cdot^{(i)}(t)$ aggregates representations at weekly, monthly and quarterly frequencies respectively. Intermediate layers are then concatenated at each layer before passing through a 2-layer MLP to generate outputs, i.e.:

$$\mathbf{s}_t^{(i)} = \begin{bmatrix} \mathbf{s}_{\text{weekly}}^{(i)}(t) \\ \mathbf{s}_{\text{monthly}}^{(i)}(t) \\ \mathbf{s}_{\text{quarterly}}^{(i)}(t) \end{bmatrix} \tag{25}$$

$$\mathbf{h}_t^{(i)} = \tanh(\mathbf{W}_h \mathbf{s}_t^{(i)} + \mathbf{b}_h) \tag{26}$$

$$Z_t^{(i)} = g\left(\mathbf{W}_z \mathbf{h}_t^{(i)} + \mathbf{b}_z\right). \tag{27}$$

State sizes for each intermediate layers $\mathbf{s}_{\text{weekly}}^{(i)}(t)$, $\mathbf{s}_{\text{monthly}}^{(i)}(t)$, $\mathbf{s}_{\text{quarterly}}^{(i)}(t)$ and the MLP hidden state $\mathbf{h}_t^{(i)}$ 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 above also excludes the LSTM block (i.e. the context stack) described in [40], focusing purely on the merits of the dilated CNN model.

Long Short-term Memory (LSTM): Traditionally used in sequence prediction for natural language processing, recurrent neural networks – specifically long short-term memory (LSTM) architectures [39] – have been increasing used in time series prediction

tasks. The equations for the LSTM in our model are provided below:

$$\mathbf{f}_t^{(i)} = \sigma(\mathbf{W}_f \mathbf{u}_t^{(i)} + \mathbf{V}_f \mathbf{h}_{t-1}^{(i)} + \mathbf{b}_f) \quad (28)$$

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

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

$$\mathbf{c}_t^{(i)} = \mathbf{f}_t^{(i)} \odot \mathbf{c}_{t-1}^{(i)} + \mathbf{i}_t^{(i)} \odot \tanh(\mathbf{W}_c \mathbf{u}_t^{(i)} + \mathbf{V}_c \mathbf{h}_{t-1}^{(i)} + \mathbf{b}_c) \quad (31)$$

$$\mathbf{h}_t^{(i)} = \mathbf{o}_t^{(i)} \odot \tanh(\mathbf{c}_t^{(i)}) \quad (32)$$

$$Z_t^{(i)} = g\left(\mathbf{W}_z \mathbf{h}_t^{(i)} + \mathbf{b}_z\right), \quad (33)$$

where $\odot$ is the Hadamard (element-wise) product, $\sigma(.)$ is the sigmoid activation function, $\mathbf{W}_.$ and $\mathbf{V}_.$ are weight matrices for the different layers, $\mathbf{f}_t^{(i)}, \mathbf{i}_t^{(i)}, \mathbf{o}_t^{(i)}$ correspond to the forget, input and output gates respectively, $\mathbf{c}_t^{(i)}$ is the cell state, and $\mathbf{h}_t^{(i)}$ 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.

B. Training Details

Model calibration was undertaken using minibatch stochastic gradient descent with the `Adam` optimiser [44], based on the loss functions defined in Section III-B. Backpropagation was performed up to a maximum of 100 training epochs using 90% of a given block of training data, and the most recent 10% retained as a validation dataset. Validation data is 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 optimisation was conducted using 50 iterations of random search, with full details provided in Appendix B. For additional information on the deep neural network calibration, please refer to [13].

Dropout regularisation [45] 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 (22), and outputs, Equation (26), of the convolutional layers in the WaveNet architecture. For the LSTM, we adopted the same dropout masks as in [46] – applying dropout to the RNN inputs, recurrent states and outputs.

V. Performance Evaluation

A. 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 [47]. These contracts spanned across 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 Appendix A.

B. Backtest Description

Throughout our backtest, the models were recalibrated from scratch every 5 years – re-running the entire hyperparameter optimisation procedure using all data available up to the recalibration point. Model weights were then fixed for signals generated over the next 5 year period, ensuring that tests were performed out-of-sample.

For the Deep Momentum Networks, we incorporate a series of useful features adopted by standard time series momentum strategies in Section III-A to generate predictions at each step:

1) *Normalised Returns* – Returns over the past day, 1-month, 3-month, 6-month and 1-year periods are used, normalised by a measure of daily volatility scaled to an appropriate time scale. For instance, normalised annual returns were taken to be $r_{t-252,t}^{(i)}/(\sigma_t^{(i)}\sqrt{252})$.
2) *MACD Indicators* – We also include the MACD indicators – i.e. trend estimates $Y_t^{(i)}$ – 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 ($X_t^{(i)} = 1$)
2) Sgn(Returns) – Moskowitz et al. 2012 [1]
3) MACD Signal – Baz et al. 2015 [4]

Finally, performance was judged based on the following metrics:

1) *Profitability* – Expected returns ($\mathbb{E}$[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 $\left(\frac{\text{Ave. P}}{\text{Ave. L}}\right)$.

C. Results and Discussion

Aggregating the out-of-sample predictions from 1995 to 2015, we compute performance metrics for both the strategy returns based on Equation (1) (Exhibit 2), as well as that 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 Table 2, this rescaling also helps to facilitates 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 Section VI, where we undertake a deeper analysis of signal turnover. More detailed results can also be found in Appendix C, which echo the findings below.

Focusing on the raw signal outputs, the Sharpe ratio-optimised LSTM outperforms all benchmarks as expected, improving the best neural network model (Sharpe-optimised 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 of using models which capture non-linear 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 (22) to (24) 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. Analysing the relative performance within each model class, we can see that models which 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-optimised models outperform returns-optimised ones, with standard regression and classification benchmarks taking third and fourth place respectively.

From Exhibit 3, while 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-optimised MLPs to outperform returns-optimised ones, and even leading to Sharpe-optimised 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 later around 10%. However, Sharpe-optimised LSTMs still retained the lowest MDD across 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 optimisation 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 which use standard regression and classification methods for trend estimation – hinting at the difficulties faced in selecting an appropriate position sizing function, and in optimising models to generate positions without accounting for risk. This is particularly relevant for binary classification

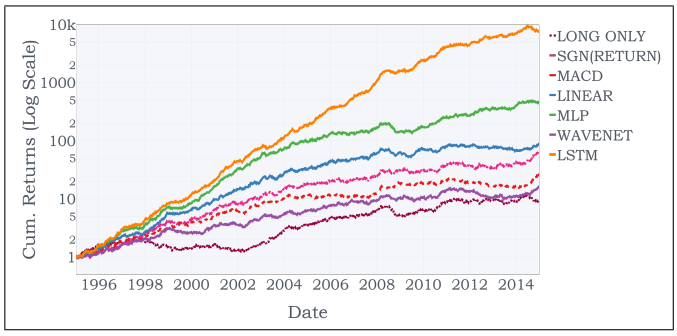
Exhibit 2: Performance Metrics – Raw Signal Outputs

	E[Return]	Vol.	Downside Deviation	MDD	Sharpe	Sortino	Calmar	% of +ve Returns	$\frac{\text{Ave. P}}{\text{Ave. L}}$
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

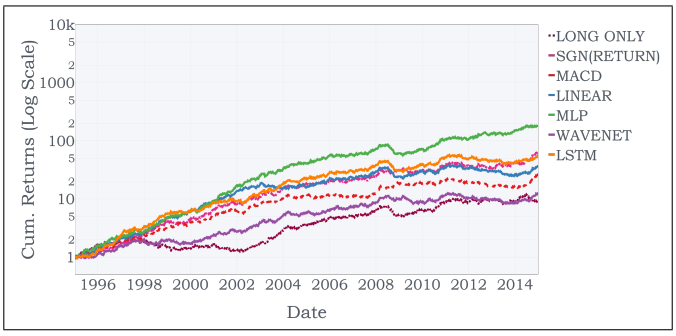
Exhibit 3: Performance Metrics – Rescaled to Target Volatility

	E[Return]	Vol.	Downside Deviation	MDD	Sharpe	Sortino	Calmar	% of +ve Returns	$\frac{\text{Ave. P}}{\text{Ave. L}}$
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

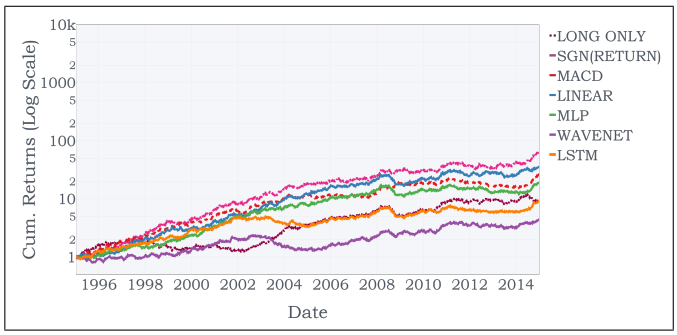
Exhibit 4: Cumulative Returns - Rescaled to Target Volatility



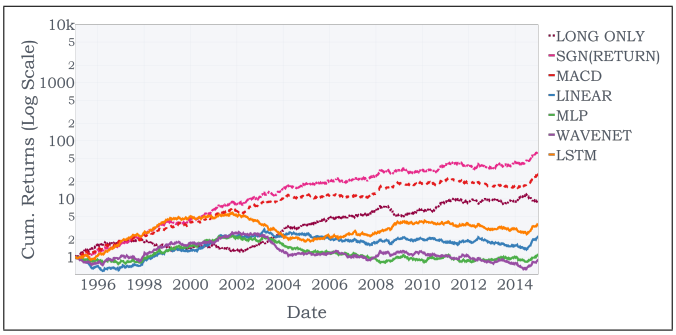
(a) Sharpe Ratio



(b) Average Returns



(c) MSE

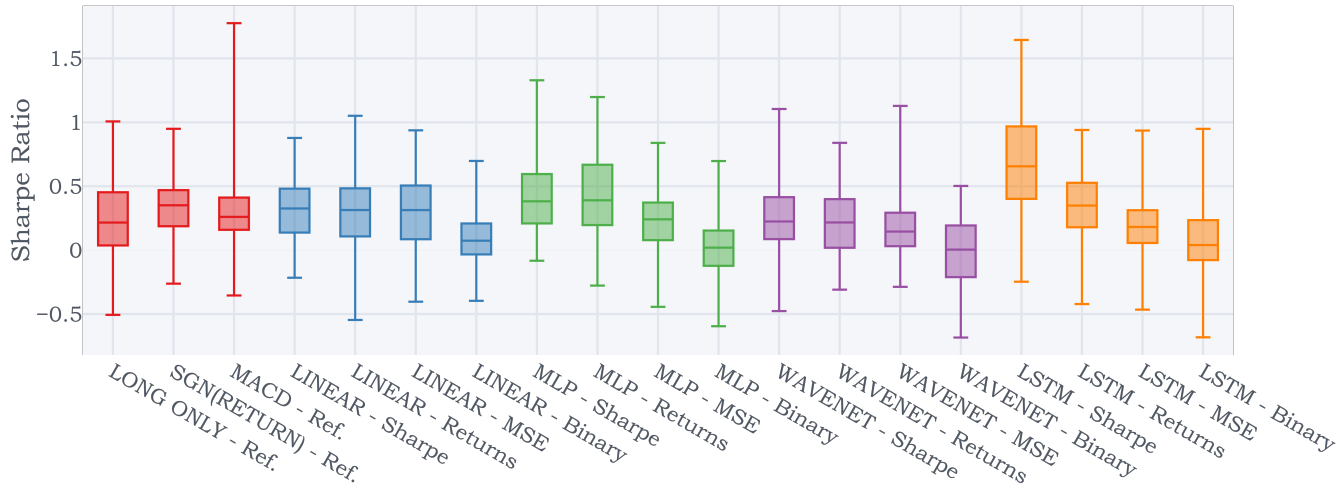


(d) Binary

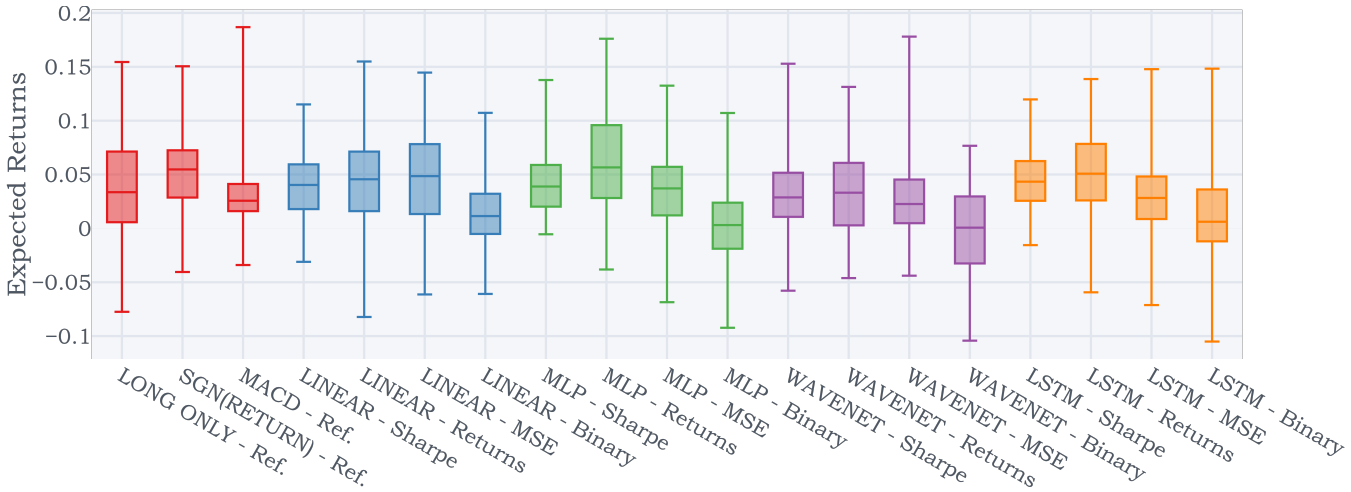
methods, which produce relatively flat equity lines and underperform reference benchmarks in general. Some of these poor results can be explained by the implicit decision threshold adopted. From the percentage of positive returns captured in Exhibit 3, most binary classification models have about a 50% accuracy which, while expected of a classifier with a 0.5 probability threshold, is far below the accuracies 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 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 positions 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 one metric per category in Section V-B, although similar results can be seen for other performance ratios are documented in Appendix C. In general, the Sharpe ratio plots in Exhibit 5a echo previous findings, with direct output methods performing better than indirect trend estimation models. However, as seen in Exhibit 5c, this is mainly attributable to significant reduction in signal volatility for the Sharpe-optimised methods, despite a comparable range of average returns in Exhibit 5b. 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 $\text{sgn}(.)$ position sizer. As such, the combination of volatility scaling, direct outputs and Sharpe ratio optimisation were all key to performance gains in Deep Momentum Networks.

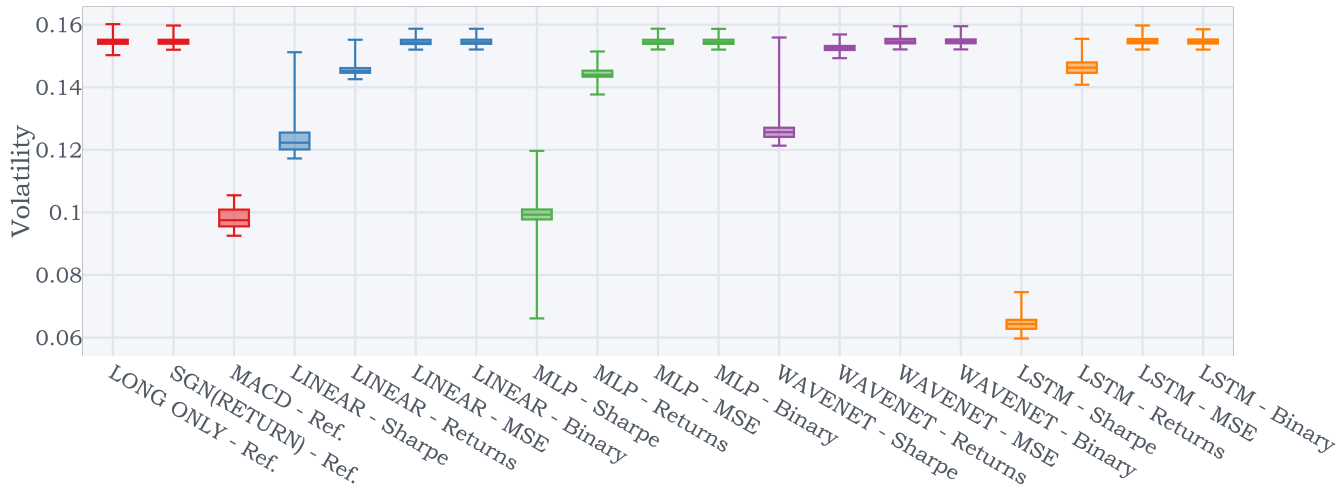
Exhibit 5: Performance Across Individual Assets



(a) Sharpe Ratio



(b) Average Returns



(c) Volatility

VI. Turnover Analysis

To investigate how transaction costs affect strategy performance, we first analyse the daily position changes of the signal – characterised for asset i by daily turnover $\zeta_t^{(i)}$ as defined in [8]:

$$\zeta_t^{(i)} = \sigma_{\text{tgt}} \left| \frac{X_t^{(i)}}{\sigma_t^{(i)}} - \frac{X_{t-1}^{(i)}}{\sigma_{t-1}^{(i)}} \right| \tag{34}$$

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

Exhibit 6a 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 while 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 6a 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 quantity the impact of transaction costs on performance, we also compute the ex-cost Sharpe ratios – using the rebalancing costs defined in [8] to adjust our returns for a variety of transaction cost assumptions . For the results in Exhibit 7, the top of each bar chart marks the maximum cost-free Sharpe ratio of the strategy, with each coloured 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 5bps), with the profitability across most machine learning models persisting only up to 4bps. However, we still obtain higher cost-adjusted Sharpe ratios with the Sharpe-optimised LSTM for up to 2-3 bps, demonstrating its suitability for trading more liquid instruments.

A. Turnover Regularisation

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 below:

$$\tilde{r}_{t,t+1}^{TSMOM} = \frac{\sigma_{\text{tgt}}}{N_t} \sum_{i=1}^{N_t} \left(\frac{X_t^{(i)}}{\sigma_t^{(i)}} \, r_{t,t+1}^{(i)} - c \left| \frac{X_t^{(i)}}{\sigma_t^{(i)}} - \frac{X_{t-1}^{(i)}}{\sigma_{t-1}^{(i)}} \right| \right), \tag{35}$$

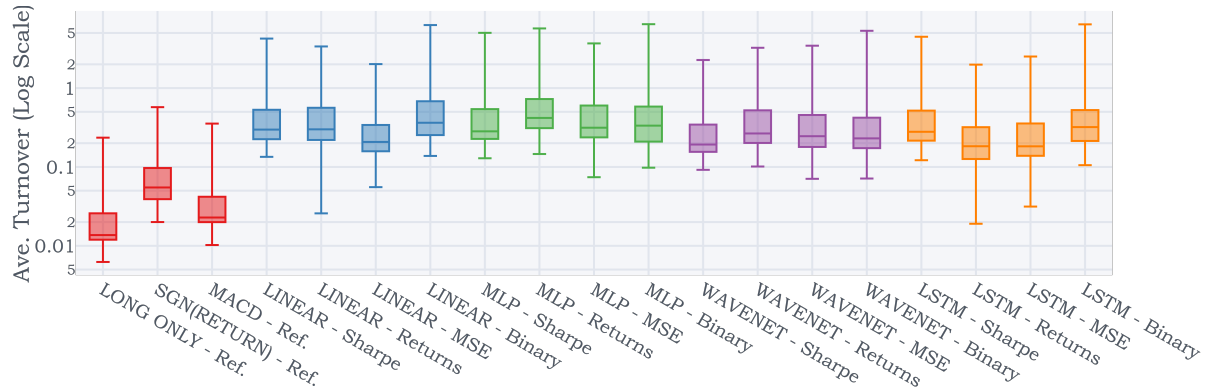
where c is a constant reflecting transaction cost assumptions. As such, using $\tilde{r}_{t,t+1}^{TSMOM}$ in Sharpe ratio loss functions during training corresponds to optimising the ex-cost risk-adjusted returns, and $c \left| \frac{X_t^{(i)}}{\sigma_t^{(i)}} - \frac{X_{t-1}^{(i)}}{\sigma_{t-1}^{(i)}}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

Given that the Sharpe-optimised LSTM is still profitable in the presence of small transactions costs, we seek to quantify the effectiveness of turnover regularisation when costs are prohibitively high – considering the extreme case where $c = 10$bps in our investigation. Tests were focused on the Sharpe-optimised LSTM with and without the turnover regulariser (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 8, we can see that the turnover regularisation does help improve the LSTM in the presence of large costs, leading to slightly better performance ratios when compared to the reference benchmarks.

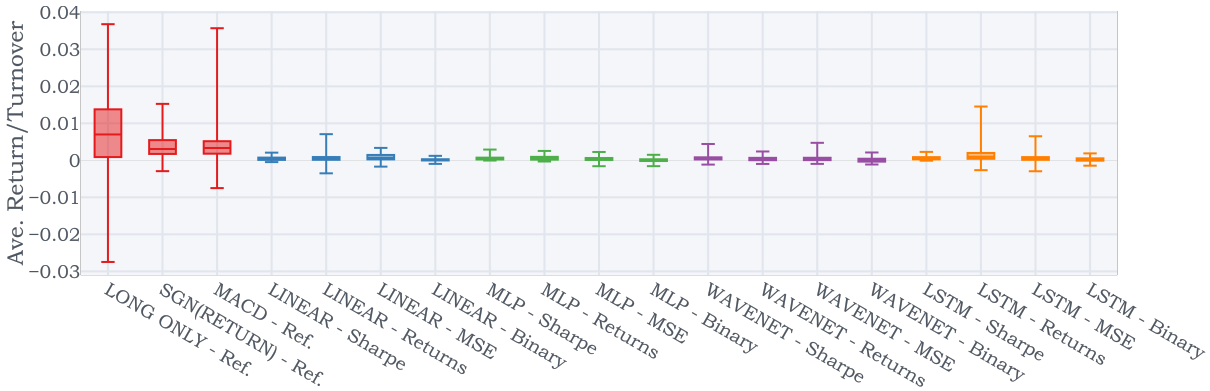
VII. Conclusions

We introduce Deep Momentum Networks – a hybrid class of deep learning models which retain 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. Firstly, 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. Secondly, trading rules were directly generated as outputs from the model, which we calibrate by maximising 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

Exhibit 6: Turnover Analysis



(a) Average Strategy Turnover



(b) Average Returns / Average Turnover

Exhibit 7: Impact of Transaction Costs on Sharpe Ratio

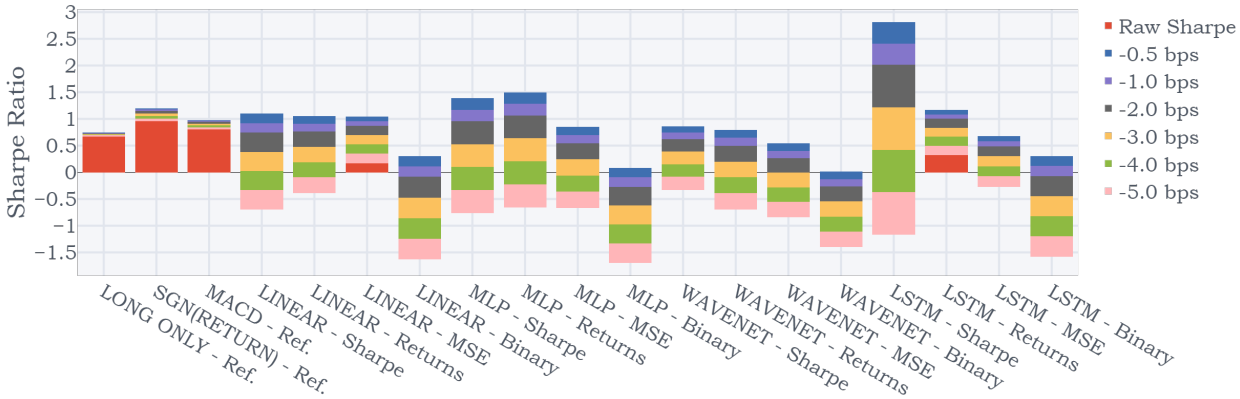


Exhibit 8: Performance Metrics with Transaction Costs ($c = 10$bps)

	E[Return]	**Vol.**	**Downside Deviation**	**MDD**	**Sharpe**	**Sortino**	**Calmar**	**% of +ve 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

ratio – where the LSTM model achieved best results. Incorporating transaction costs, the Sharpe-optimised LSTM outperforms benchmarks up to 2-3 basis points of costs, demonstrating its suitability for trading more liquid assets. To accommodate high costs settings, we introduce a turnover regulariser 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 non-stationarity in the data, such as using the recently introduced Recurrent Neural Filters [48]. Another direction of future work focuses on the study of time series momentum at the microstructure level.

VIII. Acknowledgements

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

[1] T. J. Moskowitz, Y. H. Ooi, and L. H. Pedersen, "Time series momentum," *Journal of Financial Economics*, vol. 104, no. 2, pp. 228 – 250, 2012, Special Issue on Investor Sentiment.

[2] B. Hurst, Y. H. Ooi, and L. H. Pedersen, "A century of evidence on trend-following investing," *The Journal of Portfolio Management*, vol. 44, no. 1, pp. 15–29, 2017.

[3] Y. Lemprire, C. Deremble, P. Seager, M. Potters, and J.-P. Bouchaud, "Two centuries of trend following," *Journal of Investment Strategies*, vol. 3, no. 3, pp. 41–61, 2014.

[4] J. Baz, N. Granger, C. R. Harvey, N. Le Roux, and S. Rattray, "Dissecting investment strategies in the cross section and time series," *SSRN*, 2015. [Online]. Available: https://ssrn.com/abstract=2695101

[5] A. Levine and L. H. Pedersen, "Which trend is your friend," *Financial Analysts Journal*, vol. 72, no. 3, 2016.

[6] B. Bruder, T.-L. Dao, J.-C. Richard, and T. Roncalli, "Trend filtering methods for momentum strategies," *SSRN*, 2013. [Online]. Available: https://ssrn.com/abstract=2289097

[7] A. Y. Kim, Y. Tse, and J. K. Wald, "Time series momentum and volatility scaling," *Journal of Financial Markets*, vol. 30, pp. 103 – 124, 2016.

[8] N. Baltas and R. Kosowski, "Demystifying time-series momentum strategies: Volatility estimators, trading rules and pairwise correlations," *SSRN*, 2017. [Online]. Available: https://ssrn.com/abstract=2140091

[9] C. R. Harvey, E. Hoyle, R. Korgaonkar, S. Rattray, M. Sargaison, and O. van Hemert, "The impact of volatility targeting," *SSRN*, 2018. [Online]. Available: https://ssrn.com/abstract=3175538

[10] N. Laptev, J. Yosinski, L. E. Li, and S. Smyl, "Time-series extreme event forecasting with neural networks at uber," in *Time Series Workshop – International Conference on Machine Learning (ICML)*, 2017.

[11] B. Lim and M. van der Schaar, "Disease-atlas: Navigating disease trajectories using deep learning," in *Proceedings of the 3rd Machine Learning for Healthcare Conference (MLHC)*, ser. Proceedings of Machine Learning Research, vol. 85, 2018, pp. 137–160.

[12] Z. Zhang, S. Zohren, and S. Roberts, "DeepLOB: Deep convolutional neural networks for limit order books," *IEEE Transactions on Signal Processing*, 2019.

[13] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[14] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[15] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

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

[17] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "The M4 competition: Results, findings, conclusion and way forward," *International Journal of Forecasting*, vol. 34, no. 4, pp. 802 – 808, 2018.

[18] S. Smyl, J. Ranganathan, , and A. Pasqua. (2018) M4 forecasting competition: Introducing a new hybrid es-rnn model. [Online]. Available: https://eng.uber.com/m4-forecasting-competition/

[19] M. Binkowski, G. Marti, and P. Donnat, "Autoregressive convolutional neural networks for asynchronous time series," in

Proceedings of the 35th International Conference on Machine Learning, ser. Proceedings of Machine Learning Research, vol. 80, 2018, pp. 580–589.

[20] S. S. Rangapuram, M. W. Seeger, J. Gasthaus, L. Stella, Y. Wang, and T. Januschowski, "Deep state space models for time series forecasting," in *Advances in Neural Information Processing Systems 31 (NeurIPS)*, 2018.

[21] M. Fraccaro, S. Kamronn, U. Paquet, and O. Winther, "A disentangled recognition and nonlinear dynamics model for unsupervised learning," in *Advances in Neural Information Processing Systems 30 (NIPS)*, 2017.

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

[23] S. Gu, 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 2018*, 2017. [Online]. Available: https://ssrn.com/abstract=3159577

[24] S. Kim, "Enhancing the momentum strategy through deep regression," *Quantitative Finance*, vol. 0, no. 0, pp. 1–13, 2019.

[25] J. Sirignano and R. Cont, "Universal features of price formation in financial markets: Perspectives from deep learning," *SSRN*, 2018. [Online]. Available: https://ssrn.com/abstract=3141294

[26] S. Ghoshal and S. Roberts, "Thresholded ConvNet ensembles: Neural networks for technical forecasting," in *Data Science in Fintech Workshop – Conference on Knowledge Discover and Data Mining (KDD)*, 2018.

[27] W. Bao, J. Yue, and Y. Rao, "A deep learning framework for financial time series using stacked autoencoders and long-short term memory," *PLOS ONE*, vol. 12, no. 7, pp. 1–24, 2017.

[28] P. Barroso and P. Santa-Clara, "Momentum has its moments," *Journal of Financial Economics*, vol. 116, no. 1, pp. 111 – 120, 2015.

[29] K. Daniel and T. J. Moskowitz, "Momentum crashes," *Journal of Financial Economics*, vol. 122, no. 2, pp. 221 – 247, 2016.

[30] R. Martins and D. Zou, "Momentum strategies offer a positive point of skew," *Risk Magazine*, 2012.

[31] P. Jusselin, 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. [Online]. Available: https://ssrn.com/abstract=3042173

[32] M. Potters and J.-P. Bouchaud, "Trend followers lose more than they gain," *Wilmott Magazine*, 2016.

[33] L. M. Rotando and E. O. Thorp, "The Kelly criterion and the stock market," *The American Mathematical Monthly*, vol. 99, no. 10, pp. 922–931, 1992.

[34] W. F. Sharpe, "The sharpe ratio," *The Journal of Portfolio Management*, vol. 21, no. 1, pp. 49–58, 1994.

[35] N. Jegadeesh and S. Titman, "Returns to buying winners and selling losers: Implications for stock market efficiency," *The Journal of Finance*, vol. 48, no. 1, pp. 65–91, 1993.

[36] ——, "Profitability of momentum strategies: An evaluation of alternative explanations," *The Journal of Finance*, vol. 56, no. 2, pp. 699–720, 2001.

[37] J. Rohrbach, S. Suremann, and J. Osterrieder, "Momentum and trend following trading strategies for currencies revisited - combining academia and industry," *SSRN*, 2017. [Online]. Available: https://ssrn.com/abstract=2949379

[38] Z. Zhang, S. Zohren, and S. Roberts, "BDLOB: Bayesian deep convolutional neural networks for limit order books," in *Bayesian Deep Learning Workshop – Conference on Neural Information Processing (NeurIPS)*, 2018.

[39] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[40] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu, "WaveNet: A generative model for raw audio," *CoRR*, vol. abs/1609.03499, 2016.

[41] D. Silver, 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, "Mastering the game of Go without human knowledge," *Nature*, vol. 550, pp. 354–, 2017.

[42] P. N. Kolm and G. Ritter, "Dynamic replication and hedging: A reinforcement learning approach," *The Journal of Financial Data Science*, vol. 1, no. 1, pp. 159–171, 2019.

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

[44] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, 2015.

[45] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[46] Y. Gal and Z. Ghahramani, "A theoretically grounded application of dropout in recurrent neural networks," in *Advances in Neural Information Processing Systems 29 (NIPS)*, 2016.

[47] "Pinnacle Data Corp. CLC Database," https://pinnacledata2.com/clc.html.

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

APPENDIX

A. Dataset Details

From the full 98 ratio-adjusted continuous futures contracts in the Pinnacle Data Corp CLC Database, we extract 88 which have $< 10\%$ of its data missing – with a breakdown by asset class below:

1) Commodities:

Identifier	Description
BC	BRENT CRUDE OIL, composite
BG	BRENT GASOIL, comp.
BO	SOYBEAN OIL
CC	COCOA
CL	CRUDE OIL
CT	COTTON #2
C_	CORN
DA	MILK III, Comp.
FC	FEEDER CATTLE
GC	GOLD (COMMEX)
GI	GOLDMAN SAKS C. I.
HG	COPPER
HO	HEATING OIL #2
JO	ORANGE JUICE
KC	COFFEE
KW	WHEAT, KC
LB	LUMBER
LC	LIVE CATTLE
LH	LIVE HOGS
MW	WHEAT, MINN
NG	NATURAL GAS
NR	ROUGH RICE
O_	OATS
PA	PALLADIUM
PL	PLATINUM
RB	RBOB GASOLINE
SB	SUGAR #11
SI	SILVER (COMMEX)
SM	SOYBEAN MEAL
S_	SOYBEANS
W_	WHEAT, CBOT
ZA	PALLADIUM, electronic
ZB	RBOB, Electronic
ZC	CORN, Electronic
ZF	FEEDER CATTLE, Electronic
ZG	GOLD, Electronic
ZH	HEATING OIL, electronic
ZI	SILVER, Electronic
ZK	COPPER, electronic
ZL	SOYBEAN OIL, Electronic
ZM	SOYBEAN MEAL, Electronic
ZN	NATURAL GAS, electronic
ZO	OATS, Electronic
ZP	PLATINUM, electronic
ZR	ROUGH RICE, Electronic
ZS	SOYBEANS, Electronic
ZT	LIVE CATTLE, Electronic
ZU	CRUDE OIL, Electronic
ZW	WHEAT, Electronic
ZZ	LEAN HOGS, Electronic

2) Equities:

Identifier	Description
AX	GERMAN DAX INDEX
CA	CAC40 INDEX
EN	NASDAQ, MINI
ER	RUSSELL 2000, MINI
ES	S & P 500, MINI
HS	HANG SENG
LX	FTSE 100 INDEX
MD	S&P 400 (Mini electronic)
SC	S & P 500, composite
SP	S & P 500, day session
XU	DOW JONES EUROSTOXX50
XX	DOW JONES STOXX 50
YM	Mini Dow Jones ($5.00)

3) Fixed Income:

Identifier	Description
AP	AUSTRALIAN PRICE INDEX
DT	EURO BOND (BUND)
FA	T-NOTE, 5yr day session
FB	T-NOTE, 5yr composite
GS	GILT, LONG BOND
TA	T-NOTE, 10yr day session
TD	T-NOTES, 2yr day session
TU	T-NOTES, 2yr composite
TY	T-NOTE, 10yr composite
UA	T-BONDS, day session
UB	EURO BOBL
US	T-BONDS, composite

4) FX:

Identifier	Description
AD	AUSTRALIAN $$, day session
AN	AUSTRALIAN $$, composite
BN	BRITISH POUND, composite
CB	CANADIAN 10YR BOND
CN	CANADIAN $$, composite
DX	US DOLLAR INDEX
FN	EURO, composite
FX	EURO, day session
JN	JAPANESE YEN, composite
MP	MEXICAN PESO
NK	NIKKEI INDEX
SF	SWISS FRANC, day session
SN	SWISS FRANC, composite

To reduce the impact of outliers, we also winsorise the data by capping/flooring it to be within 5 times its exponentially weighted moving (EWM) standard deviations from its EWM average – computed using a 252-day half life.

Exhibit 9: 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 Regularisation Weight (α)	10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}	Lasso Regression Only

B. Hyperparameter Optimisation

Hyperparameter optimisation was applied using 50 iterations of random search, with the full search grid documented in Exhibit 9, with the models fully recalibrated every 5 years using all available data up to that point. For LSTM-based models, time series were subdivided into trajectories of 63 time steps ($\approx$ 3 months), with the LSTM unrolled across the length of the trajectory during backpropagation.

C. Additional Results

In addition to the selected results in Section V, we also present a full list of results was presented for completeness – echoing the key findings reported in the discuss. Detailed descriptions of the plots and tables can be found below:

1) Cross-Validation Performance: The testing procedure in Section V can also be interpreted as a cross-validation approach – splitting the original dataset into six 5-year blocks (1990-2015), calibrating using an expanding window of data, and testing out-of-sample on the next block outside the training set. As such, for consistency with machine learning literature, we present our results in a cross validation format as well – reporting the average value across all blocks $\pm$ 2 standard deviations. Furthermore, this also gives an indication of how signal performance varies across the various time periods.

- **Exhibit 10** – Cross-validation results for raw signal outputs.
- **Exhibit 11** – Cross-validation results for signals which have been rescaled to target volatility at the portfolio level.

2) Metrics Across Individual Assets: We also provide additional plots on performance of other risk metrics and performance ratios across individual assets, as described below:

- **Exhibit 12** – Full performance ratios (Sharpe, Sortino, Calmar).
- **Exhibit 13** – Box plots including additional risk metrics (volatility, downside deviation and maximum drawdown). We note that our findings for other risk metrics are similar to volatility – with Sharpe-optimised models lowering risk across different methods.

Exhibit 10: Cross-Validation Performance – Raw Signal Outputs

	E[Return]	**Vol.**	**Downside Deviation**	**MDD**
Reference				
Long Only	0.043 ± 0.028	0.054 ± 0.016	0.037 ± 0.013	0.116 ± 0.091
Sgn(Returns)	0.047 ± 0.051	0.046 ± 0.012	0.032 ± 0.007	0.067 ± 0.041
MACD	0.026 ± 0.032	0.032 ± 0.008	0.023 ± 0.007	0.054 ± 0.048
Linear				
Sharpe	0.034 ± 0.030	0.039 ± 0.028	0.028 ± 0.020	0.072 ± 0.096
Ave. Returns	0.033 ± 0.031	0.046 ± 0.025	0.031 ± 0.018	0.110 ± 0.114
MSE	0.047 ± 0.038	0.049 ± 0.019	0.033 ± 0.013	0.100 ± 0.121
Binary	0.012 ± 0.028	0.045 ± 0.011	0.031 ± 0.009	0.109 ± 0.045
MLP				
Sharpe	0.038 ± 0.027	0.030 ± 0.041	0.021 ± 0.028	0.062 ± 0.160
Ave. Returns	**0.056 ± 0.046***	0.044 ± 0.024	0.030 ± 0.017	0.075 ± 0.150
MSE	0.037 ± 0.051	0.048 ± 0.021	0.032 ± 0.015	0.109 ± 0.134
Binary	-0.004 ± 0.028	0.042 ± 0.007	0.028 ± 0.006	0.111 ± 0.079
WaveNet				
Sharpe	0.030 ± 0.030	0.038 ± 0.019	0.027 ± 0.015	0.069 ± 0.055
Ave. Returns	0.034 ± 0.043	0.042 ± 0.003	0.030 ± 0.003	0.088 ± 0.062
MSE	0.024 ± 0.046	0.043 ± 0.010	0.030 ± 0.010	0.102 ± 0.056
Binary	-0.009 ± 0.023	0.043 ± 0.008	0.030 ± 0.008	0.159 ± 0.107
LSTM				
Sharpe	0.045 ± 0.030	**0.017 ± 0.004***	**0.012 ± 0.003***	**0.019 ± 0.005***
Ave. Returns	0.045 ± 0.050	0.048 ± 0.018	0.034 ± 0.011	0.104 ± 0.119
MSE	0.023 ± 0.037	0.048 ± 0.022	0.033 ± 0.017	0.116 ± 0.082
Binary	-0.005 ± 0.088	0.042 ± 0.003	0.027 ± 0.006	0.151 ± 0.211

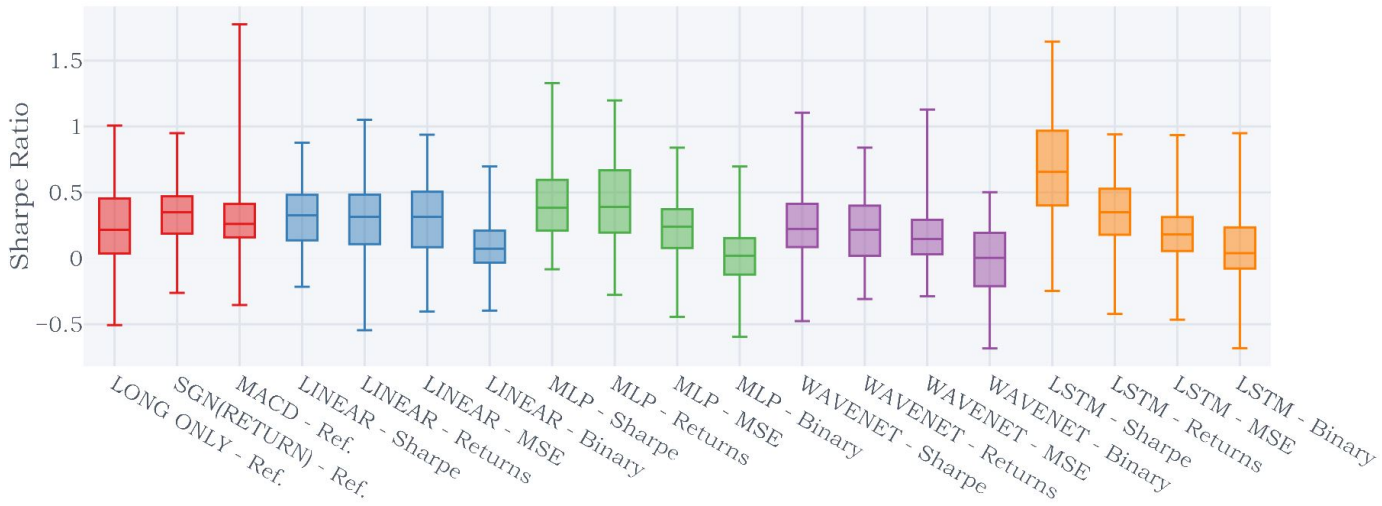
	Sharpe	**Sortino**	**Calmar**	**Fraction of +ve Returns**	**$\frac{\text{Ave. P}}{\text{Ave. L}}$**
Reference					
Long Only	0.839 ± 0.786	1.258 ± 1.262	0.420 ± 0.490	0.546 ± 0.025	0.956 ± 0.135
Sgn(Returns)	1.045 ± 1.230	1.528 ± 1.966	0.864 ± 1.539	0.543 ± 0.061	1.002 ± 0.067
MACD	0.839 ± 1.208	1.208 ± 1.817	0.625 ± 1.033	0.532 ± 0.030	1.016 ± 0.079
Linear					
Sharpe	1.025 ± 1.530	1.451 ± 2.154	0.800 ± 1.772	0.544 ± 0.042	1.000 ± 0.100
Ave. Returns	0.757 ± 0.833	1.150 ± 1.378	0.397 ± 0.686	0.530 ± 0.009	1.005 ± 0.100
MSE	1.012 ± 1.126	1.532 ± 1.811	0.708 ± 1.433	0.540 ± 0.024	1.008 ± 0.096
Binary	0.288 ± 0.729	0.434 ± 1.113	0.123 ± 0.313	0.506 ± 0.027	1.024 ± 0.051
MLP					
Sharpe	1.669 ± 2.332	2.420 ± 3.443	1.665 ± 2.738	0.554 ± 0.063	1.069 ± 0.151
Ave. Returns	1.415 ± 1.781	2.127 ± 2.996	1.520 ± 2.761	0.553 ± 0.043	1.022 ± 0.134
MSE	0.821 ± 1.334	1.270 ± 2.160	0.652 ± 1.684	0.525 ± 0.025	1.036 ± 0.127
Binary	-0.099 ± 0.648	-0.180 ± 0.956	-0.013 ± 0.304	0.500 ± 0.042	0.986 ± 0.064
WaveNet					
Sharpe	0.780 ± 0.538	1.118 ± 0.854	0.477 ± 0.610	0.535 ± 0.022	0.990 ± 0.094
Ave. Returns	0.809 ± 1.113	1.160 ± 1.615	0.501 ± 1.036	0.543 ± 0.059	0.963 ± 0.069
MSE	0.513 ± 0.991	0.744 ± 1.477	0.276 ± 0.509	0.527 ± 0.033	0.979 ± 0.077
Binary	-0.220 ± 0.523	-0.329 ± 0.768	-0.043 ± 0.145	0.499 ± 0.011	0.969 ± 0.043
LSTM					
Sharpe	**2.781 ± 2.081***	**3.978 ± 3.160***	**2.488 ± 1.921***	**0.593 ± 0.054***	**1.104 ± 0.199***
Ave. Returns	0.961 ± 1.268	1.397 ± 1.926	0.679 ± 1.552	0.547 ± 0.039	0.972 ± 0.118
MSE	0.451 ± 0.526	0.668 ± 0.812	0.184 ± 0.170	0.520 ± 0.026	0.996 ± 0.048
Binary	-0.114 ± 2.147	-0.191 ± 3.435	0.227 ± 1.241	0.495 ± 0.077	1.002 ± 0.077

Exhibit 11: Cross-Validation Performance – Rescaled to Target Volatility

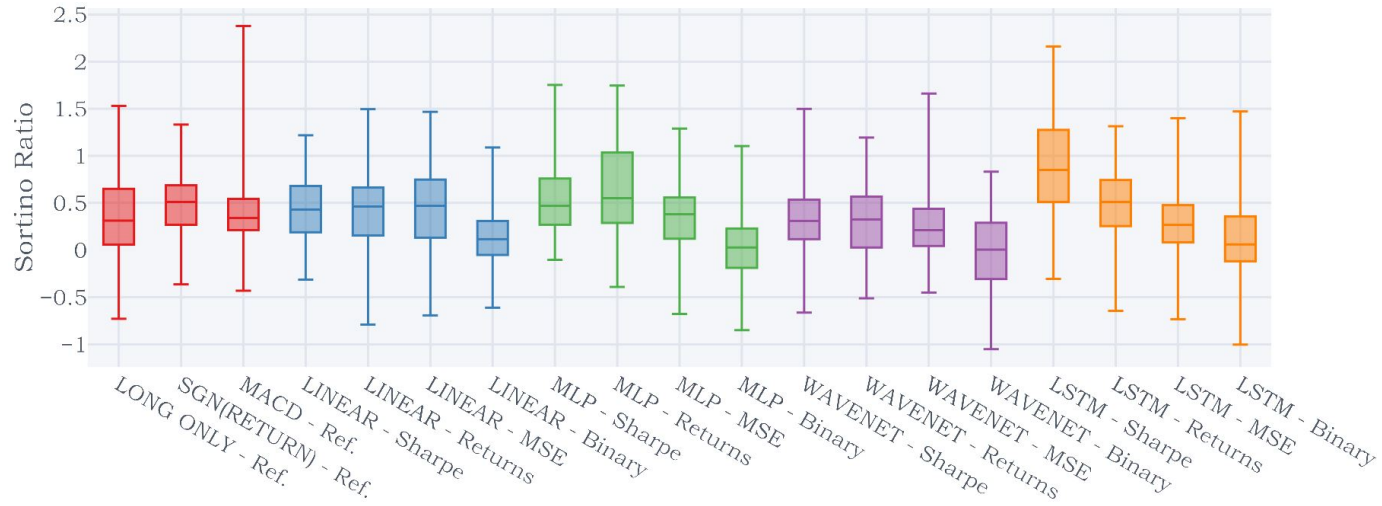
	E[Return]	**Vol.**	**Downside Deviation**	**MDD**
Reference				
Long Only	0.131 ± 0.142	0.154 ± 0.001	0.104 ± 0.014	0.304 ± 0.113
Sgn(Returns)	0.186 ± 0.184	0.154 ± 0.002	0.101 ± 0.012	0.194 ± 0.126
MACD	0.140 ± 0.166	0.154 ± 0.002	0.105 ± 0.010	0.243 ± 0.129
Linear				
Sharpe	0.182 ± 0.273	0.155 ± 0.003	0.105 ± 0.007	0.232 ± 0.175
Ave. Returns	0.127 ± 0.141	0.154 ± 0.003	0.101 ± 0.009	0.318 ± 0.177
MSE	0.170 ± 0.189	0.154 ± 0.003	**0.099 ± 0.006***	0.256 ± 0.221
Binary	0.049 ± 0.170	0.155 ± 0.002	0.104 ± 0.013	0.351 ± 0.114
MLP				
Sharpe	0.271 ± 0.375	0.154 ± 0.008	0.104 ± 0.000	0.186 ± 0.259
Ave. Returns	0.233 ± 0.270	0.154 ± 0.003	0.101 ± 0.010	0.194 ± 0.277
MSE	0.148 ± 0.178	0.154 ± 0.003	0.100 ± 0.009	0.268 ± 0.262
Binary	-0.011 ± 0.117	0.154 ± 0.002	0.102 ± 0.018	0.377 ± 0.221
WaveNet				
Sharpe	0.131 ± 0.103	0.154 ± 0.002	0.104 ± 0.009	0.254 ± 0.164
Ave. Returns	0.142 ± 0.196	0.154 ± 0.002	0.103 ± 0.003	0.262 ± 0.204
MSE	0.087 ± 0.150	**0.153 ± 0.003***	0.101 ± 0.009	0.307 ± 0.247
Binary	-0.030 ± 0.099	0.155 ± 0.001	0.105 ± 0.006	0.485 ± 0.283
LSTM				
Sharpe	**0.435 ± 0.342***	0.155 ± 0.002	0.108 ± 0.012	**0.164 ± 0.077***
Ave. Returns	0.157 ± 0.202	**0.153 ± 0.002***	0.102 ± 0.011	0.285 ± 0.196
MSE	0.087 ± 0.091	0.154 ± 0.003	0.100 ± 0.006	0.310 ± 0.130
Binary	-0.008 ± 0.332	0.155 ± 0.002	0.100 ± 0.009	0.428 ± 0.495

	Sharpe	**Sortino**	**Calmar**	**Fraction of +ve Returns**	$\frac{\text{Ave. P}}{\text{Ave. L}}$
Reference					
Long Only	0.847 ± 0.915	1.287 ± 1.475	0.445 ± 0.579	0.546 ± 0.025	0.958 ± 0.164
Sgn(Returns)	1.213 ± 1.205	1.856 ± 1.944	1.098 ± 1.658	0.543 ± 0.061	1.028 ± 0.070
MACD	0.911 ± 1.086	1.361 ± 1.733	0.643 ± 0.958	0.532 ± 0.030	1.023 ± 0.074
Linear					
Sharpe	1.176 ± 1.772	1.752 ± 2.615	1.060 ± 2.376	0.544 ± 0.042	1.025 ± 0.139
Ave. Returns	0.826 ± 0.914	1.287 ± 1.504	0.471 ± 0.777	0.530 ± 0.009	1.016 ± 0.116
MSE	1.101 ± 1.220	1.729 ± 2.037	0.890 ± 1.787	0.540 ± 0.024	1.022 ± 0.116
Binary	0.321 ± 1.105	0.509 ± 1.720	0.169 ± 0.585	0.506 ± 0.027	1.031 ± 0.127
MLP					
Sharpe	1.757 ± 2.405	2.623 ± 3.626	2.091 ± 3.474	0.554 ± 0.063	1.085 ± 0.176
Ave. Returns	1.516 ± 1.764	2.336 ± 2.923	1.771 ± 2.889	0.553 ± 0.043	1.038 ± 0.141
MSE	0.960 ± 1.163	1.510 ± 1.927	0.864 ± 1.999	0.525 ± 0.025	1.059 ± 0.103
Binary	-0.071 ± 0.756	-0.140 ± 1.133	0.000 ± 0.372	0.500 ± 0.042	0.991 ± 0.072
WaveNet					
Sharpe	0.849 ± 0.663	1.270 ± 1.060	0.575 ± 0.680	0.535 ± 0.022	1.000 ± 0.121
Ave. Returns	0.920 ± 1.271	1.376 ± 1.915	0.738 ± 1.591	0.543 ± 0.059	0.979 ± 0.066
MSE	0.565 ± 0.972	0.854 ± 1.513	0.364 ± 0.665	0.527 ± 0.033	0.986 ± 0.081
Binary	-0.196 ± 0.641	-0.298 ± 0.946	-0.044 ± 0.206	0.499 ± 0.011	0.974 ± 0.067
LSTM					
Sharpe	**2.803 ± 2.195***	**4.084 ± 3.469***	**2.887 ± 3.030***	**0.593 ± 0.054***	**1.106 ± 0.216***
Ave. Returns	1.023 ± 1.312	1.564 ± 2.131	0.706 ± 1.440	0.547 ± 0.039	0.980 ± 0.127
MSE	0.563 ± 0.580	0.865 ± 0.901	0.284 ± 0.269	0.520 ± 0.026	1.014 ± 0.016
Binary	-0.050 ± 2.152	-0.122 ± 3.381	0.190 ± 1.152	0.495 ± 0.077	1.012 ± 0.048

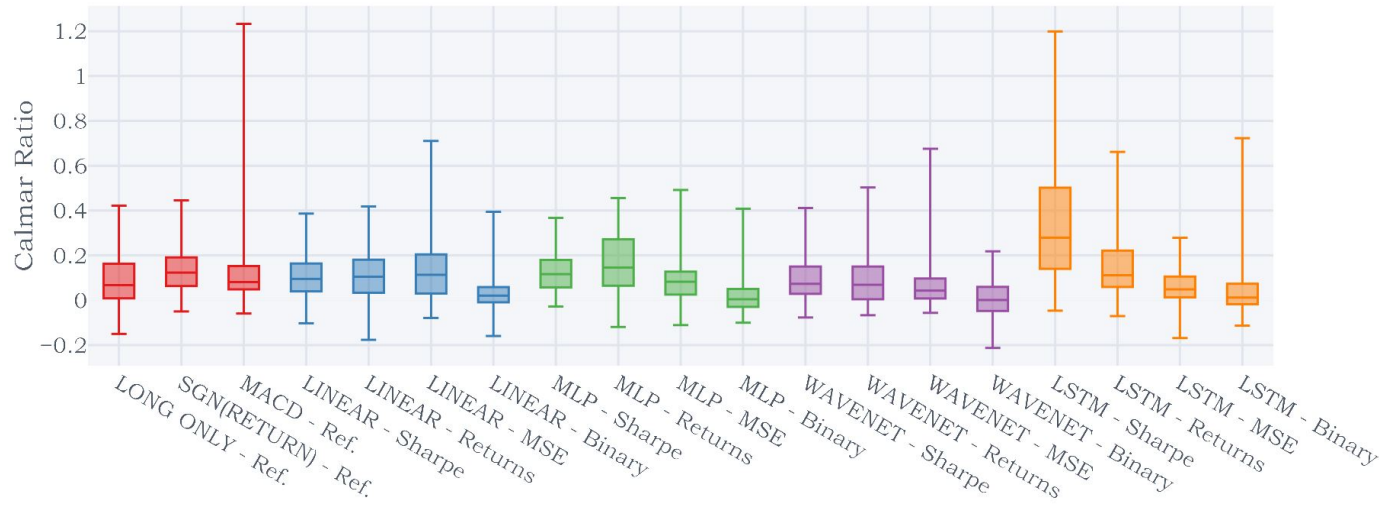
Exhibit 12: Performance Ratios Across Individual Assets



(a) Sharpe Ratio

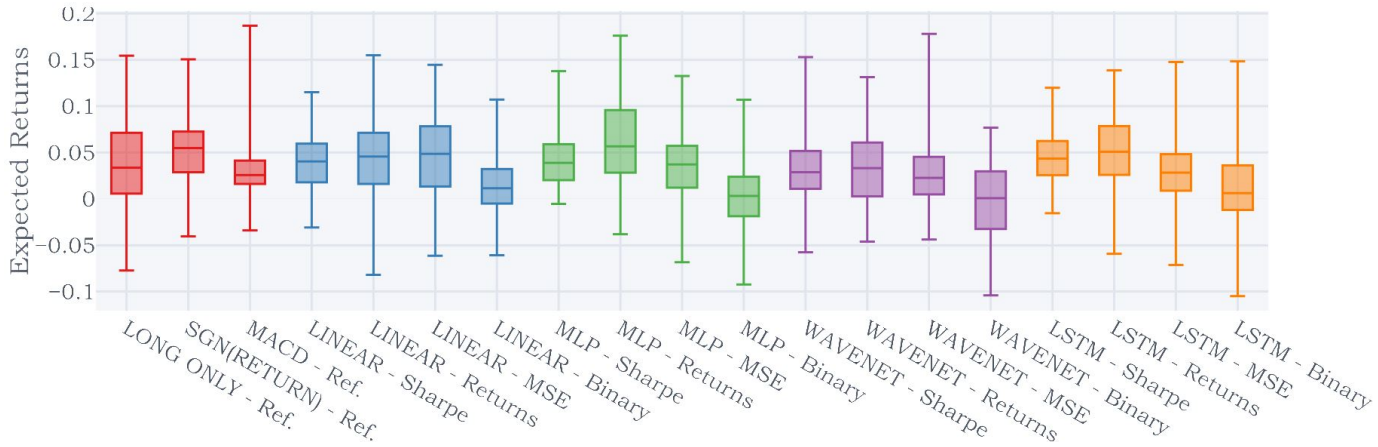


(b) Sortino Ratio

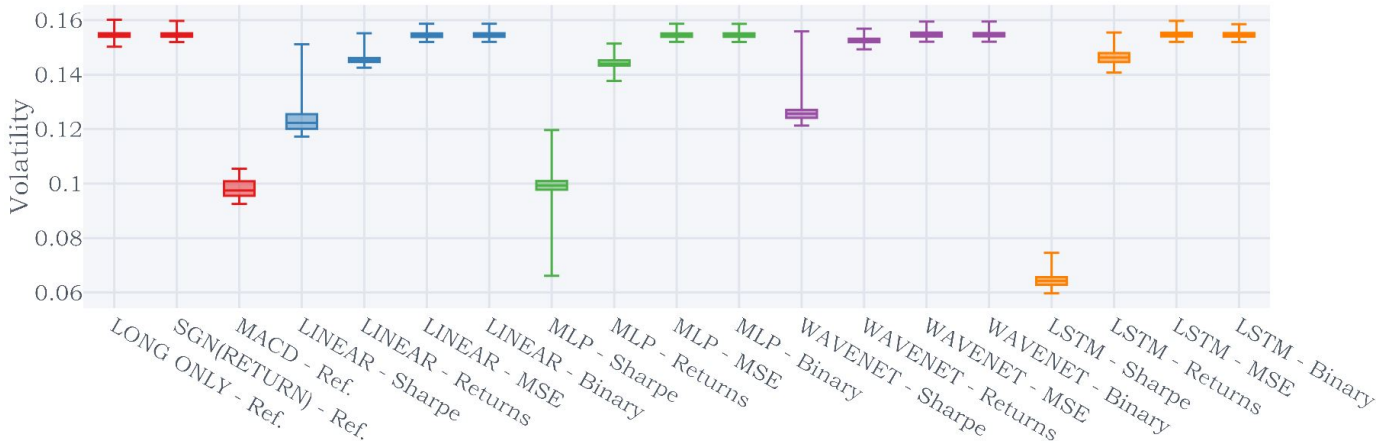


(c) Calmar Ratio

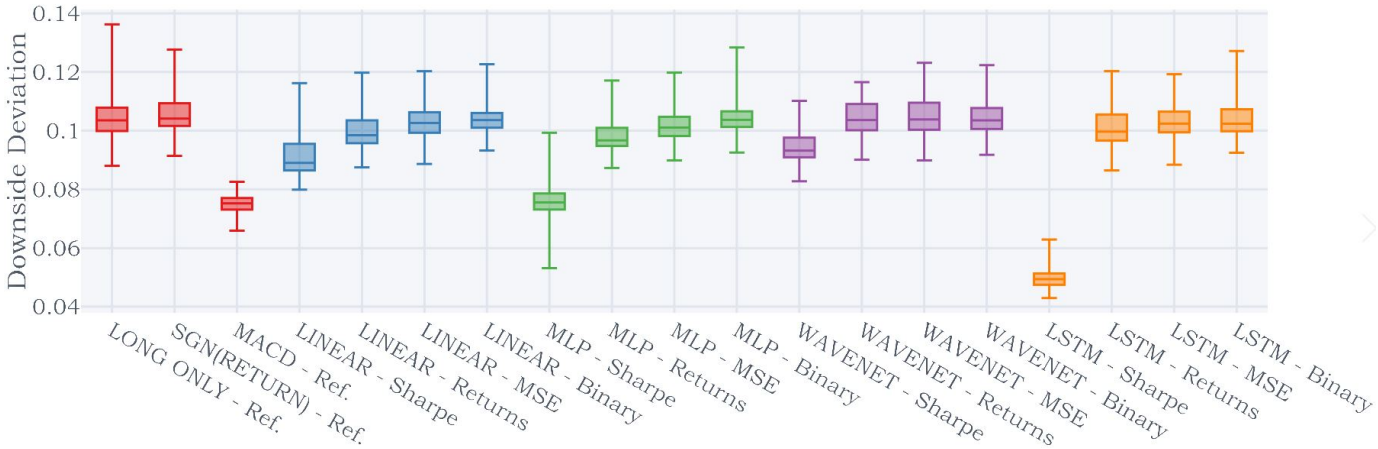
Exhibit 13: Reward vs Risk Across Individual Assets



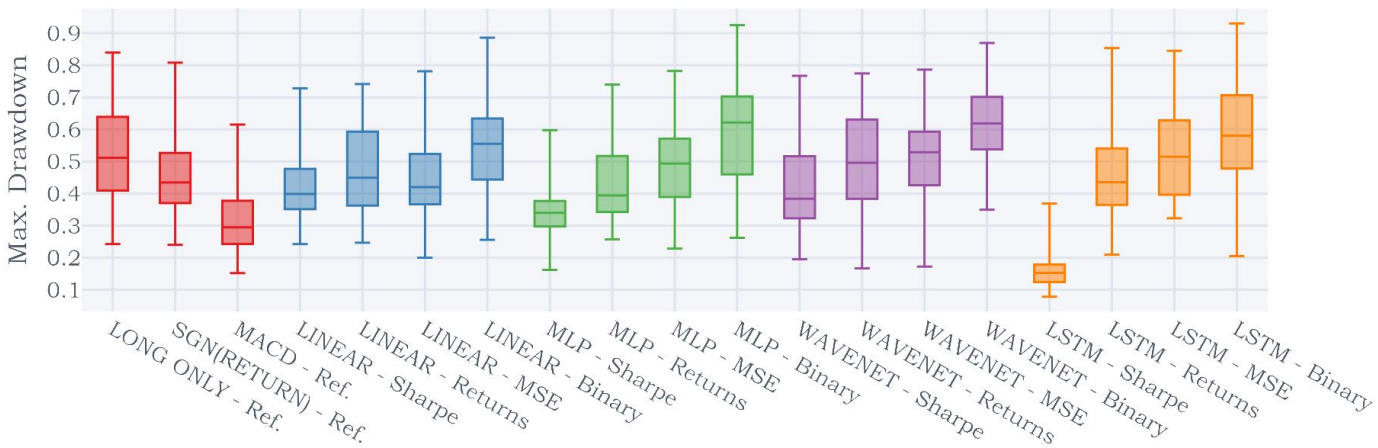
(a) Expected Returns



(b) Volatility



(c) Downside Deviation



(d) Max. Drawdown