Slow Momentum with Fast Reversion: A Trading Strategy Using Deep Learning and Changepoint Detection

Kieran Wood, Stephen Roberts, and Stefan Zohren

Kieran Wood
is a DPhil student with the Machine Learning Research Group and the Oxford-Man Institute of Quantitative Finance at the University of Oxford in Oxford, UK.
kieran.wood@eng.ox.ac.uk

Stephen Roberts
is the RAEng/Man professor of machine learning at the University of Oxford and the director of the Oxford-Man Institute of Quantitative Finance at the University of Oxford in Oxford, UK.
sjrob@robots.ox.ac.uk

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

KEY FINDINGS

- Momentum strategies, including deep learning–based deep momentum networks, have underperformed in recent years owing to difficulties in adjusting to rapid changes in the market, such as when a trend reverses from an uptrend to a downtrend, or vice versa.
- Inserting an online changepoint detection module into a deep momentum network pipeline leads to large performance gains, especially during periods of significant nonstationarity, as observed in recent years.
- The model achieves superior risk-adjusted returns by blending a slow momentum strategy with a fast mean-reversion strategy, with the changepoint detection module helping to balance the two in a data-driven manner.

ABSTRACT

Momentum strategies are an important part of alternative investments and are at the heart of the work of commodity trading advisors. These strategies have, however, been found to have difficulties adjusting to rapid changes in market conditions, such as during the 2020 market crash. In particular, immediately after momentum turning points, when a trend reverses from an uptrend (downtrend) to a downtrend (uptrend), time-series momentum strategies are prone to making bad bets. To improve the responsiveness to regime change, the authors introduce a novel approach, in which they insert an online changepoint detection (CPD) module into a deep momentum network pipeline, which uses a long short-term memory deep-learning architecture to simultaneously learn both trend estimation and position sizing. Furthermore, their model is able to optimize the way in which it balances (1) a slow momentum strategy that exploits persisting trends but does not overreact to localized price moves and (2) a fast mean-reversion strategy regime by quickly flipping its position and then swapping back again to exploit localized price moves. The CPD module outputs a changepoint location and severity score, allowing the model to learn to respond to varying degrees of disequilibrium, or smaller and more localized changepoints, in a data-driven manner. The authors back test their model over the period 1995–2020, and the addition of the CPD module leads to a 33% improvement in the Sharpe ratio. The module is especially beneficial in periods of significant nonstationarity; in particular, over the most recent years tested (2015–2020), the performance boost is approximately 66%. This is especially interesting because traditional momentum strategies underperformed in this period.

Time-series momentum (TSMOM) (Moskowitz, Ooi, and Pedersen 2012) strategies are derived from the philosophy that strong price trends tend to persist. These trends have been observed to hold across a range of timescales, asset classes, and time periods (Lempérière et al. 2014; Baz et al. 2015; Hurst, Ooi, and Pedersen 2017). Momentum strategies are often referred to as *follow the winner* because it is assumed that winners will continue to be winners in the subsequent period.

Momentum strategies are an important part of alternative investments and are at the heart of the work of commodity trading advisors. Much effort goes into quantifying the magnitude of trends (Bruder et al. 2013; Baz et al. 2015; Levine and Pedersen 2016) and sizing traded positions accordingly (Kim, Tse, and Wald 2016; Baltas and Kosowski 2017; Harvey et al. 2018). Rather than using handcrafted techniques to identify trends and select positions, Lim, Zohren, and Roberts (2019) introduced *deep momentum networks* (DMNs), in which a long short-term memory (LSTM) (Hochreiter and Schmidhuber 1997) deep learning architecture achieves this task by directly optimizing on the Sharpe ratio of the signal. Deep learning has been widely used for time-series forecasting (Lim and Zohren 2020), achieving a high level of accuracy across various fields, including the field of finance for daily data (Bao, Yue, and Rao 2017; Gu, Kelly, and Xiu 2017; Lim, Zohren, and Roberts 2019; Kim 2019; Poh et al. 2021) and in a high-frequency setting, using limit order book data (Sirignano and Cont 2018; Zhang, Zohren, and Roberts 2019). In recent years, implementation of such deep learning models has been made accessible via extensive open-source frameworks such as TensorFlow (Abadi et al. 2015) and PyTorch (Paszke et al. 2017).

Momentum strategies aim to capitalize on persisting price trends; however, occasionally these trends break down, which we label *momentum turning points*. At these turning points, momentum strategies are prone to performing poorly because they are unable to adapt quickly to this abrupt change in regime. This concept is explored by Garg et al. (2021) who blended a slow momentum signal based on a long lookback window (LBW), such as 12 months, with a fast momentum signal based on a short LBW, such as 1 month. This approach is a balancing act between reducing noise and being quick enough to respond to turning points. Adopting the terminology from Garg et al. (2021), a bull or bear market is when the two momentum signals agree on a long or short position, respectively. If slow momentum suggests a long (short) position and fast momentum a short (long) position, we term this a *correction (rebound) phase*.

Correction and rebound phases, in which the momentum assumption breaks down, are examples of mean reversion (De Bondt and Thaler 1985; Poterba and Summers 1988; Jegadeesh 1991) regimes. Mean-reversion trading strategies, often referred to as *follow the loser* strategies, assume losers (winners) over some LBW will be winners (losers) in the subsequent period. If we observe the positions taken by a DMN, alongside exploiting persisting trends, the model also exploits fluctuations in return data at a shorter time horizon by regularly flipping its position and then quickly changing back again. We argue that the high Sharpe ratio achieved by DMNs can be largely attributed to this fast mean-reversion property.

Changepoint detection (CPD) is a field that involves the identification of abrupt changes in sequential data, in which the generative parameters for our model after the changepoint are independent of those that come before. The nonstationarity of real-world time series in fields such as finance, robotics, and sensor data has led to a plethora of research in this field. To respond to CPD in real time, we require an online algorithm, which processes each data point as it becomes available, as opposed to offline algorithms that consider the entire dataset at once and detect changepoints retrospectively. First introduced by Adams and MacKay (2007), Bayesian approaches to online CPD, which naturally accommodate to noisy, uncertain, and incomplete time-series data, have proven to be very successful. Assuming a changepoint model of the parameters, the Bayesian approach integrates out the uncertainty for these parameters as opposed to using a point estimate. Gaussian processes (GPs) (Williams and Rasmussen 1996; Rasmussen 2003), which are collections of random variables any finite number of which have joint Gaussian distributions, are well suited to time-series modeling (Roberts et al. 2013). GPs are often referred to as a Bayesian nonparametric model and have the ability to handle changepoints (Garnett et al. 2010; Saatçi, Turner, and Rasmussen 2010; Lloyd et al. 2014). Rather than comparing slow and fast momentum signals to detect regime change, we use GPs as a more principled method for detecting momentum turning points. For our experiments, we use the Python package GPflow (Matthews et al. 2017) to build Gaussian process models, which leverage the TensorFlow framework.

In this article, we introduce a novel approach, in which we add an online CPD module to a DMN pipeline to improve overall strategy returns. By incorporating the CPD module, we optimize our response to momentum turning points in a data-driven manner by passing outputs from the module as inputs to a DMN, which in turn learns trading rules and optimizes positions based on some finance value function, such as the Sharpe ratio (Sharpe 1994). This approach helps to correctly identify when we are in a bull or bear market and select the momentum strategy accordingly. With the addition of the CPD module, the new model learns how to exploit, but not overreact to, noise at a shorter time scale. Our strategy is able to exploit the fast reversion we observe in DMNs but effectively balance this with a slow momentum strategy and improve returns across an entire bull or bear regime. Effectively, the new pipeline has more knowledge on how to respond to abrupt changes, or a lack of changes, in a data-driven way.

We argue that the CPD is an artificial construct that can have varying degrees of severity and is dependent on choices such as the length of the lookback horizon. Rather than specifying regimes based on some criterion or threshold, we use our CPD module to quantify, or score, the level of disequilibrium, allowing the model to consider smaller or more localized regime changes. The length of the LBW is the most sensitive design choice for the CPD module—if the lookback horizon is too long, we miss smaller but still potentially significant regime changes, and if the horizon is too short, the data become too noisy and are of little value. We introduce the LBW length as a structural hyperparameter that we optimize using the outer optimization loop of our model. This allows the module to be more tightly coupled with our LSTM module, thus helping us to maximize the efficiency of the CPD and allowing us to tweak the LSTM hyperparameters in conjunction with the LBW.

It can be noted that the performance of DMNs, without CPD, deteriorates in more recent years. The deterioration in performance is especially notable in the 2015–2020 period, which exhibits a greater degree of turbulence, or disequilibrium, than the preceding years. One possible explanation for deterioration in momentum strategies in recent years is the concept of *factor crowding*, which is discussed in depth by Baltas (2019), who argued that arbitrageurs inflict negative externalities on one another. By using the same models, and hence taking the same positions, a coordination problem is created, pushing the price away from fundamentals. It is argued that momentum strategies are susceptible to this scenario. Impressively, the addition of a CPD module helps to alleviate the deterioration in performance, and our model significantly outperforms the standard DMN model during the 2015–2020 period. A similar phenomenon can be observed from around 2003, when electronic trading was becoming more common, where the deep learning–based strategies start to significantly outperform classic TSMOM strategies.

CHANGEPOINT DETECTION USING GAUSSIAN PROCESSES

A classic univariate regression problem of the form $y(x) = f(x) + \epsilon$, where ϵ is an additive noise process, has the goal of evaluating the function f and the probability distribution $p(y_*|x_*)$ of some point y_* given some x_*. Our daily time-series data, for asset i, consist of a sequence of observations for (closing) price $\{p_t^{(i)}\}_{t=1}^{T}$, up to time T. Because financial time series are nonstationary in the mean, for each time t we take the first difference of the time series, otherwise known as the arithmetic returns

$$r_{t-1,t}^{(i)} = \frac{p_t^{(i)} - p_{t-1}^{(i)}}{p_{t-1}^{(i)}} \tag{1}$$

in an attempt to remove any linear trend in the mean. Throughout this article, for brevity, we will refer to $r_{t-1,t}$ simply as r_t. For the purposes of CPD, it is not computationally feasible, nor is it necessary, to consider the entire time series; hence, we consider the series $\{r_t^{(i)}\}_{t=T-l}^{T}$, with lookback horizon l from time T. For every CPD window, where $\mathcal{T}=\{T-l, T-l+1, \dots, T\}$, we standardize our returns as

$$\hat{r}_t^{(i)} = \frac{r_t^{(i)} - \mathbb{E}_{\mathcal{T}}\left[r_t^{(i)}\right]}{\sqrt{\mathrm{Var}_{\mathcal{T}}\left[r_t^{(i)}\right]}} \tag{2}$$

This step is taken for two reasons: We can assume that the mean over our window is zero, and with unit variance, we have more consistency across all windows when we run our CPD module.

Our approach to changepoint detection involves a curve-fitting approach for input–output pairs $(t, \hat{r}_t^{(i)})$ via the use of GP regression (Rasmussen 2003). GP regression is a probabilistic, nonparametric method, popular in the fields of machine learning and time-series analysis (Roberts et al. 2013). It is a kernel-based technique in which the $\mathcal{GP}$ is specified by a covariance function $k_\xi(\cdot)$, which is in turn parameterized by a set of hyperparameters ξ. In its common guise, the GP has a stationary kernel; however, it should be noted that GPs can readily work well even when the time series is nonstationary (Brahim-Belhouari and Bermak 2004). We define the GP as a distribution over functions where

$$\hat{r}_t^{(i)} = f(t) + \epsilon_t, f \sim \mathcal{GP}(0, k_\xi), \epsilon_t \sim \mathcal{N}(0, \sigma_n^2) \tag{3}$$

given noise variance σ_n, which helps to deal with noisy outputs that are uncorrelated.

Rizvi (2018) and Liu, Kiskin, and Roberts (2020) demonstrated that a Matérn 3/2 kernel is a good choice of covariance function for noisy financial data, which tend to be highly nonsmooth and not infinitely differentiable. This problem setting favors the least smooth of the Matérn family of kernels, which is the 3/2 kernel. We parametrize our Matérn 3/2 kernel as

$$k(x, x') = \sigma_h^2 \left(1 + \frac{\sqrt{3}|x - x'|}{\lambda}\right) e^{\left(-\frac{\sqrt{3}|x-x'|}{\lambda}\right)} \tag{4}$$

with kernel hyperparameters $\xi_M = (\lambda, \sigma_h, \sigma_n)$, where λ is the input scale and σ_h the output scale. We define our covariance matrix for a set of locations $\mathrm{x} = [x_1, x_2, \dots x_n]$ as

$$\mathbf{K}(\mathbf{x},\mathbf{x}) = \begin{pmatrix} k(x_1,x_1) & \cdots & k(x_1,x_n) \\ \vdots & \ddots & \vdots \\ k(x_n,x_1) & \cdots & k(x_n,x_n) \end{pmatrix} \tag{5}$$

Using $\hat{\mathbf{r}} = [\hat{r}_{t-l}, \ldots, \hat{r}_t]$, we integrate out the function variables to give $p(\hat{\mathbf{r}} \mid \xi) = \mathcal{N}(\mathbf{0}, \mathbf{V})$, with $\mathbf{V} = \mathbf{K} + \sigma_n^2 \mathbf{I}$. Because $p(\xi|\hat{\mathbf{r}})$ is intractable, we instead apply Bayes' rule

$$p(\xi|\hat{\mathbf{r}}) = \frac{p(\hat{\mathbf{r}}|\xi)p(\xi)}{p(\hat{\mathbf{r}})} \tag{6}$$

and perform type II maximum likelihood on $p(\hat{\mathbf{r}}|\xi)$. We minimize the negative log marginal likelihood:

$$\text{nlml}_\xi = \min_\xi \left(\frac{1}{2} \hat{\mathbf{r}}^\top \mathbf{V}^{-1} \hat{\mathbf{r}} + \frac{1}{2} \log |\mathbf{V}| + \frac{l+1}{2} \log 2\pi \right) \tag{7}$$

We use the GPflow framework to compute the hyperparameters ξ, which in turn uses the L-BFGS-B optimization algorithm (Zhu et al. 1997) via the scipy.optimize.minimize package.

Garnett et al. (2010) and Roberts et al. (2013) assumed that our function of interest is well behaved, except for a drastic change, or changepoint, at $c \in \{t - l + 1, t - l + 2, \ldots, t - 1\}$, after which all observations before c are completely uninformative about the observations after this point. It is important to note that the LBW l for this approach needs to be prespecified, and it is assumed that it contains a single changepoint. Each of the two regions is described by different covariance functions $k_{\xi 1}$, $k_{\xi 2}$, in our case Matérn 3/2 kernels, which are parameterized by hyperparameters ξ_1 and ξ_2, respectively. The region-switching kernel is

$$k_{\xi_R}(x,x') = \begin{cases} k_{\xi_1}(x,x') & x, x' < c \\ k_{\xi_2}(x,x') & x, x' \geq c \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

with a full set of hyperparameters $\xi_R = \{\xi_1, \xi_2, c, \sigma_n\}$. Here, a changepoint can take multiple forms, with these cases being a drastic change in covariance, a sudden change in the input scale, or a sudden change in the output scale. In the context of financial time series, we can think of these cases as a change in correlation length, a change in mean-reversion length, or a change in volatility.

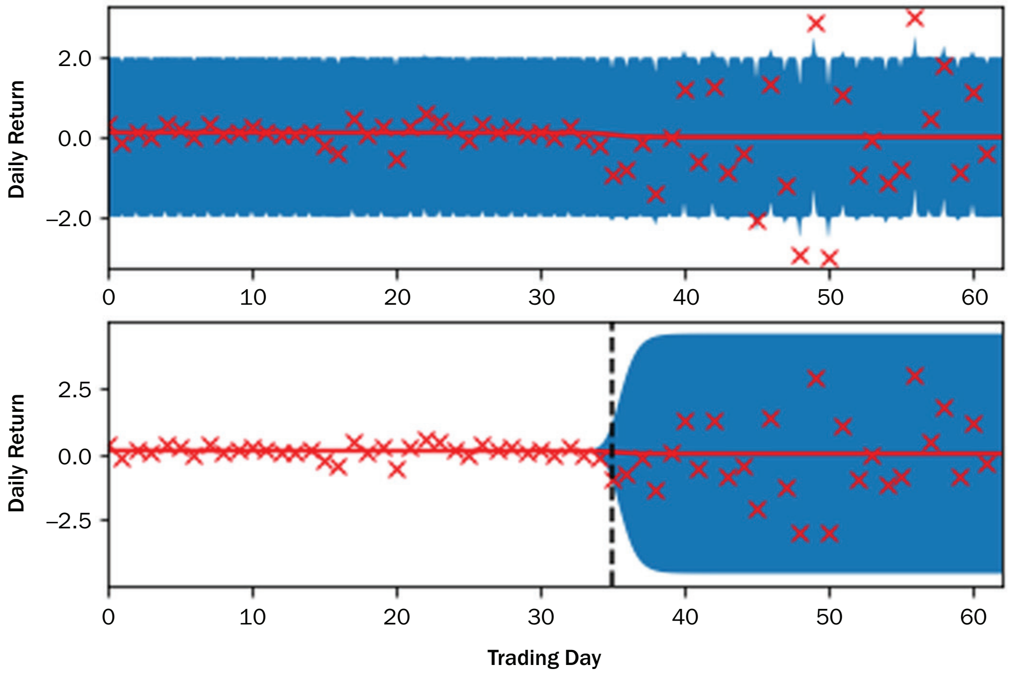
It is computationally inefficient to fit $2(l - 1)$ GPs, to minimize $\text{nlml}_{\xi R}$ as in Equation 7, owing to the introduction the discrete hyperparameter c. We instead borrow an idea from Lloyd et al. (2014) and approximate the abrupt change of covariance in Equation 8 using a sigmoid function $\sigma(x) = 1/(1 + e^{-s(x-c)})$, which has the properties $\sigma(x, x') = \sigma(x)\sigma(x')$ and $\bar{\sigma}(x, x')(1 - \sigma(x))(1 - \sigma(x'))$. Here, $c \in (t - l, t)$ is the changepoint location, and $s > 0$ is the steepness parameter. Our changepoint kernel is

$$k_{\xi_C}(x,x') = k_{\xi_1}(x,x')\sigma(x,x') + k_{\xi_2}(x,x')\bar{\sigma}(x,x') \tag{9}$$

with a full set of hyperparameters $\xi_C = \{\xi_1, \xi_2, c, s, \sigma_n\}$. We can compute $\text{nlml}_{\xi C}$ by optimizing the parameters a single GP, which is significantly more efficient than computing

EXHIBIT 1

Fitting the Matérn 3/2 (top) and Changepoint (bottom) Kernels to Daily Return Data



nlml_{ξ_R}, despite having additional hyperparameters. This new kernel has the added benefit of capturing more gradual transitions from one covariance function to another, owing to the addition of the steepness parameter s. We implement Equation 9 in GPflow via the gpflow.kernels.ChangePoints class, adding the constraint $c \in (t - l, t)$, which is not enforced by default.

To quantify the level of disequilibrium, we look at the reduction in negative log marginal likelihood achieved via the introduction of the changepoint kernel hyperparameters through comparison to nlml_{ξ_M}. If the introduction of additional hyperparameters leads to no reduction in negative log marginal likelihood, then the level of disequilibrium is low. Conversely, a large reduction indicates significant disequilibrium, or a stronger changepoint, because the data are better described by two covariance functions. Our changepoint score $\nu_t^{(i)} \in (0,1)$ and location $\gamma_t^{(i)} \in (0,1)$ are

$$\nu_t^{(i)} = 1 - \frac{1}{1 + e^{-(\text{nlmn}_{\xi_C} - \text{nlmn}_{\xi_M})}}, \quad \gamma_t^{(i)} = \frac{c - (t - l)}{l} \tag{10}$$

which are both normalized values, which helps to improve stability and performance of our LSTM module.

Exhibit 1 shows plots of daily returns for the S&P 500 composite ratio-adjusted continuous futures contract during the first quarter of 2020, in which returns have been standardized as per Equation 2. The top plot fits a GP using the Matérn 3/2 kernel, and the bottom uses the changepoint kernel specified in Equation 9. The shaded blue region covers ±2 standard deviations from the mean, and we can see that the top plot is dominated by the white noise term $\sigma_n \approx 1$. The black dotted line indicates the location of the changepoint hyperparameter c after minimizing negative log marginal likelihood, which aligns with the COVID-19 market crash. The negative log marginal likelihood is reduced from 88.0 to 47.9, which corresponds to $\nu_t^{(i)} \approx 1$.

MOMENTUM STRATEGIES REVIEW

Classical Strategies

In this article, we focus on univariate time-series approaches (Moskowitz, Ooi, and Pedersen 2012), as opposed to cross-sectional (Jegadeesh and Titman 1993) strategies, which trade assets against each other and select a portfolio based on relative ranking. Volatility scaling (Kim, Tse, and Wald 2016; Harvey et al. 2018) has been proven to play a crucial role in the positive performance of TSMOM strategies, including deep learning strategies (Lim, Zohren, and Roberts 2019). We scale the returns of each asset by its volatility so that each asset has a similar contribution to the overall portfolio returns, ensuring that our strategy targets a consistent amount of risk. The consistency over time and across assets has the added benefit of allowing us to benchmark strategies. Targeting an annualized volatility σ_{tgt}, which we take to be 15% in this article, the realized return of our strategy from day t to $t + 1$ is

$$R_{t+1}^{\text{TSMOM}} = \frac{1}{N}\sum_{i=1}^{N} R_{t+1}^{(i)}, \quad R_{t+1}^{(i)} = X_t^{(i)} \frac{\sigma_{\text{tgt}}}{\sigma_t^{(i)}} r_{t+1}^{(i)} \tag{11}$$

where X_t is our position size, N the number of assets in our portfolio, and $\sigma_t^{(i)}$ the ex ante volatility estimate of the ith asset. We compute $\sigma_t^{(i)}$ using a 60-day exponentially weighted moving standard deviation.

The simplest trading strategy for which we benchmark performance is long only, for which we always select the maximum position $X_t^{(i)} = 1$. The original article on time-series momentum (Moskowitz, Ooi, and Pedersen 2012), which we will refer to as *Moskowitz*, selects a position as $X_t^{(i)} = \text{sgn}(r_{t-252,t})$, where we are using the volatility scaling framework and $r_{t-252,t}$ is annual return. In an attempt to react more quickly to momentum turning points, Garg et al. (2021) blended a slow signal based on annual returns and a fast signal based on monthly returns to give an intermediate strategy:

$$X_t = (1-w)\text{sgn}(r_{t-252,t}) + w\,\text{sgn}(r_{t-21,t}) \tag{12}$$

We control the relative contribution of the fast and slow signal via $w \in [0, 1]$, with the case $w = 0$ corresponding to the Moskowitz strategy. We additionally use moving average convergence/divergence (MACD) (Baz et al. 2015) as a benchmark; for details on the implementation, we invite the reader to see Lim, Zohren, and Roberts (2019).

Deep Learning

We adopt a number of key choices that lead to the improved performance of DMNs.

LSTM architecture. Of the deep-learning architectures assessed by Lim, Zohren, and Roberts (2019), the LSTM (Hochreiter and Schmidhuber 1997) architecture yields the best results. LSTM is a special kind of recurrent neural network (RNN) (Goodfellow, Bengio, and Courville 2016), initially proposed to address the vanishing and exploding gradient problem (Bengio, Simard, and Frasconi 1994). An RNN takes an input sequence and, through the use of a looping mechanism in which information can flow from one step to another, can be used to transform this into an output sequence while taking into account contextual information in a flexible way. An LSTM operates with cells, which store both short-term memory and long-term memory, using gating mechanisms to summarize and filter information. Internal memory states are sequentially updated with new observations at each step. The resulting model has fewer trainable parameters, is able to learn representations of long-term relationships, and typically achieves better generalization results.

Trading signal and position sizing. Trading signals are learned directly by DMNs, removing the need to manually specify both the trend estimator and maps this into a position. The output of the LSTM is followed by a time-distributed, fully connected layer with a activation function tanh(·), which is a squashing function that directly outputs positions $X_t^{(i)} \in (-1, 1)$. The advantage of this approach is that we learn trading rules and position sizing directly from the data. Once our hyperparameters θ have been trained via backpropagation (LeCun et al. 2012), our LSTM architecture $g(\cdot;\theta)$ takes input features $\mathbf{u}_{T-\tau+1:T}^{(i)}$ for all time steps in the LSTM looking back from time T with τ steps and directly outputs a sequence of positions:

$$\mathbf{X}_{T-\tau+1:T}^{(i)} = g(\mathbf{u}_{T-\tau+1:T}^{(i)};\theta) \tag{13}$$

In an online prediction setting, only the final position in the sequence $X_T^{(i)}$ is of relevance to our strategy.

Loss function. It has been observed (Potters and Bouchaud 2016) that correctly predicting the direction of a stock move does not translate directly into a positive strategy return, because the driving moves can often be large but infrequent. Furthermore, we want to account for trade-offs between risk and reward; hence, we explicitly optimize networks for risk-adjusted performance metrics. One such metric used by DMNs is the Sharpe ratio (Sharpe 1994), which calculates the return per unit of volatility. Our Sharpe loss function is

$$\mathcal{L}_{sharpe}(\theta) = -\frac{\sqrt{252}\,\mathbb{E}_{\Omega}[R_t^{(i)}]}{\sqrt{Var_{\Omega}[R_t^{(i)}]}}, \tag{14}$$

where Ω is the set of all asset–time pairs $\{(i, t) | i \in \{1, 2, \ldots, N\}, t \in \{T - \tau + 1, \ldots, T\}\}$. Automatic differentiation is used to compute gradients for backpropagation (Goodfellow, Bengio, and Courville 2016), which explicitly optimizes networks for our chosen performance metric.

Model inputs. For each time step, our model can benefit from inputting signals from various time scales. We normalize returns to be $r_{t-t',t}^{(i)}/\sigma_t^{(i)}\sqrt{t'}$, given a time offset of t' days. We use offsets $t' \in \{1, 21, 63, 126, 256\}$, corresponding to daily, monthly, quarterly, biannual, and annual returns. We also encode additional information by inputting MACD indicators (Baz et al. 2015). MACD is a volatility-normalized moving-average convergence–divergence signal, defining the relationship between a short and long signal. For implementation details, please refer to Lim, Zohren, and Roberts (2019). We use pairs in {(8, 24), (16, 28), (32, 96)}. We can think of these indicators as performing a function similar to a convolutional layer.

TRADING STRATEGY

Strategy Definition

Because we are using a data-driven approach, we split our training data as a first step, setting aside the first 90% for training and the last 10% for validation for each asset. We calibrate our model using the training data by optimizing on the Sharpe loss function (Equation 14) via minibatch stochastic gradient descent (SGD), using the Adam (Kingma and Ba 2015) optimizer. We observe validation loss after each epoch, which is a full pass of the data, to determine convergence. We also use the validation set for the outer optimization loop, in which we tune our model hyperparameters. The hyperparameter optimization process is detailed in Appendix B.

It is necessary to precompute the CPD location $\gamma_t^{(i)}$ and severity $\nu_t^{(i)}$ parameters as detailed by Equation 10. We do this for each time–asset pair in our training and validation set. It is necessary to do this for a chosen $l \in \{10, 21, 63, 126, 252\}$, corresponding to two weeks, a month, a quarter, half a year, and a full year. We selected these LBW sizes to correspond to input return timescales, with the exception of the 10-day LBW, which was selected to be as close to daily return data as reasonably possible. We reinitialize our Matérn 3/2 kernel for each time step, with all hyperparameters set to 1. This approach was found to be more stable than borrowing parameters from the previous time step. For our changepoint kernel, we initialize the hyperparameters as $c = t - \frac{l}{2}$ and $s = 1$. All other parameters are initialized as the equivalent parameter from fitting the Matérn 3/2 kernel, initializing $k_{\xi 1}$ and $k_{\xi 2}$ with the same values. In the rare case this process fails, we try again by reinitializing all changepoint kernel parameters to 1, with the exception of setting $c = t - \frac{l}{2}$. In the event the module still fails for a given time step, we fill the outputs $\nu_t^{(i)}$ and $\gamma_t^{(i)}$ using the outputs from the previous time step, noting that we need to increment the changepoint location by an additional step.

For each LSTM input, we pass in the normalized returns from the different time scales, our MACD indicators, and CPD severity and location for a chosen l. We can either fix l for our strategy or introduce it as a structural hyperparameter, which is tuned by the outer optimization loop. By doing this, we have information exchange from our CPD module all the way through to our Sharpe ratio loss function and traded positions. Once our model has been fully trained, we can run it online by computing the CPD module for the most recent data points and then using our LSTM module to select positions to hold for the next day for each asset.

Experiments via Backtesting

For all of our experiments, we used a portfolio of 50 liquid, continuous futures contracts over the period 1990–2020. The combination of commodities, equities, fixed income, and FX futures was selected to make up a well-balanced portfolio. The data were extracted from the Pinnacle Data Corp. CLC database (Pinnacle Data Corp. 2021), and the selected futures contracts are listed in Appendix A. All of the selected assets have less than 10% of data missing.

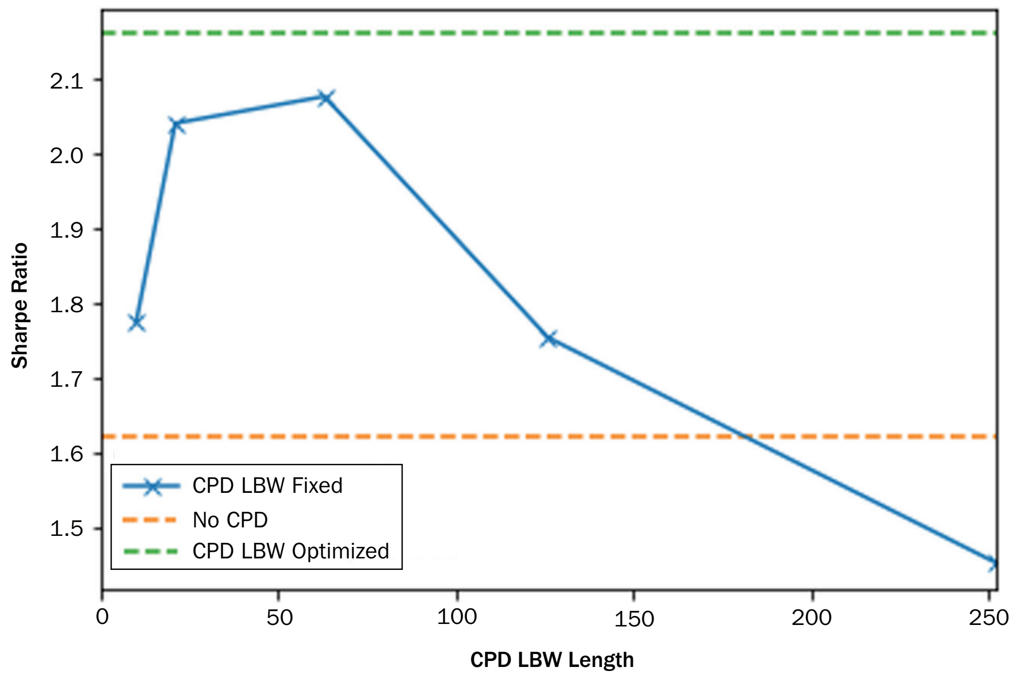
To back test our model, we use an expanding window approach, in which we start by using 1990–1995 for training/validation and then test out of sample on the period 1995–2000. With each successive iteration, we expand the training/validation window by an additional five years, perform the hyperparameter optimization again, and test on the subsequent five-year period. Data were not available from 1990 for every asset, and we only use an asset if there is enough data available in the validation set for at least one LSTM sequence. All of our results are recorded as an average of the test windows. We test our LSTM with the CPD strategy using an LBW $l \in \{10, 21, 63, 126, 252\}$ and then with the optimized l for each window, based on validation loss.

We benchmark our strategy against those we have discussed, in which we choose $w \in \{0, 0.5, 1\}$ for the intermediate strategy. We also compare our strategy to a DMN that does not have the CPD module. To maintain consistency with previous work by Lim, Zohren, and Roberts (2019), we benchmark strategy

1. **profitability** through annualized returns and percentage of positive captured returns;

EXHIBIT 2

Risk-Adjusted Strategy Returns for Different Changepoint LBW Lengths



2. **risk** through annualized volatility, annualized downside deviation, and maximum drawdown (MDD); and
3. **risk-adjusted performance** through annualized Sharpe, Sortino, and Calmar ratios.

We provide results for both the raw signal output and then with an additional layer of volatility rescaling to the target of 15%, for ease of comparison between strategies. It should be noted that this article selects a more realistic 50-asset portfolio instead of the full 88 assets previously selected by Lim, Zohren, and Roberts (2019). We focus on the raw predictive power of the model and do not account for transaction costs at this stage; however, this is a simple adjustment and can easily be incorporated into the loss function. We have included some details and analysis of transaction costs in Appendix C. For further information on the implementation and the effects of transaction costs, please refer to Lim, Zohren, and Roberts (2019).

RESULTS AND DISCUSSION

Our aggregated out-of-sample prediction results, averaged across all five-year windows from 1995–2020, are recorded in Exhibit 3 and again in Exhibit 4 using volatility rescaling. We plot the effect of CPD LBW size on the average Sharpe ratio in Exhibit 2 and demonstrate how optimizing on this as a hyperparameter can improve overall performance. Impressively, due to our GP framework for CPD, we are able to achieve superior results with limited data and hence very small LBWs. There is a notable performance boost from only a two-week LBW, and performance almost maxes out after only one month, with an LBW of one quarter leading to the highest Sharpe ratio. As we approach an LBW of one year, we lose the benefit of the CPD module because it places too much emphasis on larger changepoints that are further

EXHIBIT 3

Strategy Performance Benchmark—Raw Signal Output

	Returns	Vol.	Sharpe	Downside Deviation	Sortino	MDD	Calmar	% of +ve Returns	Ave. P / Ave. L
Reference									
Long Only	2.30%	5.22%	0.44	3.59%	0.64	3.12%	0.79	52.45%	0.975
MACD	2.65%	3.58%	0.77	2.57%	1.09	2.56%	0.95	53.34%	1.002
TSMOM									
$w = 0$	4.41%	4.80%	0.94	3.44%	1.32	3.22%	1.35	54.28%	0.990
$w = 0.5$	3.29%	3.78%	0.89	2.80%	1.23	2.70%	1.16	53.88%	0.998
$w = 1$	2.17%	4.71%	0.48	3.29%	0.68	3.24%	0.67	51.48%	1.026
LSTM	3.53%	2.52%	1.62	1.71%	2.46	1.72%	2.79	55.23%	1.075
LSTM w/CPD									
10-day LBW	3.04%	1.57%	1.77	**1.07%**	2.74	1.09%	2.78	55.50%	1.096
21-day LBW	**3.68%**	1.81%	2.04	1.21%	3.07	1.08%	**3.75**	**56.43%**	1.095
63-day LBW	3.51%	**1.72%**	2.08	1.10%	3.27	**1.06%**	3.58	55.61%	1.140
126-day LBW	3.37%	2.28%	1.75	1.59%	2.66	1.52%	2.88	54.95%	1.117
252-day LBW	2.81%	2.24%	1.45	1.57%	2.19	1.54%	2.32	54.00%	1.101
LBW Optimized	3.64%	1.73%	**2.16**	1.17%	**3.33**	1.14%	3.50	56.22%	**1.133**

EXHIBIT 4

Strategy Performance Benchmark—Rescaled to Target Volatility of 15%

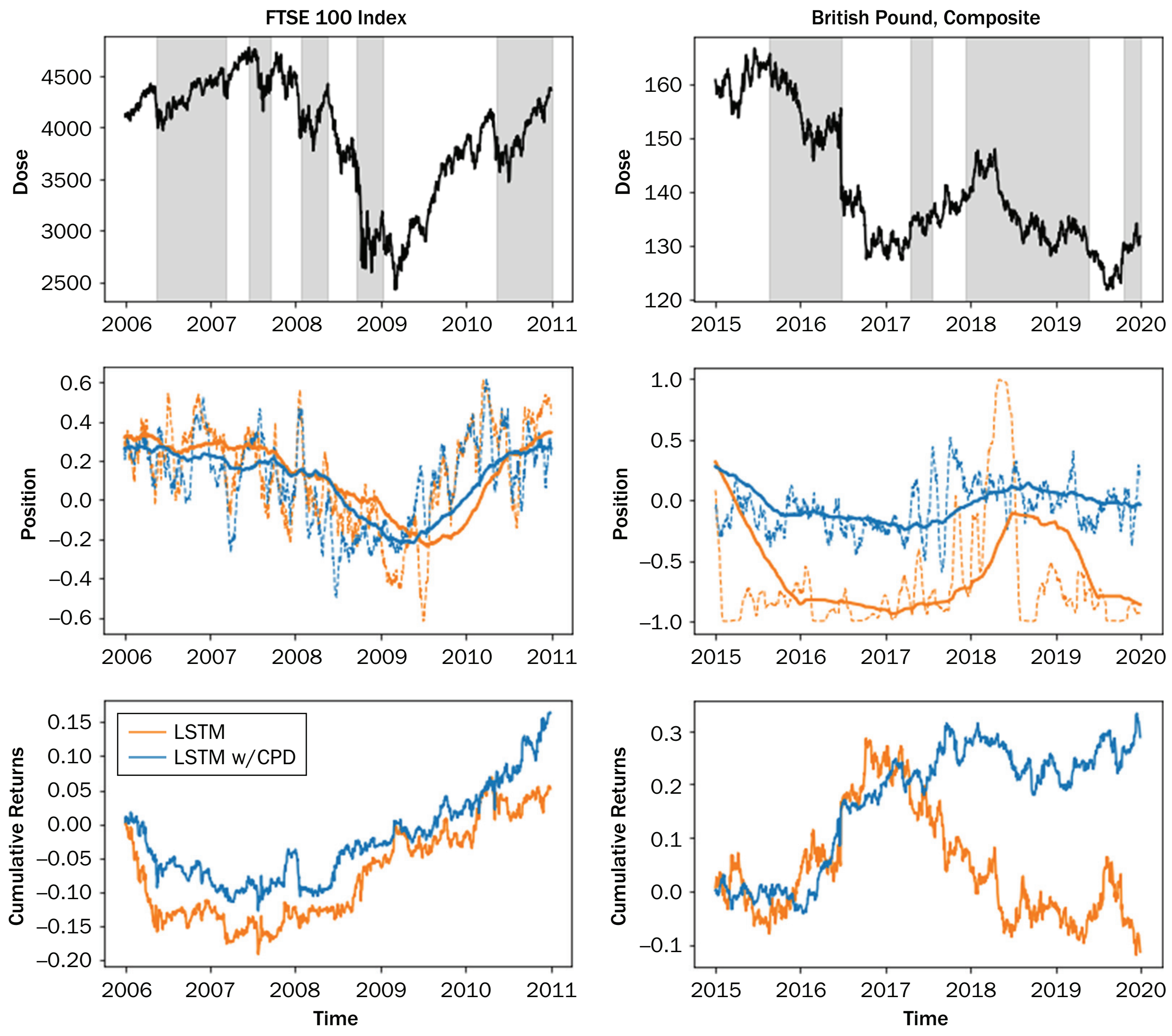
	Returns	Vol.	Sharpe	Downside Deviation	Sortino	MDD	Calmar	% of +ve Returns	Ave. P / Ave. L
Reference									
Long Only	6.62%	15.00%	0.44	10.32%	0.64	8.96%	0.79	52.45%	0.975
MACD	11.08%	15.00%	0.77	10.74%	1.09	10.72%	0.95	53.34%	1.002
TSMOM									
$w = 0$	13.79%	15.00%	0.94	10.74%	1.32	10.05%	1.35	54.28%	0.990
$w = 0.5$	13.06%	15.00%	0.89	11.10%	1.23	10.72%	1.16	53.88%	0.998
$w = 1$	6.89%	15.00%	0.48	10.46%	0.68	10.32%	0.67	51.48%	1.026
LSTM	21.03%	15.00%	1.62	10.15%	2.46	10.24%	2.79	55.23%	1.075
LSTM w/CPD									
10-day LBW	29.01%	15.00%	1.77	10.18%	2.74	10.39%	2.78	55.50%	1.096
21-day LBW	30.57%	15.00%	2.04	10.06%	3.07	**9.01%**	**3.75**	**56.43%**	1.095
63-day LBW	30.71%	15.00%	2.08	**9.65%**	3.27	9.22%	3.58	55.61%	1.140
126-day LBW	22.16%	15.00%	1.75	10.44%	2.66	9.99%	2.88	54.95%	1.117
252-day LBW	18.82%	15.00%	1.45	10.54%	2.19	10.32%	2.32	54.00%	1.101
LBW Optimized	**31.52%**	15.00%	**2.16**	10.10%	**3.33**	9.88%	3.50	56.22%	**1.133**

in the past. We also note that the CPD computation becomes more intensive for $l \in \{126, 252\}$. If we introduce LBW as a hyperparameter to be reevaluated as the training window continues to expand, we observe an additional 4% increase in the Sharpe ratio, leading to a total increase of 33% over the LSTM baseline.

Another idea involved passing in outputs from multiple CPD modules with different LBWs in parallel as inputs to the LSTM. This was not found to improve the model and

EXHIBIT 5

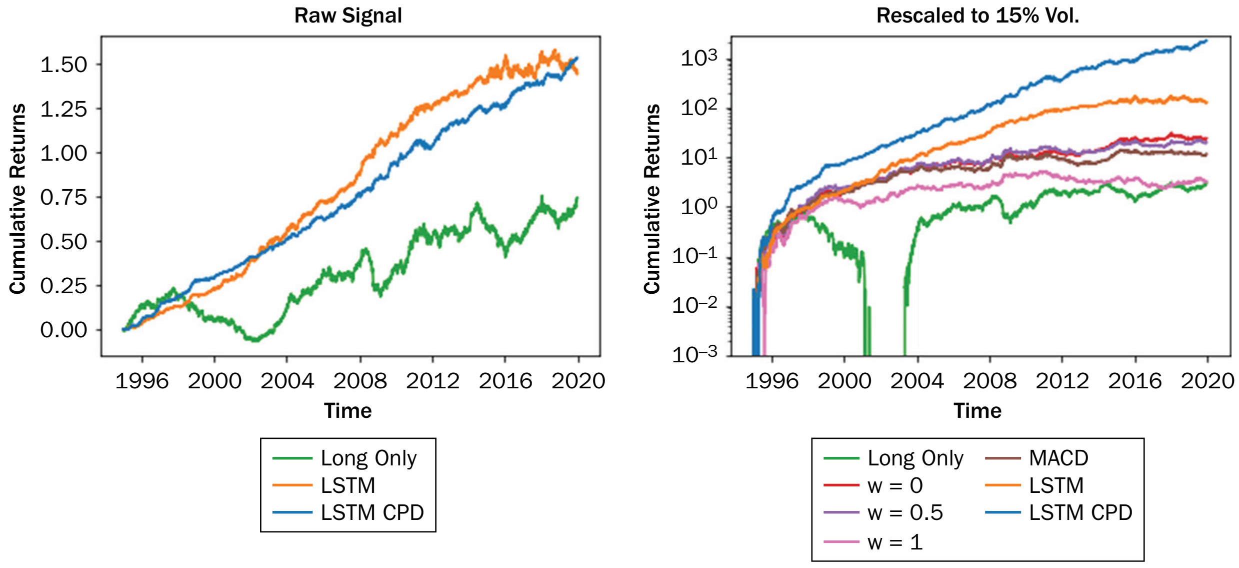
Slow Momentum and Fast Reversion Strategies Happening Simultaneously



actually resulted in degraded performance. Multiple LBWs could be useful if using a more complex deep learning architecture than LSTM.

In Exhibit 5, we observe slow momentum and fast reversion strategies happening simultaneously. By introducing CPD, we are able to achieve superior returns because we are better able to learn the timing of these strategies and when to place more emphasis on one of them, using a data-driven approach. These plots examine the positions our DMN takes for single assets during periods of regime change, providing a comparison of a DMN with and without the CPD module. The top plots track the daily closing price, with the alternating white and gray regions indicating regimes separated by significant changepoints. CPD is performed online with a 63-day LBW, with the changepoint severity $\nu_t^{(i)} \geq 0.9995$ on the left plot and $\nu_t^{(i)} \geq 0.995$ on the right plot indicating a changepoint. Each case uses a 63-day burn-in time before we can classify a subsequent changepoint. The middle plots compare the moving averages of position size taken for over a long time scale of one year, indicated by the solid lines, and a shorter timescale of one month, indicated by the dashed line.

EXHIBIT 6
Benchmarking Strategy Performance



The bottom plots indicate cumulative returns for each strategy. The plots on the left look at the FTSE 100 Index during the lead up to the 2008 final crash and its aftermath. With the addition of CPD, our strategy is able to exploit persisting trends with better timing. It is quicker to react to the first dip in 2008, taking short positions to exploit the bear market with a slow momentum strategy, and is similarly able to react to adapt to the bull market established in 2009 by more quickly moving to a long strategy. Both approaches exhibit a fast reverting strategy; however, after the addition of CPD, the strategy is slightly less aggressive with positions taken in response to localized changes. The plots on the right look at the British pound exchange rate in the lead up to the Brexit vote in 2016 and its aftermath. Here, the bull and bear regimes are both less defined, and there is a higher level of nonstationarity. With the addition of the CPD module, our model takes a much more conservative slow momentum strategy and instead opts to focus more on achieving positive returns via a fast mean-reverting strategy.

Our results demonstrate that, via the introduction of the CPD module, we outperform the standard DMN in all performance metrics. Our model correctly classifies the direction of the return more often and has a higher average profit-to-loss ratio. We can see that the CPD module helps to reduce risk, thus reducing volatility, downside deviation, and MDD while still achieving slightly higher raw returns. This translates to an improvement in risk-adjusted performance, improving the Sortino ratio by 35% and the Calmar ratio by 25%. These metrics suggest that the CPD module makes our model more robust to market crashes. We observe an improvement in the Sharpe ratio, our target metric, of 33%, which translates to an improvement of 130% in comparison to the best-performing TSMOM strategy.

We plot the raw and rescaled signals to benchmark strategies in Exhibit 6. The plot on the left, of raw signal output, demonstrates that via the introduction of the CPD module, we are able to reduce the strategy volatility, especially during the market nonstationarity of more recent years. With the exception of long only, we omit the reference strategies in this plot to avoid clutter. The plot on the right, of signal with rescaled volatility, demonstrates that our strategy outperforms all benchmarks with risk-adjusted performance. We show intermediate strategy output for $w \in \{0, 0.5, 1\}$.

EXHIBIT 7

Hyperparameter Search Range

Hyperparameters	Random Search Grid
Dropout Rate	0.1, 0.2, 0.3, 0.4, 0.5
Hidden Layer Size	5, 10, 20, 40, 80, 160
Minibatch Size	64, 128, 256
Learning Rate	10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}
Max Gradient Norm	10^{-2}, 10^{0}, 10^{2}
*CPD LBW Length	10, 21, 63, 126, 252

NOTES: *CPD LBW length can be either a hyperparameter or fixed.

We can see the difficulties of trying to address regime change with handcrafted techniques such as the intermediate $w = 0.5$, which in our experiments actually fails to outperform the $w = 0$ Moskowitz strategy on all risk-adjusted performance ratios.

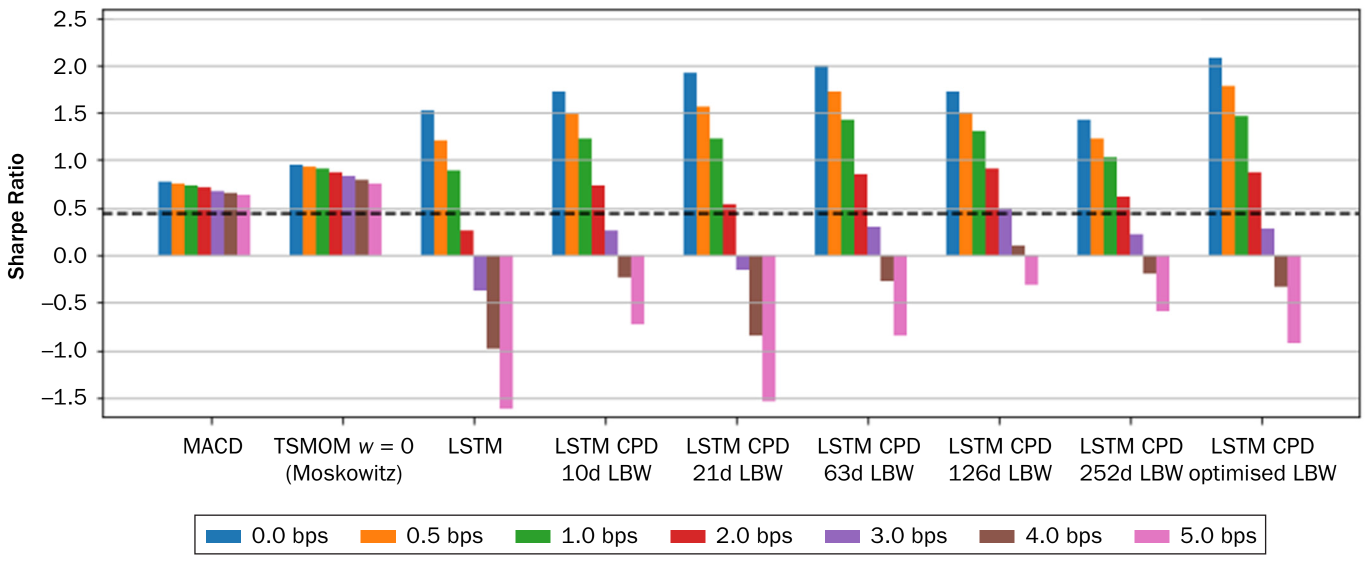
We note that up until about 2003, when the uptake of electronic trading was becoming much more widespread, the traditional TSMOM and MACD strategies are comparable to the results achieved via the LSTM DMN architecture. At this point, the LSTM starts to significantly outperform these traditional strategies until more recent years, when we see volatility increase and performance, especially risk-adjusted performance, drop significantly. This drop in performance can be largely attributed to increased market nonstationarity. Impressively, with the addition of the CPD module, our DMN pipeline continues to perform well even during the market nonstationarity of the 2015–2020 period. Using five repeated trials of the entire experiment, with and without CPD, the average improvement for the Sharpe ratio in this period is 70%, for LBW $l = 21$.

CONCLUSIONS

We have demonstrated that the introduction of an online CPD module is a simple, yet effective, way to significantly improve model performance, specifically DMNs. Our model is able to blend different strategies at different timescales, learning to do so in a data-driven manner directly based on our desired risk-adjusted performance metric. In periods of stability, our model is able to achieve superior returns by focusing on slow momentum while exploiting but not overreacting to local mean reversion. The impressive performance increase in periods of nonstationarity, such as recent years,

EXHIBIT 8

Looking at the Impact of Increasing Average Transaction Cost *C* from 0 to 5 bps and Comparing with Long-Only Benchmark (dashed line)



can be attributed to the fact that we (1) can effectively incorporate CPD online with a very short LBW because we do so using GP and (2) pass changepoint score $\nu_t^{(i)}$ from our CPD module to the DMN, helping our model learn how to respond to varying degrees of disequilibrium. As a result, we enhance performance in such conditions in which we observe a more conservative slow momentum strategy with a focus on fast mean reversion.

Future work includes incorporating a CPD module into other deep learning architectures or performing CPD on a model representation as opposed to model inputs. The work in this article has natural parallels to the field of continual learning, which is a paradigm whereby an agent sequentially learns new tasks. Another direction of work will involve using continual learning for momentum trading, in which CPD is used to determine task boundaries.

APPENDIX A

EXHIBIT A1

Dataset Details

Identifier_Description_Backtest From		
Commodities		
CC	Cocoa	1995
DA	Milk III, composite	2000
GI	Goldman Saks C. I.	1995
JO	Orange Juice	1995
KC	Coffee	1995
KW	Wheat, KC	1995
LB	Lumber	1995
NR	Rough Rice	1995
SB	Sugar #11	1995
ZA	Palladium, Electronic	1995
ZC	Corn, Electronic	1995
ZF	Feeder Cattle, Electronic	1995
ZG	Gold, Electronic	1995
ZH	Heating Oil, Electronic	1995
ZI	Silver, Electronic	1995
ZK	Copper, Electronic	1995
ZL	Soybean Oil, Electronic	1995
ZN	Natural Gas, Electronic	1995
ZO	Oats, Electronic	1995
ZP	Platinum, Electronic	1995
ZR	Rough Rice, Electronic	1995
ZT	Live Cattle, Electronic	1995
ZU	Crude Oil, Electronic	1995
ZW	Wheat, Electronic	1995
ZZ	Lean Hogs, Electronic	1995

Identifier_Description_Backtest From		
Equities		
CA	CAC40 Index	2000
EN	NASDAQ, Mini	2005
ER	Russell 2000, Mini	2005
ES	S&P 500, Mini	2000
LX	FTSE 100 Index	1995
MD	S&P 400 (Mini Electronic)	1995
SC	S&P 500, Composite	2000
SP	S&P 500, Day Session	1995
XU	Dow Jones EUROSTOXX50	2005
XX	Dow Jones STOXX 50	2005
YM	Mini Dow Jones ($5.00)	2005
Fixed Income		
DT	Euro Bond (Bund)	1995
FB	T-Note, 5yr composite	1995
TY	T-Note, 10yr composite	1995
UB	Euro BOBL	2005
US	T-Bonds, Composite	1995
FX		
AN	Australian $$, Composite	1995
BN	British Pound, Composite	2000
CN	Canadian $$, Composite	1995
DX	US Dollar Index	1995
FN	Euro, Composite	1995
JN	Japanese Yen, Composite	1995
MP	Mexican Peso	2000
NK	NIKKEI Index	1995
SN	Swiss Franc, Composite	1995

APPENDIX B

EXPERIMENT DETAILS

We split our data into training and validation datasets using a 90%/10% split. We winsorize our data by limiting them to be within five times their exponentially weighted moving (EWM) standard deviations from their EWM average, using a 252-day half-life. We calibrate our model using the training data by optimizing on the Sharpe loss function via minibatch SGD, using the Adam optimizer. We limit our training to 300 epochs, with an early stopping patience of 25 epochs, meaning training is terminated if there is no decrease in validation loss during this time period. The model is implemented via the Keras API in TensorFlow. Our LSTM sequence length was set to 63 for all experiments. For training and validation, in an attempt to prevent overfitting, we split our data into non-overlapping sequences, rather than using a sliding window approach. A stateless LSTM is used, meaning the last state from the previous batch is not used as the initial state for the subsequent batch. Keeping the order of each individual sequence intact, we shuffle the order in which each sequence appears in an epoch. We employ dropout regularization (Srivastava et al. 2014) as another technique to avoid overfitting, applying it to LSTM inputs and outputs.

We tune our hyperparameters, with options listed in Exhibit 7, using an outer optimization loop. We achieve this via 50 iterations of random grid search to identify the optimal model. We perform the full experiment for each choice of CPD LBW length and then use the model that achieved the lowest validation loss for the optimized CPD model.

APPENDIX C

TRANSACTION COSTS

In Exhibit 8, we demonstrate the impact of transaction costs on our raw signal, in which we increase the average transaction cost from 0 to 5 bps. The black dotted line indicates the long-only reference. Our strategy outperforms classical strategies for transaction costs of up to 2 bps, at which point it rapidly deteriorates owing to the fast reverting component. We note that a larger CPD LBW size becomes favorable as we increase *C*. We suspect this is because the model focuses on larger long-term changepoints and favors slow momentum over fast reversion. For larger average transaction costs greater than 1 bp, we suggest incorporating turnover-adjusted returns into the loss function (Equation 14). This adjustment is detailed by Lim, Zohren, and Roberts (2019), who demonstrated that it works well when transaction costs are high. Assuming an average transaction cost of *C*, we calculate turnover adjusted returns as

$$\bar{R}_{t+1}^{(i)} = R_{t+1}^{(i)} + -C\sigma_{\text{tgt}} \left| \frac{X_t^{(i)}}{\sigma_t^{(i)}} - \frac{X_{t-1}^{(i)}}{\sigma_{t-1}^{(i)}} \right| \tag{C1}$$

ACKNOWLEDGMENT

We would like to thank the Oxford-Man Institute of Quantitative Finance for financial and computing support.

REFERENCES

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

Adams, R. P., and D. J. C. MacKay. 2007. “Bayesian Online Changepoint Detection.” *arXiv* 0710.3742.

Baltas, N. 2019. “The Impact of Crowding in Alternative Risk Premia Investing.” *Financial Analysts Journal* 75 (3): 89–104.

Baltas, N., and R. Kosowski. 2017. “Demystifying Time-Series Momentum Strategies: Volatility Estimators, Trading Rules and Pairwise Correlations.” SSRN, https://ssrn.com/abstract=2140091.

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

Baz, J., N. Granger, C. R. Harvey, N. Le Roux, and S. Rattray. 2015. “Dissecting Investment Strategies in the Cross Section and Time Series.” SSRN, https://ssrn.com/abstract=2695101.

Bengio, Y., P. Simard, and P. Frasconi. 1994. “Learning Long-Term Dependencies with Gradient Descent Is Difficult.” *IEEE Transactions on Neural Networks* 5 (2): 157–166.

Brahim-Belhouari, S., and A. Bermak. 2004. “Gaussian Process for Nonstationary Time Series Prediction.” *Computational Statistics & Data Analysis* 47 (4): 705–712.

Bruder, B., T. L. Dao, J. C. Richard, and T. Roncalli. 2013. “Trend Filtering Methods for Momentum Strategies.” SSRN, https://ssrn.com/abstract=2289097.

De Bondt, W. F. M., and R. Thaler. 1985. “Does the Stock Market Overreact?” *The Journal of Finance* 40 (3): 793–805.

Garg, A., C. L. Goulding, C. R. Harvey, and M. Mazzoleni. 2021. “Momentum Turning Points.” https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3489539.

Garnett, R., M. A. Osborne, S. Reece, A. Rogers, and S. J. Roberts. 2010. “Sequential Bayesian Prediction in the Presence of Changepoints and Faults.” *The Computer Journal* 53 (9): 1430–1446.

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

Gu, S., B. T. Kelly, and D. Xiu. 2017. “Empirical Asset Pricing via Machine Learning.” Research paper no. 18-04, Chicago Booth, https://ssrn.com/abstract=3159577.

Harvey, C. R., E. Hoyle, R. Korgaonkar, S. Rattray, M. Sargaison, O. Van Hemert. 2018. “The Impact of Volatility Targeting.” SSRN, https://ssrn.com/abstract=3175538.

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

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

Jegadeesh, N. 1991. “Seasonality in Stock Price Mean Reversion: Evidence from the US and the UK.” *The Journal of Finance* 46 (4): 1427–1444.

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

Kim, S. 2019. “Enhancing the Momentum Strategy through Deep Regression.” *Quantitative Finance* 0 (0): 1–13.

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

Kingma, D., and J. Ba. 2015. “Adam: A Method for Stochastic Optimization.” International Conference on Learning Representations.

LeCun, Y. A., L. Bottou, G. B. Orr, and K. R. Muller. 2012. “Efficient BackProp.” In *Neural Networks: Tricks of the Trade*, pp. 9–48. Berlin: Springer.

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

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

Lim, B., and S. Zohren. 2020. “Time Series Forecasting with Deep Learning: A Survey.” *arXix* 2004.13408.

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.

Liu, B., I. Kiskin, and S. Roberts. 2020. “An Overview of Gaussian Process Regression for Volatility Forecasting.” 2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC), pp. 681–686. IEEE.

Lloyd, J. R., D. Duvenaud, R. Grosse, J. B. Tenenbaum, and Z. Ghahramani. 2014. “Automatic Construction and Natural-Language Description of Nonparametric Regression Models.” *Proceedings of the AAAI Conference on Artificial Intelligence* 28 (1).

Matthews, A. G. G., M. van der Wilk, T. Nickson, K. Fujii, A. Boukouvalas, P. Leon-Villagra, Z. Ghahramani, and J. Hensman. 2017. “GPflow: A Gaussian Process Library Using TensorFlow.” *Journal of Machine Learning Research* 18 (40): 1–6.

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

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

Pinnacle Data Corp. 2021. *Pinnacle Data Corp. CLC Database.* https://pinnacledata2.com/clc.html.

Poh, D., B. Lim, S. Zohren, and S. Roberts. 2021. “Building Cross-Sectional Systematic Strategies by Learning to Rank.” *The Journal of Financial Data Science* 3 (2): 70–86.

Poterba, J. M., and L. H. Summers. 1988. “Mean Reversion in Stock Prices: Evidence and Implications.” *Journal of Financial Economics* 22 (1): 27–59.

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

Rasmussen, C. E. 2003. “Gaussian Processes in Machine Learning.” In: *Summer School on Machine Learning*, pp. 63–71. New York: Springer.

Rizvi, S. A. A. “Analysis of Financial Time Series Using Non-Parametric Bayesian Techniques.” PhD Thesis, University of Oxford, 2018.

Roberts, S., M. Osborne, M. Ebden. S. Reece, N. Gibson, and S. Aigrain. 2013. “Gaussian Processes for Time-Series Modelling.” *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 371: 20110550.

Saatçi, Y., R. D. Turner, and C. E. Rasmussen. 2010. "Gaussian Process Change Point Models." *ICML*.

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

Sirignano, J., and R. Cont. 2018. "Universal Features of Price Formation in Financial Markets: Perspectives from Deep Learning." SSRN, https://ssrn.com/abstract=3141294.

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

Williams, C. K. I., and C. E. Rasmussen. 1996. "Gaussian Processes for Regression." NeurIPS Proceedings.

Zhang, Z., S. Zohren, and S. Roberts. 2019. "DeepLOB: Deep Convolutional Neural Networks for Limit Order Books." *IEEE Transactions on Signal Processing*.

Zhu, C., R. H. Byrd, P. Lu, and J. Nocedal. 1997. "Algorithm 778: L-BFGS-B: Fortran Subroutines for Large-Scale Bound-Constrained Optimization." *ACM Transactions on Mathematical Software (TOMS)* 23 (4): 550–560.