Deep learning enhanced quantitative trading strategies

Stefan Zohren

Oxford Man Institute of Quantitative Finance
University of Oxford

August 12, 2022

Special thanks to:

Associate Member

Bryan Lim

Student

Kieran Wood

Student

Daniel Poh

Visiting Student

Fernando Moreno-Pino

Student

Alvaro Arroyo Nuñez

Post-doctoral Researcher

Zihao Zhang

Forthcoming book

Deep Learning in Quantitative Trading

Series Name

DOI: 10.xxxx/xxxxxxxx (do not change)
First published online: MMM dd YYYY (do not change)

Zihao Zhang
University of Oxford

Stefan Zohren
University of Oxford

Outline

An Overview of Research at the Oxford-Man Institute

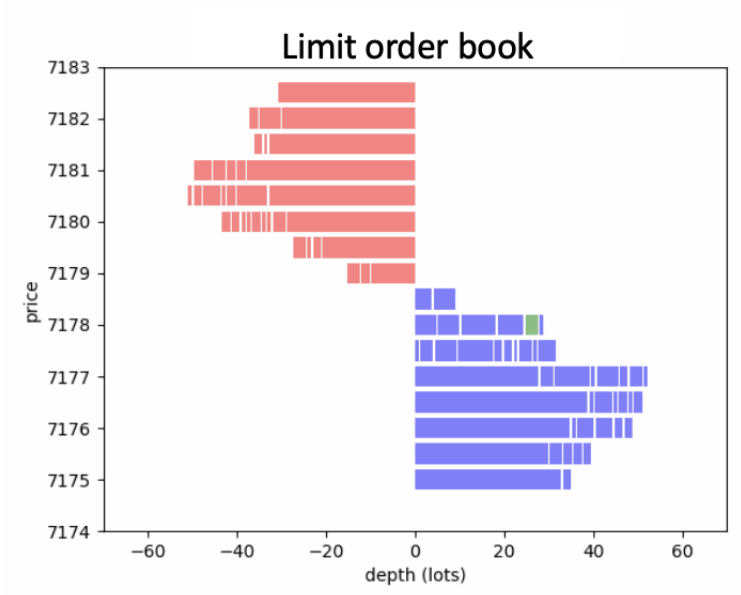
The Oxford-Man Institute of Quantitative Finance

- A unique collaboration between Oxford University and Man Group plc
  - Opened Summer 2007
  - Focus: Machine Learning in quantitative finance & alternative investments
  - Vision to become world leading research centre

- Man Research Laboratory
  - Embedded commercial research laboratory
  - Undertakes Man AHL's research
  - Opportunity for collaboration and academic publication

Research Topics: Modelling High-Frequency Data

- Institute intensely engaged in:
  - Modelling of limit order book data
  - Deep learning models for forecasting from HFT observations
  - Key contributions such as DeepLOB have been adopted across industry in market making and execution



Search **Bloomberg**

Technology

Man Group-Oxford Quants Say Their AI Can Predict Stock Moves

Journals & Magazines > IEEE Transactions on Signal P... > Volume: 67 Issue: 11

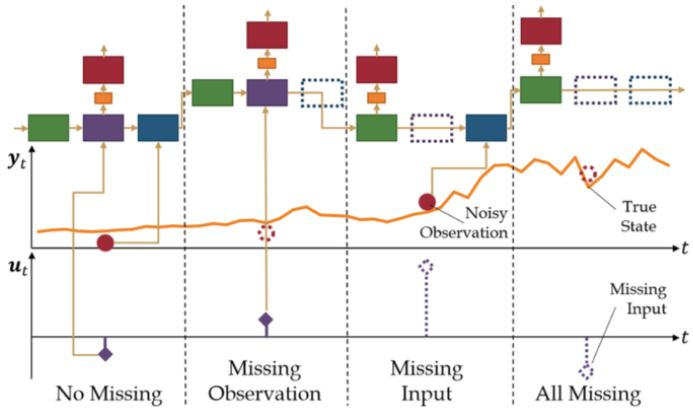
DeepLOB: Deep Convolutional Neural Networks for Limit Order

Publisher: IEEE | Cite This | PDF

Zihao Zhang ; Stefan Zohren ; Stephen Roberts All Authors

Research Topics: Deep Learning for Time Series

- We have done extensive research on deep learning for time-series modelling with applications in finance and beyond
- Members of the group were involved in one of the first transformer based architectures for time-series, the temporal fusion transformer

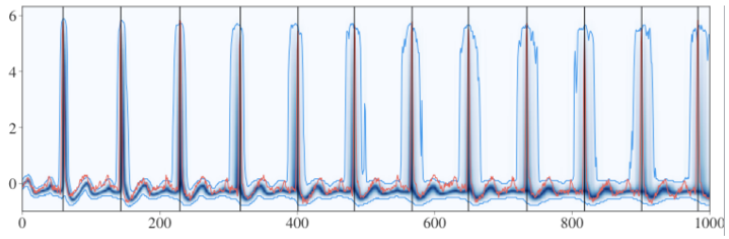


Review articles

Time-series forecasting with deep learning: a survey

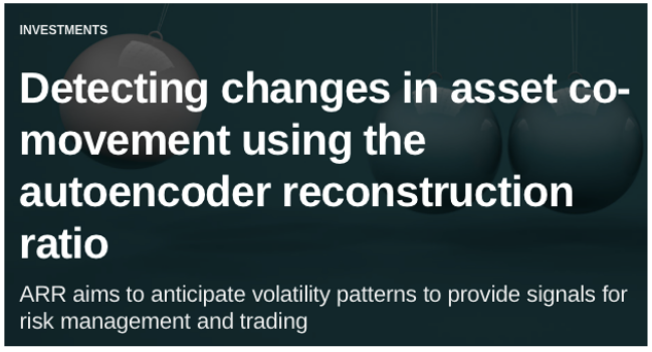
Bryan Lim ✉ and Stefan Zohren

Published: 15 February 2021 | https://doi.org/10.1098/rsta.2020.0209



Research Topics: Deep Learning for Quant Trading

- A major focus has been on deep learning for quant trading, including work on:
  - Portfolio optimisation
  - Reinforcement learning for trading
  - Autoencoder based risk/fragility measures
  - Enhancing traditional quant trading strategies (e.g. time-series momentum and cross-sectional momentum strategies)



Deep Learning for Portfolio Optimization

Zihao Zhang, Stefan Zohren and Stephen Roberts

The Journal of Financial Data Science Fall 2020, 2 (4) 8-20; DOI: https://doi.org/10.3905/jfds.2020.1.042

the journal of financial data science

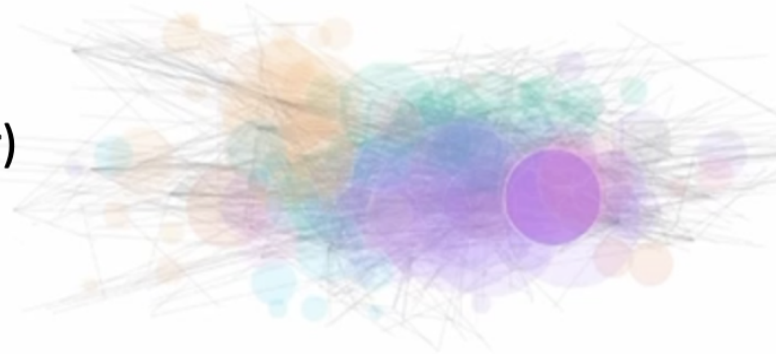
Building Cross-Sectional Systematic Strategies by Learning to Rank

Daniel Poh, Bryan Lim, Stefan Zohren and Stephen Roberts

The Journal of Financial Data Science Spring 2021, 3 (2) 70-86; DOI: https://doi.org/10.3905/jfds.2021.1.060

Research Topics: Networks and NLP in Finance

- Institute engages in research on:
  - Financial news networks
  - Sentiment analysis and propagation
  - Virality in Social Media (Reddit, Twitter)
  - Financial word embeddings
- Organised some key workshops in field



nature > scientific reports > articles > article

Article | Open Access | Published: 04 February 2021

Sentiment correlation in financial news networks and associated market movements

Xingchen Wan, Jie Yang, Slavi Marinov, Jan-Peter Calliess, Stefan Zohren & Xiaowen Dong

Scientific Reports **11**, Article number: 3062 (2021) | Cite this article

An Introduction to Deep Learning for Time-Series Forecasting

Basic Building Blocks

- Time series forecasting models predict future values of a target $y_{i,t}$ for a given entity i at time t.
- In the simplest case, one-step-ahead forecasting models take the form:

$$\hat{y}_{i,t+1} = f(y_{i,t-k:t}, \boldsymbol{x}_{i,t-k:t}, \boldsymbol{s}_i), \tag{1}$$

- Deep neural networks learn predictive relationships by using a series of non-linear layers to construct intermediate feature representations [1].
- In time series settings, this can be viewed as encoding relevant historical information into a latent variable $\boldsymbol{z}_t$, with the final forecast produced using $\boldsymbol{z}_t$ alone:

$$f(y_{t-k:t}, \boldsymbol{x}_{t-k:t}, \boldsymbol{s}) = g_{\text{dec}}(\boldsymbol{z}_t), \tag{2}$$

$$\boldsymbol{z}_t = g_{\text{enc}}(y_{t-k:t}, \boldsymbol{x}_{t-k:t}, \boldsymbol{s}), \tag{3}$$

Convolutional Neural Networks

- Convolutional neural networks (CNNs) extract local relationships that are invariant across spatial dimensions.
- To adapt CNNs to time series datasets, one can make use of convolutional filters designed to ensure only past information is used for forecasting.
- For an intermediate feature at hidden layer l, each causal convolutional filter takes the form:

$$\boldsymbol{h}_t^{l+1} = A\Big((\boldsymbol{W} * \boldsymbol{h})(l,t) \Big), \tag{4}$$

where:

$$(\boldsymbol{W} * \boldsymbol{h})(l,t) = \sum_{\tau=0}^{k} \boldsymbol{W}(l,\tau)\boldsymbol{h}_{t-\tau}^{l}. \tag{5}$$

Convolutional Neural Networks

- Equation (5) bears a strong resemblance to finite impulse response (FIR) filters in digital signal processing, which has several implications:
    1. Temporal CNNs assume that relationships are time-invariant, using the same set of filter weights at each time step and across all time.
    2. CNNs are only able to use inputs within its defined lookback window (receptive field) to make forecasts.
- A single causal CNN layer with a linear activation function is equivalent to an auto-regressive (AR) model.

Dilated Convolutions

- Using standard convolutional layers can be computationally challenging where long-term dependencies are significant, as the number of parameters scales directly with the size of the receptive field.
- To alleviate this, modern architectures frequently make use of dilated covolutional layers (*van den Oord et. al.* [2]), which extend Equation (5) as:

$$(\boldsymbol{W} * \boldsymbol{h})(l, t, d_l) = \sum_{\tau=0}^{\lfloor k/d_l \rfloor} \boldsymbol{W}(l, \tau) \boldsymbol{h}^l_{t-d_l\tau}, \tag{6}$$

- Dilated convolutions can be interpreted as convolutions of a down-sampled version of the lower layer features – reducing resolution to incorporate information from the distant past.
- By increasing the dilation rate with each layer, dilated convolutions can gradually aggregate information at different time blocks, allowing for more history to be used in an efficient manner.

Dilated Convolutions

- As an example, in the WaveNet architecture (*van den Oord et. al.* [2]) dilation rates are increased in powers of 2 with adjacent time blocks aggregated in each layer – allowing for 2^l time steps to be used at layer l:

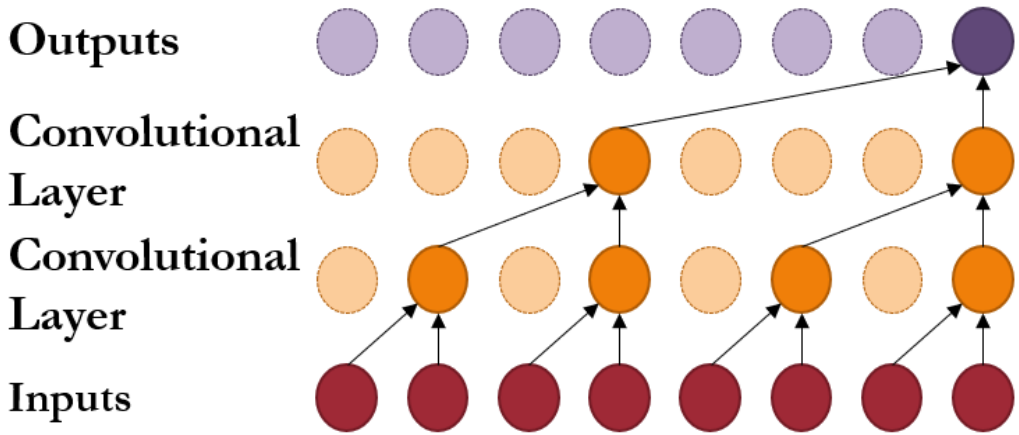


Figure: CNN model.

Recurrent Neural Networks

- Recurrent neural networks (RNNs) have historically been used in sequence modelling (*Bengio et. al.* [3]).
- Given the natural interpretation of time series data as sequences of inputs and targets, RNN-based architectures naturally lend themselves to temporal forecasting applications.
- RNN cells contain an internal memory state which acts as a compact summary of past information. The memory state is recursively updated with new observations at each time step:

$$\boldsymbol{z}_t = \nu\left(\boldsymbol{z}_{t-1}, y_t, \boldsymbol{x}_t, \boldsymbol{s}\right) \tag{7}$$

- RNNs do not require the explicit specification of a lookback window as per the CNN case. From a signal processing perspective, the main recurrent layer resembles a non-linear version of infinite impulse response (IIR) filters.

Long Short Term Memory Networks

- Due to the infinite lookback window, older variants of RNNs can suffer from limitations in learning long-range dependencies in the data due to exploding and vanishing gradients (*Bengio et. al.* [3]).
- Long Short-Term Memory networks (LSTMs) (*Hochreiter et. al.* [4]) were hence developed to address these limitations, by improving gradient flow within the network.
- This is achieved through the use of a cell state $\boldsymbol{c}_t$ which stores long-term information, modulated through a series of gates:

$$\textit{Input gate:} \quad \boldsymbol{i}_t = \sigma(\boldsymbol{W}_{i_1}\boldsymbol{z}_{t-1} + \boldsymbol{W}_{i_2}y_t + \boldsymbol{W}_{i_3}\boldsymbol{x}_t + \boldsymbol{W}_{i_4}\boldsymbol{s} + \boldsymbol{b}_i), \tag{8}$$

$$\textit{Output gate:} \quad \boldsymbol{o}_t = \sigma(\boldsymbol{W}_{o_1}\boldsymbol{z}_{t-1} + \boldsymbol{W}_{o_2}y_t + \boldsymbol{W}_{o_3}\boldsymbol{x}_t + \boldsymbol{W}_{o_4}\boldsymbol{s} + \boldsymbol{b}_o), \tag{9}$$

$$\textit{Forget gate:} \quad \boldsymbol{f}_t = \sigma(\boldsymbol{W}_{f_1}\boldsymbol{z}_{t-1} + \boldsymbol{W}_{f_2}y_t + \boldsymbol{W}_{f_3}\boldsymbol{x}_t + \boldsymbol{W}_{f_4}\boldsymbol{s} + \boldsymbol{b}_f), \tag{10}$$

where $\boldsymbol{z}_{t-1}$ is the hidden state of the LSTM.

Long Short Term Memory Networks

- The gates modify the hidden and cell states of the LSTM as:

$$\textit{Hidden state:} \quad \boldsymbol{z}_t = \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t), \tag{11}$$

$$\textit{Cell state:} \quad \boldsymbol{c}_t = \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \tanh(\boldsymbol{W}_{c_1}\boldsymbol{z}_{t-1} + \boldsymbol{W}_{c_2}y_t + \boldsymbol{W}_{c_3}\boldsymbol{x}_t + \boldsymbol{W}_{c_4}\boldsymbol{s} + \boldsymbol{b}_c), \tag{12}$$

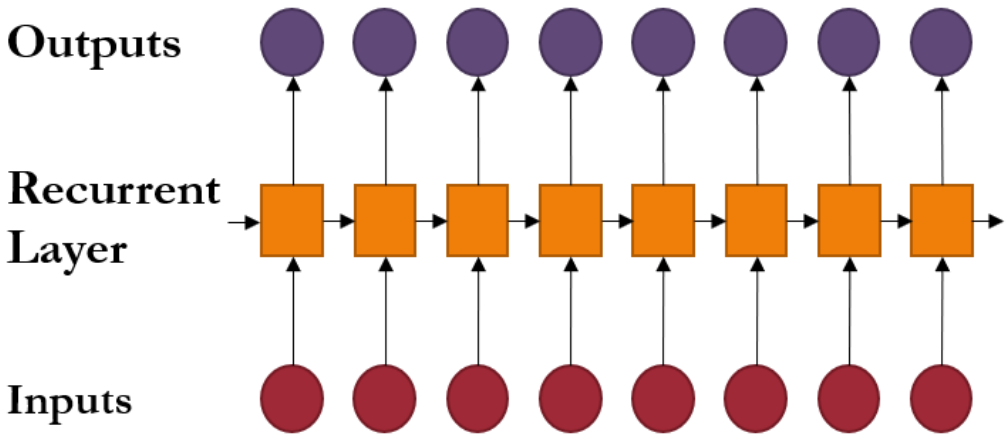


Figure: RNN model.

Attention and Self-Attention

- The development of attention mechanisms (*Bahdanau et. al.*[5]) has also lead to improvements in long-term dependency learning.
- Attention layers aggregate temporal features using dynamically generated weights, allowing the network to directly focus on significant time steps in the past.
- Conceptually, attention is a mechanism for a key-value lookup based on a given query, taking the form:

$$\boldsymbol{h}_t = \sum_{\tau=0}^{k} \alpha(\boldsymbol{\kappa}_t, \boldsymbol{q}_\tau)\boldsymbol{v}_{t-\tau}, \tag{13}$$

- An example of the usage of attention in time-series forecasting can be seen in [**?**], who aggregate features extracted by RNN encoders with attention weights produced as:

$$\boldsymbol{\alpha}(t) = \text{softmax}(\boldsymbol{\eta}_t), \tag{14}$$

$$\boldsymbol{\eta}_t = \mathbf{W}_{\eta_1} \tanh(\mathbf{W}_{\eta_2}\boldsymbol{\kappa}_{t-1} + \mathbf{W}_{\eta_3}\boldsymbol{q}_\tau + \boldsymbol{b}_\eta). \tag{15}$$

Attention and Self-Attention

- From a time series modelling perspective, attention provides two key benefits.
  1. Networks with attention are able to directly attend to any significant events that occur.
  2. Attention-based networks can also learn regime-specific temporal dynamics, by using distinct attention weight patterns for each regime (*Lim. et. al.* [6]).

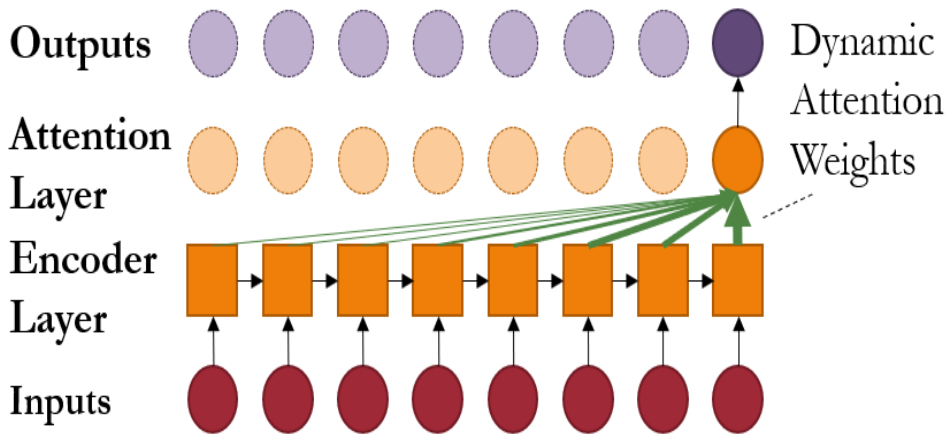


Figure: Attention model.

Transformers

- Have led to state-of-the-art performance in diverse fields, such as of natural language processing, computer vision, and speech processing (see *Lin et al.* [7]).
- The vanilla Transformer (*Vaswani et. al. [8]*) introduces the concept multi-head self-attention (defined in the next slide) in an architecture with a encoder-decoder structure.
- Unlike RNNs, Transformers has no recurrence and no convolution. Instead, it utilizes the positional encoding added in the input embeddings, to model the sequence information.
- In the original implementation, positional encoding are implemented as:

$$PE(t)_i = \begin{cases} sin(\omega_i t) & i\%2 = 1 \\ cos(\omega_i t) & i\%2 = 0 \end{cases} \tag{16}$$

Transformers

- Scaled Dot-Product attention scales values $\mathbf{V} \in \mathbb{R}^{n \times d_v}$ for queries $\mathbf{Q} \in \mathbb{R}^{n \times d_{\text{att}}}$ and keys $\mathbf{K} \in \mathbb{R}^{n \times d_{\text{att}}}$, with d_{m} the dimension of the embedding.

$$\text{Att}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\mathbf{Q}\mathbf{K}^{\top}\mathbf{U}/\sqrt{d_{\text{m}}}\right)\mathbf{V} \tag{17}$$

- This idea can be expanded to multiple heads for different representation subspaces.

$$\text{MHA}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = [\text{head}_1, ..., \text{head}_h]\,\mathbf{W}^{\mathbf{O}} \tag{18}$$

$$\text{head}_i = \text{Att}(\mathbf{Q}\mathbf{W}_i^{\mathbf{Q}}, \mathbf{K}\mathbf{W}_i^{\mathbf{K}}, \mathbf{V}\mathbf{W}_i^{\mathbf{V}}) \tag{19}$$

$$\mathbf{Y}(t) = \text{MHA}(\Theta(t), \Theta(t), \Theta(t)). \tag{20}$$

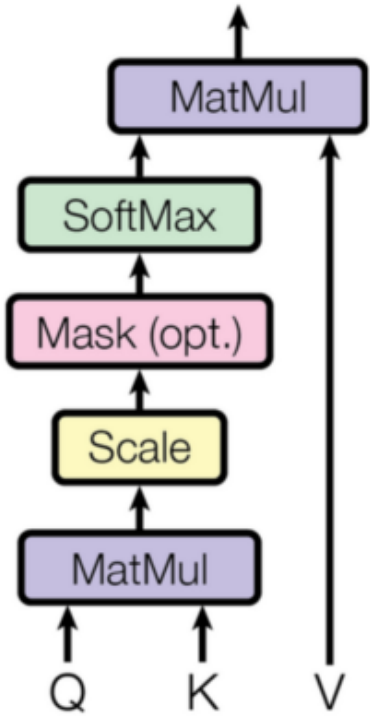


Figure: Scaled Dot-Product Attention

Transformers

- Transformers have recently have been harnessed for time-series modelling (*Li et al.* [9] *Lim et al.* [6], *Zhuo et al.* [10]).
- Different enhancements proposed in the recent literature summarized in (*Wen. et. al. [11]*):

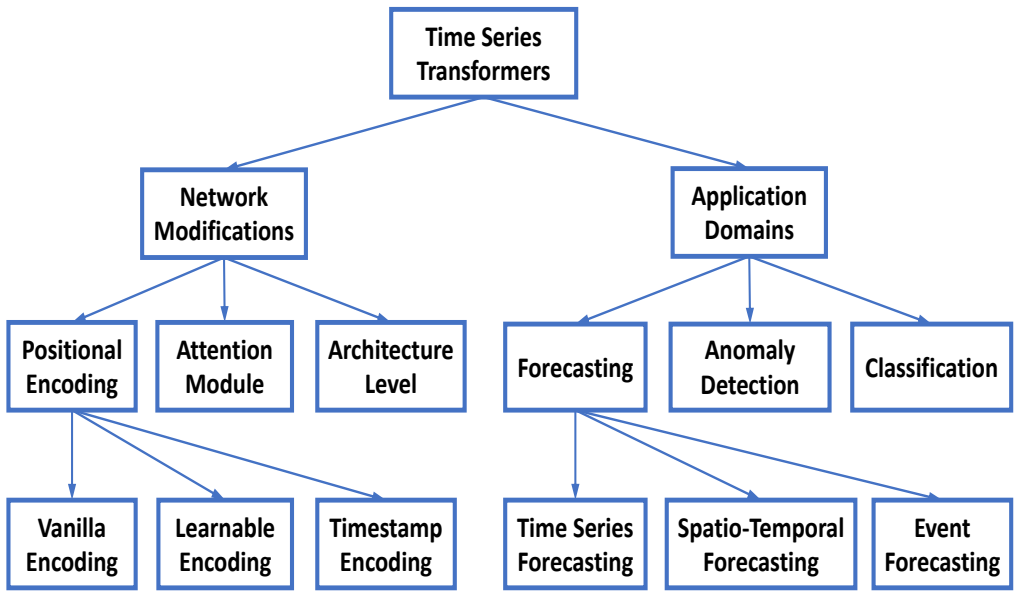


Figure: Developments in Transformers for time-series forecasting. Figure taken from (*Wen. et. al. [11]*)

Time for a break!

Papers:
Time Series Forecasting With Deep Learning: A Survey [2004.13408]

stefan.zohren@eng.ox.ac.uk

Enhancing Time Series Momentum Strategies with Deep Learning

Deep Momentum Networks

Momentum Strategies

- Time-series Momentum (TSMOM) (*Moskowitz et al.* [12]) is derived from the philosophy that strong price trends have a tendency to persist.
- Often known as 'follow the winner' because it is assumed that winners will continue to be winners in the subsequent period.
- TSMOM is a univariate approach as opposed to cross-sectional (*Jegadeesh et al.* [13]) momentum strategies, which trade assets against each other and select a portfolio based on relative ranking.
- Strategies involve 1) estimation of a trend, and 2) sizing positions accordingly.
- Momentum strategies are an important part of alternative investments and are at the heart of commodity trading advisors (CTAs).

Classical Strategies

- Volatility scaling has been proven to play a crucial role in the positive performance of TSMOM strategies (*Kim et al.* [14]).
- Where $X_t^{(i)}$ is position size of the i-th asset, N the number of assets in our portfolio, $\sigma_t^{(i)}$ the ex-ante volatility estimate and σ_{tgt} the target volatilty,

$$R_{t+1}^{\text{TSMOM}} = \frac{1}{N}\sum_{i=1}^{N} R_{t+1}^{(i)}, \quad R_{t+1}^{(i)} = X_t^{(i)} \frac{\sigma_{\text{tgt}}}{\sigma_t^{(i)}} r_{t+1}^{(i)}. \tag{21}$$

- *Moskowitz et al.* [12], selects 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.
- Moving Average Convergence Divergence (*MACD*) is a volatility normalised trend-following momentum indicator that describes the relationship between two moving averages of a security's price, functioning as a trigger for buy and sell signals (see *Baz et al.* [15]).

Deep Momentum Networks

- Previous Deep learning approaches either only learnt the trend or alternatively performed classification, taking a maximum long or short position.
- In our first work with B. Lim and S. Roberts, we proposed a framework termed as *Deep Momentum Networks* (DMNs) which resulted in significantly better risk-adusted-returns.
- Rather than estimating the trend and then using a rule based approach to size positions, DMNs learn the trend in a data-driven manner and directly output position sizes.
- A squashing function $\tanh(\cdot)$ directly outputs positions $X_t^{(i)} \in (-1, 1)$
- DMNs also benefit from the volatility scaling framework.
- Inputs are normalised returns at different timescales and different MACD indicators.

Deep Momentum Networks

- We found that the LSTM, a type of Recurrent Neural Network used for sequence modelling, produced the best results.
- Since we want to maximise risk-adjusted-returns, DMNs use a Sharpe Ratio Loss function

$$\mathcal{L}_{\text{sharpe}}(\boldsymbol{\theta}) = -\frac{\sqrt{252}\,\mathbb{E}_{\Omega}\left[R_t^{(i)}\right]}{\sqrt{\text{Var}_{\Omega}\left[R_t^{(i)}\right]}}. \tag{22}$$

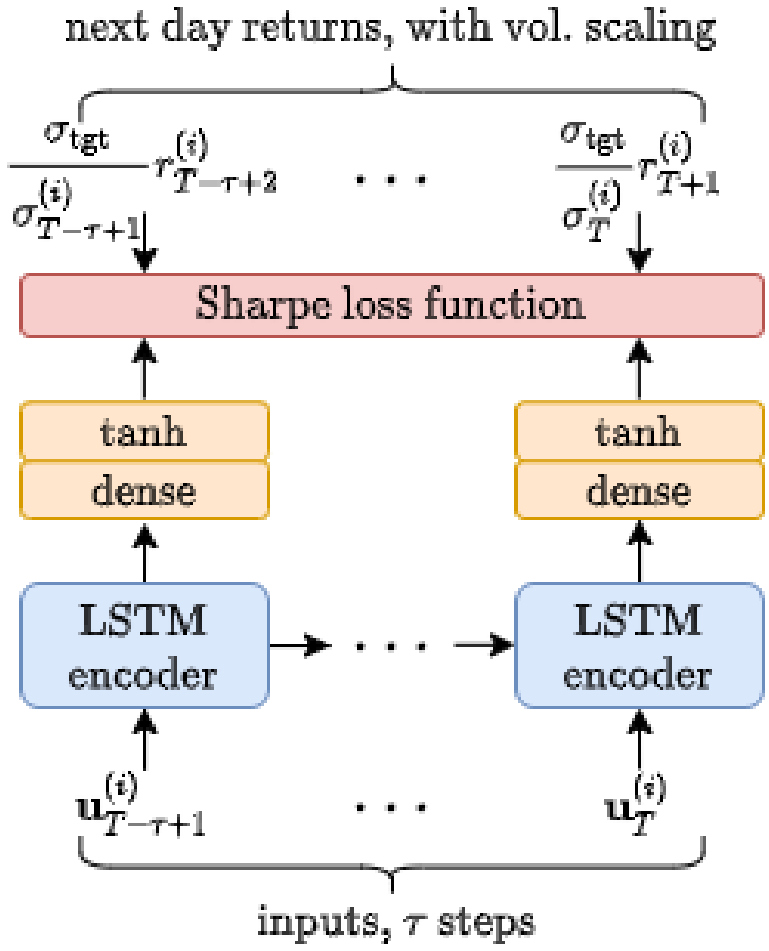
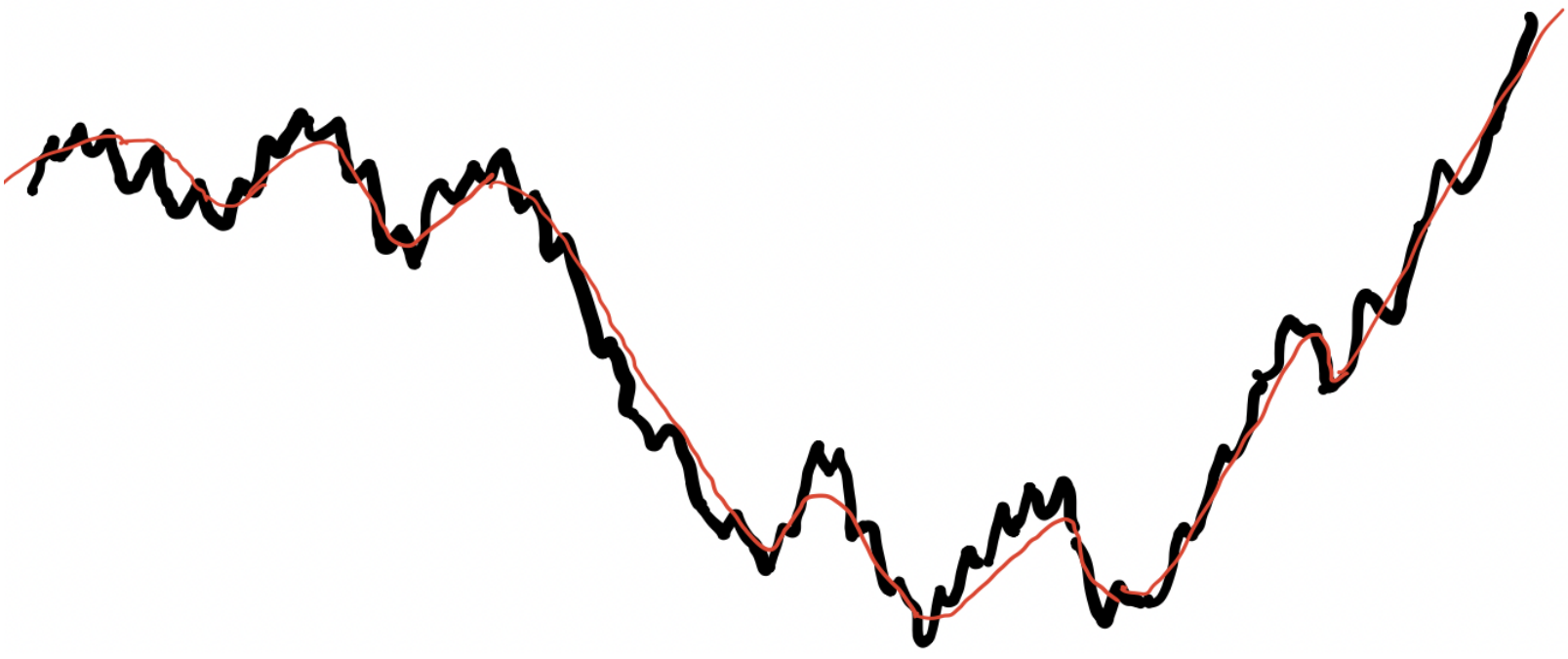


Figure: LSTM Deep Momentum Network architecture

Interpreting the results of DMNs

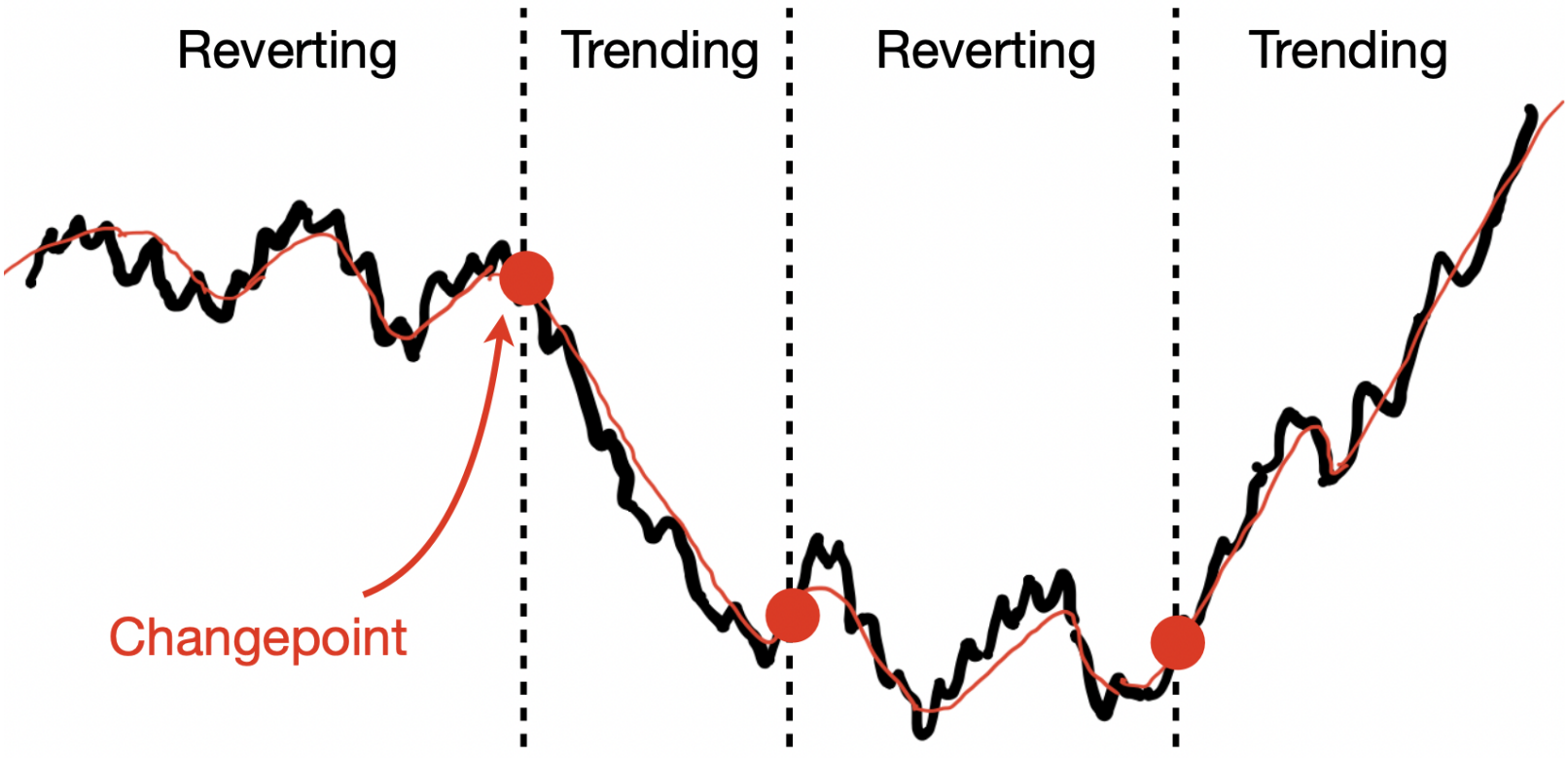
- DMN's strong performance attributed to learning simultaneously to exploit slow momentum and fast reversion
- Following small reversions in regimes of strong trend can lead to larger transaction costs



Deep Momentum Networks with Changepoint Detection

Momentum Turning Points and Changepoint Detection

- Ideally identify regimes – trending and reverting market – then learn to switch to exploit each state.
- Can be done with changepoint detection as proposed in earlier work with K. Woods and S. Roberts



Motivation from Momentum Turning Points

- Immediately after momentum turning points, where a trend reverses from an uptrend (downtrend) to a downtrend (uptrend), time-series momentum (TSMOM) strategies are prone to making bad bets.
- We require an approach which is a balancing act between being quick enough to respond to turning points, but not over-reacting to noise.
- *Garg et al.* [16] proposed an *Intermediate* stategy where a slow momentum signal based on a long lookback window, such as one year, is blended with a fast momentum signal based on a short lookback window, such as one month.

$$X_t = (1 - w)\,\mathrm{sgn}(r_{t-252,t}) + w\,\mathrm{sgn}(r_{t-21,t}). \tag{23}$$

- *MACD* can often produce false positives and signal a possible reversal without one actually happening.

Changepoint Detection

- Changepoint detection (CPD) is a field which involves the identification of abrupt changes in sequential data.
- To enable us to respond to CPD in real time, we require an 'online' algorithm, which processes each data point as it becomes available.
- First introduced by *Adams et al.* [17], Bayesian approaches to online CPD, which naturally accommodate to noisy, uncertain and incomplete time-series data, have proven to be very successful.
- We focus on approaches using Gaussian Processes (GPs) (*Williams et al.* [18]) a Bayesian non-parametric model which has a proven track record for time-series forecasting, is principled and is robust to noisy inputs.

Changepoint Detection with Gaussian Processes

- For daily return $\hat{r}_t^{(i)}$, normalised over some look-back window (we'll revisit this), we define the GP as a distribution over functions,

$$\hat{r}_t^{(i)} = f(t) + \epsilon_t, f \sim \mathcal{GP}(0, k_\xi), \epsilon_t \sim \mathcal{N}(0, \sigma_n^2), \tag{24}$$

  where ϵ is an additive noise process and $\mathcal{GP}$ is specified by a covariance function $k_\xi(\cdot,\cdot)$, which is in turn parameterised by a set of hyperparameters ξ. Noise variance σ_n, helps to deal with noisy outputs which are uncorrelated.

- The Matérn 3/2 kernel is a good choice of covariance function for noisy financial data, with kernel hyperparameters $\xi_M = (\lambda, \sigma_h, \sigma_n)$, with λ the input scale and σ_h the output scale.

- A changepoint can either be a drastic change in covariance, a sudden change in the input scale, or a sudden change in the output scale

Changepoint Detection with Gaussian Processes

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

$$p(\xi|\hat{\mathbf{r}}) = \frac{p(\hat{\mathbf{r}}|\xi)p(\xi)}{p(\hat{\mathbf{r}})} \tag{25}$$

- Then, perform type II maximum likelihood on $p(\hat{\mathbf{r}}|\xi)$, by minimising the negative log marginal likelihood,

$$\text{nlml}_\xi = \min_\xi \left(\frac{1}{2}\hat{\mathbf{r}}^\top \mathbf{V}^{-1}\hat{\mathbf{r}} + \frac{1}{2}\log|\mathbf{V}| + \frac{l+1}{2}\log 2\pi \right). \tag{26}$$

- We use the `GPflow` framework to compute the hyperparameters ξ, which in turn uses the L-BFGS-B optimisation algorithm via the `scipy.optimize.minimize` package.

Changepoint Kernel

- *Garnett et al.* introduced the **Region-switching kernel**, where it is assumed there is 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,

$$k_{\xi_R}(x,x') = \begin{cases} k_{\xi_1}(x,x') & x,x' < c \\ k_{\xi_2}(x,x') & x,x' \geq c \\ 0 & \text{otherwise.} \end{cases} \tag{27}$$

- The lookback window (LBW) l for this approach needs to be prespecified and it is assumed to contain a single changepoint.
- A more flexible approach is the **Changepoint kernel**, where $c \in (t-l, t)$ is the changepoint location, $s > 0$ is the steepness parameter and $\sigma(x,x') = \sigma(x)\sigma(x')$, $\bar{\sigma}(x,x')(1-\sigma(x))(1-\sigma(x'))$,

$$k_{\xi_C}(x,x') = k_{\xi_1}(x,x')\sigma(x,x') + k_{\xi_2}(x,x')\bar{\sigma}(x,x'). \tag{28}$$

- We use Matérn 3/2 for the left and right kernels.

Changepoint Detection Module Outputs

- We consider the series $\{r_{t'}^{(i)}\}_{t'=t-l}^{t}$, with lookback horizon l from time t. For every CPD window, where $\mathcal{T} = \{t-l, t-l+1, \ldots, t\}$, we standardise our returns for consistency.
- For each time step, our changepoint detection module outputs,
  1. changepoint detection location $\gamma_t^{(i)} \in (0,1)$, indicating how far in the past the changepoint is, and,
  2. changepoint score $\nu_t^{(i)} \in (0,1)$, which measures the level of disequilibrium, measured by the reduction in negative log marginal likelihood achieved via the introduction of the changepoint kernel hyperparameters.

$$\nu_t^{(i)}(l) = 1 - \frac{1}{1 + \mathrm{e}^{-(\mathrm{nlmn}_{\xi_C} - \mathrm{nlmn}_{\xi_M})}}, \quad \gamma_t^{(i)}(l) = \frac{c - (t-l)}{l}, \quad (29)$$

- Both values are normalised to help improve stability and performance of our LSTM module.

Changepoint Kernel

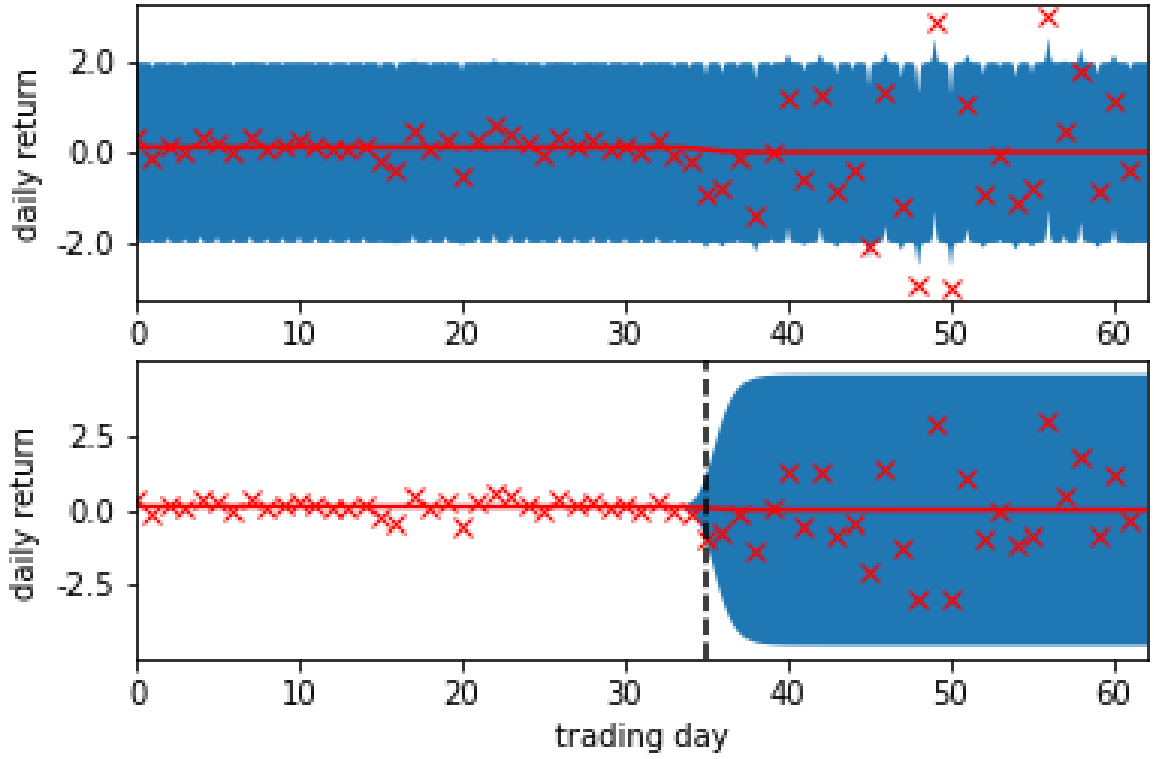


Figure: Plots of daily returns for S&P 500, composite ratio-adjusted continuous futures contract during the first quarter of 2020, where returns have been standardised. The top plot fits a GP, using the Matérn 3/2 kernel and the bottom using the Changepoint kernel specified in (28).

DMNs with Changepoint Detection Model

- In our second paper with K. Wood and S. Roberts, we introduce online CPD based on GPs into DMNs.
- Precisely, the input $u_t^{(i)}$ for each time-step of LSTM sequence consists of past returns, MACD signals, as well as changepoint location and severity scores:
  1. $\left\{ r_{t-t',t}^{(i)} / \sigma_t^{(i)} \sqrt{t'} \,|\, t' \in \{1, 21, 63, 126, 252\} \right\}$,
  2. $\{\text{MACD}(i, t, S, L) | (S.L) \in \{(8, 24), (16, 28), (32, 96)\}\}$,
  3. $\nu_t^{(i)}(l)$ and $\gamma_t^{(i)}(l)$ for $l \in \{10, 21, 63, 126, 252\}$
- The LSTM is not complex enough to handle multiple CPD lookback-windows (LBW) and we optimise l as part of the hyperparameter tuning process.
- Later work also demonstrats that multiple LBWs (short and long) work well in conjunction with a variable selection network.

Data and Experimental Setting

- Portfolio consisting of 50 of the most liquid, ratio-adjusted continuous futures contracts over the period 1990–2020.
- Includes daily Commodities, FX, Fixed Income and Equities data, extracted from the Pinnacle Data Corp CLC database.
- We use an expanding window approach, where we start by using 1990–1995 for training/validation, 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.
- We use a 90%/10% split for training/validation data, training on the Sharpe loss function via minibatch Stochastic Gradient Descent (SGD), using the validation set to tune the hyper-parameters and for early stopping.
- The outer optimisation loop tunes dropout rate, hidden layer size, minibatch size, learning rate, max gradient norm and CPD LBW length, with 50 iterations of random grid search.

Performance Results

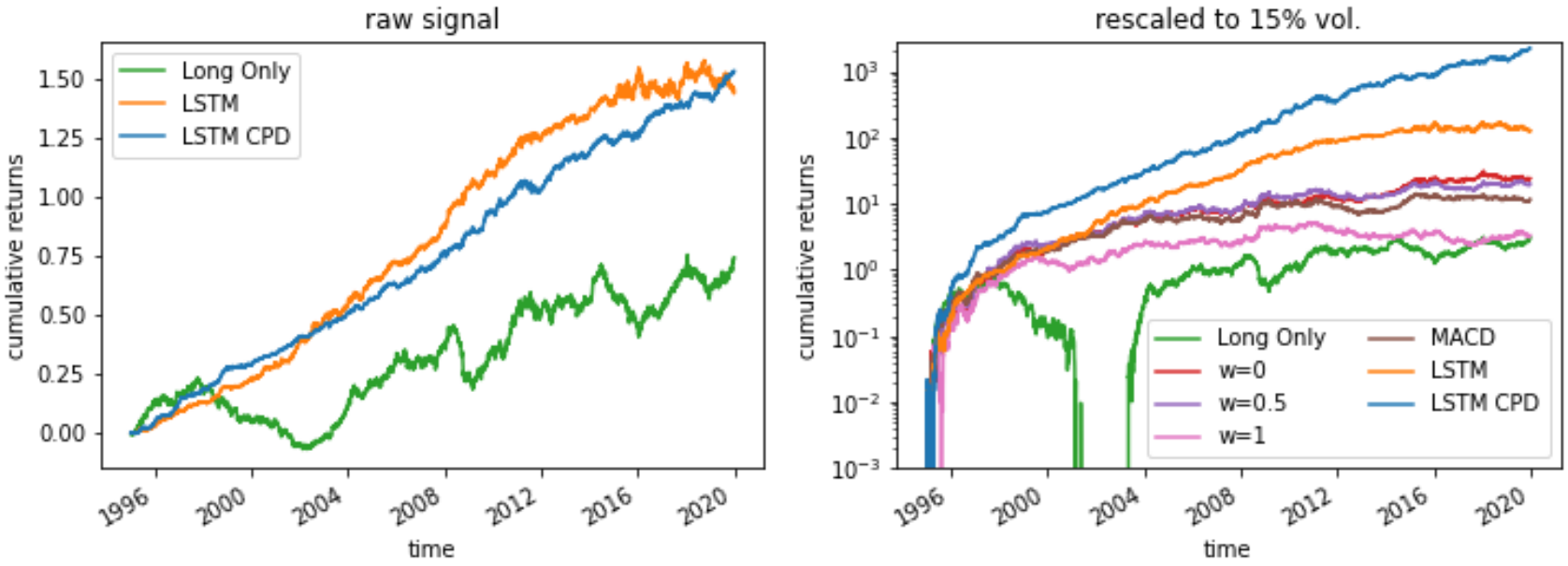


Figure: Benchmarking DMNs against *Intermediate* strategy $w \in \{0, 0.5, 1\}$, *Long Only* and *MACD*.

Performance Results

	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 Optimised	3.64%	1.73%	**2.16**	1.17%	**3.33**	1.14%	3.50	56.22%	**1.133**

Figure: Strategy performance benchmark for raw signal output.

Slow Momentum with Fast Reversion

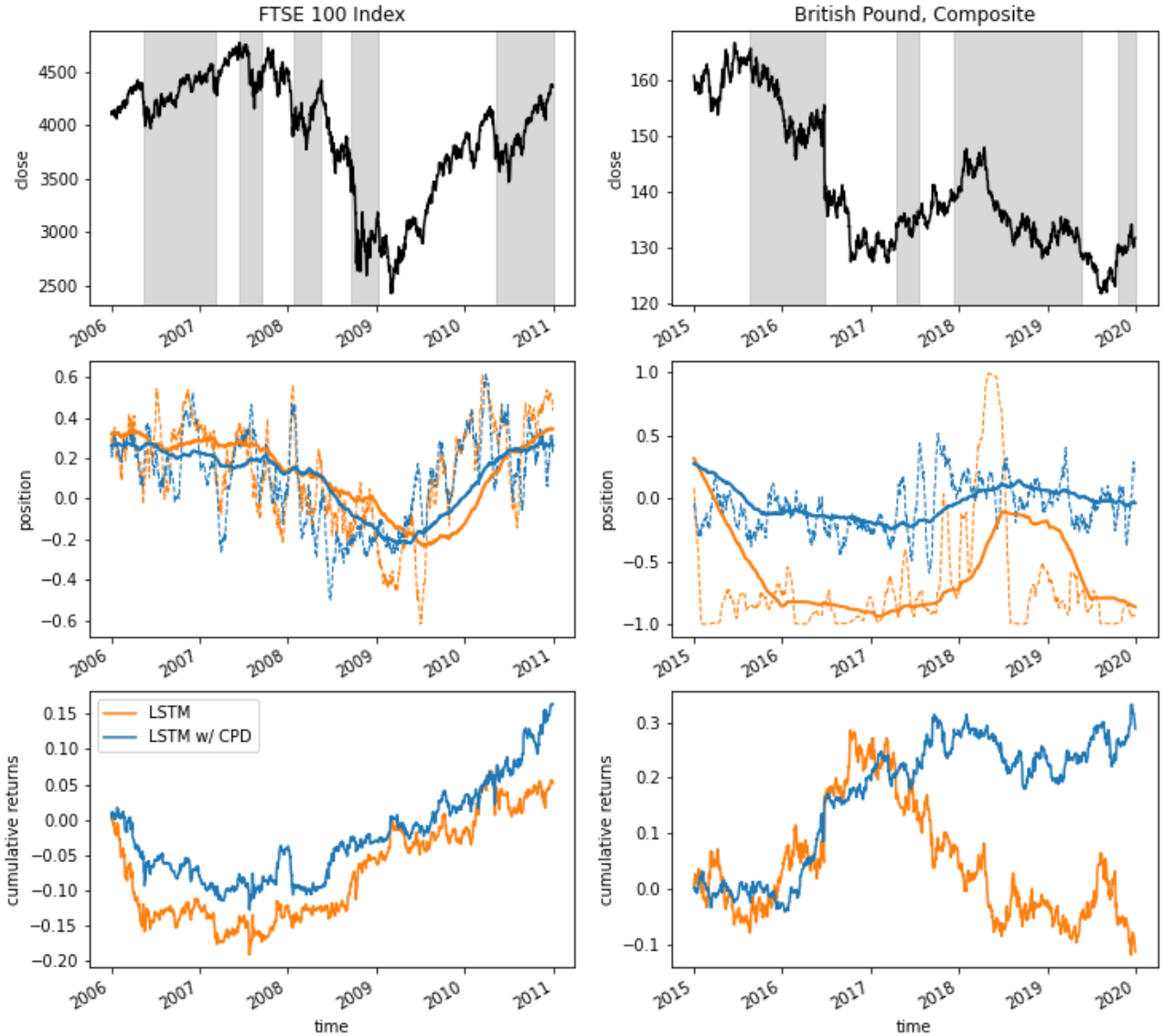


Figure: Slow momentum and fast reversion happening simultaneously.

Momentum Transformer

Transformers

- Based on the concept ‘attention is all you need’, doing away with convolutions and recurrent neural networks (RNNs).
- The attention-based architecture allows the network to focus on significant time steps in the past and longer-term patterns
- Have led to state-of-the-art performance in diverse fields, such as of natural language processing, computer vision, and speech processing (see *Lin et al.* [7]).
- Have recently have been harnessed for time-series modelling (*Li et al.* [9] *Lim et al.* [6], *Zhuo et al.* [10]).
- Naturally adapts to new market regimes, such as during the SARS-CoV-2 crisis.

Base Architectures Tested in the Momentum Transformer

- **Transformer:** (*Vaswani et al.* [8]) consists of encoder and decoder – each consisting of l identical layers of a (multi) self-attention mechanism, followed by a position-wise feed-forward network and a residual connection between these two components.
- **Decoder-Only Transformer:** (*Li et al.* [9]) only the decoder side.
- **Convolutional Transformer:** *Li et al.* [9] incorporates convolutional and log-sparse self-attention.
- **Informer Transformer:** *Zhuo et al.* [10] replaces the naive sparsity rule of the Conv. Transformer with a measurement based on the Kullback-Leibler divergence to distinguish essential queries, referred to as *ProbSparse* self-attention.
- **Decoder-Only Temporal Fusion Transformer (TFT):** an attention-LSTM hybrid which uses recurrent LSTM layers for local processing and interpretable self-attention layers for long-term dependencies. We consider the Decoder-Only version of the original TFT (*Lim et al.* [6]).

Momentum Transformer (Decoder-Only TFT)

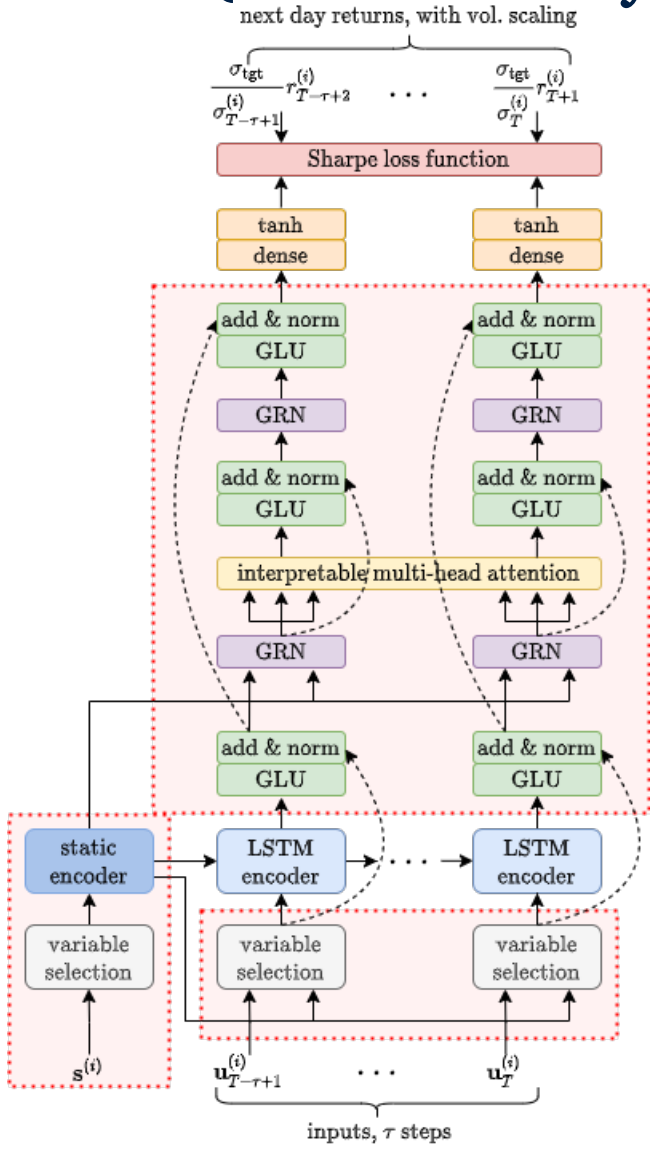


Figure: Decoder-Only TFT

Results

	Returns	Vol.	Sharpe	Downside Deviation	Sortino	MDD	Calmar	% of +ve Returns	Ave. P / Ave. L
Average 2015–2020									
Long-Only	1.73%	5.00%	0.37	3.59%	0.51	11.41%	0.15	51.97%	0.982
TSMOM	0.97%	4.38%	0.24	3.19%	0.33	8.25%	0.12	52.82%	0.931
LSTM	1.23%	1.85%	0.82	1.32%	1.19	3.55%	0.66	53.38%	1.004
Transformer	1.98%	1.29%	1.53	0.85%	2.32	1.07%	1.86	54.76%	1.071
Decoder-Only Trans.	1.37%	1.97%	0.72	1.37%	1.03	2.63%	0.60	52.76%	1.012
Conv. Transformer	1.85%	1.92%	0.98	1.30%	1.47	3.14%	0.77	52.93%	1.056
Informer	1.67%	1.09%	1.51	0.72%	2.30	1.17%	1.44	54.39%	1.089
Decoder-Only TFT	1.99%	1.23%	1.71	0.82%	2.61	1.17%	2.06	55.72%	1.073
Decoder-Only TFT CPD	**2.06%**	**1.02%**	**2.00**	**0.66%**	**3.10**	**0.82%**	**2.53**	**55.74%**	**1.120**
SARS-CoV-2									
Long-Only	-1.46%	6.73%	-0.19	5.64%	-0.22	12.32%	-0.12	57.28%	0.720
TSMOM	0.90%	4.73%	0.21	3.14%	0.32	4.17%	0.22	50.00%	1.041
LSTM	-4.15%	2.82%	-1.50	2.52%	-1.67	5.35%	-0.78	52.29%	0.643
Transformer	4.42%	**1.28%**	**3.38**	**0.83%**	**5.55**	**0.84%**	7.31	**64.85%**	1.066
Decoder-Only Trans.	**8.02%**	2.58%	3.01	1.42%	**5.55**	1.05%	**8.56**	58.83%	**1.243**
Conv. Transformer	3.13%	1.99%	1.81	1.40%	2.74	1.61%	3.17	57.48%	1.058
Informer	4.30%	1.60%	2.71	1.00%	4.45	1.07%	4.28	59.61%	1.137
Decoder-Only TFT	1.81%	1.75%	1.22	1.37%	1.74	2.14%	1.57	60.39%	0.831
Decoder-Only TFT CPD	3.39%	1.51%	2.47	1.03%	4.08	1.15%	5.92	59.90%	1.068

Figure: Strategy Performance Benchmark – Raw Signal Output

Results

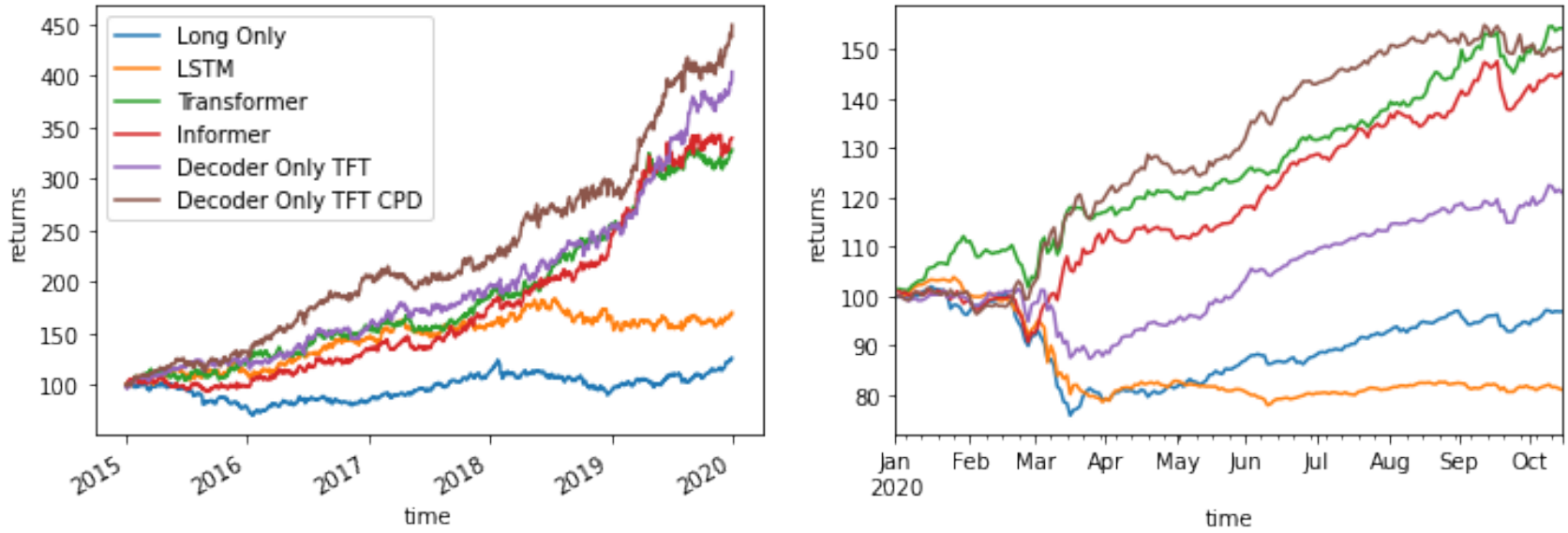


Figure: These plots benchmark our strategy performance for the 2015–2020 scenario (left) and the SARS-CoV-2 scenario (right). For each plot we start with $100 and we re-scale returns to 15% volatility. Since we ran each experiment five times, we plot the repeat which resulted in the median Sharpe ratio, across the entire experiment.

Attention Patterns

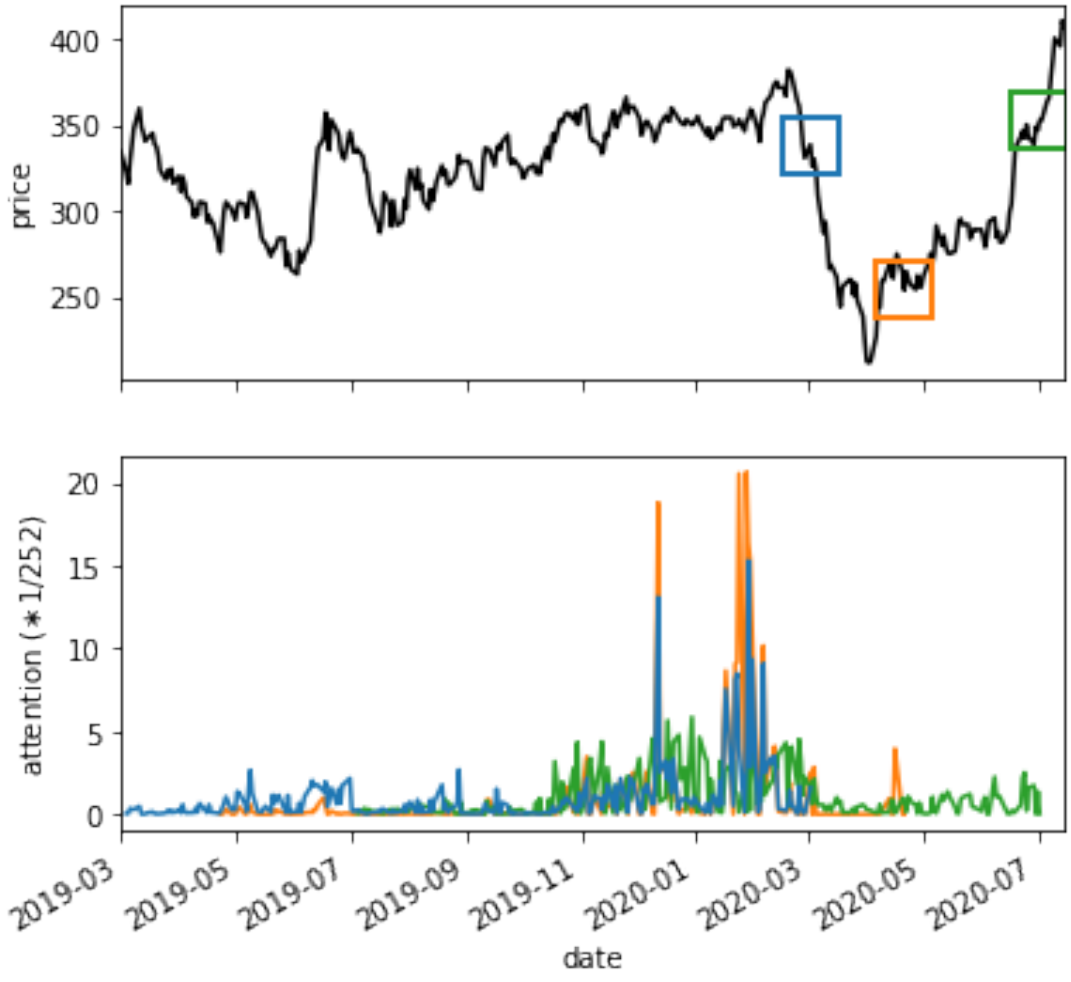


Figure: Lumber future price during SARS-CoV-2 crisis and the associated attention pattern when making a prediction at 1 March 2020 (blue), 21 April 2020 (orange), and 2 July 2020 (green).

Attention Patterns

- We observe significant structure in attention patterns.
- The attention on momentum turning points is pronounced, segmenting the time series into regimes.
- Our model focuses on previous time-steps which are in a similar regime.

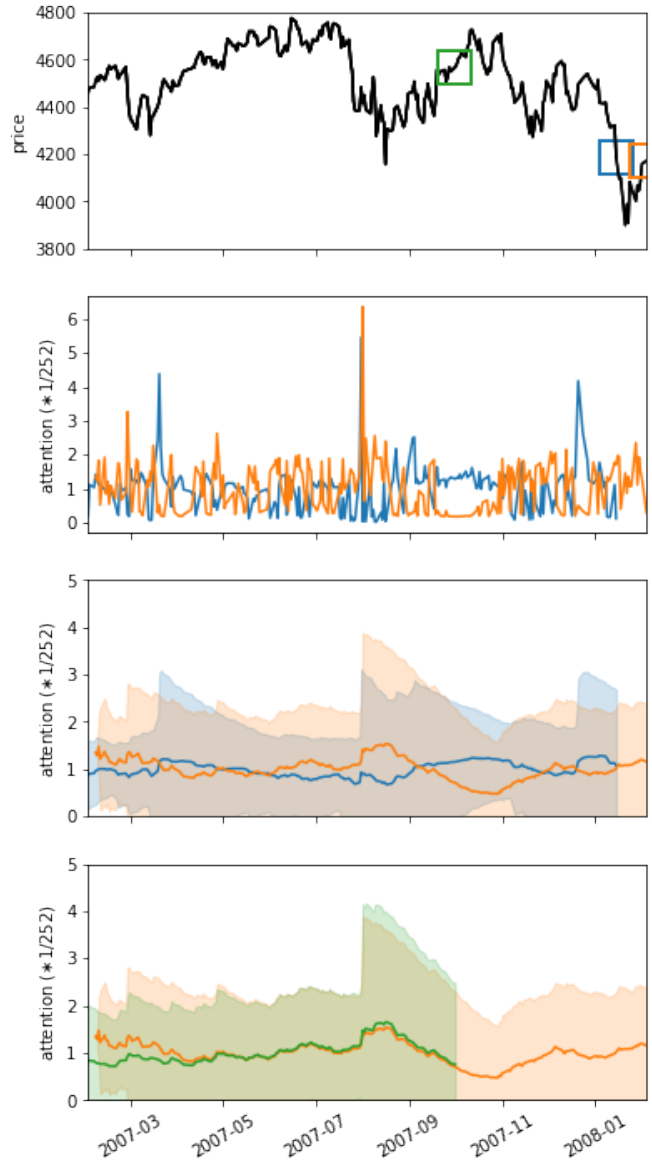


Figure: FTSE 100 future prior to 2008.

Variable Importance

- Our model intelligently blends different classical strategies at different points in time.
- We observe that the strategy changes with the addition of CPD, placing left emphasis on returns at timescales in between daily (shortest) and annual (longest).

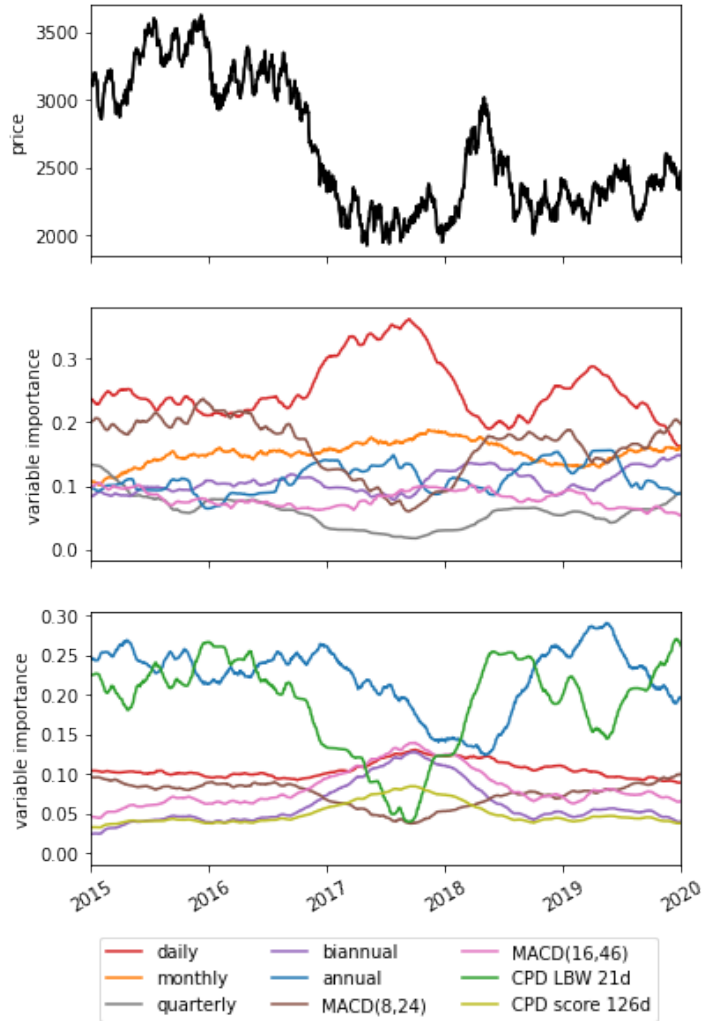


Figure: Variable importance for Cocoa future for Decoder-Only TFT (middle) and with CPD (bottom).

Transaction Cost Impact

C	0bps	0.5bps	1bps	1.5bps	2bps	2.5bps	3bps
LSTM							
Indv. CM	0.12	0.09	0.05	0.01	-0.02	-0.06	-0.10
Indv. EQ	**0.37**	0.32	0.27	0.22	0.16	0.11	0.06
Indv. FI	0.09	-0.11	-0.32	-0.53	-0.74	-0.94	-1.15
Indv. FX	0.11	0.01	-0.08	-0.18	-0.27	-0.37	-0.46
Portfolio	0.82	0.51	0.20	-0.12	-0.43	-0.74	-1.05
Transformer							
Indv. CM	0.27	0.23	0.19	0.15	0.11	0.08	0.04
Indv. EQ	**0.37**	**0.33**	**0.28**	**0.23**	**0.19**	**0.14**	**0.10**
Indv. FI	0.23	0.03	**-0.16**	**-0.35**	**-0.55**	**-0.74**	**-0.93**
Indv. FX	-0.17	-0.24	-0.32	-0.39	-0.47	-0.55	-0.62
Portfolio	1.53	1.26	0.99	0.72	**0.45**	**0.18**	**-0.09**
Informer							
Indv. CM	0.28	0.24	0.19	0.15	0.10	0.06	0.01
Indv. EQ	0.34	0.28	0.22	0.16	0.10	0.04	-0.02
Indv. FI	0.08	-0.13	-0.35	-0.56	-0.78	-0.99	-1.20
Indv. FX	-0.14	-0.24	-0.33	-0.43	-0.53	-0.62	-0.72
Portfolio	1.51	1.17	0.83	0.49	0.15	-0.19	-0.53
Decoder-Only TFT							
Indv. CM	0.44	0.40	0.35	0.31	0.26	0.22	0.17
Indv. EQ	0.25	0.19	0.13	0.07	0.02	-0.04	-0.10
Indv. FI	**0.30**	**0.05**	-0.20	-0.45	-0.69	-0.94	-1.18
Indv. FX	**0.28**	**0.18**	**0.08**	**-0.02**	**-0.12**	**-0.22**	**-0.32**
Portfolio	1.71	1.36	1.01	0.67	0.32	-0.03	-0.37
Decoder-Only TFT CPD							
Indv. CM	**0.55**	**0.50**	**0.45**	**0.40**	**0.35**	**0.30**	**0.25**
Indv. EQ	0.18	0.12	0.05	-0.01	-0.07	-0.14	-0.20
Indv. FI	0.23	-0.03	-0.29	-0.55	-0.81	-1.07	-1.33
Indv. FX	0.24	0.13	0.02	-0.09	-0.20	-0.30	-0.41
Portfolio	**2.00**	**1.61**	**1.22**	**0.83**	0.44	0.04	-0.35

Figure: Transaction cost impact on Sharpe over 2015–2020 for individual assets, averaged by asset class, and for diversified portfolio.

Conclusions

- Deep Momentum Networks are novel models which directly output trading signals which are optimised for Sharpe ratio
- The original deep Momentum Networks based on LSTMs perform well by exploiting a blend of momentum and mean reversion
- We introduce Changepoint detection to this model to more intelligently adapt to changes from trending to more reverting regimes
- We further improve the model by considering transformer based architectures
- The attention-based architectures, which we tested, are robust to significant events, such as during the SARS-CoV-2 market crash and tend to focus less on mean-reversion and more on longer term trends.

Time for a break!

Papers:
Deep Momentum Networks [1904.04912]
DMNs with Changepoints [2105.13727]
Momentum Transform [2112.08534]

stefan.zohren@eng.ox.ac.uk

Cross-section Momentum Strategies with Learning-to-Rank

Learning to Rank Algorithms for Cross-Sectional Systematic Strategies

Cross-Sectional Strategies

- Cross-sectional (CS) strategies involves buying 'winners' and selling 'losers' over fixed points in time – where winners/losers here respectively refer to some group of assets with the highest/lowest exposure to some specific risk factor.
- Popular style of systematic trading with many different 'flavors' – differentiated by risk factors, asset class, time horizon.
- Different from Time-series strategies which narrowly focuses on individual assets.
- Cross-sectional strategies instead concentrates on the *relative* performance across assets.

Cross-Sectional Strategies – Pipeline

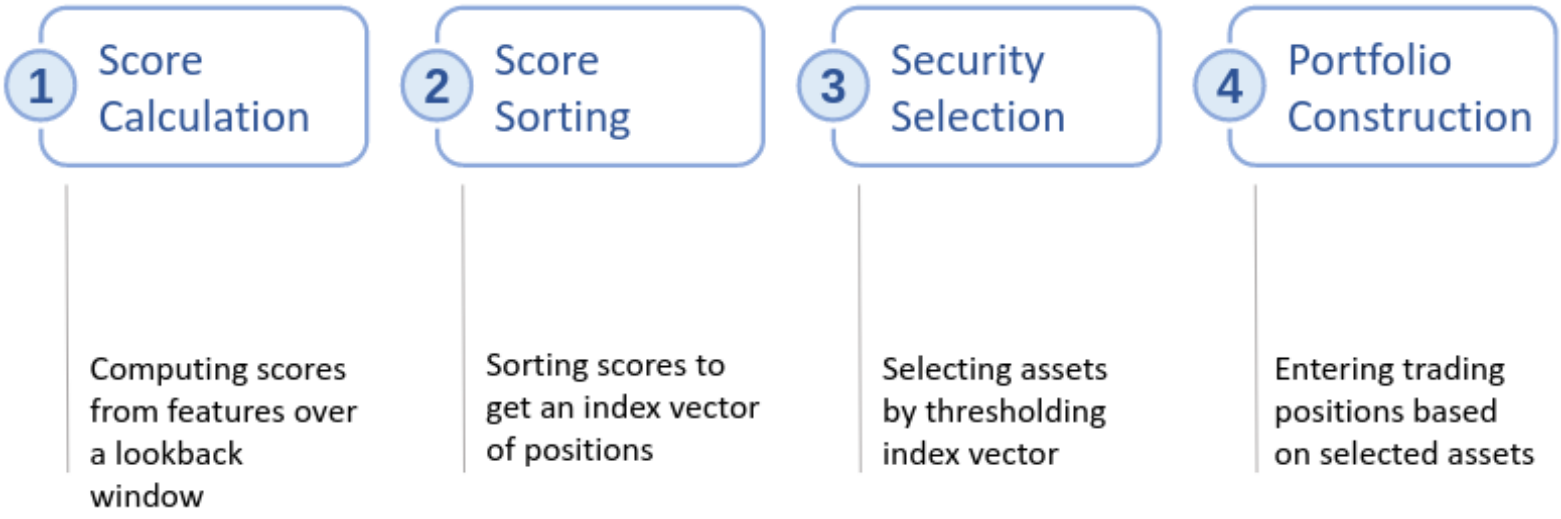


Figure: Pipeline for Cross-sectional Strategies: 4 key steps, with numerous works directed at improving various stages of the pipeline (1 and 4 are extremely active).

Cross-Sectional Momentum

- Momentum is based on the idea that returns persists over time.
- Cross-sectional momentum (CSMOM) is based on the seminal paper by *Jegadeesh et al.* [13]: First rank stocks by returns over past 12 months. Next, buy assets with highest returns (winners) and and sell assets with the lowest (losers).
- CSMOM is a classical long/short equity strategy employed by hedge funds.
- Ranking is core of strategy – numerous approaches aimed at performing this step, ranging from simple heuristics to sophisticated deep neural architectures.

Motivation

- Earlier approaches rank assets with heuristics or with advanced models trained on mean-squared error (MSE) – essentially optimising a proxy instead of actual rankings.
- Not directly targeting measure of interest leads to suboptimal trading performance.
- Directly optimising rank-based metrics difficult since their loss landscapes are problematic – non-differentiable in many areas.

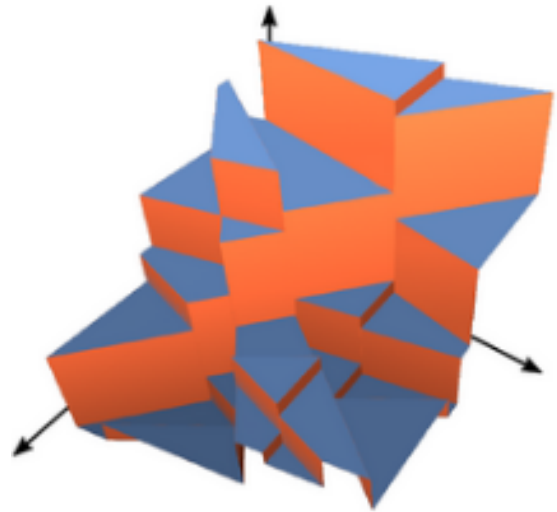


Figure: Ranking loss landscape: Challenging to optimize with gradient based measures (image from [19]).

Learning to Rank Algorithms

- Learning to Rank (LTR) is a supervised technique aimed at training algorithms to sort un-ordered objects to maximise utility.
- LTR is an active research area in academia (for e.g., information retrieval (IR)) and industry (Amazon, Spotify, etc.) – with typical use-cases being search and recommender systems.
- Importantly, LTR algorithms provides a way to *avoid* directly optimizing rank-based metrics.

Learning to Rank Algorithms

- Differentiated by *underlying ranking* and *loss functions*. For e.g., ListNet uses a NN and listwise loss, with latter given as:

$$\ell(\mathbf{y}, f(\mathbf{x})) = -\sum_i \text{softmax}(\mathbf{y}_i) \times \log\big(\text{softmax}(f(\mathbf{x}))_i\big) \tag{30}$$

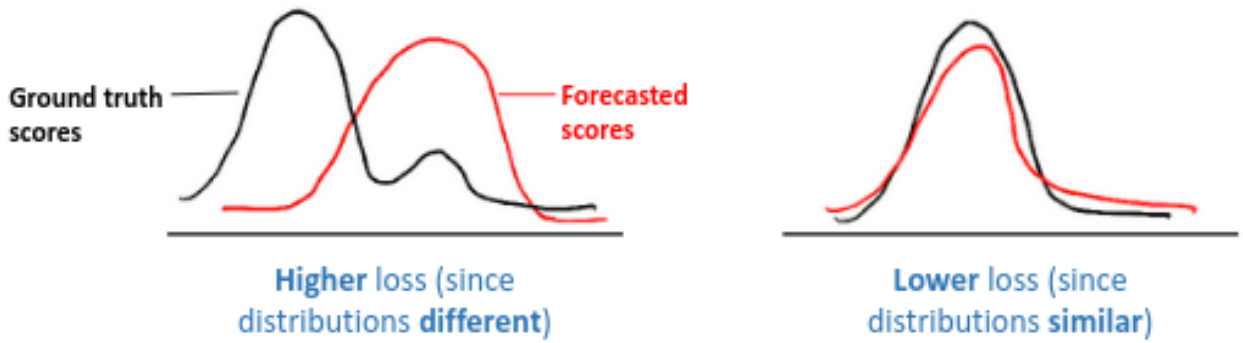


Figure: ListNet loss aims to minimise the statistical difference between the forecasted and ground truth distributions. Analogous to minimizing the Kullback–Leibler Divergence.

Learning to Rank for Cross-Sectional Momentum

- With D. Poh, B. Lim and S. Roberts, we demonstrate how to adapt LTR algorithms (RankNet, LambdaMART, etc.) for CSMOM, and provide a general framework for applying LTR to Cross-sectional strategies [20].
- LTR optimise models using ranking losses. Minimising the ranking loss, improves the accuracy of selecting winner and loser assets – thus boosting overall CSMOM performance.
- Analogous to work done with Deep Momentum Networks, where we directly optimise the metric of interest (i.e. Sharpe Ratio).

Data and Experimental Settings

- Data on obtained from Center for Research in Security Prices (CRSP) spanning 1980 to 2019. Focus on closing prices of publicly traded US equities.
- At each rebalancing interval, only use stocks $> \$1$ and actively trading over previous year. Out of around 1000+ stocks, we long the top 100 names and sell bottom 100.
- We use an expanding window approach of 5-year blocks, and a 90%/10% split for training/validation.
- Benchmark models rank randomly, or by using heuristics, with an advanced trend indicator, or by optimising MSE. All LTR models rank using ranking losses (pairwise, listwise).
- Where applicable, each model's hyperparameters (dropout rate, minibatch size, learning rate, hidden layer size, tree-depth, etc.) are calibrated with Bayesian optimisation.

Performance Results

Exhibit 2: Performance Metrics – Rescaled to Target Volatility (Updated for 2020).

	Benchmarks				Learning to Rank Models			
	Rand	JT	Baz	MLP	RNet	LM	LNet	LMLE
E[returns]	0.040	0.087	0.103	0.051	0.239	***0.353**	0.295	0.255
Volatility	0.154	0.167	0.163	0.165	0.165	0.168	***0.161**	0.163
Sharpe	0.257	0.519	0.634	0.308	1.450	***2.095**	1.833	1.566
Downside Dev.	0.099	0.107	0.101	0.111	0.080	***0.071**	0.074	0.073
MDD	0.554	0.328	0.343	0.637	0.292	***0.231**	0.262	0.245
Sortino	0.402	0.813	1.018	0.456	3.004	***4.946**	4.002	3.468
Calmar	0.072	0.265	0.301	0.079	0.820	***1.528**	1.126	1.041
% +ve Returns	0.552	0.576	0.597	0.537	0.690	***0.773**	0.710	0.672
Avg. P / Avg. L	1.112	1.112	1.108	1.092	1.418	1.439	***1.572**	1.561

Figure: Strategy performance comparison – LTR vs. Benchmarks. LTR models comfortably outperform the benchmarks models, with LambdaMART (LM) being the best ranker.

Performance Results

- LTR optimises the ranking loss function – improving ranking accuracy and thus boosting overall CSMOM performance.

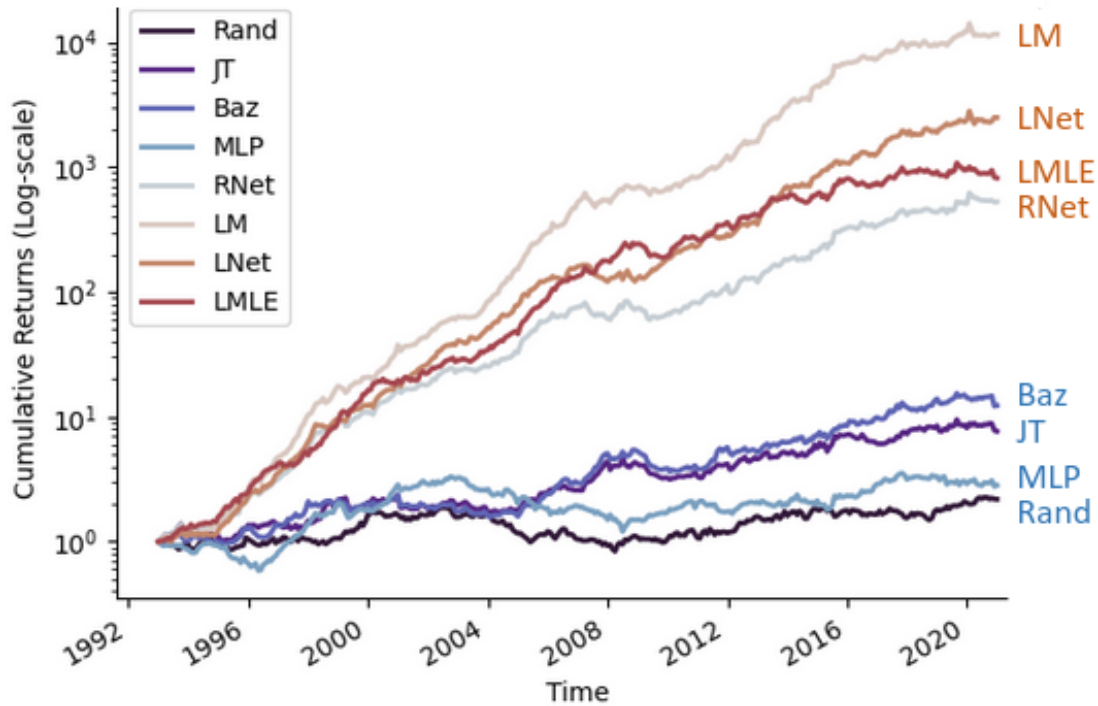


Figure: LTR (LM, LNet, LMLE, RNet) based strategies as a group collectively outperform traditional models on US equities.

Performance Results

- Optimising the ranking loss function improves ranking accuracy and boosts overall CSMOM performance.

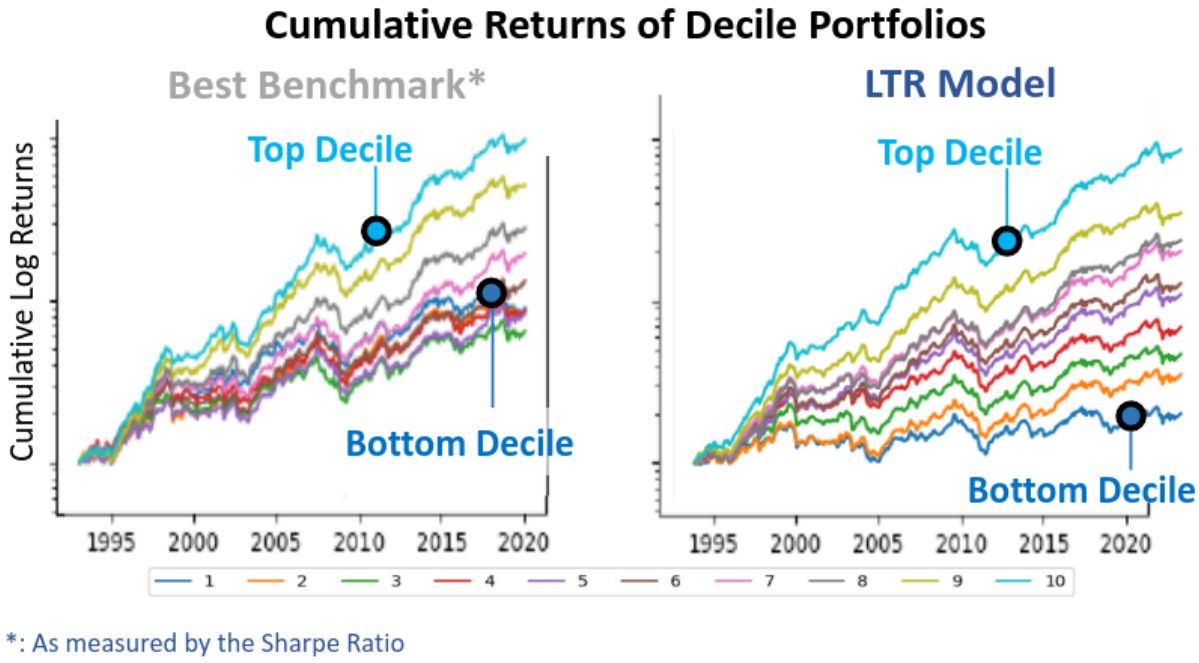


Figure: Decile portfolio returns across LTR (right) and best benchmark models. The ability of the LTR model to rank assets more accurately is reflected in the more fanned out decile returns streams.

Context-aware Learning to Rank with Self-Attention

Motivation

- Problem: In IR, LTR models trained globally – not optimal as it ignores differences between feature distributions across queries.
- Consequence: Globally learned models rank accurately on average, but may be suboptimal for individual queries.

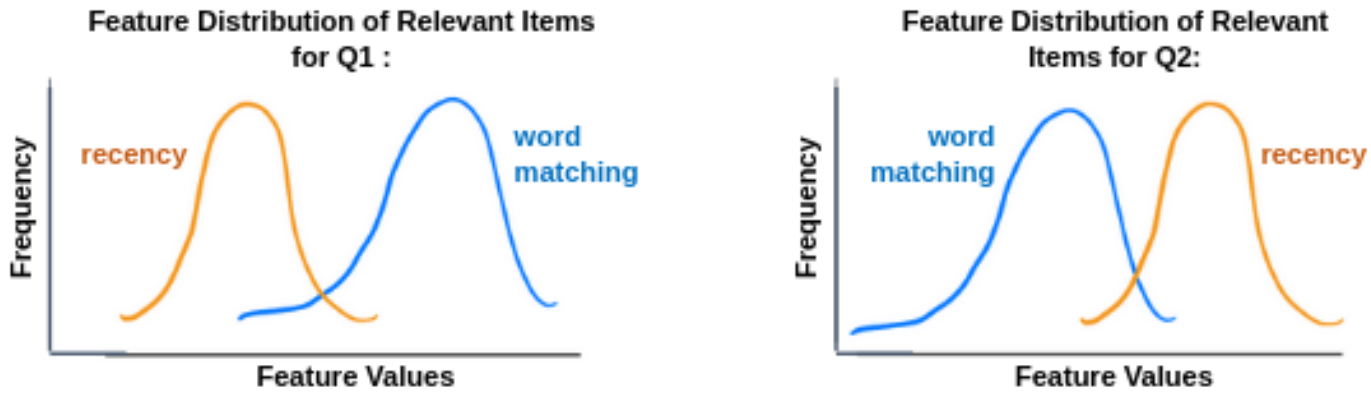


Figure: Distribution of features ('recency' and 'word matching') can shift for different Queries Q1 and Q2.

Motivation

- With LTR in CSM, the algorithms rank accurately on average but may suffer from poor accuracy over certain (*local*) instances in time when features assume a different distribution.
- Some of these local instances may coincide with *risk-off* episodes which are associated with sharp price moves – detrimental for trading and portfolio performance.

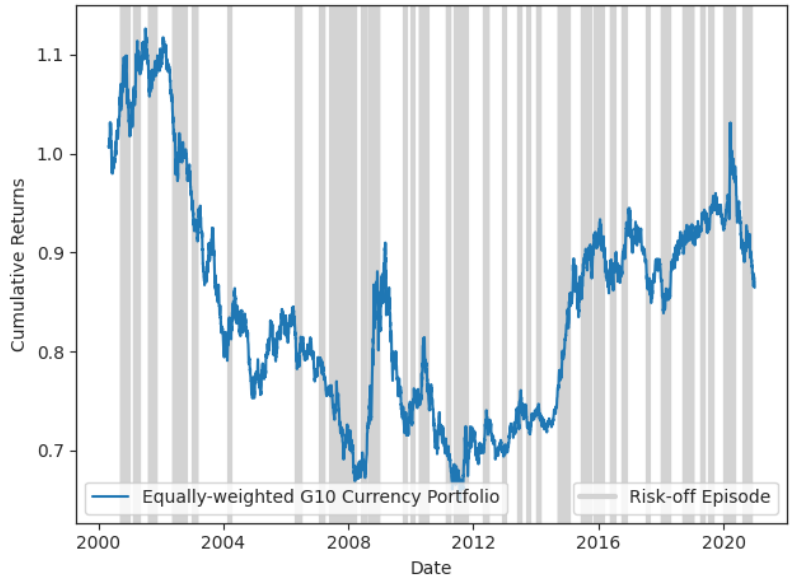


Figure: Equal weighted G10 FX portfolio. Risk-off periods (gray vertical bars) coincide with abrupt price moves.

Approach

- We make use of related research in IR – that the *features* of top-ranked items (local ranking context) contains useful information that can be used to refine an already sorted list.
- We capture this information using the Transformer's encoder module. The self-attention mechanism allows inter-item dependencies to be accounted for – not only during training at the loss level, but also at inference time.

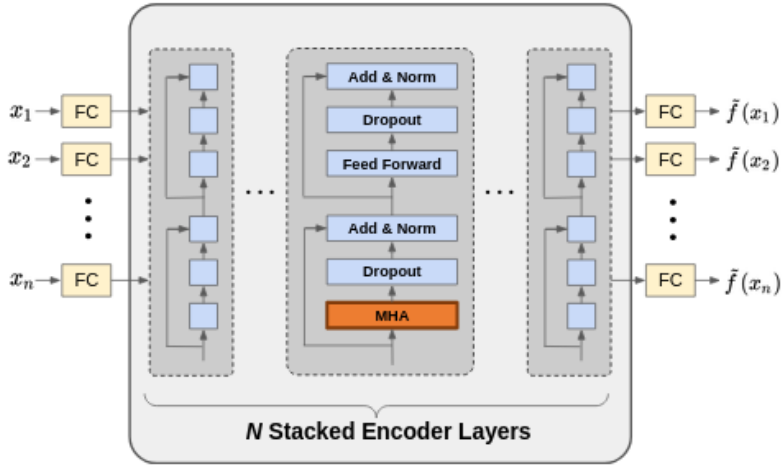


Figure: Transformer Encoder Module.

Approach

- Our final model follows that used by *Pobrotyn et al.* [21] who demonstrate, in an IR setting with web search results, that the transformer's encoder module can be used to re-rank an already sorted list of objects. This further improves the accuracy of the sorted list.
- This encoder-only architecture is referred to as the Context-aware ranking model with Self-Attention [21].

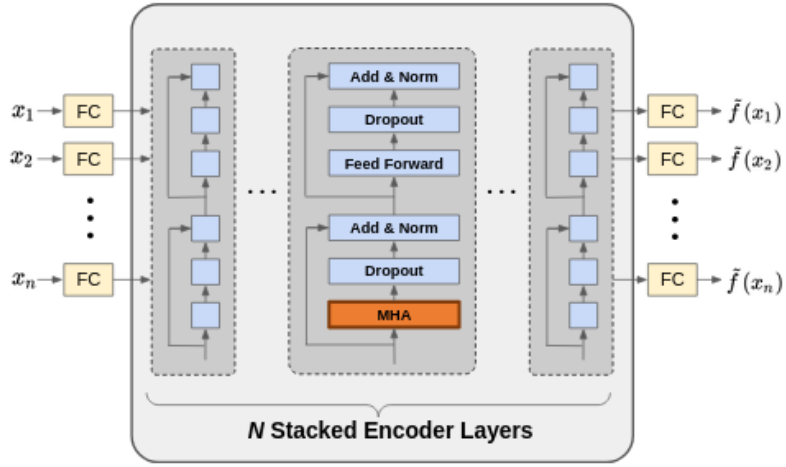


Figure: Transformer Encoder Module.

Re-ranking Pipeline

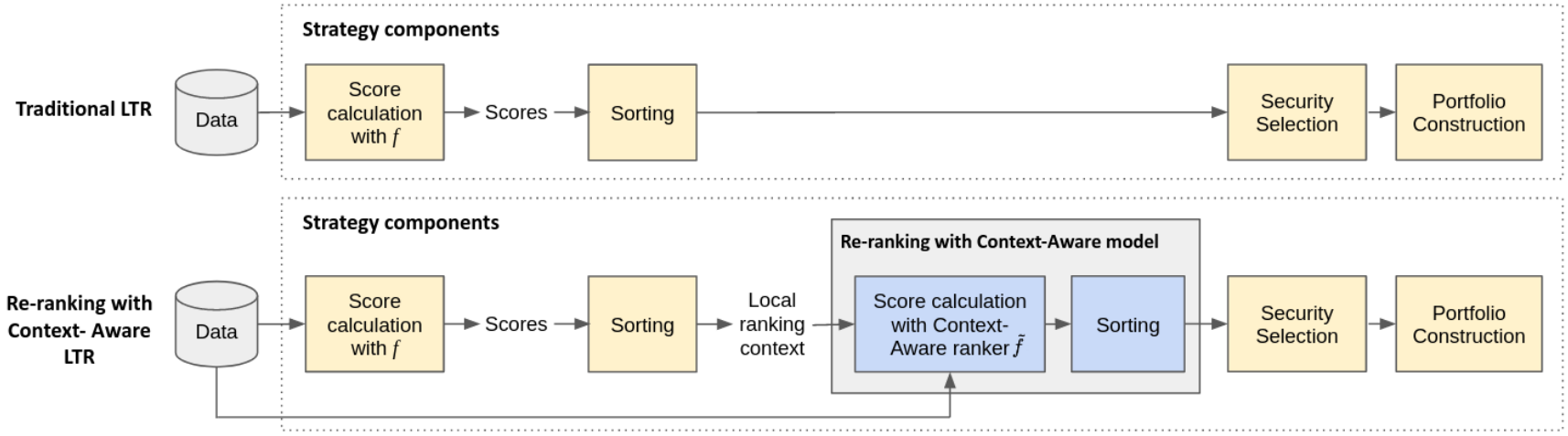


Figure: Standard LTR vs Context-aware LTR pipelines. Under the latter pipeline, the top and bottom instruments of an already-sorted list is *re-ranked* with the Context-aware LTR algorithm. This step produces a new score which we then sort and go on to construct portfolios.

Data and Experimental Settings

- 30 FX pairs (G10, EMEA, LATAM, etc.) obtained from the *Bank of Int'l Settlements (BIS)* [22]. Our data spans 2000-2020.
- VIX data obtained from *Cboe Global Markets* [23]. We define a risk-off event as instances when the VIX is 5% higher than its rolling 60-day MA. All other market states are normal states.
- Portfolio rebalanced daily. We long the top 3 FX and short the bottom 3.
- Benchmarks: Heuristics-based models, LTR models (e.g. ListNet (LN), LambdaMART (LM), etc.)
- All machine learning models use an expanding window approach of 5-year blocks, and a 90%/10% split for training/validation.

Performance Results

	PW-based models			ML-based models			LN-based models			LM-based models		
	PW	PW+P	PW+L	ML	ML+P	ML+L	LN	LN+P	LN+L	LM	LM+P	LM+L
E[returns]	0.098	**0.141**	0.117	0.096	0.116	**0.120**	0.098	**0.157**	0.133	0.129	0.148	****0.157**
Volatility	0.157	**0.156**	0.157	0.164	0.158	**0.157**	**0.162**	0.165	0.165	0.157	0.157	****0.156**
Sharpe Ratio	0.624	**0.905**	0.746	0.586	0.729	**0.764**	0.602	**0.948**	0.806	0.822	0.941	****1.008**
Downside Dev.	0.110	**0.107**	0.108	**0.106**	0.110	0.107	**0.108**	0.109	0.110	0.110	0.109	****0.106**
Max Drawdown	0.240	****0.234**	0.287	0.538	**0.533**	0.540	0.326	0.285	**0.239**	0.264	**0.236**	0.279
Sortino Ratio	0.890	**1.315**	1.086	0.907	1.050	**1.119**	0.905	**1.431**	1.210	1.179	1.357	****1.479**
Calmar Ratio	0.407	**0.604**	0.410	0.179	0.217	**0.223**	0.299	0.549	**0.559**	0.490	****0.626**	0.564
Hit-rate	0.523	**0.532**	0.518	0.510	0.519	**0.523**	0.515	**0.531**	0.514	0.531	****0.535**	0.535
AP/AL	1.016	1.025	**1.056**	**1.068**	1.050	1.038	1.047	1.047	****1.093**	1.018	1.022	**1.032**

Figure: Performance of standard LTR models and models that have been re-ranked with Context-aware LTR. Re-ranking with the Context-aware LTR improves the baseline models.

Performance Results

	PW-based models			ML-based models			LN-based models			LM-based models		
	PW	PW+P	PW+L	ML	ML+P	ML+L	LN	LN+P	LN+L	LM	LM+P	LM+L
Normal	0.705	**0.872**	0.724	0.691	0.775	**0.870**	0.573	**0.936**	0.735	0.913	1.041	****1.065**
Risk-off	-0.652	**1.495**	1.137	-1.174	**0.109**	-1.049	1.151	1.200	****2.126**	-0.706	-0.720	**0.003**

Figure: Sharpe ratios of standard LTR models and models that have been re-ranked with Context-aware LTR conditioned on normal and risk-off market states (for each grouping of 3 columns, the second and third columns refer to the re-ranked version of the model in the first column). We see that by re-ranking the original benchmark model with the Context-aware LTR, we improve the performance of the model over both normal and risk-off market states.

Performance Results

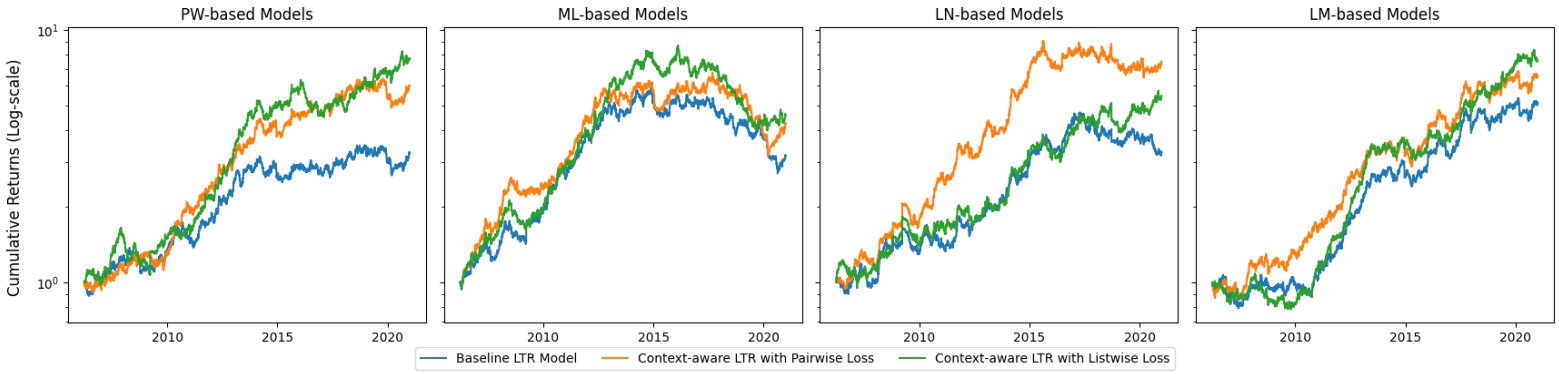


Figure: Log cumulative returns of reference baseline (blue) against the same models that undergo an additional re-ranking step (orange, green) on 2000-2020 FX data. Re-ranking is able to improve the cumulative returns profiles of the baseline models.

Transfer Ranking in Finance

Motivation

- Deep learning based LTR strategies have been used successfully in data-rich settings involving mature assets with long histories (or in a high-frequency context).
- Deploying these models on instruments with limited samples is super challenging given that DL models are well-known for being data-hungry.
- This often results in over-fitted models with degraded trading performance.

Approach

- In our up and coming paper with D. Poh and S. Roberts, we introduce the Fused Encoder Networks (FEN) – a hybrid transfer ranking (TR) model.
- Knowledge embedded in source model (trained on a larger and related upstream data set) is incorporated into a target model, which helps mitigate the problem of data scarcity.
- We focus on cryptocurrencies given its wide academic and institutional interest. Its limited sample sizes makes them a challenging and pragmatic use-case.
- In our study, we construct CSMOM portfolios and rebalance at the *weekly* frequency – further raising the bar for our algorithm.

Transfer Ranking Pipeline

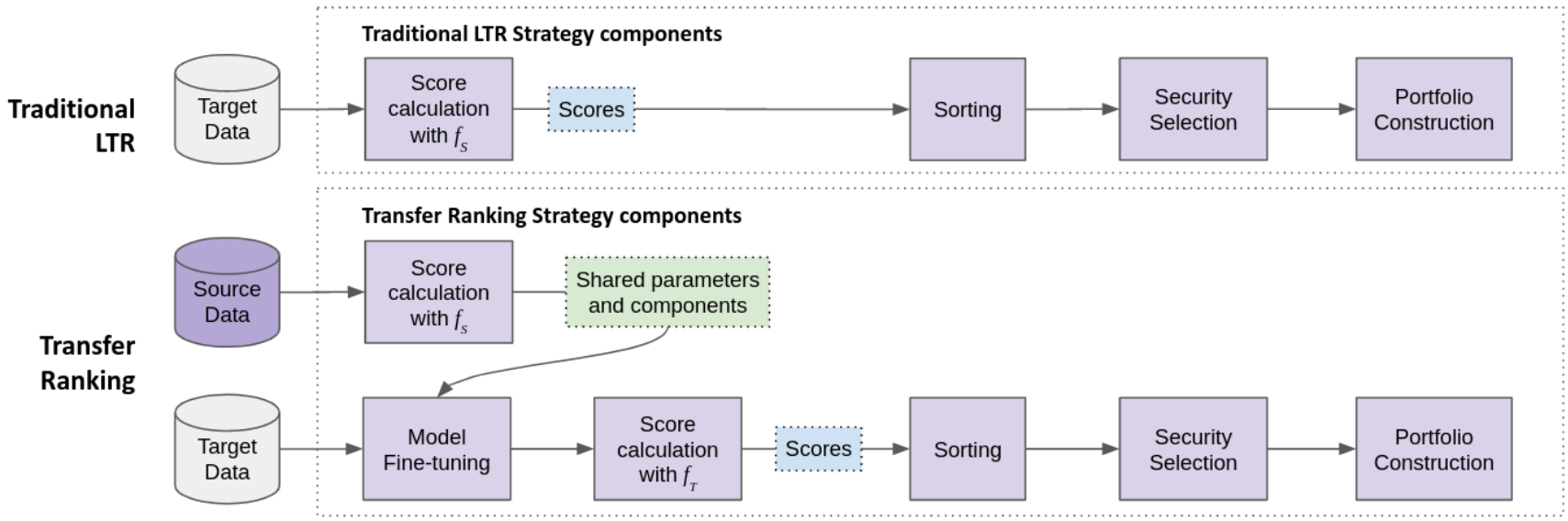


Figure: Comparing traditional LTR and TR pipelines. The latter involves first training a source model on a different but related data set, and then transferring the calibrated components to the target model. Apart from requiring an additional fine-tuning step, the remainder of the target model's workflow is similar to that of traditional LTR.

Data and Experimental Settings

- Data: Daily closing prices obtained from CoinMarketCap resampled to weekly frequency.
- Instruments: Top 10 cryptocurrencies by market cap by the end of Dec-2020 and with at least 4 years of data. We fix this universe going forward to minimise the impact of survivorship bias. Full data set runs from Jan-2016 to Dec-2021.
- We rebalance on a weekly basis, trading long/short the top/bottom 2 cryptocurrencies (quintiles).
- Benchmarks: 1-week momentum, various LTR models, transfer learning ranker constructed using the parameter-sharing.

Performance Results

	Benchmarks						Proposed
	1WR	MLP	LN	LM	SAR	SAR+ps	FEN
E[returns]	0.053	0.080±0.043	-0.094±0.132	0.076±0.109	0.084±0.100	-0.006±0.074	***0.141±0.080**
Volatility	0.221	0.201±0.033	0.211±0.087	0.205±0.068	***0.176±0.015**	0.178±0.012	0.187±0.012
Sharpe Ratio	0.238	0.403±0.240	-0.397±0.539	0.456±0.546	0.496±0.583	-0.035±0.406	***0.749±0.410**
Sortino Ratio	0.484	0.692±0.436	-0.405±0.745	0.943±1.061	0.937±1.070	0.028±0.597	***1.526±1.023**
Calmar Ratio	0.190	0.373±0.231	-0.109±0.364	0.439±0.546	0.454±0.543	0.027±0.205	***0.828±0.543**
Downside Deviation	0.109	0.125±0.039	0.173±0.098	0.132±0.081	0.115±0.029	0.122±0.022	***0.102±0.017**
Max. Drawdown	0.276	0.248±0.084	0.443±0.153	0.285±0.115	0.272±0.097	0.356±0.113	***0.188±0.036**
Avg. Profit / Avg. Loss	1.417	1.175±0.155	0.840±0.194	1.172±0.239	1.190±0.211	1.068±0.187	***1.342±0.275**
Hit Rate	0.435	0.499±0.030	0.481±0.035	0.505±0.026	0.500±0.029	0.472±0.019	***0.509±0.031**

Figure: Performance comparison across 2018-2021, rescaled to match 15% annual volatility target. All models except 1WR computed based on 10 runs. FEN outperform benchmarks on nearly all measures.

Attention Heatmaps: Normal / Risk-off Market States

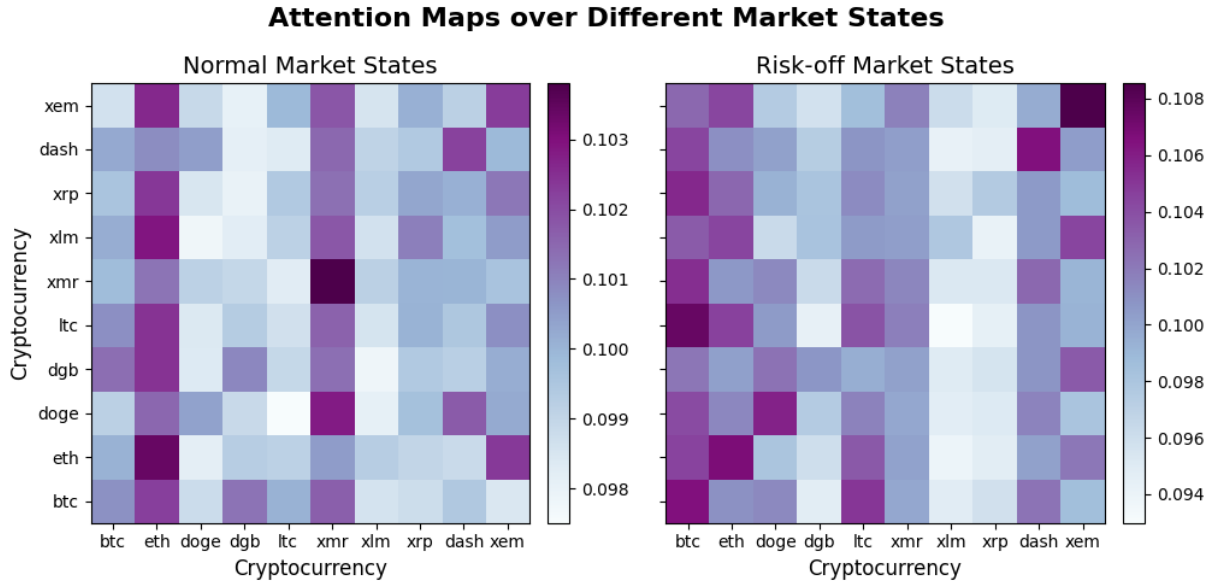


Figure: Aggregated attention heatmaps over different market states. Darker cells correspond areas which contribute more to the FEN's prediction. Darker cells appear more dispersed over risk-off states than normal states – highlighting the broad nature of these events, where multiple instruments are affected.

Attention Heatmaps: Specific Cryptocurrency Events

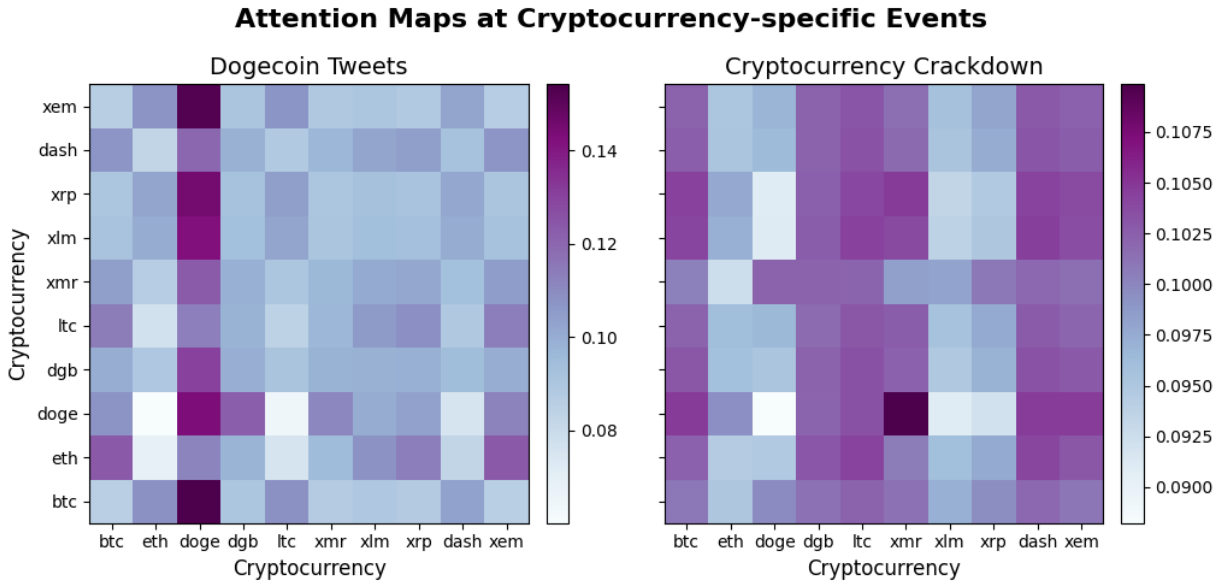


Figure: Aggregated attention heatmaps over crypto-specific events. Elon Musk's dogecoin tweets in May 2021 is reflected in the 'doge' column highlighted (left). The pattern of diffused darker blocks around late May 2019 coincides with China's regulators declaring its intent to crackdown on cryptos – reflecting the risk-off sentiments inherent in this event (right).

Conclusion

- LTR is popular area both in academia (primarily IR) and industry.
- The finance community is starting to pick up on LTR – we demonstrate a general framework how this can be done in our first work. In short, LTR improves the ranking accuracy of CS models which then boosts strategy performance.
- While LTR models rank accurately on average, they may fail to produce precise rankings when accuracy is most needed. This might be over risk-off periods which exposes the portfolio to unwanted drawdowns. We rectify this with Context-aware LTR with Self-attention.
- Our most recent work addresses the data-scarcity problem associated with low-frequency financial time-series data with FEN – a transfer ranking model. FEN outperforms standard benchmarks, and appears to possess an interpretable attention structure.

Time for a break!

Papers:
LTR for Cross-Sectional Systematic Strategies [2012.07149]
Context-aware LTR with Self-Attention [2105.10019]
Transfer Ranking in Finance [Coming soon!]

stefan.zohren@eng.ox.ac.uk

DeepVol: Volatility Forecasting via Dilated Causal Convolutions

An Introduction to Volatility Forecasting

- **Time-series volatility forecasting** plays a central role among **equity risk models**, essential tools to measure and adapt to the uncertainty characterising financial markets.
- Volatility measures are used as portfolio's risk proxy measure, increasing the usage of **volatility conditional portfolios**, which:
  - Increase portfolios' Sharpe ratio [24],
  - Reduce the likelihood of observing extreme heavy-tailed returns[25].
- Most of the models used by practitioners are based on classic methodologies, such as the GARCH model [26].

An Introduction to Volatility Forecasting

- Predictions of traditional models have been improved through the usage of **realised measures** [1] as predictors for the realised volatility [27].
  - The integration of these realised measures into the forecast requires pre-processing steps.
- Models like EGARCH [28] and High-Frequency-Based Volatility (HEAVY) [29] are of special appeal among industry practitioners [30]:
  - Based on insights from the ARCH architecture.
  - Reporting outstanding results with respect to other classical benchmarks.
  - Producing accurate and interpretable results.

[1] Non-parametric estimators of the variation of an asset's price during a time gap that summarises and exploit high-frequency data

An Introduction to Volatility Forecasting

- In spite of their popularity among practitioners, **realised measures-based models** are not able to directly use high-frequency data, which exposes them to several **disadvantages**:
  - The usage of realised measures artificially **limits the amount of information** used while forecasting the day-ahead volatility.
  - Some of the most popular and utilised realised measures **lack robustness to noise** [31]:
    - The trained models may be based on **biased data**.
  - Reported performance may be worsen due to:
    - **Dependence on data's handcrafted features**, as the realised measures' design itself.
    - **Model misspecification**, common to traditional model-based approaches.

An Introduction to Volatility Forecasting

- Motivated by the improved performance of classical methods caused by the integration of realised measures, we propose using **Dilated Causal Convolution(DCC)-based models** for **day-ahead volatility forecasting**:
  - The proposed architecture, DeepVol, takes inspiration from classical econometric methodologies by **integrating realised measures into the realised volatility forecast**.
  - DeepVol extracts relevant information from intraday's financial data by means of dilated convolutional filters:
    - **Data-driven approach** to naturally mimick the established methodologies' incorporation of realised measures into the forecast.
    - The model takes advantage of the abundance of stock market data, avoiding classical models' limitations.

Preliminaries: Data & Models Inputs

- The analysis conducted will be based on **financial returns** instead of the assets' price evolution.
- This allows us to transform the original price trend time-series into a **quasi-stationary process**, through the following transformation:

$$r_{i,t} = \log\left(\frac{p_{i,t}}{p_{i-1,t}}\right), \tag{31}$$

where $p_{i,t}$ is the closing price of an asset in the i-th interval on day t, and $r_{i,t}$ is the close to close return at the specified sampling frequency, i.e. 1, 5, 15, 30 or 60 minutes.

Preliminaries: Data & Models Inputs

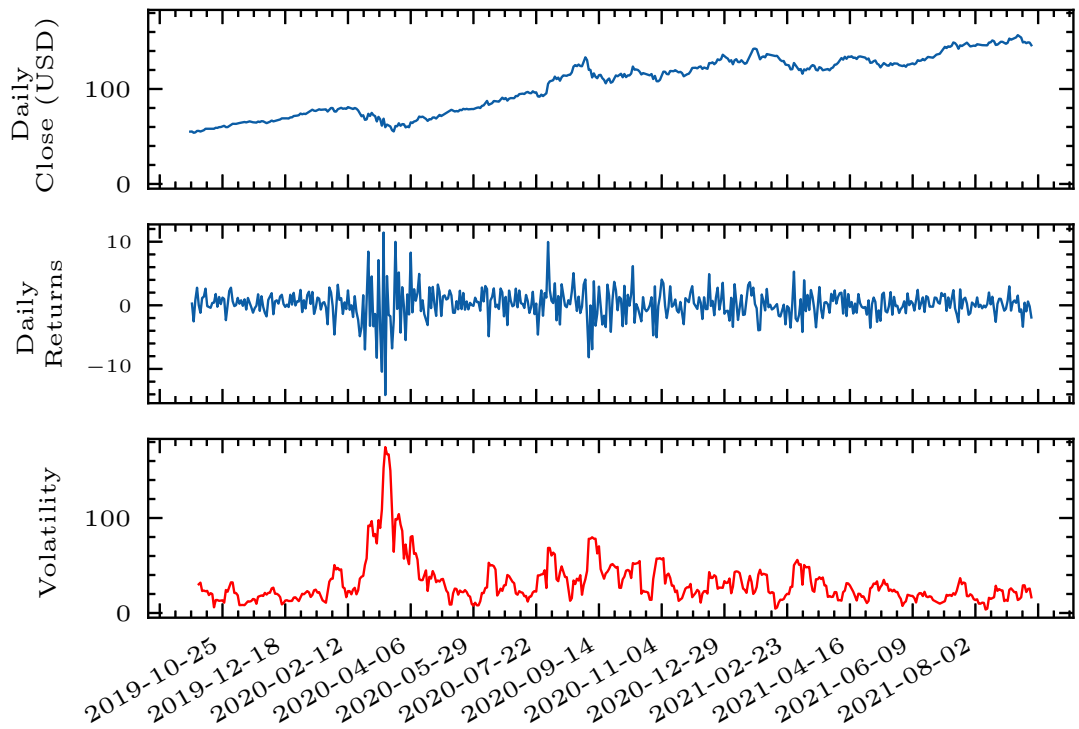


Figure: Apple's daily data. The top row shows the **price trend**, the second row the associated **daily returns**, and the bottom row shows a **volatility estimation** calculated from a 5-days moving window over the daily returns.

Preliminaries: Data & Models Inputs

- Classical volatility forecasting models can be divided into:
  - Those that solely use the daily returns to perform the day ahead volatility forecast.
  - Those who also take advantage of the realised measures to do so. The realised variance, a proxy measure of the volatility, is obtained as follows:

$$RV_t = \sum_{i=1}^{I} r_{i,t}^2, \tag{32}$$

where $r_{i,t}$ is the i-th intraday return for day t, obtained through Eq. (31).

- Trade-off: we need to **maximise the usage of high-frequency data while minimizing the microstructure-noise implicit to higher sampling frequencies**: a 5-minutes granularity is usually accepted as the optimal value for the realised measures calculation [32].

Preliminaries: Baseline Models

- Generalized Autoregressive Conditional Heteroscedastic (GARCH) model [26] operates under the assumption of volatility clustering [33], presuming that large shocks in the price tend to cluster together.
- A GARCH(p, q) process can be characterised as follows:

$$\sigma_t^2 = \omega + \sum_{i=1}^{p} \alpha_i \varepsilon_{t-i}^2 + \sum_{j=1}^{q} \beta_j \sigma_{t-j}^2, \tag{33}$$

where p is the order of the observed volatility, σ_t^2; and q is the order of the innovations, ε_t. In turn, the returns are related to the innovations as:

$$r_t = \mu + \varepsilon_t, \tag{34}$$

being μ is the mean of the process (usually set to zero). The volatility is defined from the innovations as:

$$\varepsilon_t = \sigma_t e_t, \quad e_t \sim \mathcal{N}(0, \sigma_t^2). \tag{35}$$

- The model's parameters $\{\mu, \omega, \alpha, \beta\}$ can be estimated by maximising the log-likelihood.

Preliminaries: Baseline Models

- GARCH models lead to several variations:
  - Fractionally Integrated GARCH (FIGARCH) process [34]:

$$\sigma_t^2 = \omega + \left[1 - \beta L - \phi L(1-L)^d\right] \varepsilon_t^2 + \beta \sigma_t^2, \tag{36}$$

    where $0 < d < 1$ is known as the fractional differencing parameters and L is a fractional difference operator (Lag operator).
  - Threshold ARCH (TARCH) models [**?**]

$$\sigma_t^2 = \omega + \sum_{i=1}^{p} \alpha_i \varepsilon_{t-i}^2 + \sum_{j=1}^{q} \beta_j \varepsilon_{t-i}^2 \mathbb{I}_{\varepsilon_{t-i}<0}, \tag{37}$$

  - Exponential GARCH (EGARCH) model [28]:

$$\ln \sigma_t^2 = \omega + \sum_{i=1}^{p} \alpha_i \left(|\varepsilon_{t-i}| - \sqrt{2/\pi} \right) + \sum_{k=1}^{o} \gamma_k \varepsilon_{t-k} + \sum_{j=1}^{q} \beta_j \ln \sigma_{t-j}^2. \tag{38}$$

  - IGARCH, FIGARCH, TARCH, TGARCH, APARCH, etc.

Preliminaries: Baseline Models

- Among proposals including the usage of realised measures, the HEAVY model has proven superior forecasting capabilities [35]:

$$\begin{aligned} \operatorname{var}\left(r_t \mid \mathcal{F}_{t-1}^{\mathrm{HF}}\right) &= \sigma_t^2 = \omega + \alpha \mathrm{RM}_{t-1} + \beta \sigma_{t-1}^2, \\ \mathbb{E}\left(\mathrm{RM}_t \mid \mathcal{F}_{t-1}^{\mathrm{HF}}\right) &= \mu_t = \omega_R + \alpha_R \mathrm{RM}_{t-1} + \beta_R \mu_{t-1}, \end{aligned} \tag{39}$$

  where r_t denotes daily returns, RM_t the daily realised measures, and $\mathcal{F}_{t-1}^{\mathrm{HF}}$ the high-frequency data. Additionally, we observe:

$$\begin{aligned} r_t &= \sqrt{\sigma_t^2 z_t}, \\ x_t &= \mu_t z_{RV,t}^2, \end{aligned} \tag{40}$$

  with:

$$\begin{pmatrix} z_t \\ z_{RV,t} \end{pmatrix} \sim \mathcal{N}(0, I). \tag{41}$$

- HEAVY consists of two parts:
  - σ_t^2 explains the development of the unobserved conditional variance.
  - μ_t is responsible for explaining the development of the realised measures.

DeepVol: Model Definition

- Considering a **set of assets**: $\Delta : (C(\mathbb{R}) \to \mathbb{R}^m), m \in \mathbb{N}$.
- With $T \in \mathbb{N}$ days' **intraday high-frequency data associated** to them: $\{\mathbf{r}_t^{1:J}\}_{t=1}^{T}$, where $\mathbf{r}_t^{1:J} = (r_t^1, r_t^2, \ldots, r_t^J)$ are the intraday returns of the t-th day, and being $J \in \mathbb{N}$ the length of each day's intraday data.
- We aim to **forecast the day-ahead ($T+1$) realised volatility**:

$$\hat{\sigma}_{T+1}^2 = f_\theta \left(r_{t=1}^1, r_{t=1}^2, \ldots, r_{t=1}^J, r_{t=2}^1, \ldots, r_{t=T}^1, \ldots, r_{t=T}^J \right), \tag{42}$$

where $f_\theta(\Delta) \to \mathbb{R}^m$ is a function implemented through a Dilated Causal Convolutions-based neural network, with $\theta \in \Theta$ the learnable parameters of the model from a set $\Theta \in \mathbb{R}^n$, for some $n \in \mathbb{N}$:

$$\hat{\theta} = \underset{\theta \in \Theta}{\operatorname{argmin}} \, \mathcal{L} \left(f_\theta(\Delta), \sigma_t^2(\Delta) \right), \tag{43}$$

DeepVol: Dilated Causal Convolutions

- DeepVol uses DCCs as a technique to integrate the high-frequency information into the realised volatility prediction.
- DeepVol consists of L convolutional layers. The convolution operation performed by the first layer, between the input sequences x and the kernel k, can be defined as follows:

$$F^{(l=1)}(t) = \left(x *_d k^{(l)}\right)(t) = \sum_{\tau=0}^{s-1} k_\tau^{(l)} \cdot x_{t-d\tau}, \tag{44}$$

  with d being the dilation factor and k the filter size.
- For each of the rest l-th layers, we can define the convolution operation as:

$$F^{(l)}(t) = \left(F^{(l-1)} *_d k^{(l)}\right)(t) = \sum_{\tau=0}^{s-1} k_\tau^{(l)} \cdot F_{t-d\tau}^{l-1}(t). \tag{45}$$

- Each of the layers in this hierarchical structure defines the kernel operation as an affine function acting between layers:

$$k^{(l)} : \mathbb{R}^{N_l} \longrightarrow \mathbb{R}^{N_{l+1}}, 1 \leq l \leq L. \tag{46}$$

DeepVol: Dilated Causal Convolutions

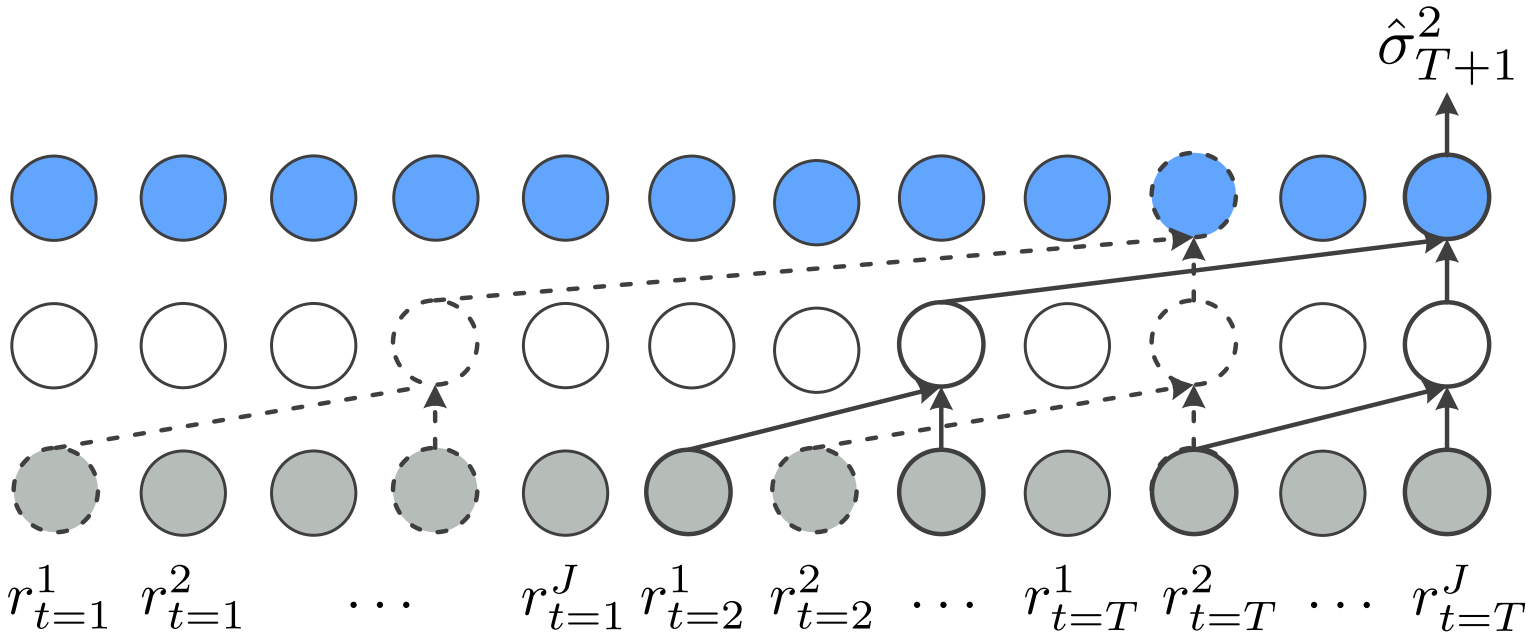


Figure: DeepVol intrinsic architecture. The dilation factor grows exponentially, allowing an increase in the receptive field without increasing the model's complexity.

DeepVol Experiments

- Experiments conducted using **two years of NASDAQ-100 data**, starting September 30, 2019, through September 30, 2021.
- For this specific time frame, **high-frequency data of different sampling frequencies** (granularities), i.e., 1, 5, 15, 30, and 60 minutes, is available and will be used through various experiments.
- **DeepVol** will directly perform its prediction from **raw high-frequency data**, unlike the baseline models, which previously to training require the estimation of daily statistics.

DeepVol Experiments: Evaluation Metrics

$$
\begin{aligned}
\ell_{MAE}(\sigma_t^2, \hat{\sigma}_t^2) &= \frac{1}{T}\sum_{t=1}^{T}\left|\sigma_t^2 - \hat{\sigma}_t^2\right|, \\
\ell_{rmse}(\sigma_t^2, \hat{\sigma}_t^2) &= \sqrt{\frac{1}{T}\sum_{t=1}^{T}(\sigma_t^2 - \hat{\sigma}_t^2)}, \\
\ell_{SMAPE}(\sigma_t^2, \hat{\sigma}_t^2) &= \frac{1}{T}\sum_{t=1}^{T}\frac{\left|\sigma_t^2 - \hat{\sigma}_t^2\right|}{(\sigma_t^2 - \hat{\sigma}_t^2)/2}, \\
\ell_{ME}(\sigma_t^2, \hat{\sigma}_t^2) &= max\left(\left|\sigma_t^2 - \hat{\sigma}_t^2\right|\right), \\
\ell_{MedAE}(\sigma_t^2, \hat{\sigma}_t^2) &= median(\sum_{t=1}^{T}\left|\sigma_t^2 - \hat{\sigma}_t^2\right|), \\
\ell_{QLIKE}(\sigma_t^2, \hat{\sigma}_t^2) &= \frac{1}{T}\sum_{t=1}^{T}\log\left(\hat{\sigma}_t^2\right) + \frac{\sigma_t^2}{\hat{\sigma}_t^2},
\end{aligned}
\tag{47}
$$

where $\hat{\sigma}_t^2$ and σ_t^2 represent the volatility forecast and the volatility proxy measure, respectively, with T the total amount of rolling forecasts.

DeepVol Experiments: Out-of-Sample Forecast
Out-of-Sample Forecast

- NASDAQ-100 dataset splitted into two folds:
  - Training: September 30, 2019, to December 31, 2020 (15 months).
  - Testing: January 1, 2021, to September 30, 2021 (9 months).
- A sampling frequency of 5 minutes for the intraday data and a receptive field of one day (using t-day intraday's data to predict the realised volatility for day $t + 1$) are utilised.

DeepVol Experiments: Out-of-Sample Forecast

Table: Out-of-sample forecast: experiments results for the NASDAQ-100 dataset.

Method	MAE	RMSE	SMAPE	QLIKE	ME	MedAE
martingale	5.180	11.410	0.324	747.480	96.654	**1.614**
TARCH	4.849	10.320	0.301	351.310	71.659	2.804
IGARCH	5.008	10.534	0.302	351.702	72.048	2.797
FIGARCH	4.631	10.356	0.294	349.245	71.050	2.460
APARCH	4.730	10.096	0.299	349.974	**70.088**	2.859
AGARCH	4.833	10.304	0.324	351.217	71.215	2.819
EGARCH	4.793	10.180	0.300	348.614	70.615	2.917
HEAVY	4.565	10.239	0.292	343.490	72.404	2.545
DeepVol	**3.903**	**8.457**	**0.279**	**340.779**	71.779	2.008

DeepVol Experiments: Out-of-Sample Forecast

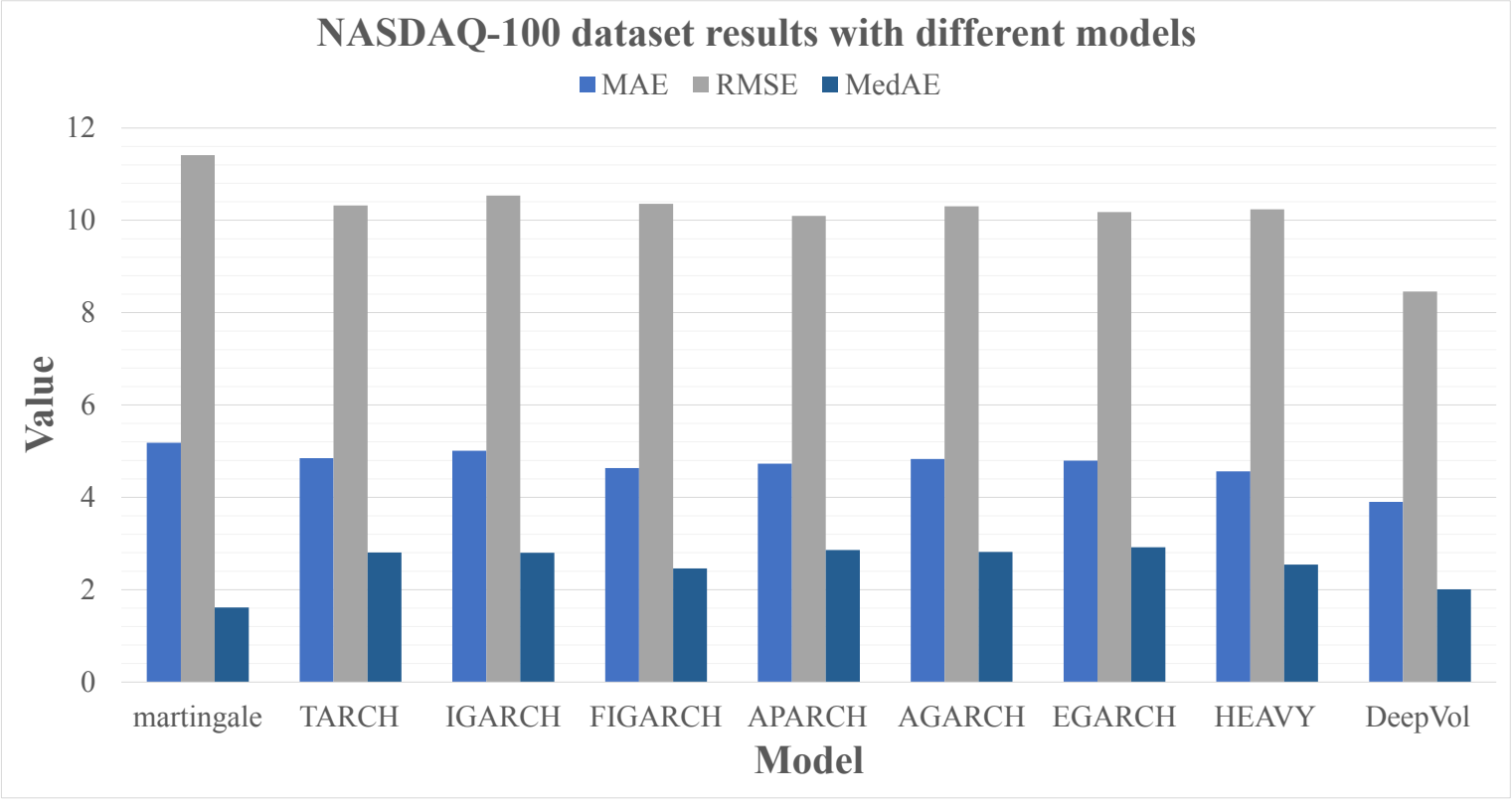


Figure: Out-of-sample forecast comparison.

DeepVol: Experiments
Out-of-Sample Forecast

Table: Out-of-sample forecast: percentage of improvement/degradation over the martingale process and the HEAVY model, for each of the evaluated models.

Method	MAE	RMSE	SMAPE	QLIKE	ME	MedAE
	Improvement over martingale (%)					
martingale	-	-	-	-	-	-
TARCH	6.398	9.553	7.099	53.001	25.860	-73.730
IGARCH	3.320	7.677	6.790	52.948	25.458	-73.296
FIGARCH	10.598	9.238	9.259	53.277	26.490	-52.410
APARCH	8.687	11.516	7.716	53.179	**27.486**	-77.138
AGARCH	6.699	9.693	0.000	53.013	26.319	-74.659
EGARCH	7.471	10.780	7.407	53.361	26.940	-80.731
HEAVY	11.873	10.263	9.877	54.047	25.089	-57.677
DeepVol	**24.653**	**25.881**	**13.889**	**54.410**	25.736	**-24.411**
	Improvement over HEAVY (%)					
martingale	-13.472	-11.437	-10.959	-117.613	-33.492	**36.579**
TARCH	-6.212	-0.791	-3.082	-2.277	1.029	-10.181
IGARCH	-9.704	-2.881	-3.425	-2.391	0.492	-9.906
FIGARCH	-1.446	-1.143	-0.685	-1.675	1.870	3.340
APARCH	-3.614	1.397	-2.397	-1.888	**3.120**	-12.342
AGARCH	-5.871	-0.635	-10.959	-2.250	1.642	-10.771
EGARCH	-4.995	0.576	-2.740	-1.492	2.471	-14.621
HEAVY	-	-	-	-	-	-
DeepVol	**14.502**	**17.404**	**4.452**	**0.789**	0.863	21.097

Generalisation and Transfer Learning Analysis

- Half the tickers of NASDAQ-100 are used during training, while the other half are utilised for testing.
- We evaluate our model's generalisation capabilities, predicting the day-ahead volatility for tickers that were not previously available.

Table: Out-of-sample-stocks forecast: experiments results for the NASDAQ-100 dataset.

Method	MAE	RMSE	SMAPE	QLIKE	ME	MedAE
martingale	8,571	31,422	0,321	2601,648	345,1567	**1,920**
TARCH	7,279	23,379	0,293	1056,421	252,682	2,876
IGARCH	7,733	23,746	0,295	1058,268	254,225	3,026
FIGARCH	7,223	23,399	0,292	1054,647	238,189	2,901
APARCH	7,019	23,165	0,291	1052,759	**237,210**	2,964
AGARCH	7,289	23,430	0,293	1056,927	258,720	2,877
EGARCH	7,207	26,647	0,291	1053,993	272,3615	2,999
HEAVY	7,258	21,139	0,289	**1035,666**	252,183	2,757
DeepVol	**6,218**	**18,410**	**0,286**	1055,625	247,700	2,052

Generalisation and Transfer Learning Analysis

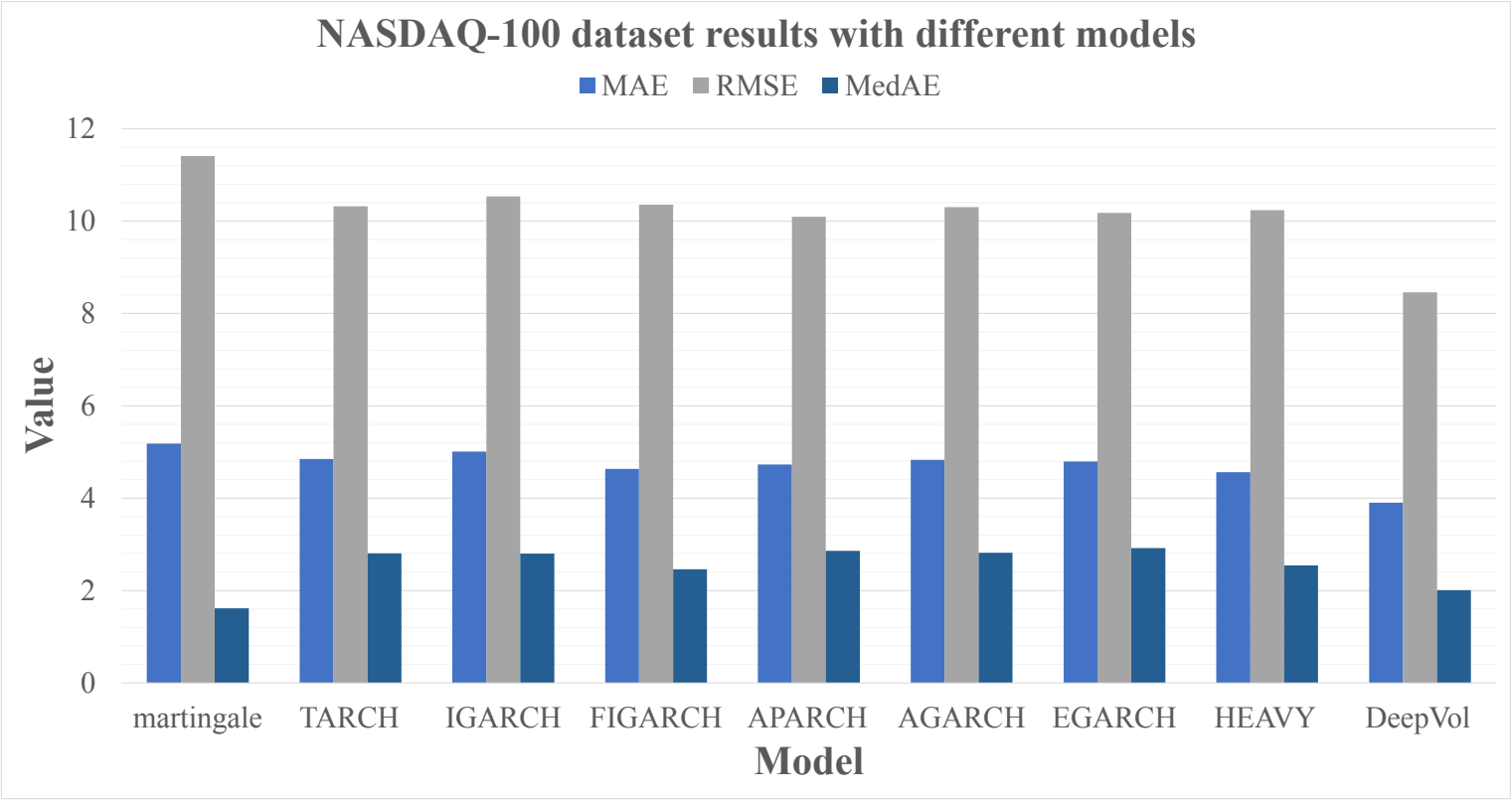


Figure: Out-of-sample-stocks forecast comparison.

DeepVol: Experiments
Generalisation and Transfer Learning Analysis

Table: Out-of-sample-stocks forecast: percentage of improvement/degradation over the martingale process and the HEAVY model, for each of the evaluated models.

Method	MAE	RMSE	SMAPE	QLIKE	ME	MedAE
	Improvement over martingale (%)					
martingale	-	-	-	-	-	-
TARCH	15,074	25,597	8,723	59,394	26,792	-49,792
IGARCH	9,777	24,429	8,100	59,323	26,345	-57,604
FIGARCH	15,727	25,533	9,034	59,462	30,991	-51,094
APARCH	18,108	26,278	9,346	59,535	**31,275**	-54,375
AGARCH	14,957	25,434	8,723	59,375	25,043	-49,844
EGARCH	15,914	18,379	9,346	59,487	21,090	-56,198
HEAVY	15,319	32,725	9,969	**60,192**	26,937	-43,594
DeepVol	**27,453**	**41,410**	**10,903**	59,425	28,235	**-6,875**
	Improvement over HEAVY (%)					
martingale	-18,090	-48,645	-11,073	-151,205	-36.868	**30,359**
TARCH	-0,289	-10,597	-1,384	-2,004	-0,198	-4.316
IGARCH	-6.545	-12.333	-2.076	-2.182	-0.810	-9.757
FIGARCH	0,482	-10,691	-1,038	-1,833	5,549	-5,523
APARCH	3,293	-9,584	-0.692	**-1,650**	**5,937**	-7,508
AGARCH	-0,427	-10,838	-1,384	-2,053	-2,592	-4,353
EGARCH	0,703	-21,326	-0,692	-1,770	-8,002	-8,778
HEAVY	-	-	-	-	-	-
DeepVol	**14,329**	**12,910**	**1,038**	-1,927	1,778	25,571

Discussion of Results

- DeepVol generally outperforms traditional autoregressive architectures.
- It exhibits **quick adaptation to volatility shocks** while showing robustness in their presence.
  - **Experiments** were **conducted in high volatility regimes**, such as the 2020 stock crisis caused by the COVID-19 pandemic.
- **Non-recurrent**: DeepVol only needs previous trading day's data for predicting the day-ahead realised volatility with high accuracy.

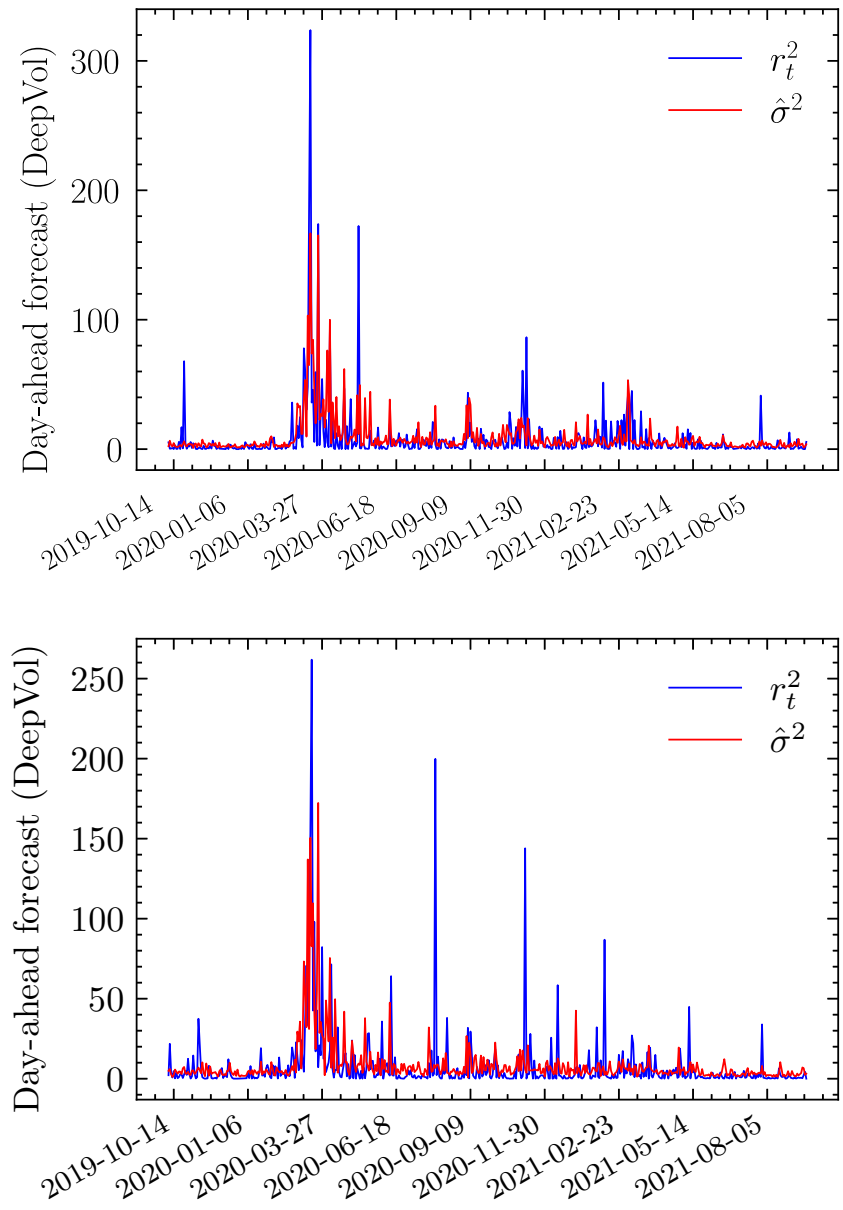


Figure: Out-of-sample-stocks: DeepVol's forecast on PYPL & QCOM.

Conclusions

- DeepVol takes advantage of Dilated Causal Convolutions to **bypass the estimation of the realised measures**.
- It tackles the problem of volatility forecasting from a pure **data-driven perspective**, removing human interaction during the process.
- DeepVol can **exponentially increase its input window** through the usage of dilated convolutions, enabling a similar operational to how handcrafted realised measures condense high-frequency information.
- It **takes advantage of the abundance of stock market intraday's data**, avoiding limitations of classical methods such as:
  - Model misspecification.
  - Hand-crafted noisy realised measures usage.
- DeepVol showed its ability to **extract universal features** and transfer learning to out-of-distribution data.
- The reported empirical results suggest that the proposed deep learning-based reports more accurate predictions than traditional methodologies, leading to more appropriate risk measures.

Thanks!

Papers:
DeepVol: Volatility Forecasting via Dilated Causal Convolutions
[Coming soon!]

stefan.zohren@eng.ox.ac.uk

[1] 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.

[2] 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.

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

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

[5] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

[6] B. Lim, S. Ö. Arık, N. Loeff, and T. Pfister, "Temporal fusion transformers for interpretable multi-horizon time series forecasting," *International Journal of Forecasting*, 2021.

[7] T. Lin, Y. Wang, X. Liu, and X. Qiu, "A survey of Transformers," *arXiv preprint arXiv:2106.04554*, 2021.

[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *arXiv preprint arXiv:1706.03762*, 2017.

[9] S. Li, X. Jin, Y. Xuan, X. Zhou, W. Chen, Y.-X. Wang, and X. Yan, "Enhancing the locality and breaking the memory bottleneck of Transformer on time series forecasting," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 32, pp. 5243–5253, 2019.

[10] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, "Informer: Beyond efficient Transformer for long sequence time-series forecasting," in *The Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Virtual Conference*, vol. 35, pp. 11106–11115, AAAI Press, 2021.

[11] Q. Wen, T. Zhou, C. Zhang, W. Chen, Z. Ma, J. Yan, and L. Sun, "Transformers in time series: A survey," *arXiv preprint arXiv:2202.07125*, 2022.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] A. Garg, C. L. Goulding, C. R. Harvey, and M. Mazzoleni, "Momentum turning points," *Available at SSRN 3489539*, 2021.

[17] R. P. Adams and D. J. MacKay, "Bayesian online changepoint detection," *arXiv preprint arXiv:0710.3742*, 2007.

[18] C. K. Williams and C. E. Rasmussen, "Gaussian processes for regression," 1996.

[19] M. Rolínek, V. Musil, A. Paulus, M. Vlastelica, C. Michaelis, and G. Martius, "Optimizing Rank-based Metrics with Blackbox Differentiation," *arXiv:1912.03500 [cs, stat]*, Mar. 2020. arXiv: 1912.03500.

[20] D. Poh, B. Lim, S. Zohren, and S. Roberts, "Building Cross-Sectional Systematic Strategies by Learning to Rank," *The Journal of Financial Data Science*, pp. 70–86, Mar. 2021.