Transfer Ranking in Finance: Applications to Cross-Sectional Momentum with Data Scarcity

Daniel Poh  
Department of Engineering Science,  
Oxford-Man Institute of Quantitative Finance, University of Oxford  
Oxford, United Kingdom  
dp@robots.ox.ac.uk

Stephen Roberts  
Department of Engineering Science,  
Oxford-Man Institute of Quantitative Finance, University of Oxford  
Oxford, United Kingdom  
sjrob@robots.ox.ac.uk

Stefan Zohren  
Department of Engineering Science,  
Oxford-Man Institute of Quantitative Finance, University of Oxford  
Oxford, United Kingdom  
stefan.zohren@eng.ox.ac.uk

ABSTRACT
Modern cross-sectional strategies incorporating sophisticated neural architectures outperform traditional counterparts when applied to mature assets with long histories. However, deploying them on instruments with limited samples generally produces over-fitted models with degraded performance. In this paper, we introduce Fused Encoder Networks – a hybrid parameter-sharing transfer ranking model which fuses information extracted using an encoder-attention module from a source dataset with a similar but separate module operating on a smaller target dataset of interest. This approach mitigates the issue of models with poor generalisability. Additionally, the self-attention mechanism enables interactions among instruments to be accounted for at the loss level during model training and inference time. We demonstrate the effectiveness of our approach by applying it to momentum strategies on the top ten cryptocurrencies by market capitalization. Our model outperforms state-of-the-art benchmarks on most measures and significantly improves the Sharpe ratio. It continues to outperform baselines even after accounting for the high transaction costs associated with trading cryptocurrencies.

KEYWORDS
Deep Learning, Transfer Learning, Information Retrieval, Learning to Rank, Neural Networks, Data Scarcity, Factor Investing, Cross-sectional Strategies, Portfolio Construction, Cryptocurrencies

1 INTRODUCTION
Cross-sectional strategies are a popular class of trading strategies that involve first sorting assets by some ranking criteria and then trading the winners against losers. Since the seminal work of [44] documenting the strategy and its performance trading US equities, numerous technical refinements have been introduced. For instance, the ranking step, which is at the core of the model, evolved from being performed with heuristics [44] to involving more sophisticated DL (Deep Learning) prediction models [37, 46] and modern ranking algorithms such as LambdaMART [14, 66, 79].

Fundamental to calibrating these models is the availability of sufficient data. While this is somewhat mitigated for DL-based strategies operating in the high-frequency domain or for mature assets with longer trading histories, deploying these strategies on newer instruments is challenging due to their small sample sizes, and their degraded performance will likely result in substantial trading losses. Cryptocurrencies, which have recently seen broad interest in academia and industry, are an example of such a class of instruments.

This dilemma reflects the broader challenge of training DL models that generalise well when confronted with limited data. Although numerous approaches (reducing model complexity [12], few-shot learning [81] and data augmentation techniques [33, 72]) exist for tackling this issue, Transfer Learning stands out as a promising methodology. Transfer Learning focuses on transferring the knowledge obtained across different but related source or upstream domains in order to boost the performance of a decision-making function on some target or downstream domains (we will use ‘source’ and ‘upstream’ interchangeably, the same applies to ‘target’ and ‘downstream’) of interest [29, 87]. It is a broad and active research area, and given the nuances that different problems entail, it overlaps several fields such as domain adaptation and multi-task learning. The multitude of transfer learning successes in computer vision [47, 71] and NLP (natural language processing) [30] tasks have boosted its popularity both as a go-to solution for knowledge transfer problems, as well as a research topic in its own right.

In the space of IR (information retrieval), transfer learning and its closely associated fields have been documented to significantly improve the accuracy of sorting algorithms in target domains with limited samples [20, 21, 35, 52]. However, unlike the computer vision and NLP communities, which have rapidly adopted transformer-based solutions in response to their domain-specific transfer learning problems, we note that a corresponding development in IR is surprisingly absent. Inspired by recent works of [65] and [63] which exploit the Transformer’s attention module to improve ranking performance as well as a first adaptation in the context of finance [67], we similarly make use of this mechanism and propose the Fused Encoder Networks (FEN). Central to the model’s architecture is the usage of dual encoder blocks – one of which is optimised on a larger but related upstream dataset and then deployed in conjunction with the other encoder block calibrated on a smaller downstream dataset of interest. We focus on applying our transfer ranking model to cryptocurrencies, where relatively short trading histories (relative to more mature financial instruments) and recent institutional interest promote them as a practical use-case. Applying cross-sectional momentum across ten cryptocurrencies with the largest market capitalisation, we provide evidence of the superiority of our approach over a set of benchmarks which includes state-of-the-art

Domains adaptation is used when the source and target datasets belong in different domains [78]. Multi-task learning is used when the goal is to jointly learn a group of related tasks by exploiting the interconnections between tasks [87].
LTR (Learning to Rank) algorithms. We obtain a threefold boost in the Sharpe ratio over risk-adjusted momentum and a gain of around 50% over the best baseline model. We structure the rest of the paper as follows: The next section discusses related works applying transfer learning to finance and ranking tasks. We next move on to sections that define the problem and describe the proposed model’s key elements. We subsequently evaluate the results of our study and finally conclude this work.

2 RELATED WORKS

2.1 Transfer Learning in Finance

Transfer learning focuses on producing an effective model for a target task with scarce training samples by transferring knowledge across different but related source domain(s) [29, 87]. It is an active area of study in machine learning, enjoying success and widespread adoption in various applications [11, 28, 31, 48]. Given the richness of the literature, numerous categorisation schemes have been established to distinguish the diverse yet interconnected strands of transfer learning research. For instance, the work in [61] states that depending on the availability of label information, problems in transfer learning can be categorised as transductive, inductive, and unsupervised.

Alternatively, depending on the source and target domains represented in similar feature or label spaces, transfer learning can be loosely classed as either homogeneous or heterogeneous [87]. Using the schema in [61], transfer learning techniques divide into four groups: instance-based, feature-based, parameter-based, and relational-based approaches. Despite these attempts to provide a comprehensive taxonomy of the research landscape, several terminological inconsistencies persist – for example, phrases such as domain adaptation and transfer learning are used interchangeably to refer to the same processes [82]. We point our readers to [61], [82], and [87] for a better understanding of this broad field.

In the context of transfer learning for finance, most works focus on trading and employ an inductive approach with a parameter-based focus – incorporating the source architecture’s weights or model components into the target model to optimise the task of interest. In one of the earliest examples of transfer learning in finance, [86] propose DeepLOB, a hybrid deep neural model that predicts stock price movements using high-frequency limit order data. The authors document DeepLOB’s ability to extract universal features by showing that it can successfully predict the direction of price movement of a target set of stocks that are not part of the source training set. [60] develops the RIC-NN (Rank Information Coefficient Neural Net) – a multi-factor approach for stock price prediction using deep transfer learning. They demonstrate that the lower layers of the optimised model’s weights in one region (represented by MSCI North America) can initialise the same model for prediction in a different region (MSCI Asia Pacific). In a similar vein but taking on a broader coverage, [49] propose QuantNet, which fuses information by training on different markets with a global bottleneck shared across multiple autoencoder-based architectures. The model then employs a separate decoder for each target market to generate market-specific trading strategies. The work in [45] develops a deep Q-learning reinforcement learning trading agent that they combine with pre-trained weights to prevent over-fitting from insufficient financial data. Concentrating on financial time-series forecasting, [41] tackles the poor performance of training deep learning models on limited data with multi-source transfer learning. In a risk-management setting but utilising a similar pre-trained weights approach, [73] investigate the utility of knowledge transfer methods in evaluating credit risk.

Unlike earlier studies applying transfer learning to the problems in finance, ours differ in a few key aspects. Firstly, [49] and [60] primarily concentrate on transferring knowledge and do not address model overfitting. Additionally, while [45] and [41] are closer in spirit to our work (i.e., they aim to alleviate overfitting that surfaces when calibrating on the target task with limited samples), both do not explicitly connect their research to momentum effects.

Although momentum effects have been documented on numerous asset classes [2, 57, 58, 64], and with numerous sophisticated prediction models developed to enhance the profitability of momentum-based strategies [46, 53, 84], the literature on pre-training these advanced models and applying them to settings where the dataset is small, is scarce. With cryptocurrencies, numerous studies (e.g., [8] and [39]) apply sophisticated machine learning for momentum-based trading. However, few, if any, explore augmenting these trading models by transferring knowledge from different datasets and examining their application over lower frequency settings, such as over weeks (which exacerbates the data scarcity problem). We address this deficiency in the research literature with our work in this paper.

2.2 Transfer Ranking

Transfer Ranking (TR) is defined as the application of transfer learning to LTR algorithms [51, 54]. While a plethora of TR algorithms have been developed to address the extensive nature of ranking tasks [19], the primary objectives of these algorithms are essentially similar, i.e., to boost the ranking model’s accuracy on a new collection of items (which we refer to as the target or downstream dataset) by leveraging labelled data from another collection (source or upstream dataset) [51]. From the literature classification in [51], TR techniques are either supervised or unsupervised, with each category further divided into homogeneous or heterogeneous depending on the similarity between the source and target feature spaces. Given that our work resembles model adaptation within the space of homogeneous supervised TR techniques, we review works related to this form and point the reader to [51] for details on other setups.

Model adaptation TR concentrates on adapting the existing source model with a few labelled training data from the target collection. Most algorithms concentrating on this form treat the
source ranker as a base model or exploit its as a feature for training [51]. These methods respectively resemble the parameter- and feature-based transfer learning methods in the scheme proposed by [61]. The work in [34], for instance, explores two classes of model adaptation techniques for web search ranking: (i) model interpolation and (ii) error-driven learning approaches based on boosting. The first method learns an ensemble involving both source and target models; In the second, the authors propose LambdaBoost and LambdaSMART [15], which at each boosting iteration adapt the source model to target data (the former searches the space of finite basis functions, while the latter searches the space of regression trees). Other research directions involve calibrating the source model to better align with the target data. For example, [22] proposes Trada, which is a tree-based model that tunes both weights and splits based on a combination of source and target data. Trada is further improved by incorporating pairwise information to refine the model’s adaptation process [3]. Lastly, [77] adopts a different approach by coefficient transformation. Concretely, the authors develop an algorithm that generates personalised results by learning a transformation matrix that manipulates the weights of a global ranker.

Surveying the literature surrounding model adaptation TR, solutions employing trees are common. This development is likely owing to their flexibility and robustness, which has been validated by the performant tree-boosting ranking algorithm, LambdaMART [14]. While recent LTR breakthroughs, such as [68] and [65] provide clear evidence that transformer-based rankers can outperform their tree-based counterparts, we note a lack of models employing this architecture in TR. To this end, we propose the Fused Encoder Networks (FEN) – a novel and hybrid parameter-sharing transfer ranking model inspired by the Transformer. The FEN fuses and leverages information flexibly extracted by a pair of encoder-attention modules operating on both source and target domain data. With this design, the model inherits the advantages of self-attention and can account for dependencies across instruments at the loss level during training and inference time. By incorporating the components of the source model in the FEN model, the networks then determine the optimal combination of source and target information for the ranking task.

3 PROBLEM DEFINITION

This section formalises the problem setting. To get a better idea of the LTR framework as applied to cross-sectional momentum and how TR builds on top of LTR, please refer to the section titled Learning to Rank and Transfer Ranking in the Appendix.

3.1 Linking Returns and Model Formulation

Given a portfolio of cryptocurrencies that is rebalanced weekly, the returns for a cross-sectional momentum (CSM) strategy at week $t$ can be expressed as follows:

$$ r_{t, t+1}^{CSM} = \frac{1}{n} \sum_{i=1}^{n} S_t^{(i)} (\sigma_{tgt}^{(i)}) S_{t+1}^{(i)} $$

where $r_{t, t+1}^{CSM}$ refers to the portfolio’s realised returns over weeks $t$ to $t+1$, $n$ denotes the number of cryptocurrencies and $S_t^{(i)} \in \{-1, 0, 1\}$ characterises the CSM signal or trading rule for instrument $i$. We fix the annualised volatility target $\sigma_{tgt}$ at 15% and scale returns with $\sigma_t^{(i)}$, which is an estimator for ex-ante weekly volatility. The latter is computed as the rolling exponentially weighted standard deviation with a 26-week (approximately half a year) span on weekly returns.

The CSM signal $S_t$ in Equation (1) can be obtained using heuristics [44] or by minimising the empirical loss of some prediction model $f$ parameterised by $\theta$:

$$ L(\theta) = \frac{1}{|\Psi|} \sum_{(x,y) \in \Psi} \ell(y, f(x; \theta)) $$

With slight abuse of notation, we take each item $x_i$ to be a k-sized real-valued feature vector. Since $x$ is a list of $n$ items, we have $x \in \mathbb{R}^{n \times k}$. The training data is denoted as $\Psi = \{(x, y) \in \mathbb{R}^{n \times k} \times \mathbb{Z}^n\}$, where $y$ is a list of $n$ bins for the same $x$ sorted by realised returns for the next period. When the prediction model $f$ adopts a regression-rank approach such as those proposed in Gu et al. [37] and Kim [46], then the loss function $\ell$ is usually the pointwise MSE (mean squared error). With LTR models, pairwise and listwise losses that account for the mutual interaction among objects are usually used for $\ell$.

3.2 Model Formulation under the Transfer Ranking Framework

With parameter-based transfer learning approaches, an upstream or source model is first trained on the source data. This process can be expressed as:

$$ L(\theta_S) = \frac{1}{|\Psi_s|} \sum_{(x,y) \in \Psi_s} \ell(y, f_S(x; \theta_S)) $$

where the terms $f_S$, $\theta_S$ and $\Psi_S$ are respectively the prediction model, its parameter vector, and the associated dataset. The tuned upstream weights are then incorporated into the downstream or target model:

$$ L(\theta_T) = \frac{1}{|\Psi_T|} \sum_{(x,y) \in \Psi_T} \ell(y, f_T(x; \theta_S, \theta_T)) $$

where adhering to the form laid out by Equation (2), $f_T$ refers to the ranking function parameterised by both source and target parameters $\theta_T$ as well as $\theta_T$, and $\ell$ is either a pairwise or listwise loss. In the final fine-tuning step, minimising the loss can be formulated as:

$$ L(\Theta_S, \theta_T) = \frac{1}{|\Psi_T|} \sum_{(x,y) \in \Psi_T} \ell(y, f_T(x; \Theta_S, \theta_T)) $$

and $\Theta_S$ denotes the subset of parameters within $\theta_S$ that is permitted to be fine-tuned. Using this final model, we compute ranking scores which we then sort and threshold to obtain the cross-sectional trading signal $S_T$ required for portfolio construction. Figure 1 visualises the TR pipeline and compares it to traditional LTR.

4 FUSED ENCODER NETWORKS

Optimising the FEN (Fused Encoder Networks) involves training a source model on a larger dataset. The calibrated components are subsequently shared with the target model (i.e., FEN) as part of a
Figure 1: Pipelines for traditional LTR (top) and Transfer Ranking (bottom): The latter involves first training a source (upstream) model on a different but related dataset and then transferring the knowledge by sharing the calibrated weights/components with a target (downstream) model. Apart from requiring an additional fine-tuning step, the training process for the target model follows that of traditional LTR.

Figure 2: Architecture of a single Encoder Layer (left) as well as the FEN (Fused Encoder Networks) (right). Each single Encoder Layer comprises a multi-head attention module and a feed-forward network and uses dropouts and layer normalisation to learn higher-order feature representations. The FEN comprises both the source and target encoder blocks running in parallel. FC and the □ symbol represents a fully connected layer and the concatenate operator respectively. The FC block with the dashed outline is optional and is used when the target and source inputs differ in dimensions.

broader framework. The following section provides vital details and explains how various components come together.

4.1 Source Model Architecture

We use a stacked encoder block with a linear projection applied to inputs (similar to those used in Pobrotyn et al. [65] and Poh et al. [66]) as our source model. Each encoder within the stack is a composite layer housing the attention mechanism and a feed-forward
network that enables it to learn higher-order feature representations. It is an essential component in both the source and target models (See left of Figure 2).

4.2 Feed-forward Network

This component introduces non-linearity via its ELU (Exponential Linear Unit) [26] activation. It also facilitates interactions across different parts of its inputs.

4.3 Self-Attention and Multi-head Attention

The attention mechanism is a function that maps a query and a key-value pair to an output. While a few variants of attention exist, we use the form similar to that employed in the Transformer, which is the scaled dot-product attention followed by a softmax operation [76]:

$$\text{Att}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_{\text{model}}}}\right)V$$ (6)

where Q, K and V are respectively the query, key and value matrices. In self-attention, the same inputs are used for all three matrices. Combining multiple attention units produces the multi-head attention module, which boosts the model’s capacity to learn representations:

$$\text{MHA}(Q, K, V) = \text{concat}(\text{head}_1, ..., \text{head}_h)W^O$$ (7)

$$\text{head}_i = \text{Att}(QW^Q_i, KW^K_i, VW^V_i)$$ (8)

where in the above equations, each \(\text{head}_i\) (out of \(h\) heads) refers to the \(i\)th attention mechanism of Equation (6), and learned parameter matrices:

$$W^Q_i \in \mathbb{R}^{d_{\text{model}} \times d_k}, \quad W^K_i \in \mathbb{R}^{d_{\text{model}} \times d_k}, \quad W^V_i \in \mathbb{R}^{d_k \times d_{\text{model}}}$$

where typically \(d_k = d_k = d_o = d_{\text{model}}/h\) is used.

4.4 Stacking Encoder Layers

A single encoder block \(\xi(\cdot)\) can be expressed as:

$$\xi(x) = \Gamma(z + \delta(\phi(z)))$$ (9)

$$z = \Gamma(x + \delta(\text{MHA}(x)))$$ (10)

where \(\Gamma(\cdot), \delta(\cdot)\) and \(\phi(\cdot)\) refers to the layer normalisation operation, a dropout function, and a projection onto a fully-connected layer respectively. Additionally, \(\text{MHA}(\cdot)\) represents the multi-head attention module. Stacking multiple encoder layers enables the model to learn more complex representations:

$$\xi(x) = \xi_1(...(\xi_N(x)))$$ (11)

where \(\xi(x)\) in Equation (11) characterises a stacked encoder block involving a series of \(N\) encoder layers.

4.5 Target Model Architecture

With conventional parameter-sharing techniques, if a neural model is used to learn the source task, then a target model can be constructed by directly retaining most of its layers [87]. Motivated by the empirical superiority of context-aware LTR models using the attention mechanism over standard rankers [65, 67], we utilise a hybrid approach – running the pre-trained source Transformer’s encoder block \(\xi_S(\cdot)\) as an additional feature extractor operating in parallel with the target Transformer’s block \(\xi_T(\cdot)\):

$$\xi_S(x) = \xi_S(...(\xi_{SM}(x)))$$ (12)

$$\xi_T(x) = \xi_T(...(\xi_{TN}(x)))$$ (13)

where \(\xi_{SM}\) and \(\xi_{TN}\) are respectively the \(i\)th and \(j\)th encoder layer from the source and target model (See the right of Figure 2). Note the additional full-connected block with the dashed outline – this is optional and is used when the target and source inputs differ in dimensions. Against standard parameter-sharing methods, our setup has the additional advantage of being plug-and-play, allowing to be appended anywhere to any pre-trained network. The outputs from both encoder blocks are concatenated:

$$\xi_{S,T}(x) = \text{concat}(\xi_S(x), \xi_T(x))$$ (14)

before projecting onto a fully connected layer to obtain our raw score predictions \(f(x)\). Although we use a simple projection for simplicity, this can be replaced by more complex processing such as those that involve deep or cross networks [80].

4.6 Loss Function

We optimise both the source and target components with the ListNet loss [16] and note that other losses such as ListMLE [85] or metrics like ApproxNDCG [13] can be used. The ListNet loss is defined as:

$$\ell(y, f(x)) = -\sum_i \text{softmax}(y)_i \times \log(\text{softmax}(f(x))_i)$$ (15)

4.7 Target Model Fine-tuning

Deep networks can be fine-tuned in several ways. A straightforward approach is to optimise all the parameters of the model after initialising it with the pre-trained model’s parameters. Another method, driven by the findings that learned features transition from general to specific along a network [56], concentrates on re-training only the last few layers while leaving the parameters of the remaining initial layers frozen (at their pre-trained values). There are also other methods, such as those proposed by Girshick et al. [36] and Razavian et al. [69]. As the number of layers to freeze is a manual design choice that introduces unnecessary complexity, we do not freeze any layers and instead focus on re-training the entire model with a low learning rate.

5 PERFORMANCE EVALUATION DETAILS

5.1 Dataset Overview

For actual performance evaluation, all models utilise daily closing prices obtained from CoinMarketCap [27] further downsampling to the weekly frequency. Our universe is defined as the top 10 cryptocurrencies by market capitalisation as of the end of Dec-2019, with at least four years of price history. Given the limited size of the data, this selection ensures sufficient samples for model training and drawing meaningful inferences. We fix this set of cryptocurrencies from the end of Dec-2019 to the end of Dec-2021 to minimise the impact of survivorship bias. The complete data set thus runs from 1-Jan-2016 to 31-Dec-2021, with 2018 and 2021 being the first and last target test years respectively.
To train the source model needed for knowledge transfer, we make use of the same set of daily data relating to 30 currency pairs as per Baz et al. [6] and Poh et al. [67] obtained from the Bank for International Settlements (BIS) [5] spanning May-2000 to Dec-2021 which we again downsample to the weekly frequency. To measure risk aversion, we use the daily close of the VIX historical data from the Cboe Global Markets [17], where a week is labelled risk-off if it contains one or more days when the VIX is 5% higher than its 60-day moving average. A normal week is thus a non-risk-off week. For further details about the data, we refer the reader to Dataset Details in the Appendix.

5.2 Strategy and Predictor Descriptions

Our portfolios are rebalanced weekly. We construct an equally weighted long/short portfolio involving the top/bottom two cryptocurrencies as ranked by the model and calculate returns using Equation (1).

We note that there are also other approaches for constructing the cross-sectional portfolio. Within the cryptocurrency literature, our method resembles Tzouvanas et al. [74], who forms a self-financed long/short strategy based on a risk-adjusted momentum portfolio. Given that volatility-scaling is a common feature in time-series momentum [4, 59], it is plausible that our approach captures similar effects to this class of strategies. Liu et al. [55] adopt a double-sort approach – first sorting based on market capitalisation, and then by three-week momentum. The final portfolios are located at the intersection of this two-step process and are traded without volatility-scaling. This approach resembles the momentum factor portfolio of Fama and French [32] in the asset pricing research. Finally, in the equity literature, there is Jegadeesh and Titman [44] who trade equally-weighted winner and loser decile portfolios with no risk scaling.

We use a simple set of returns-based features for predictors across both source and target models. The source model is trained with the following features:

1. **Raw returns** – Raw FX returns over the past 1, 2, 3, and 4 weeks, using the findings of Menkhoff et al. [58].
2. **Normalised returns** – Raw FX returns over the previous 1, 2, 3, and 4 weeks standardised by weekly volatility scaled to the appropriate time scale.

For the target model, we again use both raw and normalised cryptocurrency returns but only over the past 1, 2, and 3 weeks as evidence of significant momentum returns in cryptocurrencies is mixed using a 1-month formation period [74].

5.3 Models and Comparison Metrics

1. **1-week Returns (1WR)** – This is risk-adjusted momentum strategy based on the findings of Tzouvanas et al. [74]. Unlike the original approach, we do not form overlapping portfolios but instead construct weekly long/short portfolios based on returns over the previous week. The final portfolio is scaled to the desired target volatility.
2. **Multi-Layer Perceptron (MLP)** – Characterises the typical Regress-then-rank model used by contemporary strategies.
3. **ListNet (LN)** – Listwise Learning to Rank model proposed by Cao et al. [16].
4. **LambdaMART (LM)** – Pairwise Learning to Rank model proposed by Burges [14]. This model is based on Gradient Boosted Decision Trees (GBDT) and is still regarded as the state-of-the-art for LTR problems in recent IR studies [68].
5. **Self-attention-based Ranker (SAR)** – This benchmark uses the Transformer’s encoder module for ranking and is applied directly to the cryptocurrency data. The use of the encoder as a scoring function for LTR was proposed by Pobroyn et al. [65]. With this approach, they establish state-of-the-art performance on the WEB30K LTR dataset and report substantial performance gains over the well-known and high performing LambdaMART. This idea was later applied to constructing cross-sectional strategies by Poh et al. [67].
6. **SAR with Parameter-Sharing (SAR+ps)** – This involves pre-training the Transformer’s encoder module on the larger FX dataset and then fine-tuning on the target cryptocurrency set. Fine-tuning pre-trained models has been documented directly to the cryptocurrency data. The use of the encoder as a scoring function for LTR was proposed by Pobroyn et al. [65]. With this approach, they establish state-of-the-art performance on the WEB30K LTR dataset and report substantial performance gains over the well-known and high performing LambdaMART. This idea was later applied to constructing cross-sectional strategies by Poh et al. [67].
7. **Fused Encoder Networks (FEN)** – The proposed model.

We assess the models over four key areas (profitability, riskiness, risk-adjusted performance, and ranking accuracy) with the following metrics:

- **Profitability**: Expected returns (\(\mathbb{E}[\text{Returns}]\)) and Hit rate (percentage of positive returns at the portfolio level obtained over the out-of-sample period).
- **Risks**: Volatility, Maximum Drawdown (MDD), and Downside Deviation.
- **Financial Performance**: Sharpe, Sortino, and Calmar ratios are used as a gauge to measure risk-adjusted performance. We also include the average profit divided by the average loss.
- **Ranking Performance**: NDCG@2. This is based on the Normalised Discounted Cumulative Gain (NDCG) [43], a measure of graded relevance. We focus on assessing the NDCG of the top/bottom two cryptocurrencies, which are linked directly to strategy performance.

5.4 Training and Backtest Details

**Benchmark Models:**

Except for 1WR and LM, all models are tuned with minibatch stochastic gradient descent with the Adam optimiser. All machine-learning-based models are calibrated using an expanding window approach in one-year blocks. Using the first iteration as an example: we retain 90% of the 2016-2017 period for training and use the remainder for cross-validation, with 2018 held out for testing. In

---

\(^1\)WEB30K or MSLR WEB30K (Microsoft Learning to Rank Datasets-30k) is a large scale dataset released by Microsoft that is used for researching and benchmarking LTR algorithms.

\(^2\)Unlike Poh et al. [67], we use the variant without positional encodings as we are not performing re-ranking.
the next iteration, the training set is expanded to include 2018 (i.e., 2016 to 2018), and the model is tested using 2019 (See also Figure 3). Backpropagation was conducted for a maximum of 100 epochs and with an expanding window approach, where 90% of a given training block is used for training and the remainder for validation. Early stopping was used to prevent over-fitting and is triggered when the validation loss does not improve for 25 epochs. Model hyperparameters were tuned over 50 iterations of search using HyperOpt [10].

Further details relating to model calibration can be found in Additional Training Details of the Appendix.

**Fused Encoder Networks:**

The FEN is optimised in three steps. The upstream model is first calibrated on the source FX data. This is performed with a one-year expanding window approach where we test our strategy on a given year and use all preceding samples for model tuning with a 90%/10% train/validation split. For the next year, we expand and shift the training and test periods respectively.

For the next step, the source model’s encoder block is transferred to the FEN and trained on the target data with the source block frozen. Like source model training, the FEN is trained using a single-year expanding window with a 90%/10% split. To minimise any detrimental effects arising from negative transfer, which is the case when the transferred source knowledge adversely impacts performance on the downstream task, we first proceed as follows: to generate predictions for each out-of-sample year $t$ on the target task, we produce ten runs of the source models evaluated on $t - 1$. We pick the source model with the highest Sharpe ratio and use it to train the target model (as we evaluate model performance (discussed later) based on ten runs, we pick the top two source models and go on to train five target models with each.). This process is illustrated in Figure 3. The consequences of transferring knowledge from models with the lowest Sharpe ratios can be found in Incorporating Knowledge from Sub-optimal Source Models in the Appendix.

The final fine-tuning phase optimises the unfrozen network with a very low learning rate — set as the smallest value in the search space for learning rates. The model is run for a maximum of 100 epochs and is stopped early if the validation loss fails to improve for ten epochs. We fine-tune only the FEN and SAR+ps. Further details can be found in Additional Training Details in the Appendix.

## 6 Results and Discussion

### 6.1 Trading Performance

Table 1 consolidates trading performance across the various strategies after ten runs††. While we compute the statistics here without transaction costs to focus on raw performance, we consider the impact of different cost assumptions later in Turnover Analysis. An additional layer of volatility scaling is applied at the portfolio level to facilitate comparison. Where applicable, each bold and asterisked figure indicate the best measurement across all models for its respective performance measures.

The statistics presented in Table 1 demonstrate the superiority of the FEN model — delivering an approximately threefold boosting of the Sharpe ratio relative to the heuristics-based 1WR and an improvement of about 50% versus the best benchmark (SAR). From a risk perspective, FEN is among the least volatile and possesses the lowest downside deviation and max drawdown figures. We also note that the instability associated with cryptocurrencies is reflected across all models running at levels higher than the 15% annual target.

Focusing on the reference baselines, SAR is the best model, while LN is the worst. While SAR’s performance is likely a result of its sophisticated underlying model, it is only marginally better than the tree-based LM. This observation is reasonable since SAR and LM have produced state-of-the-art results on different LTR tasks [65, 68]. Additionally, the consequence of over-fitting is reflected in LN’s results — which are significantly worse than the simpler 1WR model. While both SAR and SAR+ps share the same transformer-based architecture, the latter’s inferiority suggests that only relying on a network pre-trained on FX data is insufficient in the context of this work.

In FEN, the pre-trained network runs in parallel with the target encoder block. This setup leads to the model behaving like an ensemble of (potentially shallow) networks. Specifically, this setup avoids depending solely on the pre-trained module — which can be catastrophic for performance when affected by negative transfer (See Incorporating Knowledge from Sub-optimal Source Models in the Appendix) — and instead allows extracted features to be flexibly combined in a data-driven manner.

### 6.2 Ranking Performance

We assess ranking precision with the NDCG (Normalised Discounted Cumulative Gain), a popular measure in the IR literature, with higher values implying greater accuracy in sorting objects (to match some optimal ordering). Given that we trade the top/bottom quintile each time, we concentrate on the average NDCG@2 for long and short trades aggregated over all weekly rebalances. Table 2 indicates that the MLP can rank the most accurately based on this measure, and this is followed by FEN. Although better accuracy generally implies higher returns and Sharpe ratios as seen in Table 1, this does not hold for the MLP. This inconsistency can be resolved when we group FEN’s returns conditioned on the relative accuracy (over rebalances) of FEN against the MLP††. By segmenting returns and accuracy in this manner, we observe that FEN is only slightly worse off returns-wise when it is less accurate than the MLP but generates considerably higher returns when it is more accurate — which explains its higher overall returns (and Sharpe ratio). These results are presented in Table 3.

### 6.3 Attention Heatmap Patterns

Attention is an integral part of various state-of-the-art Transformer-based architectures, providing these models with an effective way to model relationships by attending to relevant regions in the data. We examine the raw attention heatmaps of our proposed model to determine if this mechanism can learn helpful information. Specifically,
Figure 3: Source and target (FEN) model training. The source and target models are trained using an expanding window of one-year blocks. To minimise any impact of negative transfer, we assess the performance of the upstream model in the year preceding the target test year. We only use models that have performed well for knowledge transfer.

Table 1: Performance metrics for all models over the out-of-sample 2018-2021 period, rescaled to match a 15% annual volatility target. All figures except those belonging to 1WR are computed based on ten runs. FEN outperforms the benchmark models on nearly all measures.

| Benchmarks Proposed | 1WR     | MLP     | LN      | LM      | SAR     | SAR+ps   | Proposed  |
|---------------------|---------|---------|---------|---------|---------|----------|-----------|
| 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±0.035 | 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±0.273 |
| 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 |

Table 2: Ranking performance for all models based on the popular NDCG (Normalised Discounted Cumulative Gain). We calculate the average NDCG@2 over long and short positions across all rebalances. All figures except those belonging to 1WR are computed based on ten runs. Higher accuracy typically translates to higher returns and Sharpe ratios (See Table 1). FEN outperforms most benchmarks in terms of NDCG@2 except for the MLP.

| Benchmarks | 1WR     | MLP     | LN      | LM      | SAR     | SAR+ps   | Proposed  |
|------------|---------|---------|---------|---------|---------|----------|-----------|
| NDCG@2     | 0.580±0.198 | **0.615±0.007** | 0.597±0.010 | 0.601±0.007 | 0.604±0.010 | 0.594±0.007 | 0.607±0.011 |

we visualise the last attention layer of the model’s target encoder stack since deeper layers tend to capture high-level representation and global relationships [62]. Against increasingly coupled markets, volatility spillovers have grown increasingly frequent such as risk transmission from the equity to cryptocurrency markets [75]. Figure 4 depicts the aggregated heatmaps across normal and global risk-off regimes‡‡. Darker cells correspond to larger weights

‡‡We define a risk-off event at the weekly frequency as an instance when the daily VIX over any one or more days within the week is 5% points higher than its 60-day moving average, and all normal states are thus defined to be non-risk-off states. This definition classifies approximately 20% of the target dataset’s evaluation period as risk-off.
Table 3: Segmented returns based on the relative ranking performances of the MLP and FEN models. We calculate the average NDCG@2 over long and short positions across all rebalances. The first two columns compare these measures at the times when FEN has an NDCG@2 lower (i.e., less accurate) than the MLP. The next pair of columns compare the same measures over instances when FEN has a higher NDCG@2. The magnitude of FEN’s gains (when more accurate than the MLP) is larger than its losses (when less accurate).

|                  | MLP more accurate | FEN more accurate |
|------------------|-------------------|-------------------|
|                  | NDCG@2           | Average Returns  |
|                  | 0.648±0.115      | 0.006±0.023       |
|                  | 0.566±0.117      | 0.000±0.020       |
|                  | 0.576±0.108      | -0.004±0.016      |
|                  | 0.648±0.105      | 0.006±0.014       |

which play a more significant role in the model’s prediction. The heatmaps show that the darker cells are more dispersed in risk-off states since multiple instruments are affected. This pattern of dispersion is less evident over normal states and is likely due to idiosyncratic developments in the cryptocurrency markets.

Beyond the impactful macro-level episodes that cause sharp changes to the VIX, we observe similar behaviour of the FEN’s attention mechanism at significant cryptocurrency-specific market events. In the left of Figure 5 for instance, a distinct darker column of cells over Dogecoin formed around Elon Musk’s tweet about the digital token. On the right heatmap, the dispersed region of darker cells reflects the risk-off sentiments inherent in China’s regulators’ intent to crack down on the use of cryptocurrencies. These plots indicate that FEN’s attention module is responsive to market dynamics and is capable of learning valuable information. More importantly, it appears to possess an interpretable structure and provide an additional perspective that helps explain the model’s strong performance against other attention-based benchmarks. We refer the interested reader to the section titled Comparing Attention Heatmaps in the Appendix to see how the heatmaps of other models compare.

To further examine the link between attention values, predicted ranks, and eventual trading positions, we compute the column-wise averages of the attention heatmaps grouped by different market states and aggregated across all runs, with the final result presented in Figure 6. The histogram for each state has values that approximately follow a ‘J’ shape. Higher averages for the upper and lower halves of the ranked list generally coincide with trading positions (long trades are labelled 9 and 10 on the horizontal axis, while shorts are 1 and 2), with the longs’ half of the histogram typically registering higher values. This pattern of higher weights near the ends of the ranked list is similarly observed in the other attention-based models SAR (Figure 11) and SAR+ps (Figure 12) – highlighting the close connection between attention, ranking outcomes, and, eventually, the traded positions generated by these models.

Before proceeding to the next section on turnover analysis, we highlight a limitation of the attention mechanism. Specifically, while it can pick up complex dependencies between the model’s input elements, it is computationally prohibitive – scaling quadratically with the length of the input sequence. Hence, directly using FEN on a broader cross-sectional slate of instruments will likely be challenging. To this end, a few workarounds have been proposed. We also note the recent work of Choromanski et al. who develop Performers, which is not just linear in both time and space complexities but can produce provably accurate estimates of softmax full-rank attention.

### 6.4 Turnover Analysis

To investigate the impact of transaction costs on strategy performance, we first compute the weekly portfolio turnover at period $t$ which we define as:

$$\zeta_t(i) = \sigma_{tgt} \sum_{i=1}^{n} \frac{\hat{S}_t(i)}{\sigma_t(i)} - \frac{\hat{S}_{t-1}(i)}{\sigma_t(i)}$$

(16)

Table 4 shows the FEN model as among the strategies with the lowest average turnover over 2018-2021. To further examine and quantify the impact of transaction cost on performance, we also compute the ex-cost Sharpe ratios and present the results in Table 5. The figures show that FEN can deliver higher and positive cost-adjusted Sharpe ratios up to 29 bps – above the 26 bps per trade assumed for cryptocurrencies and therefore demonstrating its suitability for trading these instruments.

### 7 CONCLUSION

We introduce Fused Encoder Networks (FEN) – a novel hybrid parameter-sharing transfer ranking model that overcomes the difficulties of constructing cross-sectional strategies based on neural rankers to trade instruments with short histories. Using the popular momentum strategy applied to cryptocurrencies as a demonstrative use-case, we validate the effectiveness of our proposed approach. Concretely, the FEN surpasses the reference baselines, which include state-of-the-art LTR algorithms, on most performance measures – notably delivering an approximately threefold boosting of the Sharpe ratio over risk-adjusted momentum and a gain of approximately 50% over the best benchmark. Future work includes modifying various components of the original framework, such as introducing additional processing in the output head or replacing the listwise loss with...
other loss functions. Another direction is to explore transferring and synthesising knowledge from other/multiple source domains.

ACKNOWLEDGMENTS
We would like to thank the Oxford-Man Institute of Quantitative Finance for financial and computing support. SR is grateful for support from the Royal Academy of Engineering.

REFERENCES
[1] Martin Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng.
Transfer Ranking in Finance: Applications to Cross-Sectional Momentum with Data Scarcity

Table 5: Sharpe ratios across selected profitable strategies with a 15% annual volatility target under different transaction cost assumptions. Positive Sharpe ratios for FEN persist up to around 29 bps. This is above the 26 bps per trade assumed for cryptocurrencies and thus demonstrate their suitability for trading these instruments.

| Transaction Costs | 1WR | MLP | LM | SAR | FEN |
|-------------------|-----|-----|----|-----|-----|
| 0 bps             | 0.238 | 0.403±0.240 | 0.456±0.546 | 0.496±0.583 | *0.749±0.410 |
| 5 bps             | 0.115 | 0.273±0.248 | 0.302±0.542 | 0.360±0.580 | *0.621±0.415 |
| 10 bps            | -0.011 | 0.144±0.257 | 0.153±0.532 | 0.225±0.576 | *0.493±0.418 |
| 15 bps            | -0.140 | 0.016±0.267 | 0.11±0.515 | 0.089±0.574 | *0.366±0.422 |
| 20 bps            | -0.271 | -0.111±0.276 | -0.134±0.506 | -0.046±0.571 | *0.238±0.425 |
| 25 bps            | -0.404 | -0.236±0.286 | -0.283±0.504 | -0.180±0.569 | *0.111±0.429 |
| 30 bps            | -0.538 | -0.359±0.297 | -0.433±0.504 | -0.315±0.567 | *-0.017±0.432 |

Figure 6: Column averages of the raw attention heatmaps for FEN across different market states aggregated over multiple runs. Each column average corresponds to a ranking index predicted by the model. Across the three different settings, the bars appear to roughly follow a ‘J’ shape, with higher weights for each half of the ranked list typically coinciding with traded positions (longs are labelled 9 and 10 on the horizontal axis, while shorts are 1 and 2). Values for the ranking index 2 diverge from this pattern.

2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. https://www.tensorflow.org/. Software available from tensorflow.org.

[2] Clifford S. Asness, Tobias J. Moskowitz, and Lasse Pedersen. 2013. Value and Momentum Everywhere. Journal of Finance 68, 3 (2013), 929–985.

[3] Jing Bai, Fernando Diaz, Yi Chang, Zhaohui Zheng, and Keke Chen. 2010. Cross-market model adaptation with pairwise preference data for web search ranking. In Proceedings of the 23rd international conference on computational linguistics: Posters (COLING '10). Association for Computational Linguistics, USA, 18–26. Number of pages: 9. Place: Beijing, China.

[4] Akindynos-Nikolaos Baltas and Robert Kosowski. 2012. Momentum Strategies in Futures Markets and Trend-following Funds. SSRN Electronic Journal (2012). https://doi.org/10.2139/ssrn.1968986

[5] (BIS) Bank for International Settlements. 2022. US dollar exchange rates. Calculated (or Derived) based on BIS data. (2022). https://www.bis.org/statistics/xrusd.htm

[6] Jamil Baz, Nicolas M Granger, Campbell R. Harvey, Nicolas Le Roux, and Sandy Rattray. 2015. Dissecting Investment Strategies in the Cross Section and Time Series. SSRN Electronic Journal (2015). https://doi.org/10.2139/ssrn.2695101

[7] Irwan Bello, Barret Zoph, Quoc Le, Ashish Vaswani, and Jonathon Shlens. 2019. Attention Augmented Convolutional Networks. In 2019 IEEE/CVF International Conference on Computer Vision (ICCV). IEEE, Seoul, Korea (South), 3285–3294. https://doi.org/10.1109/ICCV.2019.00338

[8] Gian Pietro Bellocca, Giuseppe Attanasio, Luca Caglieri, and Jacopo Fior. 2022. Leveraging the momentum effect in machine learning-based cryptocurrency trading. Machine Learning with Applications 8 (2022), 100310. https://doi.org/10.1016/j.mlwa.2022.100310

[9] Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The Long-Document Transformer. arXiv:2004.05150 [cs]

[10] James Bergstra, Brent Komer, Chris Eliasmith, Dan Yamins, and David D Cox. 2015. Hyperopt: A Python Library for Model Selection and Hyperparameter Optimization. Computational Science & Discovery 8, 1 (July 2015), 014008. https://
A LEARNING TO RANK AND TRANSFER RANKING

A.1 Learning to Rank for Cross-Sectional Momentum

Learning to Rank (LTR) is a supervised learning problem where the goal is to sort a list of objects to maximise some measure of utility (e.g., discounted cumulative gain). For training, we are presented with a set of dates \( B = \{b_1, \ldots, b_{t-1}\} \) where at each \( b_i \) we have \( n \) instruments \( c_i = (c_i^{(1)}, \ldots, c_i^{(n)}) \). We also have a corresponding set of quintiles \( Q \) of realised returns \( q_{i+1} \) from the next period that \( c_i \) have been assigned based on an ascending sort. Here, \( q_{i+1} = \{q_{i+1}^{(1)}, \ldots, q_{i+1}^{(n)}\} \) where \( q_{i+1}^{(j)} \in Q = \{Q_1, \ldots, Q_5\} \) for \( i, j \). Additionally, \( k > j \Rightarrow Q_k > Q_j \) for \( k, l \in \{1, \ldots, 5\} \) where \( \succ \) stands for the order relation. In other words, instruments characterised by higher realised returns are assigned higher quintiles. With each rebalance-instrument pair at period \( i \), a feature vector \( u_i^{(j)} = f(b_i, c_i^{(j)}) \) can be formed, noting that \( f(\cdot) \) characterises the feature function and that \( i \in \{1, \ldots, t\} \) and \( j \in \{1, \ldots, n\} \). Letting \( u_i = (u_i^{(1)}, \ldots, u_i^{(n)}) \), we can go on to form the broader training set \( \{u_i, q_{i+1}\}_{i=1}^{t-1} \). The goal of LTR is to learn a function \( f \) that predicts a score \( f(u_i^{(j)}) = z_i^{(j)} \) at the Score Calculation phase (refer to the Traditional LTR pipeline in Figure 1) when presented with an out-of-sample input \( u_i^{(j)} \). For more details, we point the reader to Poh et al. [66] and Li [50].

A.2 From Learning to Rank to Transfer Ranking

Transfer Ranking (TR) builds on top of the LTR framework. A domain \( D \) consists of a feature space \( X \) and a marginal distribution \( P(X) \), i.e., \( D = \{X, P(X)\} \). \( X \) represents an instance set and is defined as \( X = \{x_i \in X, i = 1, \ldots, n\} \). A task \( T \) is composed of a label space \( Y \) and a decision function \( f \), so \( T = \{Y, f\} \). The decision function \( f \) is learned from the data. Using these definitions, the goal of LTR for CSM in the preceding section can be expressed as learning the ranking function \( f \) in \( T = \{Q, f\} \) over some domain of interest \( D = \{U, P(U)\} \). The feature space and instance set are represented by \( U \) and \( U \) respectively. A single instance \( u_i \) at period \( i \) is given as \( u_i = (u_i^{(1)}, \ldots, u_i^{(n)}) \) where \( u_i^{(j)} \in U \) for \( i, j \).

In the TR setting, we are given observations corresponding to \( m^S \in \mathbb{N}^+ \) source domains and tasks, i.e., \( \{D_{S_1}, T_{S_1}\} | i = 1, \ldots, m^S \) as well as observations about \( m^T \in \mathbb{N}^+ \) target domains and tasks, i.e., \( \{D_{T_j}, T_{T_j}\} | j = 1, \ldots, m^T \). TR aims to exploit the knowledge embedded in the source domains to enhance the performance of the learned decision functions \( f_{T_j}, j = 1, \ldots, m^T \) on the target domains.

In this work, we focus on \( m^S = m^T = 1 \). Specifically, we want to enhance the performance of \( f_T \), which is the scoring function for cryptocurrencies, by incorporating knowledge extracted from the FX dataset \( f_S \). This is achieved by assigning the parameters \( \theta_S \) of the calibrated \( f_S \) with \( f_T \) as described in Problem Definition.

B DATASET DETAILS

B.1 Source Dataset

Our source dataset is composed of 31 currencies all expressed versus the USD over the period 2-May-2000 to 31-Dec-2021. We use this data to train the upstream model required for knowledge transfer. The full list of currencies is as follows:

- G10: AUD, CAD, CHF, EUR, GBP, JPY, NOK, NZD, SEK, USD
- EM Asia: HKD, INR, IDR, KRW, MYR, PHP, SGD, TWD, THB
- EM Latam: BRL, CLP, COP, PEN, MXN
- CEEMEA: CZK, HUF, ILS, PLN, RUB, TRY
- Africa: ZAR

B.2 Target Dataset

Our target dataset is focused on the following 10 cryptocurrencies: Bitcoin (BTC), Ethereum (ETH), Dogecoin (DOGE), DigiByte (DGB), Litecoin (LTC), Stellar (XLM), XRP, Monero (XMR), NEM (XEM), Dash (DASH). For this digital tokens span 1-Jan-2016 to 31-Dec-2021.

B.3 Data Preprocessing

In order to reduce the impact of outliers during training, we winsorise both foreign exchange and cryptocurrency data by capping and flooring it to be within 3 times its exponentially weighted moving (EWM) standard deviation from its EWM average that is calculated with a 26-week half-life decay.

C ADDITIONAL TRAINING DETAILS

Model Fine-tuning: We fine-tune both FEN and SAR+ps models. In this step, we unfreeze the entire network and re-train with the lowest value in the learning rate search range, i.e., \( 10^{-6} \) and over different batch sizes.

Python Libraries: LambdaMART uses XGBoost [23]. All other machine learning models are developed with TensorFlow [1].

Hyperparameter Optimisation: Hyperparameters are tuned using HyperOpt [10]. For LambdaMART, we refer to the hyperparameters as they are named in the XGBoost library. The feature search spaces for various machine learning models are as follows:

Multi-layer Perceptron (MLP):

- Dropout Rate – [0.0, 0.2, 0.4, 0.6, 0.8]
- Hidden Width – [8, 16, 32, 64, 128]
- Batch Size – [8, 16, 32, 64, 128]
- Learning Rate – \( [10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}] \)

ListNet (LN):

- Dropout Rate – [0.0, 0.2, 0.4, 0.6, 0.8]
- Hidden Width – [8, 16, 32, 64, 128]
- Batch Size – [1, 2, 4, 8, 16]
Table 6: Performance metrics for FEN and SAR+ps over the 2018-2021 period with a 15% annual volatility target. All figures are based on ten runs. Both models incorporate components from sub-optimal source models. The impact is clearly evident for both models with statistics that are worse off than their respective counterparts in Table 1.

|                      | SAR+ps | FEN   |
|----------------------|--------|-------|
| E[returns]           | -0.050±0.097 | 0.078±0.055 |
| Volatility           | 0.174±0.013  | 0.190±0.018 |
| Sharpe Ratio         | -0.277±0.569 | 0.414±0.302 |
| Sortino Ratio        | -0.260±0.819 | 0.719±0.527 |
| Calmar Ratio         | -0.009±0.321 | 0.364±0.264 |
| Downside Deviation   | 0.131±0.025  | 0.117±0.018 |
| Max. Drawdown        | -0.424±0.173 | 0.240±0.066 |
| Avg. Profit / Avg. Loss | 1.016±0.249 | 1.189±0.163 |
| Hit Rate             | 0.459±0.026  | 0.496±0.020 |

- Learning Rate – [10⁻⁶, 10⁻⁵, 10⁻⁴, 10⁻³]

LambdaMART (LM):
- ‘objective’ – ‘rank:pairwise’
- ‘eval_metric’ – ‘ndcg’
- ‘num_boost_round’ – [50, 100, 250, 500, 1000]
- ‘max_depth’ – [6, 7, 8, 9, 10]
- ‘eta’ – [10⁻⁶, 10⁻⁵, 10⁻⁴, 10⁻³, 10⁻², 10⁻¹]
- ‘reg_alpha’ – [10⁻⁶, 10⁻⁵, 10⁻⁴, 10⁻³, 10⁻², 10⁻¹]
- ‘reg_lambda’ – [10⁻⁶, 10⁻⁵, 10⁻⁴, 10⁻³, 10⁻², 10⁻¹]
- ‘tree_method’ – ‘gpu_hist’

Self-attention Ranker (SAR), Self-attention Ranker with Parameter-sharing (SAR+ps), both the source model for Fused Encoder Networks (FEN) and FEN itself:
- $d_f$ – [8, 16, 32, 64, 128]
- $d_{ff}$ – [8, 16, 32, 64, 128]
- Dropout Rate – [0.0, 0.2, 0.4, 0.6, 0.8]
- Hidden Width – [16, 32, 64, 128, 256]
- Batch Size – [2, 4, 6, 8, 10]
- Learning Rate – [10⁻⁶, 10⁻⁵, 10⁻⁴, 10⁻³]
- No. of Encoder Layers – [1, 2, 3, 4]
- No. of Attention Heads – 1

Fine-tuning step for SAR+ps and FEN:
- Batch Size – [2, 4, 6, 8, 10]
- Learning Rate – 10⁻⁶

D INCORPORATING KNOWLEDGE FROM SUB-OPTIMAL SOURCE MODELS

We investigate the impact on model performance when inappropriate source models are used for knowledge transfer. Out of 10 runs of the source model, we pick the worst 2 as measured by the strategy’s Sharpe ratio on the out-of-sample period. Using components from each, we construct their respective target models and make predictions on the corresponding target dataset. The results of this exercise are presented in Table 6. Importantly, we see that transferring knowledge from sub-optimal source models has a substantial performance drag for FEN (around 45% reduction in the Sharpe ratio).

E COMPARING ATTENTION HEATMAPS

This section compares the heatmaps of models that use the attention mechanism: the SAR, SAR+ps, and FEN. Specifically, we present the weights of each model’s last attention layer for each market state and cryptocurrency-specific event discussed in Attention Heatmap Patterns. Darker cells correspond to larger weights, contributing more to the model’s prediction. Additionally, attention weights across different models are not comparable since these are raw weights. Overall, the patterns produced by FEN appear to broadly align with what one might expect. Risk-off states, for instance, are associated with heatmaps with darker cells that are dispersed – reflecting the broad impact felt by multiple instruments. This development is observed to a lesser extent in SAR, and is almost entirely absent in SAR+ps.
Figure 7: Aggregated maps over normal market states. Cells with larger weights cluster in pockets for FEN, but are dispersed for SAR. SAR+ps interestingly places a high amount of attention weights on Monero ('xmr') and NEM ('xem'), which are relatively less popular than Bitcoin ('btc') and Ethereum ('eth').

Figure 8: Aggregated maps over risk-off market states. Darker cells are dispersed more in FEN and SAR, but less for SAR+ps.

Figure 9: Attention maps in early May 2021, around the time of Elon Musk’s Dogecoin ('doge') tweets. The 'doge' column is activated for FEN, but this pattern is absent for both SAR and SAR+ps.
Figure 10: Attention maps for the second half of May 2021, which covers China’s government regulators declaring a crackdown on the use of cryptocurrencies by domestic financial institutions on the 19th of the same month. The significance of this event is evident from the pattern of diffused shaded blocks for FEN, with multiple coins affected. This is observed to a lesser extent in SAR’s attention map and is almost completely absent in SAR+ps’s.

Figure 11: Column averages of the raw attention heatmaps for SAR across different market states aggregated over multiple runs. Each column average corresponds to a ranking index predicted by the model. Across the three different settings, the bars appear to roughly follow a ‘J’ shape, with higher weights for each half of the ranked list typically coinciding with traded positions (longs are labelled 9 and 10 on the horizontal axis, while shorts are 1 and 2).
Figure 12: Column averages of the raw attention heatmaps for SAR+ps across different market states aggregated over multiple runs. Each column average corresponds to a ranking index predicted by the model. Across the three different settings, the bars appear to roughly follow a reversed 'J' shape flipped about the y-axis, with higher weights for each half of the ranked list typically coinciding with traded positions (longs are labelled 9 and 10 on the horizontal axis, while shorts are 1 and 2).