Berkeley
UNIVERSITY OF CALIFORNIA

Deep Learning Statistical Arbitrage

(Jorge Guijarro-Ordonez, Markus Pelger, Greg Zanotti)

230ZB - Deep Learning and Applications I

Group 7 Report

Abhishek Kumar, Ajay Kumar, Gurmeet Bedi,

Rishi Kumra, Rohit Chaturvedi

Contents

1 Introduction

As a reminder, this paper presents a novel data-driven solution and comprehensive framework for statistical arbitrage combining three key ideas: arbitrage portfolio generation, arbitrage signal extraction, and arbitrage allocation decision based on the signals. The aim is to create portfolios that are similar to the target assets.

In 230 ZA, our group implemented 2 of 3 models mentioned in the paper on a portfolio of the top 500 stocks by market cap (reflecting SP500). The first model that the authors presented uses residual returns not explained by a Fama-French factor model and uses an Ornstein-Uhlenbeck process to model these residuals. The process is modeled as a rolling regression and yields 4 parameters that are used in a Feed-Forward Neural network to generate trading signals. The second model works in a similar way where it uses a Fourier transform for generation of arbitrage signals which is then again fed to Feed Forward Neural Networks for calculation of optimum weights.

In 230ZB, we have now implemented the third model which involves inputting the residuals into a convolutional neural network (CNN) with 3-day kernels representing trends in the underlying price signals. The output of this is then passed to a transformer with 4 attention heads which gives us the weights of our portfolio. In addition, we also replicated the previous 2 models and this model for FX futures instead and now have results for both equities and FX.

Key Findings:

1. **Our results are consistent with the paper:** Despite constraints on gpu and the number of epochs we could train on, we found our results were close to the findings of the paper. With transaction costs included, in our run, we found that the OU/FFN, FFT/FFN and CNN/transformer models for the Fama French 5 factor residuals had Sharpes of 0.55, 0.64, 0.67 respectively.

2. **Asset Class Difference:** We began to explore FX futures and found similar results suggesting this approach is asset-class variable. We will be dicussing this in more detail in our project report.

3. The choice of asset pricing factors has a minor effect on performance, while the time-series model for signal extraction, specifically the convolutional transformer, significantly improves results.

4. Arbitrage trading is based on local asymmetric trend and reversion patterns, with different policies for downturn and uptrend movements.

5. Residuals, rather than raw returns, are used for time-series-based trading patterns, enabling the incorporation of alternative data into the portfolio construction process.

2 Mathematics of the Paper

Here, we will discuss the mathematical details behind each of the 3 key elements of the paper: arbitrage portfolios, arbitrage signal extraction and arbitrage allocation. These equations form the fundamental functions we use in our code.

2.1 Arbitrage Portfolios

Every month, the list of stocks used to construct the arbitrage portfolios are picked based on their market capitalization values on CRSP. The idea is to pick only those stocks that exceed a minimum threshold of 0.01 percent of total market cap. On average, this leaves them with approximately the largest 550 stocks, which corresponds roughly to the SP500 index. The cross-sectional daily return data for each of these stocks is taken from January 1978 through December 2016.

Then, by assuming that asset returns can be modelled by a conditional factor model, the paper applies various factor models to the cross-sectional matrix of daily returns minus the risk free rate (excess returns).

$$R_{n,t} = \beta_{n,t-1}^T F_t + \xi_{n,t}$$

In their empirical analysis, they use Fama-French 3 and 5 factor models as well as PCA and IPCA models. The residuals from these models are then defined as the arbitrage portfolios.

$$\xi_{n,t} = R_{n,t} - \beta_{n,t-1}^T F_t$$

Arbitrage portfolios are trades relative to mimicking stock portfolios. The factor component in the residuals corresponds to a well-diversified mimicking portfolio, that is very "close" to the specific stock in the relative trade. In the case of PCA factors, the paper constructs well-diversified portfolios of stocks that have the highest correlation with each target stock in the relative trades. This is conceptually the same idea as using a clustering algorithm to construct a portfolio of highly correlated stocks. Hence,residuals of PCA factors construct relative trades based on correlations in past return time-series. In the case of IPCA factors, we intuitively construct for each stock a well-diversified mimicking portfolio that is as similar as possible in terms of the underlying firm characteristics. Hence, the mimicking portfolio represents an asset with very similar firm fundamentals.

The relative trades put on then exploits the temporal time-series deviations from the risk factors captured by the mimicking portfolios. This is the fundamental idea of the paper.

2.2 Arbitrage Signal Extraction and weight Allocation

The signal is extracted from the time-series of the residuals. An example for an arbitrage signal would be a parametric model for mean reversion that is estimated for each arbitrage portfolio. The trading strategy for each arbitrage portfolio would depend on its speed of mean reversion and its deviation from the long run mean. More generally, the arbitrage signal is the estimation of a time-series model, which can be parametric or nonparametric. This paper considers three models - parametric OU, nonparametric FFT and a CNN.

They apply the signal extraction to the time series of the last L lagged residuals (30 days in the paper), $\xi^L_{n,t-1}$ denoted as below. This is then mapped into a vector of signals $\theta_{n,t-1}$:

$$\xi^L_{n,t-1} = (\xi_{n,t-L}, \xi_{n,t-L+1}, \ldots, \xi_{n,t-1})$$

$$\xi^L_{n,t-1} \to (\theta_{n,t-1})$$

The signals $\theta^{n,t-1} \in \mathbb{R}^p$ for the arbitrage portfolio n at time t only depend on the time-series of lagged returns $\epsilon_L^{n,t-1}$. Note that the dimensionality of the signal can be the same as for the input sequence.

Models for Arbitrage Signal

2.2.1 Parametric Model - Ornstein-Uhlenbeck Process

Classic Mean reversion is implied here. The Ornstein-Uhlenbeck (OU) process is a stochastic process used to model the behavior of a system subject to random fluctuations. The OU process is defined by the following differential equation:

$$dX_t = \kappa(\mu - X_t)dt + \sigma dB_t$$

where κ is the mean reversion speed, μ is the long-term mean, σ is the volatility, and B_t is a Brownian motion.

The parameters for each residual process, the last cumulative sum and a goodness of fit measure form the signals for the OU model:

$$\theta_{\text{OU}} = (\kappa, \mu, \sigma, X_L, R^2)$$

where X_L is the last cumulative sum and R^2 is a goodness of fit measure.

$$w_\epsilon^{\text{OU}}(\theta_{\text{OU}}) = \begin{cases} -1 & \text{if } \frac{X_L - \mu}{\sqrt{\sigma^2/\kappa^2}} > c_{\text{thresh}} \text{ and } R^2 > c_{\text{crit}} \\ 1 & \text{if } \frac{X_L - \mu}{\sqrt{\sigma^2/\kappa^2}} < -c_{\text{thresh}} \text{ and } R^2 > c_{\text{crit}} \\ 0 & \text{otherwise} \end{cases}$$

An alternative to the simple weighting with thresholds above is to allow for a more flexible allocation function given the same time-series signals. The paper also considers for all our models a general feedforward neural network (FFN) to map the signal into an allocation weight. FFNs are nonparametric estimators that can capture very general functional relationships. Hence, they also consider the additional model that restricts the signal function, but allows for a flexible allocation function:

$$w^{\epsilon|OU-FFN}(\theta^{OU}) = g^{FFN}(\theta^{OU})$$

2.2.2 Pre-specified Filters - Fast Fourier Transform

A Fast Fourier Transform (FFT) provides a frequency decomposition of the original time-series and separates the movements into mean-reverting processes of different frequencies. FFT applies the filter $W_j^{\text{FFT}} = e^{\left(\frac{2\pi ij}{L}\right)}$ in the complex plane, but for real-valued time-series, it is equivalent to fitting the following model:

$$x_l = a_0 + \sum_{j=1}^{L/2-1} (a_j * \cos(\frac{2\pi ij}{L}) + b_j * \sin(\frac{2\pi ij}{L})) + a_{L/2}\cos(\pi l)$$

FFT simply represents the original time-series in a different form without losing any information. It is based on the insight that not the magnitude of the original data but the relative relationship in a time-series matters.

We use a flexible feedforward neural network, as earlier in the OU process, for the allocation function.

$$w^{\epsilon|FFT}(\theta^{FFT}) = g^{FFN}(\theta^{FFT})$$

2.2.3 Deep-Learning CNN Model

A Convolutional Neural Network leverages three important ideas that can help improve a machine learning system: sparse interactions, parameter sharing and Equivariant representations.

In a traditional neural network, every node is connected to every other node resulting in dense interactions. In contrast, Convolutional layers use filters or kernels that move across the input data with a

certain stride, selectively activating specific regions based on importance. This results in sparse architecture making CNNs computationally more efficient and capable of handling large input data.

Another key concept in CNNs is parameter sharing, where the same set of weights (parameters) is shared across multiple spatial locations in the input data. This is particularly powerful when the data contains certain features which are relevant regardless of their location. This sharing not only reduces the number of parameters, making training feasible, but also enhances the variance reduction prowess of the model.

The property of Equivariance makes the convolutional neural networks immune to changes in the input space. The spatial relationships learned by the network is preserved even if the input layers are shifted in domain.

For time series data, a CNN can learn to capture local patterns with receptive fields.For example, it might identify local trends, spikes, or recurring shapes within the time series. Through stacking multiple Convolutional layers, a CNN can learn hierarchical representations of temporal patterns. Lower layers might capture simple local patterns, while higher layers can combine these patterns to recognize more complex temporal structures. The Convolutional and pooling layers in a CNN act as feature extractors. They can abstract relevant features from the time series, allowing the network to focus on the most informative aspects of the data.

The arbitrage portfolio for the 550 stocks created above, is fed into a deep learning block. The deep learning block consists of i) **'D'** number of Convolution Neural Networks (which is a hyper-parameter), ii) **'H'** number of self-attention headers and iii) **'F'** number of Feed Forward Neural Networks. This block generates the optimum weight for these 550 stocks subject to a constraint. The most common constraints are i) Maximizing the Sharpe Ratio or ii) Maximising Mean Variance. The risk aversion γ in the second constraint is taken in the paper as 1.

Exploring the Convolution Neural Network This block is used to study the short-term dependence of the arbitrage portfolio fed into the network. For arbitrage, one is interested in whether the lags are on an increasing trend, a decreasing trend, or a mix. The filter size and the number of times it is applied determine how many lags should be considered for local learning.

$$y_{l,d}^{(0)} = b_d^{(0)} + \sum_{m=1}^{D_{\text{size}}} W_{d,m}^{(0)} x_{l-m+1}^{(0)}, \qquad x_{l,d}^{(1)} = \text{ReLU}\left(\frac{y_{l,d}^{(0)} - \mu_d^{(0)}}{\sigma_d^{(0)}}\right) \tag{1}$$

$$y_{l,d}^{(1)} = b_d^{(1)} + \sum_{m=1}^{D_{\text{size}}} \sum_{j=1}^{D} W_{d,j,m}^{(1)} x_{l-m+1,j}^{(1)}, \qquad x_{l,d}^{(2)} = \text{ReLU}\left(\frac{y_{l,d}^{(2)} - \mu_d^{(2)}}{\sigma_d^{(2)}}\right) \tag{2}$$

$$\tilde{x_{l,d}} = x_{l,d}^{(2)} + x_l^{(0)}; \quad \mu_k^{(i)} = \frac{1}{L}\sum_{l=1}^{L} y_{l,k}^{(i)}; \qquad \sigma_k^{(i)} = \sqrt{\frac{1}{L}\sum_{l=1}^{L}(y_{(l,k)}^{(i)} - \mu_k^{(i)})^2} \tag{3}$$

Here, b is bias in dimension. R^D , $W^{(0)}$ in dimension $R^{DxD_{size}}$ and $W^{(1)}$ is in dimension $R^{DxDxD_{size}}$, $y^{(0)}$ is the output of the first CNN block, and $y^{(1)}$ is the output of second CNN. $\tilde{x}_{l,d}$ belong to LxD dimensions where D is the total number of filters of length D_{size} applied to x.

2.2.4 Transformer

A transformer is a deep learning architecture that relies on the parallel multi-head attention mechanism. They have been successfully used in capturing temporal dependencies in the data mainly in natural language processing applications.

A major advantage of transformers is that they can process sequences simultaneously, avoiding the bottleneck in Recurrent Neural Networks. They operate on the principle of self attention which allows them to weight the importance of different time points in the sequence when making predictions. The multi-head attention mechanism of transformer is highly useful, making it possible for it to simultaneously focus on different aspects of the temporal pattern.

In the context of stock returns, this feature enables it to capture patterns and relationships that span various time intervals, capturing both short-term and long-term dependencies. The incorporation of positional encoding in the transformer's architecture addresses the sequential nature of time series data, allowing the model to identify the order of events. This is crucial in the context of stock returns, where the lagged order of returns is fundamental in understanding market dynamics.

This part of the deep learning network updates the weight of the input based on the long-term dependence structure of the arbitrage portfolio. The number of headers, **'H'**, is a hyper-parameter to learn global trends.

$$V_i = \tilde{x}W_i^v + b_i^V \in \mathbb{R}^{L\times D/H}, \quad K_i = \tilde{x}W_i^k + b_i^k \in \mathbb{R}^{L\times D/H}, \quad Q_i = \tilde{x}W_i^Q + b_i^Q \in \mathbb{R}^{L\times D/H} \tag{4}$$

$W_i^{V,K,Q} \in \mathbb{R}^{D\times F/H}$, $b_i^V, b_i^K, b_i^Q \in \mathbb{R}^{D/H}$ need to be estimated

$$h_{i,l} = \sum_{j=1}^{L} w_{l,j,i} V_{i,j} \in \mathbb{R}^{D/H}, \quad w_{l,j,i} = \frac{e^{K_{i,l} \cdot Q_{i,j}}}{\sum_{m=1}^{L} e^{K_{i,l} \cdot Q_{i,m}}} \tag{5}$$

Concatenation of these states and linear combination obtains the last hidden state.

2.2.5 Feed Forward Neural Network

The updated output based on local and global networks from the transformer is fed into the deep neural network to generate the weights. The weights are adjusted to maximize the two constraints.

A diagrammatic approach and dimensions are shown on the next page.

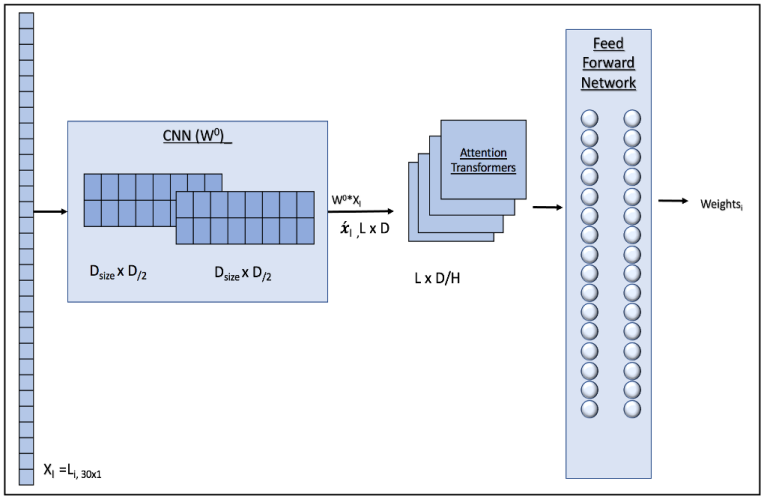
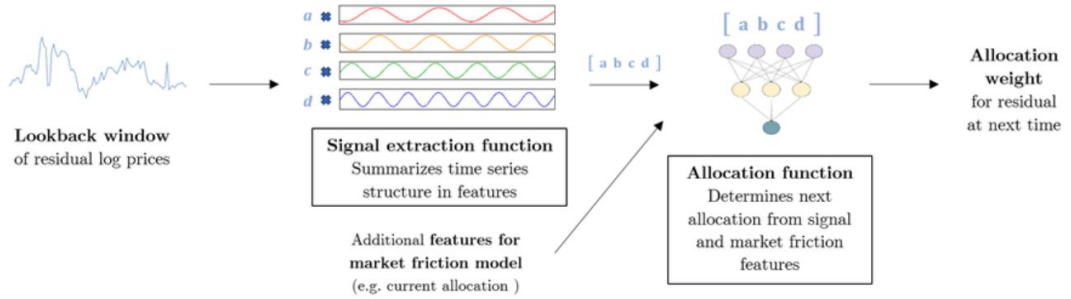


Figure 1: Deep Learning Architecture

As mentioned, all three methods use a max sharpe objective function to tweak the allocations. This objective function is subject to a no-leverage constraint on the weights such that they all add to 1, we solve for:

$$\max \mathbb{E}[w_{t-1}^T R_t] - \gamma Var(w_{t-1}^T R_t)$$



This figure illustrates the conceptual structure of our statistical arbitrage framework. The model takes as input the last L cumulative returns of a residual portfolio on a lookback window at a given time and outputs the predicted optimal allocation weight for that residual for the next time. The model is composed of a signal extraction function and an allocation function.

Figure 2: The integration of the three key elements of this paper.

3 Our Methodology

For the first course, we replicated 2 of the 3 signal extraction models of this paper:

1. Ornstein-Uhlenbeck

2. Fast Fourier Transform

Now, we've replicated the 3rd and final signal extraction model:

3. CNN + Transformer

For the CNN + transformer architecture, we first define the CNN block, This block consists of two 1D Convolutional layers (conv1 and conv2) with ReLU activation functions, and Instance Normalization (normalization1 and normalization2). We have pre-specified the kernel size of this block and the output of this block is specified extrinsically using another variable.

We then implement a new class CNNTransformer which encompasses the CNN block and the Transformer block. This consists of 3 parts a list of CNN block instances forming the convolutional part of the model, encoder - which contains a transformer encoder layer and a linear output layer which outputs the final result.

The specific parameters of the models are as follows : 2 1D CNN layers with filter size 1 and 8 and same kernel size of 2. A transformer with 4 attention heads, embedding dimension of 8 and 2*8 feedforward dimension with a dropout probability of 0.25. Finally, the output is obtained by applying a linear layer of size 8 to the last time step of the transformer output.

This is the complete architecture through which the input data is fed. The output of this CNNTransformer layer is fed into a FeedForwardNeural Network.

Moreover, we downloaded data for the 9 top FX futures in the world, extracted residuals using PCA, and fit these residuals into each of the three models above.

The following section explains how we collected our data, created functions and optimised our code.

3.1 Data

Equities

The daily returns of various stocks meeting the market cap requiremenet were taken from CRSP using the WRDS database. While the paper went through a monthly rebalance of stocks for the top 0.01 percent of market cap, we decided to pick the stocks that are directly represented in the SP500 index to accurately reflect the market. This is a proxy to the paper's data and may create some minor discrepancies. We have also removed stocks from trading if they had no returns for a particular day.

The daily returns data was taken from 1998 to 2022 for around 1100 stocks (the 500 stocks in SP500 over the time period). The original paper only went up to 2016 due to it being published a few years

ago. To reflect newer market conditions and cover a recent stress period such as Covid in 2020, our final results are shown for returns from 2011 onward.

The risk-free rate used to calculate excess returns was taken from the US-Treasury one-month rates. This was downloaded from the Kenneth French Data Library.

For the factor-models in the preliminary step of creating the arbitrage portfolios, we downloaded the Fama-French factor returns also from the Kenneth French Data Library.

FX

Daily prices/currency pair rates were taken directly from Bloomberg for 9 top currency futures:

GBP EUR CHF JPY SEK NOK CAD NZD AUD

The returns were calculated in USD and PCA was conducted to find the top 5 factors across the 9 FX futures. These factors were then subtracted from the net excess returns to obtain the residuals.

Key Python libraries used: sklearn (for linear regression), torch (for neural networks), numpy, pandas

3.2 Code

Our group's code started by trying to replicate the provided GitHub repository by the original paper's author. However, this had missing residuals data and missing functions due to licensing issues from original providers. Therefore, we had to rework some of the code to calculate residuals and adjust the code to incorporate our new data size and structure.

Below are simplified block diagrams of each of the respective files and steps in the code. Note, that this is missing the third deep learning approach as this is something we plan to implement for the ZB module.

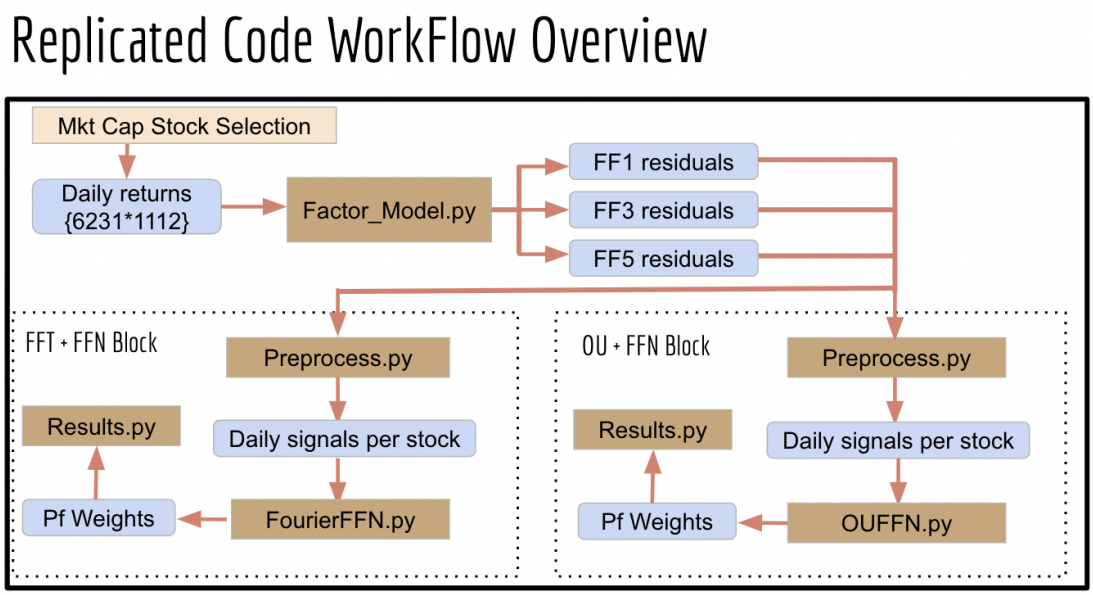


Figure 3: Overview of the code structure

Figure 3 explains the overall paper replication in one diagram. Based on our SP500 stocks (1112 stocks), we created a matrix of daily returns (6231 days) which was fed into our factor model function.

The factor model function regresses the SP500 stocks onto the Fama-French 1,3 and 5 factor returns to get residuals from each factor model. This is then fed into 1 of 2 signal processing and allocation code blocks - either the OU+FFN or FFT+FFN block. These blocks of code generate signals and allocate weights to each stock for each day.

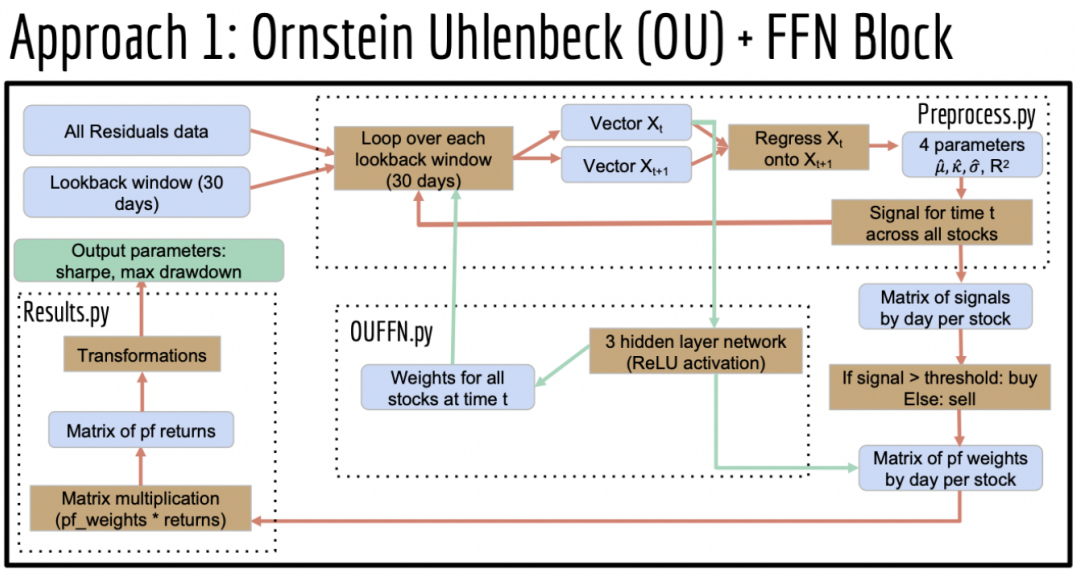


Figure 4: Overview of OU block from the previous figure

The underlying OU process is an ARMA time-series approximation. Therefore, the lagged residuals over 30 days are regressed onto one another, and these regression parameters are used to calculate our signals. This is done in 2 potential ways: by defining thresholds for buy/sell signals based on regression parameters or bypassing the regression data through a feed-forward neural network.

The Fourier Transformation process breaks down the residuals of the lookback period (in this case, 30 days) to different frequencies. Fourier transformation's intuition is converting a time domain signal to the frequency domain. In theory, a time-series signal can be broken down to infinite frequencies. We have truncated the Fourier coefficients to the first 30 coefficients in our case. These 30 coefficients form the arbitrage signals based on which the Feed Forward Neural Network learns the weights.

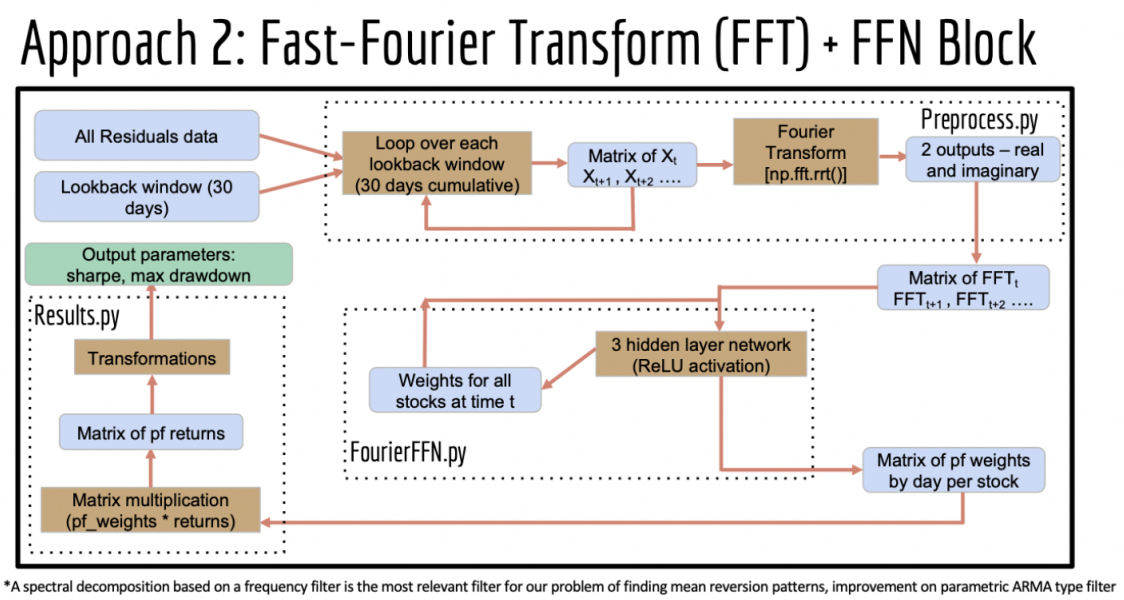


Figure 5: Overview of the FFT block from the original overview

4 Our Results

Our results are provided from 1st January 2011 to 31st December 2022. All results are annualized. The number of trading days in a year is taken as 252. The results shall be provided for the following two parametric methods:

1. Ornstein-Uhlenbeck process for generation of arbitrage signals, fed to Feed Forward Neural Network for optimum weights
2. Fourier Transformation Process for generation of arbitrage signals fed to Feed Forward Neural Networks for optimum weights

There are few cases where we see that with cost performance is better than without cost performance. We found this curious and extensively checked to see no calculation errors. This can be attributed to the non-convex nature of the problem. Also, the convergence may be achieved after possibly hundreds of random initializations. This, however, is not a function of several epochs. Also, on a positive note, we believe the transaction costs drive the weights and optimize overall Sharpe.

We shall present the results for two scenarios:

1. Without Transaction Cost
2. With Transaction Cost (5bps for transaction cost and 1 bps for shorts)

and along the following dimensions:

1. Fama French 1 Factor Residuals (K = 1)
2. Fama French 3 Factor Residuals (K = 3)
3. Fama French 5 Factor Residuals (K = 5)

The following constraints have been maximized to train the Neural Network:

1. Sharpe
2. Mean-Variance
3. Return Maximization

From the perspective of benchmarks, the S&P 500 will considered as a benchmark.

The hyperparameters used for the results are as follows (any deviation will be reported separately):

1. epochs = 250
2. batch-size = 100 days
3. Lookback = 30 days
4. training period = 1000 days
5. re-train period = 125 days
6. holding period = 1 day

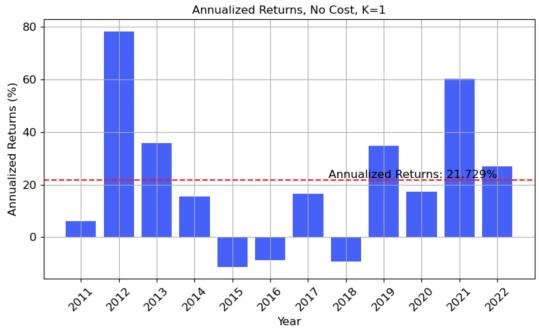
4.1 Model 1 Ornstein-Uhlenbeck with FFN

1. **Constraint is Sharpe Maximization**

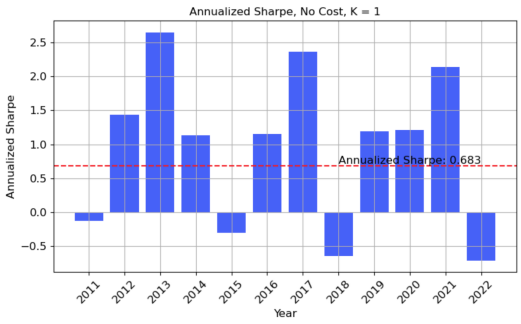
   (a) **No Cost**

Factors (K)	Sharpe	Mean Returns	Standard Deviation
1	0.683	21.729%	31.680%
3	0.685	25.101%	19.030%
5	0.229	13.718%	16.674%

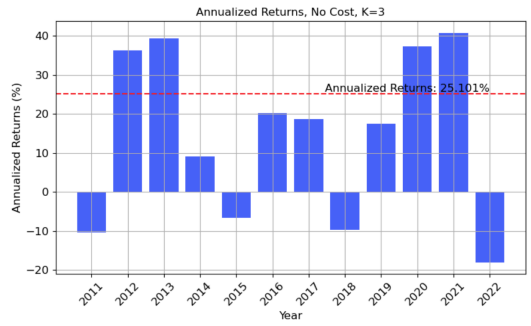
Table 1: Results for No Cost Scenario



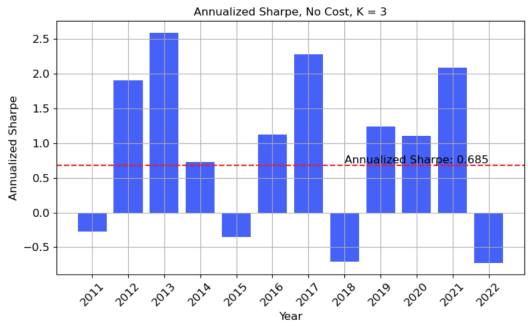
(a) 1 Factor, Annualized Returns



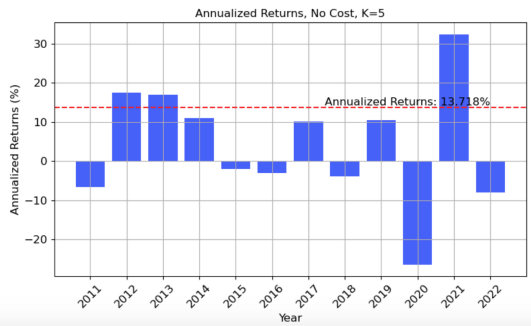
(b) 1 Factor, Annualized Sharpe



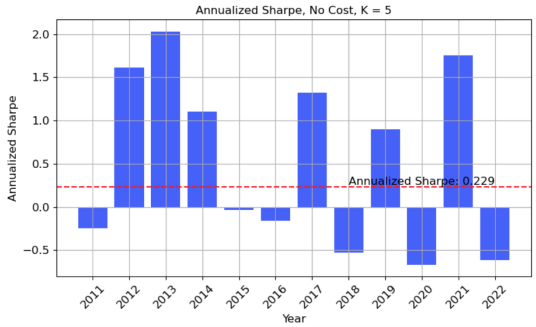
(c) 3 Factors, Annualized Returns



(d) 3 Factors, Annualized Sharpe



(e) 5 Factors, Annualized Returns



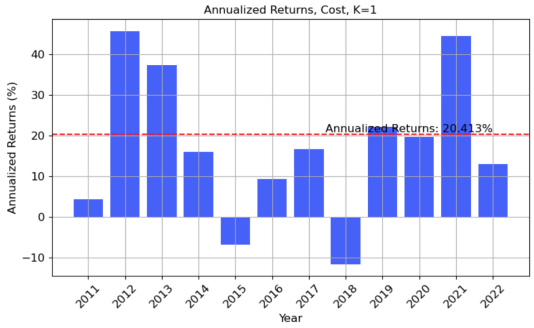
(f) 5 Factors, Annualized Sharpe

Figure 6: No Cost, Sharpe Maximisation Results

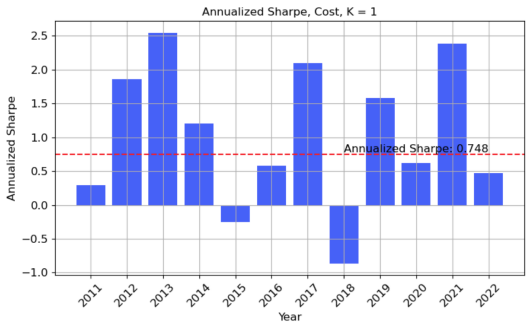
(b) **With Cost**

Factors (K)	Sharpe	Mean Returns	Standard Deviation
1	0.748	20.413%	22.636%
3	0.705	28.819%	19.336%
5	0.554	18.434%	18.628%

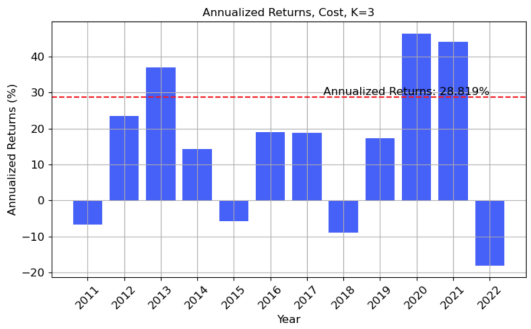
Table 2: Results with Costs



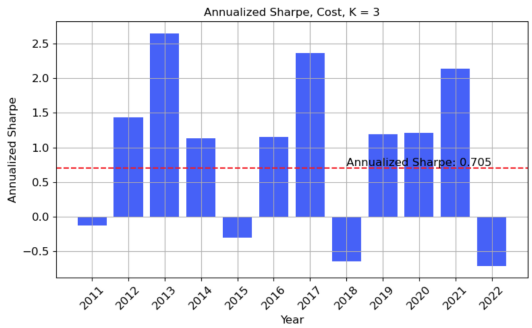
(a) 1 Factor, Annualized Returns



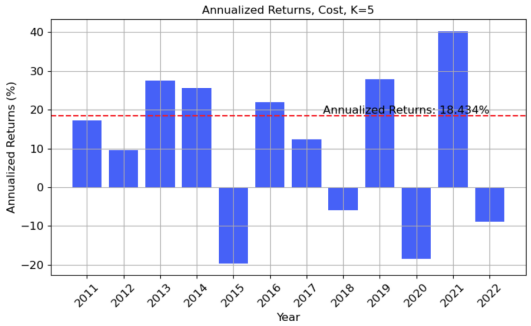
(b) 1 Factor, Annualized Sharpe



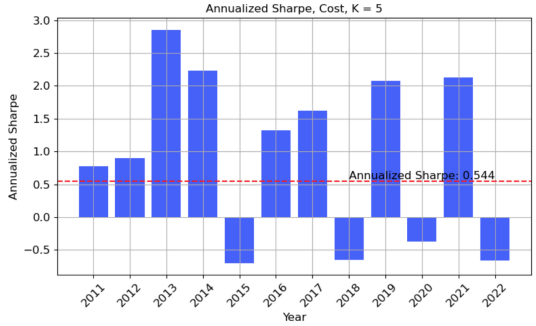
(c) 3 Factors, Annualized Returns



(d) 3 Factors, Annualized Sharpe



(e) 5 Factors, Annualized Returns



(f) 5 Factors, Annualized Sharpe

Figure 7: Cost, Sharpe Maximisation Results

(c) **Short Comparison**

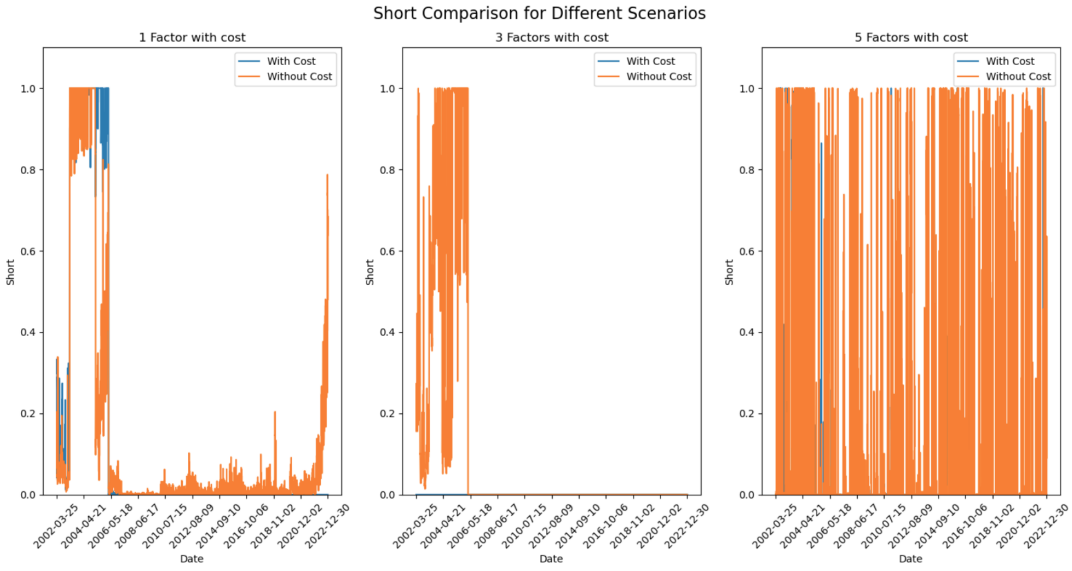


Figure 8: Short Proportion

(d) **Turnover Comparison**

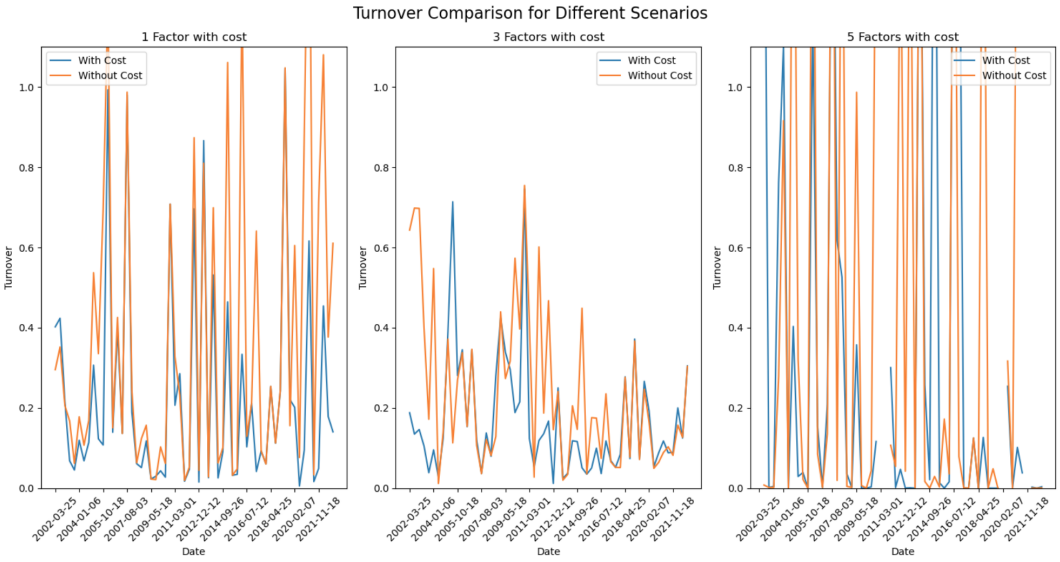


Figure 9: Turnover Proportion

2. **Constraint is Mean-Variance Maximization**

(a)

Factors (K)	Sharpe	Mean Returns	Standard Deviation
1	0.563	7.162%	28.292%
3	0.624	45.969%	51.564%
5	0.498	20.801%	18.692%

Table 3: Results for No Cost Scenario

Factors (K)	Sharpe	Mean Returns	Standard Deviation
1	0.626	24.666%	20.869%
3	0.545	26.757%	39.966%
5	0.638	26.904%	16.450%

Table 4: Results with Costs

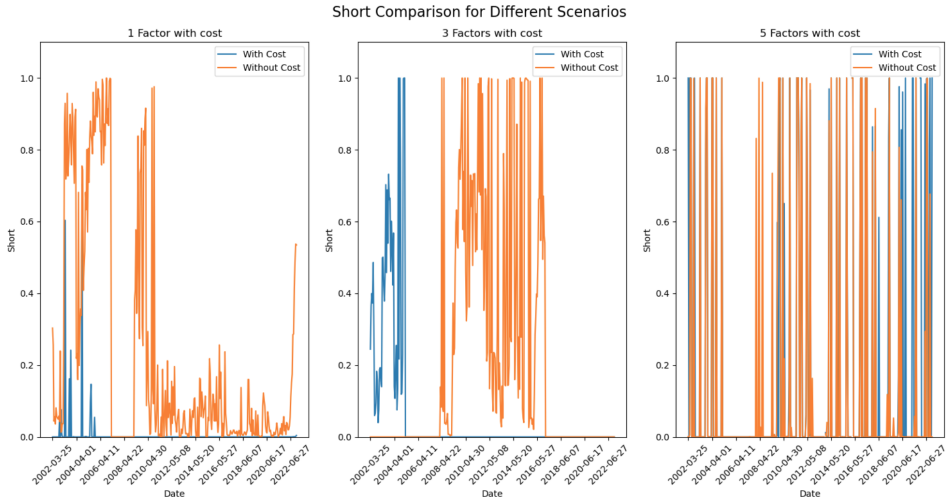


Figure 10: Short Proportion

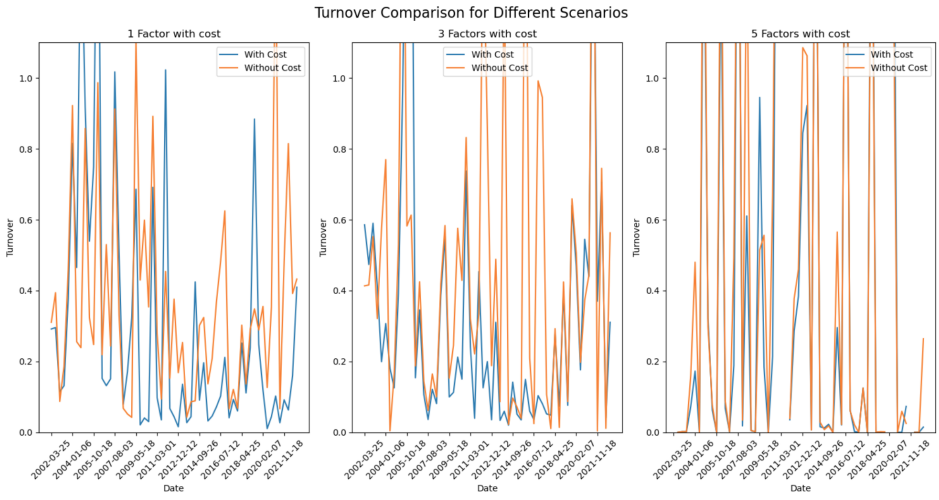


Figure 11: Turnover Proportion

4.2 Model 2 Fourier Transformation with FFN

1. **Constraint is Sharpe Maximization**

(a) **No Cost**

Factors (K)	Sharpe	Mean Returns	Standard Deviation
1	1.121	38.715%	32.895%
3	0.829	32.205%	44.803%
5	0.527	8.714%	16.084%

Table 5: Results for No Cost Scenario

(b) **With Cost**

Factors (K)	Sharpe	Mean Returns	Standard Deviation
1	0.948	37.327%	43.928%
5	1.014	16.025%	14.572%

Table 6: Results with Costs

(c) **Short Comparison**

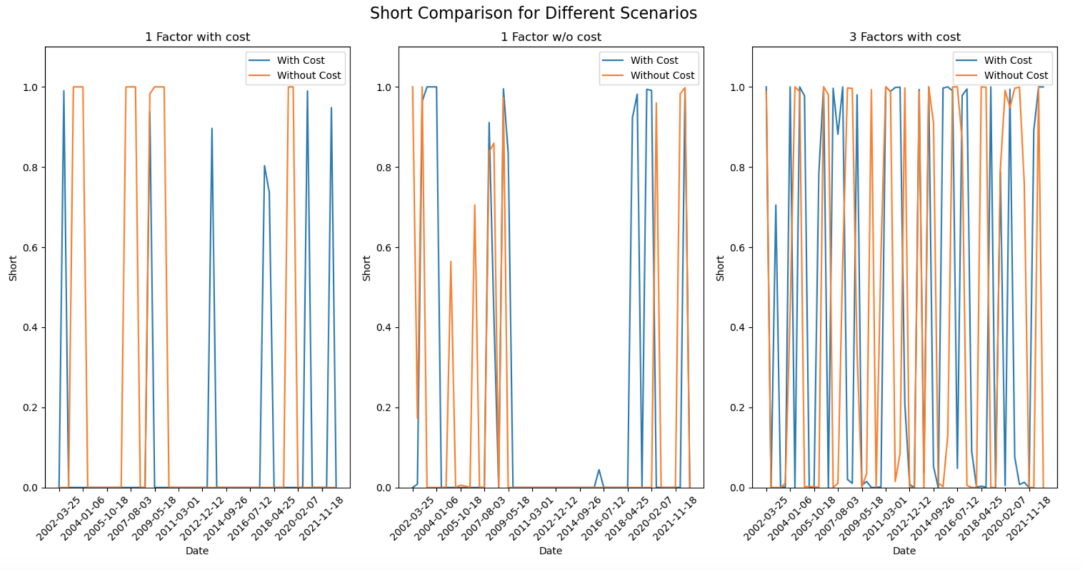


Figure 12: Short Proportion

(d) **Turnover Comparison**

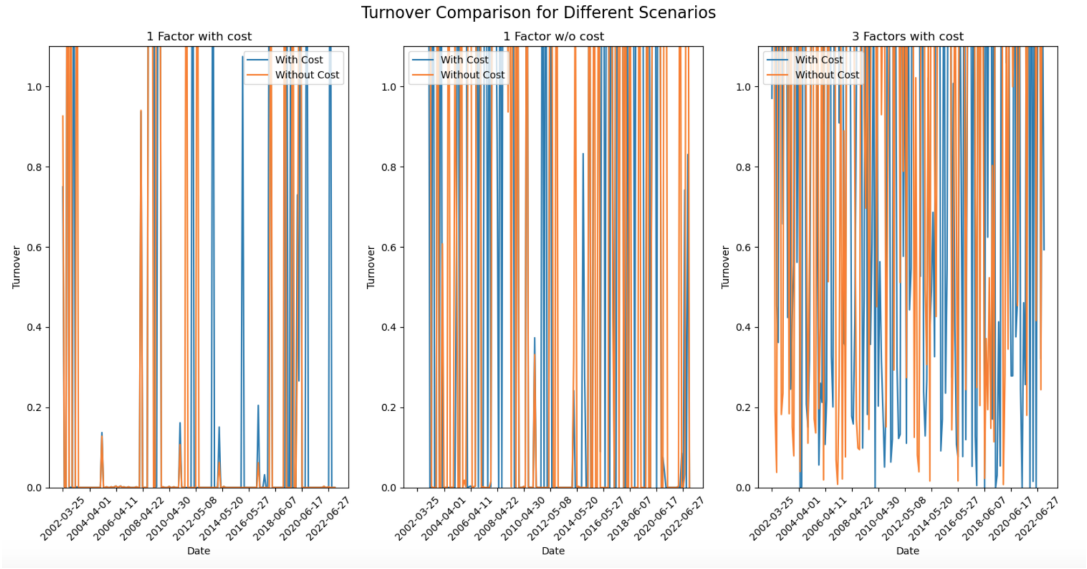


Figure 13: Turnover Proportion

2. **Constraint is Mean-Variance Maximization**

(a) **No Cost**

Factors (K)	Sharpe	Mean Returns	Standard Deviation
1	0.748	18.367%	26.641%
3	0.968	61.397%	75.15%
5	0.498	20.801%	18.692%

Table 7: Results for No Cost Scenario

(b) **With Cost**

Factors (K)	Sharpe	Mean Returns	Standard Deviation
1	0.848	24.627%	30.119%
3	0.489	14.286%	59.499%
5	0.638	26.904%	16.450%

Table 8: Results with Costs

(c) **Short Comparison**

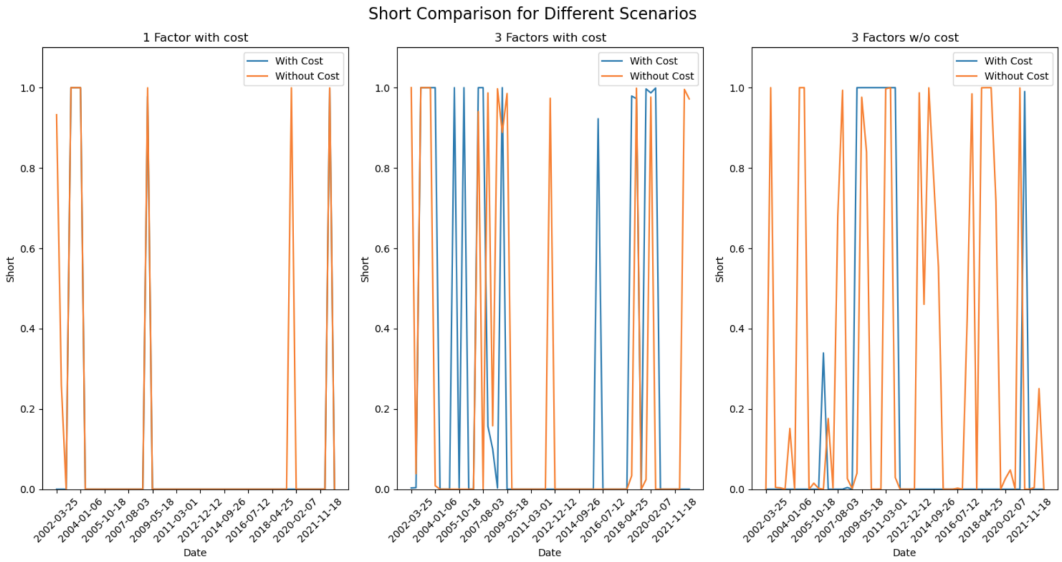


Figure 14: Short Proportion

(d) **Turnover Comparison**

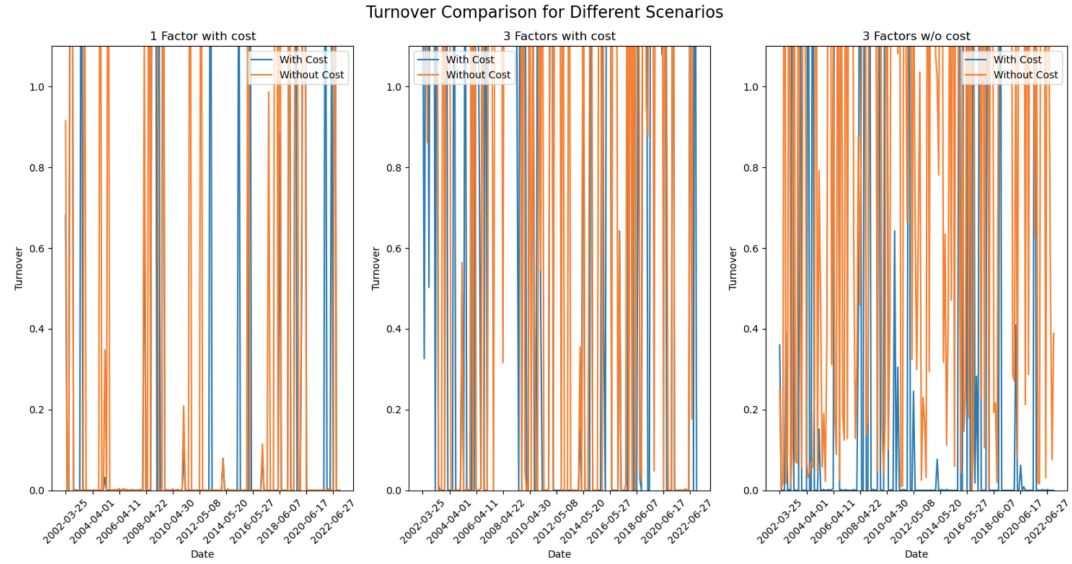


Figure 15: Turnover Proportion

4.3 Model 3 Convolution Neural Network and Transformer with FFN

1. **Constraint is Sharpe Maximization**

   (a) **No Cost**

Factors (K)	Sharpe	Mean Returns	Standard Deviation
1	0.693	9.439%	13.620%
3	0.869	4.343%	4.997%
5	0.215	2.401%	11.167%

Table 9: Results for No Cost Scenario

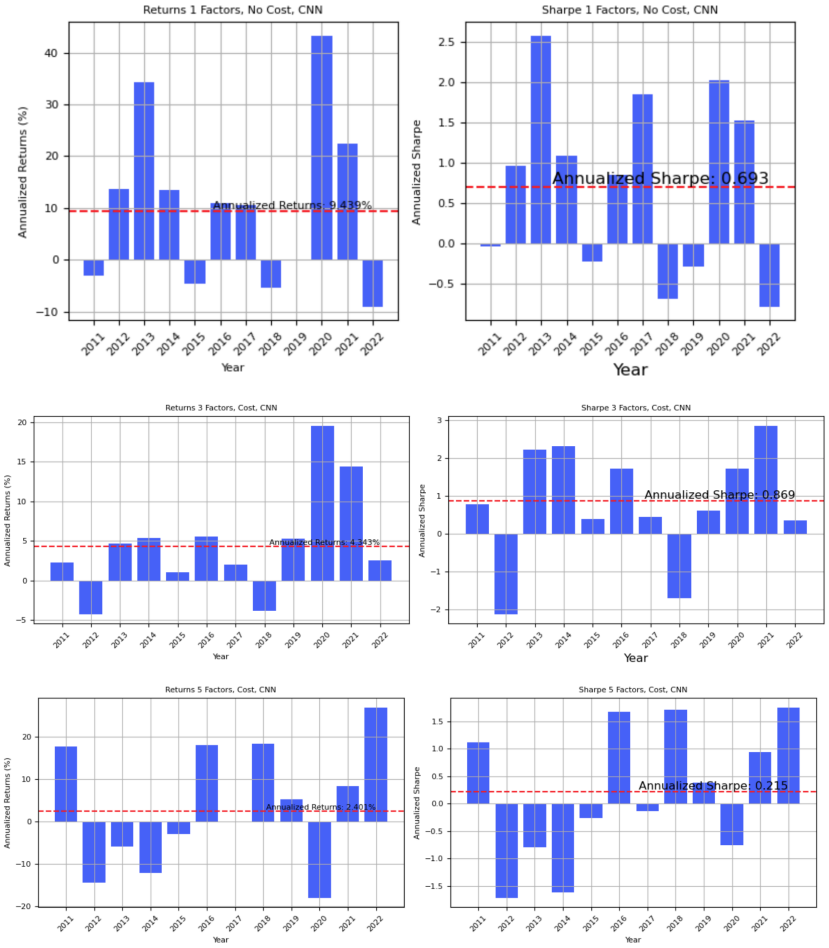


Figure 16: No Cost CNN Results

(b) **With Cost**

Factors (K)	Sharpe	Mean Returns	Standard Deviation
1	0.936	10.817%	11.556%
3	0.247	1.813%	7.340%
5	0.669	8.788%	13.136%

Table 10: Results for Cost Scenario

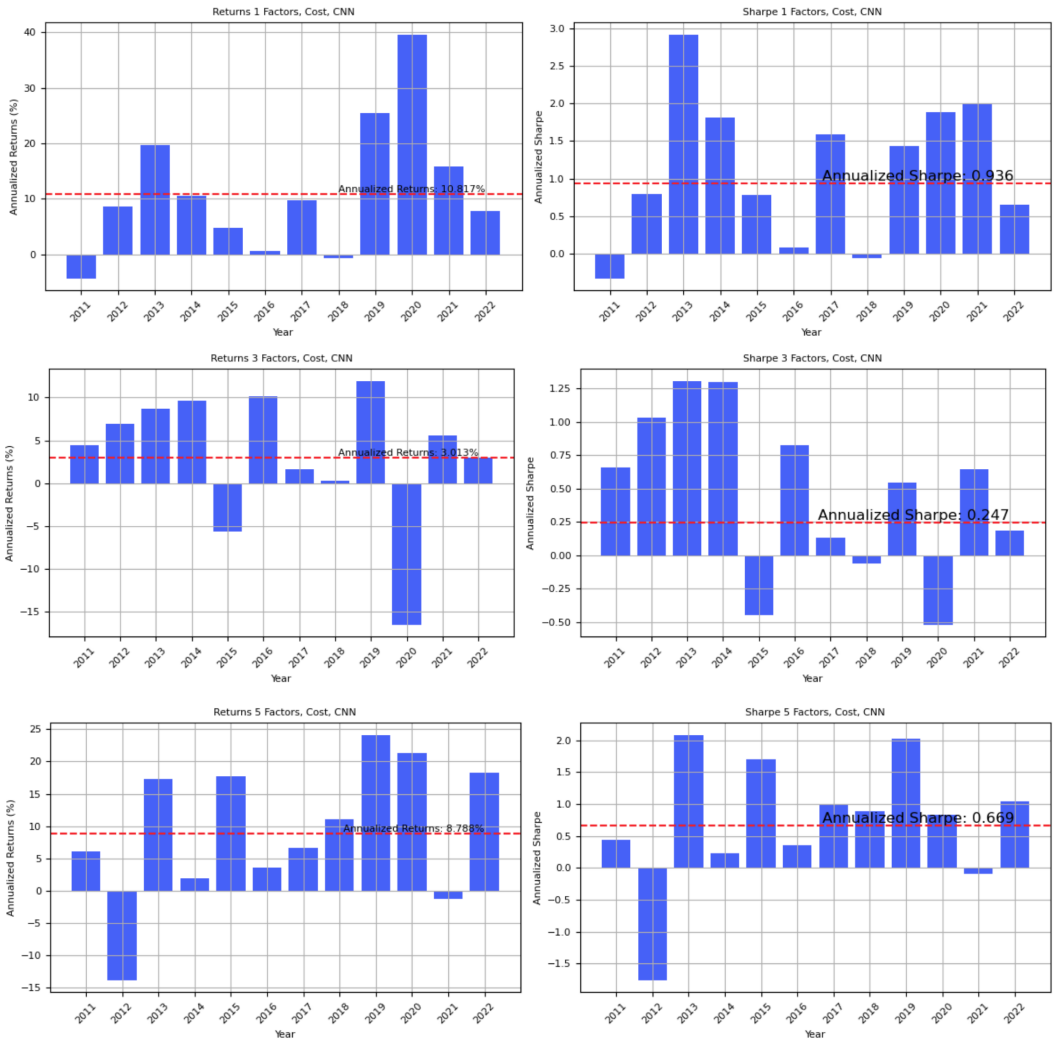
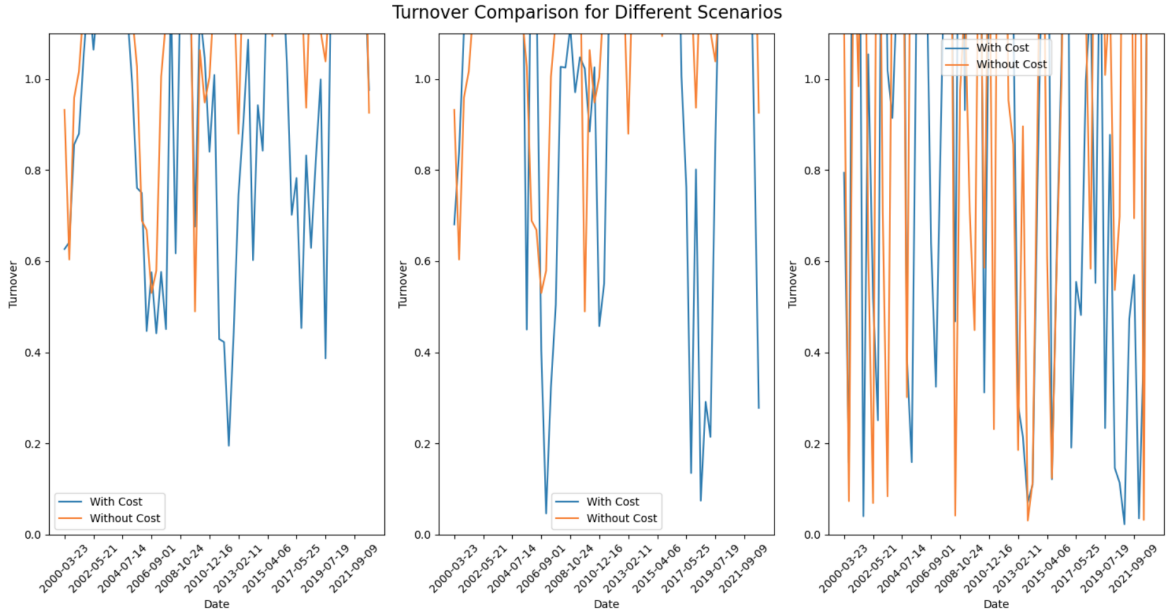


Figure 17: Cost CNN Results

(c) **Turnover**

0.9



(d) **Drawdown Comparison**

Factor	Sharpe No Cost Drawdown	Sharpe Cost Drawdown
1	15.6%	46.97%
3	12.02%	57.67%
5	11.399	26.37%

Table 11: Drawdown

(a) **Constraint is Mean Variance Maximization**

Selected results:

The good results are shown below for Mean-Variance Optimization.

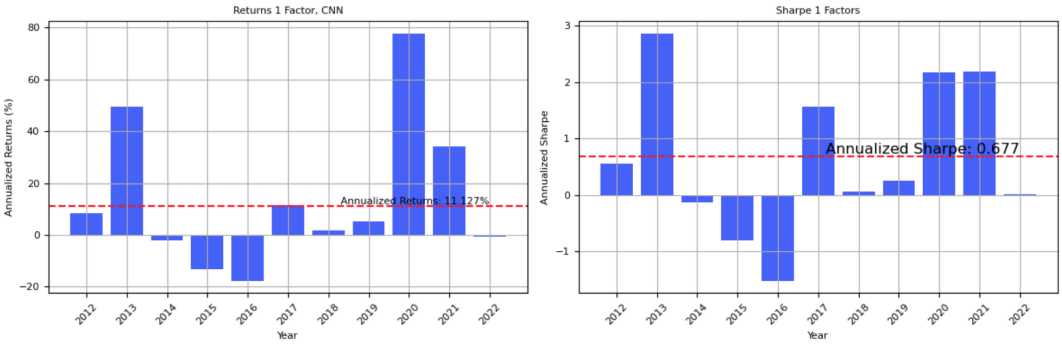


Figure 18: K= 1 Mean Var With Cost

4.4 Model 4 Results with FX

A brief introduction to results is presented here and a detailed result will be provided in the additional project report we submit.

The sharpe was observed to be 0.25, and mean returns were to the tune of 2%. A couple of scenario graphs are provided below:

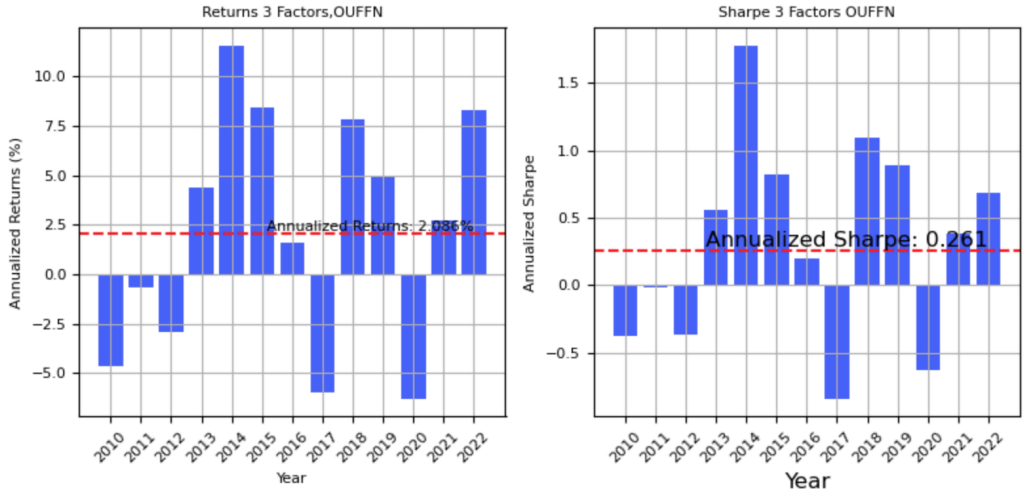


Figure 19: K=1 OUFFN Returns

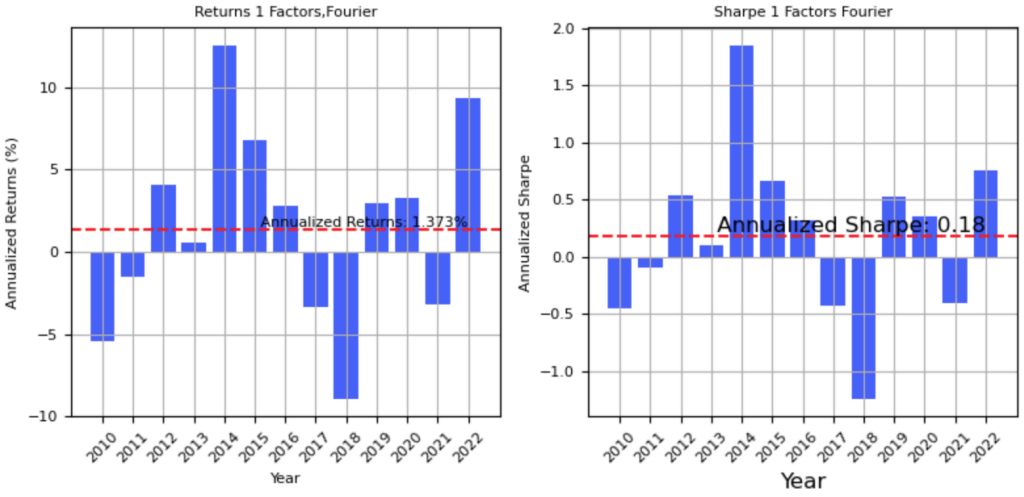


Figure 20: K=1 Fourier Returns

5 Benchmark Results

1. The S & P 500 yearly returns data spanning from 1998 to 2022 serves as a benchmark for assessing the performance of any other strategy. The S & P 500 returns on average 7 % per annum with annualized volatility of 13%. The returns are lower as compared to both of our ML based models. The OU + FFN based model returns on average 22% for 1 Fama French factor based residual and 25 % for 3 Fama French based residuals.

   Similarly, Fourier + FFN has an annualized return of 38 % in case of FF1 model and 32 % in case of a FF3 model.

2. S&P 500 has a sharpe ratio of 0.38 during the period between 1998-2022. Compared to this our OU + FFN had a sharpe ratio 0.68 for both of aforementioned cases and, Fourier + FFN had a sharpe ratio of 1.21 and 0.83 for the two cases with 1 factor and 3 factor residuals respectively.

3. This proves the superiority of our model as compared to the benchmark model of S&P 500 in both absolute and risk adjusted terms.

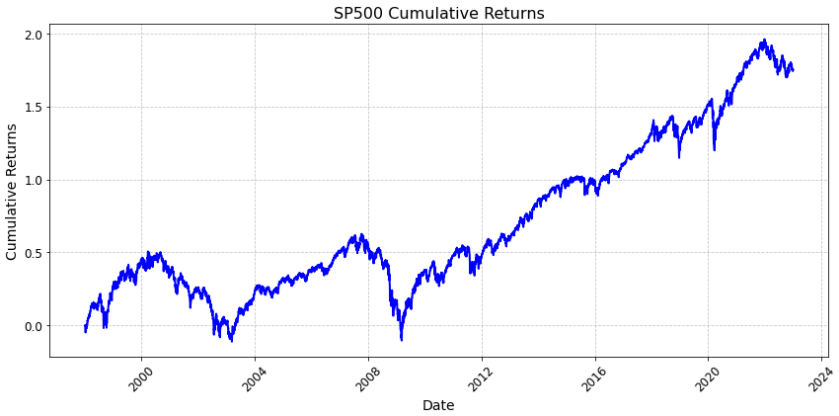


Figure 21: SP500 Cumulative Returns

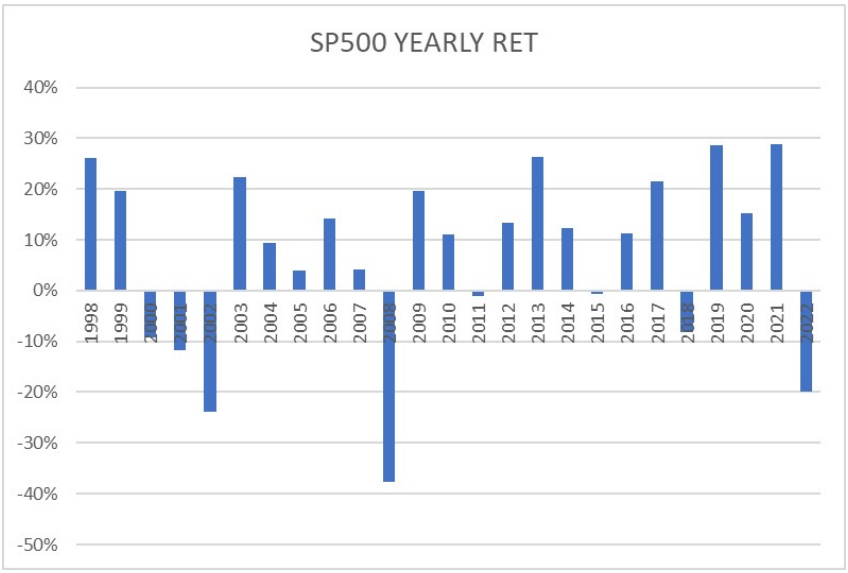


Figure 22: S&P 500 Yearly Returns

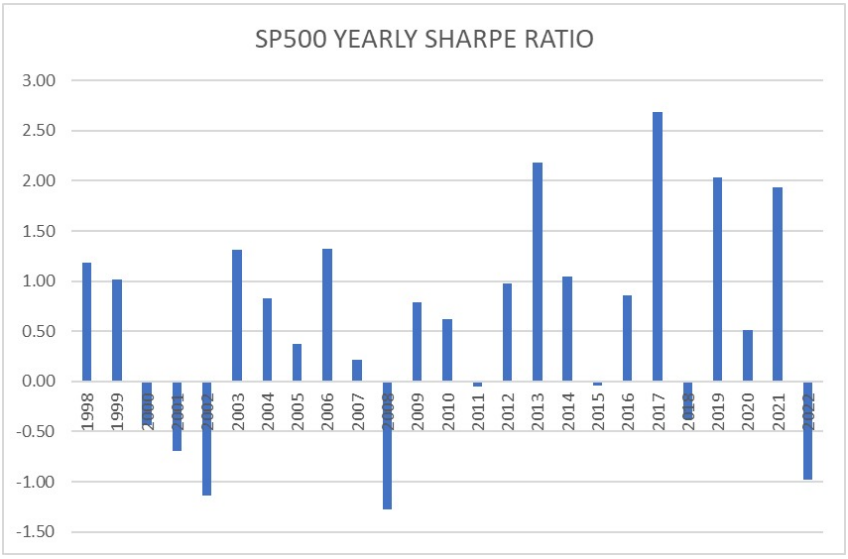


Figure 23: S&P 500 Yearly Sharpe

4. **Returns:** The Sharpe with various factors is about 0.7, and annualized means hover in the range of 18-28 %. Similarly, the annualized standard deviation hovers in the range of 18-22 %. Our numbers are slightly better than what has been reported by the original author. Some of that can be attributed to the higher epochs we ran (250).

   Short: In Figure 5, we can see that the system shorts heavily when the market is in a downturn. We see there are short positions spike in 2008.

5. **Ornstein-Uhlenbeck Versus Fourier Transform for Arbitrage Signal Generation:** Fourier has slightly better performance than Ornstein Uhlenbeck. It can be seen when comparing Table 1 with Table 3. The results (all three, Sharpe, Returns, and Standard Deviations) are higher in the case of Fourier generated Arbitrage signals.

6 Issues Faced in Replication and our solutions

1. The first roadblock we faced was the availability of replicable data. We did not receive data from the authors and hence had to go with the best proxy available in the market. Instead of 0.1% top liquid stocks, we took S&P 500 as a proxy and took data as described in the methodology section above.
2. In the absence of GPUs, the code took significantly longer to execution and, thus curtailing hyper-parameter tuning options.
3. The underlying problem seems like a non-convex problem, mainly because of the presence of a large number of stocks(1100+) over 10+ years. Hence, despite random initialization, we were getting stuck at local minima. Our solution was different seed initialization to avoid these. This helped us avoid some issues but also took significant computational resources. Because of this, the returns with cost are coming larger than without cost. Essentially, this was a 'discovery' problem.
4. The code blocks could have been made more intuitive by the authors of the paper. It took a significant amount of time to understand and debug the code. Moreover, one of the biggest issue was that code is based on residual outputs/ signals which was not entirely intuitive as to what they represented.
5. Results at lower epochs were non-intuitive including short proportions and return characteristics. We ended up getting negative Sharpe which we later improved on increasing the number of epochs

7 Future Scope

1. **CNN + Transformer:** In the extension to the project, we plan to leverage the machine learning model that combines a Convolutional Neural Network (CNN) with a Transformer architecture.

This hybrid approach is tailored to our specific problem, drawing on the strengths of both components. CNNs, known for their success in computer vision and pattern detection, serve as flexible local filters, capturing data-driven patterns within the time series. On the other hand, Transformers, widely recognized in sequence modeling like Natural Language Processing, excel in capturing complex dependencies between these local patterns. The CNN+Transformer model first generates a time-series signal by extracting local patterns and then models allocation using a flexible data-driven approach. The CNN identifies local filters of varying sizes, while the Transformer estimates temporal interactions between these filters. This innovative combination of techniques allows us to create a dynamic trading signal that encapsulates the intricate patterns within the data, ultimately improving our understanding of market dynamics.

2. **Modifications to existing sub-parts**

   (a) **Errors:** The estimation of errors is being done presently using Fama French Betas calculated over a rolling period of 60 days. A better estimation of changing beta's is using the Kalman Filter. We can also try estimating the residuals using LSTM as an adventurous route.

   (b) **Model Architecture:** While the paper explores CNN+Transformer, we are considering exploring alternative architecture to get better performance results. This includes exploring ensemble methods, GANs and reinforcement learning architecture.

   (c) **Arbitrage Signal:** Authors' parametric methods are OU and FFT. We can explore other stochastic parametric methods for arbitrage signal generation.

   (d) **Neural Network:** Hyperparameter tuning of epochs, batch size, re-train frequency, holding period, and hidden layers are all areas we intend to explore in the future. signal generation.

   (e) **Trading Frequency:** The study only considered daily data. It would be interesting to see if the results can be replicated at higher frequency of say 4-hourly or hourly. We expect to find higher alpha there.

3. **FX, Commodities**

   We intend to explore the applicability of such a trading infrastructure on expanded FX asset class. Here, we can also work towards modifying the residual definition of the FX and use it for signal prediction.

   Adapting the current codebase to Commodities is challenging because Commodities are usually traded in futures markets and futures markets have several features not present in equities like futures roll and instrument maturity. Similarly spot FX is tradable and liquid but it is not a 'lit' market unlike S&P500 equities. We have begun to explore factor models similar to Fama-French, but for FX. A paper (Appendix 9.11) explores a three-factor model which we plan to explore in the second part of this course ZB.

8 Critical Review

8.1 Model Choice

1. The model described in the paper has no explicit theoretical constraints that would prevent it from constructing highly skewed portfolios. This means that the model can generate portfolios that are heavily concentrated in a small number of assets while being very underweight in most assets. This could create several problems, such as increased risk and liquidity issues.

8.2 Underlying Principles

1. Paper uses powerful deep learning architectures that can learn complex patterns in data. However, they can also be prone to overfitting, meaning they can learn noise in the data and the real patterns. This can lead to suboptimal results, especially when the model is applied to new data that does not contain the same noise patterns. Additionally, if the number of global patterns the transformer learns is misspecified for new data, the model may break down and produce suboptimal results.
2. If a simpler model is true, then we would have wasted our efforts by creating a more complex model. This is because a simpler model would be easier to understand and interpret, and it would likely be more robust to changes in the data. Additionally, a simpler model would be more likely to generalize well to new data, as it would not be as prone to overfitting.
3. Transformers can be sensitive to the choice of hyperparameters. It can be difficult to find the optimal hyperparameters for a given task. This can lead to sub-optimal performance if the hyperparameters are not tuned carefully.
4. Model can be challenging to interpret. The underlying architecture for FFT and transformers is not very transparent. This can make it difficult to understand how the model is making its predictions.

8.3 Execution and practical implementation

1. The model described in the paper assumes that transaction costs are static at 5bps for buying and an additional 1bp for short-selling. However, transaction costs can vary depending on several factors, such as the size of the trade, the liquidity of the asset, and the time of day. This means that the assumption of static transaction costs may not accurately reflect the true cost of trading. As a result, the model may be sub-optimally designed, as it may not take into account the full cost of trading.
2. The authors did not provide data to replicate the paper.

9 Appendix

In appendix, we list down different papers which we studied to better understand other work which is being done in the field and to take best ideas for implementation.

9.1 Ensembling and Dynamic Asset Selection for Risk-Controlled Statistical Arbitrage

The authors propose a general approach for risk-controlled trading based on machine learning and StatArb. The approach employs an ensemble of regressors that are heterogeneous in three ways:
The regressors can be any state-of-the-art forecasting algorithms. The regressors are trained on a diversified feature set that includes lagged daily prices returns and a series of technical indicators. The regressors can be trained on data from individual assets or aggregated assets' data pertaining to the same industry. After the assets have been ranked in descending order, the authors propose the use of a dynamic asset selection that looks at the past and influences the ranking by removing assets with bad past behavior. The strategy then buys the bottom k assets and sells the top k assets.
The authors evaluate their approach on a number of financial datasets and show that it outperforms a number of baseline strategies.

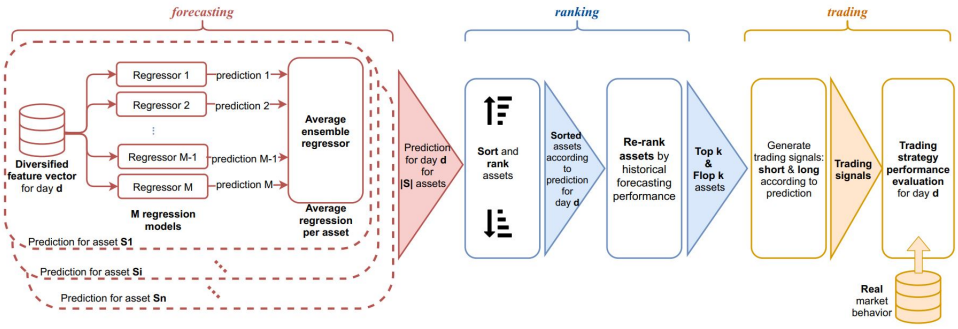


Figure 24: Architecture of the proposed general approach for risk controlled trading.

The results from the paper are shown below.
This paper is a great complement to our existing paper and provides a toolbox using ensemble ML methods for StatArb. Clearly, we can also employ dynamic allocation here using neural network to obtain superior performance.

9.2 Forecasting Returns with Network-based metrics

Introduction

The author Dr Eduard Baitinger, in the paper Forecasting Asset Returns with Network-based metrics: A Statistical and economic analysis explores latent factors not directly observable in the market to predict asset returns. He uses graph theory and topological measures to generate network-engineered parameters

	ARIMA	RF	LGB	SVR	ENS	ENS-DS $T = 30$	ENS-DS $T = 40$	ENS-DS $T = 60$	5-DAY	Buy & Hold
Return p.a	0.0569	0.0410	0.1032	0.0924	0.2198	0.2268	**0.2278**	0.2228	0.1579	0.0755
Excess return p.a.	0.0503	0.0346	0.0964	0.0856	0.2103	0.2192	**0.2202**	0.2153	0.1507	0.0681
Standard dev p.a.	0.1931	0.1775	0.1647	0.1381	0.1624	0.1606	**0.1606**	0.1607	0.2519	0.2227
Mean	0.0003	0.0002	0.0004	0.0004	0.0008	**0.0009**	**0.0009**	0.0008	0.0007	0.0004
Min	-0.0936	-0.0817	-0.0848	-0.0916	**-0.0756**	**-0.0756**	**-0.0756**	**-0.0756**	-0.1257	-0.0984
Q1	-0.0047	-0.0042	-0.0037	-0.0037	**-0.0038**	**-0.0038**	**-0.0038**	**-0.0038**	-0.0051	-0.0048
Median	0.0001	-0.0003	0.0002	0.0000	0.0005	0.0006	0.0005	0.0005	0.0001	**0.0008**
Q3	0.0052	0.0040	0.0044	0.0041	0.0049	0.0048	0.0049	0.0049	0.0055	**0.0062**
Max	0.0830	0.1385	0.1043	0.0575	0.1005	0.1005	0.1005	0.1005	**0.1827**	0.1452
Skewness	-0.0863	1.4347	0.4205	-0.1378	0.8461	1.0756	1.1136	1.0221	**1.7294**	0.2227
Kurtoisis	9.7357	23.6643	17.1581	13.2533	15.1040	15.0343	**15.5535**	14.8518	**25.5818**	13.3642
Standard error	0.0003	0.0002	0.0002	0.0002	0.0002	0.0002	0.0002	0.0002	0.0003	0.0001
t-statistic	1.1363	0.9319	2.0127	2.1021	4.5401	4.6844	4.6711	4.6199	3.2354	1.9690

Figure 25: Architecture of the proposed general approach for risk controlled trading.

that are useable to estimate expected returns.

Methodology

The author uses asset returns in the entire consideration space to construct a correlation matrix. The correlation matrix tracks the linear relationship between returns of assets with each other. The correlation matrix leads to Pearson's distance matrix, where closely correlated assets will have a lower distance and vice versa. Pearson's distance matrix is used to construct a Minimum Spanning Tree (MST) and weighted Adjacency Matrix. This MST is a breathing organism that captures latent factors of how other stocks and their characteristics impact the network. Minimum Spanning Tree leads us to various variables: closeness, betweenness, similarity, eigenvalues, degree of each node, average distance in the tree, etc. A diagrammatic representation is given below.

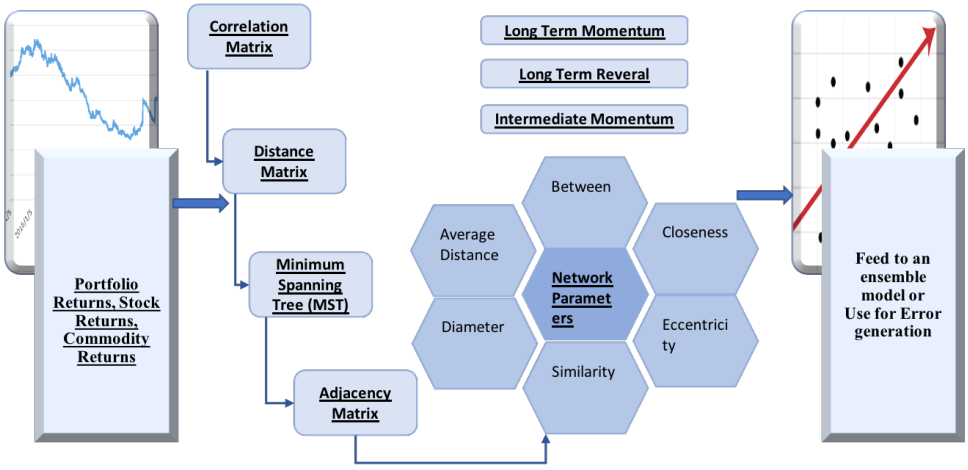


Figure 26: Architecture of the proposed general approach for Network Parameters

Multi-dimensional parameters across stock returns and fundamentals can also be used for distance matrix calculation. This adds robustness to the parameters.

Advantages

- This paper provides a novel estimation of latent factors not readily observable in the market.
- The predictive power of network theory in stock returns lies in its ability to identify systemic risk, transmission channels, and contagion effects between assets. Detecting central nodes (important assets) and understanding their interactions can provide valuable insights into market trends and potential spillover effects during periods of volatility or financial crises. These stocks are leading indicators.

9.3 Empirical Asset Pricing via Machine Learning

The paper *Empirical Asset Pricing via Machine Learning* by Shihao Gu, Bryan Kelly and Dacheng Xiu, 2018, is a seminal work in the intersection of finance and machine learning. It presents a novel approach to asset pricing by leveraging machine learning techniques, providing a fresh perspective on the empirical analysis of asset prices. In the context of our research on stock return prediction using Prophet and Random Forest algorithms, Shihao Gu, Bryan Kelly and Dacheng Xiu's work serves as a valuable reference. It not only underscores the potential of machine learning in financial research but also provides a robust framework that can be extended and refined in our study. The paper's relevance stems from its innovative methodology, which addresses the limitations of traditional asset pricing models, such as their reliance on linear relationships and assumptions of normality. By applying machine learning techniques, the authors are able to model complex, non-linear relationships and capture a wide range of empirical regularities in asset prices. Furthermore, the paper's findings have significant implications for portfolio construction and risk management. The machine learning-based asset pricing model can be used to predict returns, which are crucial inputs for these tasks.

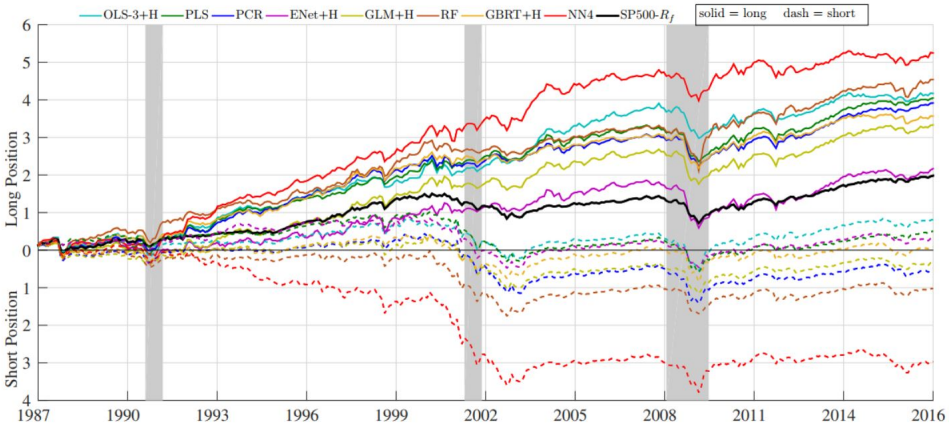


Figure 27: Empirical Asset Pricing via Machine Learning: Cumulative log returns of portfolios sorted on out-of-sample machine learning return forecasts. The solid and dash lines represent long (top decile) and short (bottom decile) positions, respectively. The shaded periods show NBER recession dates. All portfolios are value weighted

9.4 Machine Learning and Factor-Based Portfolio Optimization

This paper examined the characteristics and benefits of latent factors generated from machine learning dimensionality reduction techniques including PCA, PCS, novel approach including regularized versions that induce sparsity through penality in the objective functions and also including autoencoders as unsupervised neural network for dimensionality reduction. Their main contribution is constructing factors in machine learning and explore the impact that the proposed latent factors have on the structure of factor-based covariance matrices and to the composition and performance of minimum-variance portfolios.

The results show significant outperformance. The machine learning-based portfolios consistently outperformed the benchmark, exhibiting significantly lower standard deviation and higher Sharpe ratios. By employing a covariance matrix derived from machine learning factors, out-of-sample standard deviation decreased by up to 29% and the Sharpe ratio increased by more than 25% compared to the baseline portfolio.

Among the different machine learning methods, factors based on sparse principal component analysis (PCA) and autoencoders demonstrated the best performance. Surprisingly, shallower neural networks outperformed those with more hidden layers. Additionally, estimators allowing time-varying loadings or error covariance showed superior performance compared to static or dynamic covariance specifications.

Interestingly, the factor models that diverged the most from the sample covariance structure were the ones that generated the highest portfolio performance. Specifically, latent factor models based on PCA, sparse PCA, and an autoencoder with one hidden layer exhibited the most significant divergence from the sample estimator and achieved the best results.

Portfolio performance for different levels of transaction costs

A. Transaction costs of 5 bps

	SD	SR						
EW	4.159	0.183						
Sample	3.470***	0.203						
	Static Factor Covariance		Dynamic Beta Covariance		Dynamic Factor Covariance		Dynamic Error Covariance	
	SD	SR	SD	SR	SD	SR	SD	SR
Market	3.630***	0.203	3.346***	0.209	3.640***	0.204	3.596***	0.197
FF3	3.521***	0.212	3.327***	0.23	3.538***	0.206	3.468***	0.215
PCA	3.361***	0.230**	3.290***	0.234**	3.359***	0.230**	3.267***	0.234**
PLS	3.329***	0.224	3.276***	0.226	3.334***	0.222	3.216***	0.231
SPCA	3.373***	0.236**	3.306***	0.237**	3.377***	0.236**	3.262***	0.239**
SPLS	3.339***	0.228	3.260***	0.236*	3.346***	0.226	3.224***	0.234
AEN1	3.317***	0.235**	3.284***	0.238***	3.314***	0.234**	3.215***	0.237**
AEN2	3.351***	0.227**	3.296***	0.230**	3.349***	0.227**	3.258***	0.229*
AEN3	3.337***	0.233**	3.282***	0.234**	3.333***	0.233**	3.234***	0.238**
AEN4	3.356***	0.22	3.300***	0.221*	3.355***	0.219	3.255***	0.219

B. Transaction costs of 20 bps

	SD	SR						
EW	4.159	0.183						
Sample	3.471***	0.185						
	Static Factor Covariance		Dynamic Beta Covariance		Dynamic Factor Covariance		Dynamic Error Covariance	
	SD	SR	SD	SR	SD	SR	SD	SR
Market	3.629***	0.186	3.344***	0.182	3.641***	0.186	3.595***	0.173
FF3	3.520***	0.194	3.323***	0.197	3.537***	0.186	3.464***	0.19
PCA	3.361***	0.217	3.291***	0.218	3.359***	0.217	3.266***	0.213
PLS	3.328***	0.209	3.276***	0.208	3.333***	0.207	3.213***	0.208
SPCA	3.371***	0.223*	3.305***	0.221*	3.376***	0.222*	3.26***	0.217
SPLS	3.337***	0.213	3.260***	0.217	3.345***	0.211	3.222***	0.211
AEN1	3.315***	0.220*	3.282***	0.221*	3.312***	0.220*	3.213***	0.215
AEN2	3.349***	0.213	3.295***	0.213	3.347***	0.213	3.256***	0.208
AEN3	3.335***	0.218	3.281***	0.217	3.331***	0.218	3.232***	0.216
AEN4	3.355***	0.205	3.299***	0.204	3.354***	0.204	3.253***	0.196

This table presents monthly portfolio performance measured using the standard deviation (SD) and Sharpe ratio (SR), after transaction costs are taken into account. Panel A reports the results for transaction costs of 5 bps, while Panel B presents the results for transaction costs of 20 bps. The out-of-sample period is from January 1980 to December 2019. The results are presented for the equal-weighted portfolio (EW) and minimum-variance portfolios based on the sample estimator (Sample) and four factor-implied covariance specifications: static factor covariance, dynamic beta covariance, dynamic factor covariance and dynamic error covariance. The factor specifications are based on the single factor model (Market), the Fama-French 3-factor model (FF3), principal component analysis (PCA), partial least squares (PLS), sparse principal component analysis (SPCA), sparse partial least squares (SPLS) and autoencoders with 1, 2, 3 and 4 hidden layers (AEN). The significant outperformance of the alternative strategies from the equal-weighted strategy is denoted by: *, **, and *** for significance at the 10%, 5%, and 1% level, respectively.

Figure 28: Machine Learning and Factor-Based Portfolio Optimization

9.5 Alpha Go Everywhere: Machine Learning and International Stock Returns

Following the trail blazed by the above paper, the paper *Alpha Go Everywhere: Machine Learning and International Stock Returns* further extends the application of machine learning techniques in the realm of finance. This paper broadens the scope by applying machine learning algorithms to predict stock returns in international markets. The relevance of this extension lies in its demonstration that machine learning techniques are not confined to domestic markets but can be effectively applied globally. This underscores the versatility and robustness of machine learning in capturing complex patterns in diverse and dynamic international markets. The findings of this paper further reinforce our research direction, emphasizing the potential of machine learning, including Prophet and Random Forest algorithms, in predicting stock returns and aiding in effective stock selection for portfolio optimization.

	Equal-Weighted										Value-Weighted									
	Market	OLS-3	OLS	LASSO	RIDGE	NN1	NN2	NN3	NN4	NN5	Market	OLS-3	OLS	LASSO	RIDGE	NN1	NN2	NN3	NN4	NN5
Sharpe Ratio	0.96	1.32	2.53	2.39	2.59	3.75	3.81	3.73	3.90	3.78	0.53	0.78	0.85	0.90	1.04	1.45	1.67	1.65	1.69	1.65
Drawdowns and Turnover																				
Max DD (%)	50.78	35.7	35.69	33.99	35.73	21.04	17.90	22.60	16.42	24.24	67.53	30.61	30.89	32.29	30.35	28.53	25.82	35.15	24.83	27.41
Max 1M Loss (%)	19.41	27.35	25.58	24.57	25.58	17.17	15.81	18.58	14.47	19.86	27.89	14.61	22.71	15.54	22.71	24.14	24.95	25.62	21.57	24.25
Turnover (%)		54.16	144.9	153.19	145.03	140.05	142.25	142.28	142.58	142.89		60.27	155.71	161.08	155.64	148.31	150.66	151.29	152.52	151.87
Risk-adjusted Performance using FF5 + Mom model, 1992-2017																				
Mean Return (%)		1.66	2.81	2.65	2.82	4.04	4.11	4.12	4.16	4.10		0.99	1.54	1.31	1.54	2.10	2.13	2.03	2.19	1.95
α		0.94	2.36	2.19	2.37	3.66	3.82	3.80	3.84	3.82		1.21	1.46	1.32	1.47	2.04	2.10	2.09	2.12	2.01
$t(\alpha)$		5.39	10.43	9.44	10.47	15.84	16.11	15.20	16.86	15.27		4.17	4.49	3.97	4.52	6.13	6.75	6.53	7.06	6.96
R^2 (%)		64.59	29.64	28.66	29.60	21.14	16.65	17.06	17.93	18.44		6.74	3.77	3.03	3.79	4.45	3.29	5.64	3.65	3.84
Information Ratio		0.38	0.73	0.66	0.74	1.11	1.13	1.07	1.18	1.07		0.29	0.32	0.28	0.32	0.43	0.48	0.46	0.51	0.49
Risk-adjusted Performance using HKK model, 1990-2010																				
Mean Return (%)		1.74	3.31	3.36	3.30	4.82	4.92	4.95	4.98	4.92		0.89	1.29	1.26	1.29	2.08	2.18	2.07	2.29	2.09
α		1.43	3.22	3.23	3.21	4.74	4.78	4.87	4.76	4.86		1.00	0.98	0.92	0.98	1.87	1.89	1.87	2.16	1.96
$t(\alpha)$		5.08	10.96	10.73	11.26	16.77	16.77	16.39	16.95	15.71		3.29	3.08	2.80	3.09	5.59	6.87	5.75	7.15	6.48
R^2 (%)		30.22	7.91	6.26	8.10	5.34	5.15	5.12	6.40	5.25		2.20	5.60	5.56	5.71	5.26	3.97	5.28	4.09	3.09
Information Ratio		0.34	0.74	0.73	0.76	1.13	1.13	1.11	1.15	1.06		0.22	0.21	0.19	0.21	0.38	0.46	0.39	0.48	0.44
Risk-adjusted Performance using KW model, 1990-2010																				
Mean Return (%)		1.70	3.28	3.31	3.28	4.76	4.88	4.90	4.87	4.88		0.90	1.41	1.37	1.42	2.10	2.28	2.14	2.29	2.16
α		1.13	3.09	3.18	3.10	4.70	4.86	4.84	4.89	4.85		0.80	1.30	1.22	1.30	2.01	2.31	2.09	2.24	2.10
$t(\alpha)$		4.60	10.63	10.51	10.98	16.48	16.38	16.09	16.68	15.40		2.51	3.93	3.54	3.94	5.76	6.75	6.23	7.53	6.69
R^2 (%)		50.82	17.66	13.82	18.24	11.54	10.53	9.82	9.49	10.18		3.57	1.72	1.27	1.77	3.27	2.88	3.79	2.17	1.76
Information Ratio		0.32	0.74	0.73	0.77	1.15	1.14	1.12	1.16	1.08		0.18	0.27	0.25	0.28	0.40	0.48	0.44	0.52	0.47

Figure 29: Alpha Go: All stocks are pooled together and sorted into deciles based on predicted returns in the next month. Predictions are based on the machine learning models estimated with the data of the 36 stock characteristics (listed in Appendix A) of all stocks and 31 market dummies. We report the annualized Sharpe Ratio of equal- and value-weighted long-short portfolio returns of individual stocks. Max DD is the maximum drawdown of the long-short portfolio. Max 1M Loss is the lowest monthly return of the long-short portfolio. α is the intercept from regressing monthly long-short portfolio return onto a factor model. FF5+Mom refers to a 12-factor model that includes the international Fama-French five factors plus a momentum factor for developed markets and for emerging markets (available from 1992 to 2017). HKK refers to the 6-factor model in Hou et al. (2011) (available from 1990 to 2010), and KW refers to the partial-segmentation Carhart model in Karolyi and Wu (2018) (available from 1990 to 2017). The t-statistics of α and the R^2 of the regression are reported. Information ratio equals α divided by the standard deviation of the residuals from the regression. The testing sample is from 1992 to 2017. The model that generates the highest Sharpe Ratio, mean return, α, and information ratio, or the lowest Max DD, Max 1M Loss, Turnover, and R^2 is highlighted in color.

9.6 AlphaPortfolio: Direct Construction Through Reinforcement Learning and Interpretable AI

The paper introduces a novel portfolio management framework that outperforms traditional indirect approaches. It utilizes a multisequence, attention-based deep learning model, leveraging cutting-edge AI innovations to capture the complexities of financial data and market conditions effectively. The model is optimized through reinforcement learning (RL), allowing it to incorporate interactions between trading and market states. The resulting AlphaPortfolio demonstrates outstanding out-of-sample performance across various economic and trading restrictions in the U.S. equity market, making it a valuable tool for practitioners in trading and investment advising.

The research also highlights broader implications for the use of RL in social sciences and the importance of economically interpretable AI. RL offers a goal-directed learning approach for complex environments or action spaces with specific objectives but limited labeled data. Additionally, the economic distillation of the model reveals key firm characteristics driving AlphaPortfolio's performance, contributing to the development of interpretable AI solutions in general.

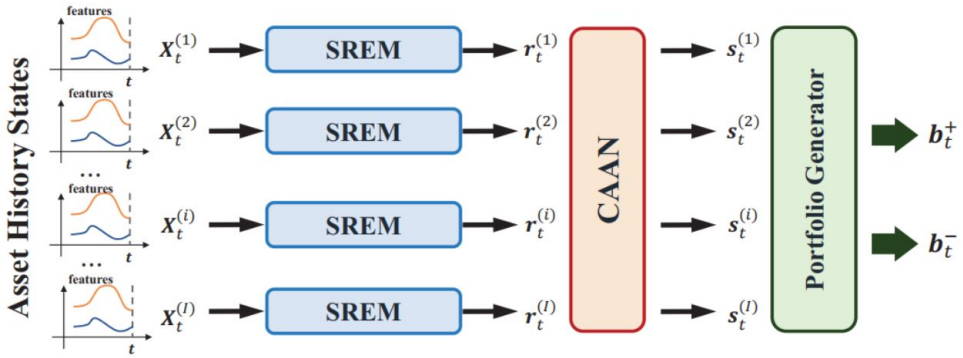


Figure 30: AlphaPortfolio Results

This table reports alphas for portfolios of long/short stocks in the highest/lowest decile of winner scores from 1990 to 2016, where (1)-(3) exclude *unrated* stocks from the portfolio and (4)-(6) exclude both *unrated* and recently *downgraded* stocks. In Panel A, portfolio returns are adjusted by the CAPM, Fama-French-Carhart 4-factor model (FFC), Fama-French-Carhart 4-factor and Pastor-Stambaugh liquidity factor model (FFC+PS), Fama-French 5-factor model (FF5), Fama-French 6-factor model (FF6), Stambaugh-Yuan 4-factor model (SY), and Hou-Xue-Zhang 4-factor model (Q4). Within (1)-(3) and within (4)-(6), the first columns present the alphas for the whole sample. The remaining four columns present alphas for subsamples excluding microcap firms in the smallest decile and quintile, respectively. Panel B presents other performance metrics in a similar fashion, with Return, Std.Dev., and Sharpe ratio all annualized. q_n symbolizes the n^{th} NYSE size percentile. "*," "**," and "***" denote significance at the 10%, 5%, and 1% level, respectively.

Panel A: Out-of-Sample Alpha Under Various Factor Models

	Excluding Unrated Firms						Excluding Unrated & Downgraded					
	(1)		(2)		(3)		(4)		(5)		(6)	
Firms	All		Size > q_{10}		Size > q_{20}		All		Size > q_{10}		Size > q_{20}	
	α(%)	R^2	α(%)	R^2	α(%)	R^2	α(%)	R^2	α(%)	R^2	α(%)	R^2
CAPM	3.4***	0.000	4.6**	0.051	5.3***	0.067	4.7***	0.000	3.8***	0.070	4.3***	0.076
FFC	3.1***	0.061	5.2***	0.380	5.5***	0.504	4.4***	0.047	4.2***	0.305	4.4***	0.437
FFC+PS	2.4*	0.070	4.6***	0.384	4.7***	0.511	3.4***	0.060	3.6***	0.309	3.4***	0.448
FF5	3.6**	0.146	4.3**	0.251	4.0***	0.375	5.0***	0.100	3.7***	0.219	2.9***	0.338
FF6	3.7***	0.147	5.6***	0.391	5.3***	0.517	5.0***	0.100	4.7***	0.308	4.0***	0.446
SY	5.3***	0.182	7.9***	0.393	7.4***	0.494	6.6***	0.149	6.9***	0.333	6.2***	0.444
Q4	3.5***	0.109	5.8***	0.294	5.6***	0.394	5.0***	0.082	5.1***	0.257	4.5***	0.350

Panel B: Other Portfolio Performance Metrics

	Excluding Unrated Firms			Excluding Unrated & Downgraded		
	(1)	(2)	(3)	(4)	(5)	(6)
Firms	All	Size > q_{10}	Size > q_{20}	All	Size > q_{10}	Size > q_{20}
Return(%)	6.18	8.30	8.99	7.45	7.54	8.06
Std.Dev.(%)	5.93	7.89	7.03	6.42	7.02	7.00
Sharpe	1.04	1.05	1.28	1.16	1.07	1.15
Skewness	-0.24	2.23	1.69	-0.89	1.18	1.18
Kurtosis	4.52	11.83	10.16	7.96	7.92	8.68
MDD	0.06	0.08	0.08	0.05	0.08	0.08

Figure 31: Alphafolio Architecture

9.7 Asset Allocation via ML and Applications to Equity Portfolio Management

This paper presents five key contributions. Firstly, it introduces a fast and convergent numerical framework applicable to various constrained optimization problems with unique solutions, without the need for specific optimization routines found in existing deep learning-based methods. Secondly, the methodology avoids estimating cross higher order moments of the asset span, which is crucial to overcome the curse of dimensionality when dealing with a large number of assets.

Thirdly, the paper suggests employing hierarchical clustering to reduce the dimension of the optimization problem in portfolios with numerous assets. Fourthly, the authors advocate the use of deep learning techniques to estimate the model input. Finally, the paper includes empirical studies on portfolio choice using a considerable number of stocks in both the U.S. and China equity markets, providing a performance analysis of the proposed approach.

Index	Return	Vol	IR	SR	MDD	CR
MVSK500	13.20%	11.27%	1.17	1.58	15.17%	0.87
MVSK1600	17.66%	14.59%	1.21	1.98	16.51%	1.07
CRRA500	14.23%	11.28%	1.26	1.78	14.10%	1.01
CRRA1600	22.49%	21.49%	1.05	2.60	16.78%	1.34
S&P500	14.23%	9.30%	1.52	2.03	13.09%	1.08

Figure 32: Performance Metrics in U.S. Stock Market

9.8 Deep Learning in Asset Pricing

The paper focuses on asset pricing and proposes a novel approach using deep neural networks to estimate a stochastic discount factor (SDF) that explains differences in asset returns.

The authors use a three-step neural network algorithm explicitly incorporating the no-arbitrage condition in the model architecture. This approach provides better risk premium signals and explains individual stock returns. Empirically, the model outperforms leading benchmark approaches and provides insights into the pricing kernel and systematic risk. The paper also includes a non-parametric adversarial estimation method to construct informative test assets using LSTM networks to extract economic conditions from complex time series.

The authors choose to use an adversarial approach to select the moment conditions that maximise the mispricing which they define as: $\min_\omega \max_g \frac{1}{N}\Sigma_{j=1}^{N}\left|\mathrm{E}\left[\left(1-\Sigma_{i=1}^{N}\omega(I_t, I_{t,i})R_{t+1,i}^{e}\right)R_{t+1,j}^{e}g(I_t, I_{t,j})\right]\right|^2$ where the function ω and g are normalized functions chosen from a specified functional class and R_i are returns of asset i.

In addition to the GAN, the authors independently constructed a FFN, and a linear and regularised linear model labeled LS and EN respectively and the respective results are below: In this and additional tables in the paper, the authors show that GAN model performs better than traditional models and is robust to training sample, stock liquidity and trading costs model.

The paper attempts to formulate a general SDF for all assets in the market and this could be used to generate model prices to be traded against in a statistical arbitrage setup especially for low liquidity assets

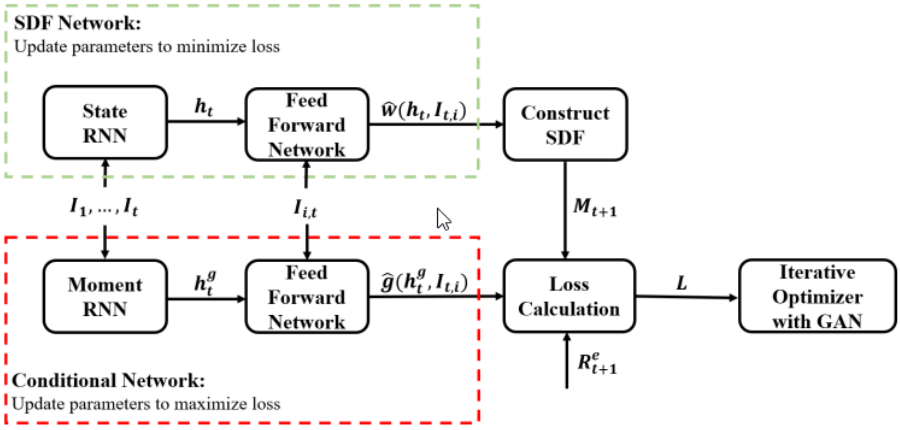


Figure 33: GAN Model Architecture: The SDF network finds a candidate SDF and the conditioning network finds the mispriced assets given the SDF

Table I: Performance of Different SDF Models

	SR			EV			XS-R^2		
Model	Train	Valid	Test	Train	Valid	Test	Train	Valid	Test
LS	1.80	0.58	0.42	0.09	0.03	0.03	0.15	0.00	0.14
EN	1.37	1.15	0.50	0.12	0.05	0.04	0.17	0.02	0.19
FFN	0.45	0.42	0.44	0.11	0.04	0.04	0.14	-0.00	0.15
GAN	2.68	1.43	0.75	0.20	0.09	0.08	0.12	0.01	0.23

This table shows the monthly Sharpe ratio (SR) of the SDF, explained time series variation (EV) and cross-sectional mean R^2 for the GAN, FFN, EN and LS model.

9.9 Deep learning with LSTM networks for financial market predictions

Deep learning with long short-term memory (LSTM) networks for financial market predictions is a research paper by Thomas Fischer and Chris Krauss that seeks to explore LSTM networks, in predicting financial market return. The paper draws its inspiration from previous research works, highlighting the capacity of machine learning to establish complex, non-linear dependencies in financial data.

It explores three key areas: methodology, data, and approach. It takes a rigorous approach to explore the LSTM's suitability in financial prediction tasks. An intro to the theory of LSTM neural networks, which are one of the most advanced deep learning techniques for sequence learning tasks is presented. It frames the prediction task by defining the features as the 240-day return sequences and developing a suitable LSTM architecture and training algorithm to generate accurate predictions. It also explores how to derive a suitable trading strategy based on the predictions.

The data set of the S&P 500 index constituents from Thomson Reuters extracted from December 1989 to September 2015 is used. The results show that LSTM networks beat the standard deep net and the logistic regression by large margins. It establishes effective trading signals with statistically and economically significant returns of 0.46% per day and sharpe ratio of 5.8 before transaction costs.

RAF – Random Forest; DNN – Deep Neural Network; LOG – Logistic Regression It also showcases how LSTM networks can effectively extract meaningful information from noisy financial time series data in

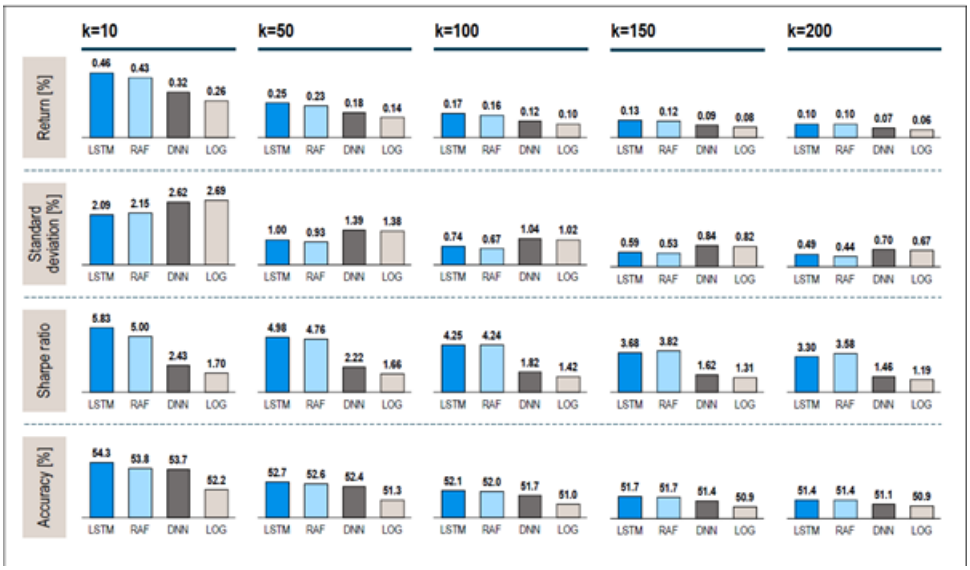


Figure 34: LSTM

a way that other popular models like random forests, deep nets, and logistic regression cannot. The authors unveil common patterns of stocks selected for profitable trading by the LSTM network, which include below-mean momentum, strong short-term reversal characteristics, high volatility, and beta. In conclusion, this paper offers a unique outlook on the prowess of LSTM neural networks in predicting financial market returns that surpasses other machine learning models previously applied in the field.

9.10 Machine Learning for Stock Selection

In "Machine Learning for Stock Selection," Keywan Christian Rasekhschaffe and Robert C. Jones juxtapose traditional quantitative finance with the new techniques of machine learning. The authors present methods to use machine learning techniques for forecasting the cross-section of stock returns while limiting overfitting, a major challenge for extracting signals. The authors aim to demonstrate the general power of machine learning for stock selection while minimizing the risk of overfitting. They describe two primary ways to reduce the risk of overfitting, namely feature engineering and forecast combinations. Feature engineering entails the application of domain knowledge to limit the risk of over-fitting, while forecast combinations focus on relationships that are robust to different forecasting techniques and training windows. The authors use a cross-sectional setting for a sample of small, medium, and large capitalization stocks from 22 developed markets and highlight the importance of feature engineering when using machine learning techniques.The factor library consisted of 194 factors assembled by IHS Markit from diverse sources. This includes 21 deep value factors, 18 relative value factors, 10 factors focused on earnings quality, 26 factors capturing earnings momentum, 26 factors focused on historical growth, 35 liquidity factors, 29 management quality and profitability factors, and 29 technical price-based factors. Excess returns are defined as returns above the riskfree rate and come from Barra. The sample

period is 1994 through 2016, with walk-forward forecasts beginning in 2004 (to allow a 10-year training period). The forecast horizon and data frequency are both monthly. These training sets are trained independently for four separate regions: the United States, Japan, Europe, and Asia ex Japan. All factors are percentiles ranked within region and industry buckets for each date. Four different algorithms are used on each of the regional training sets—a bagging estimator that used AdaBoost, a gradient boosted classification and regression tree (GBRT) algorithm, a neural network, and a bagging estimator that used a support vector machine.

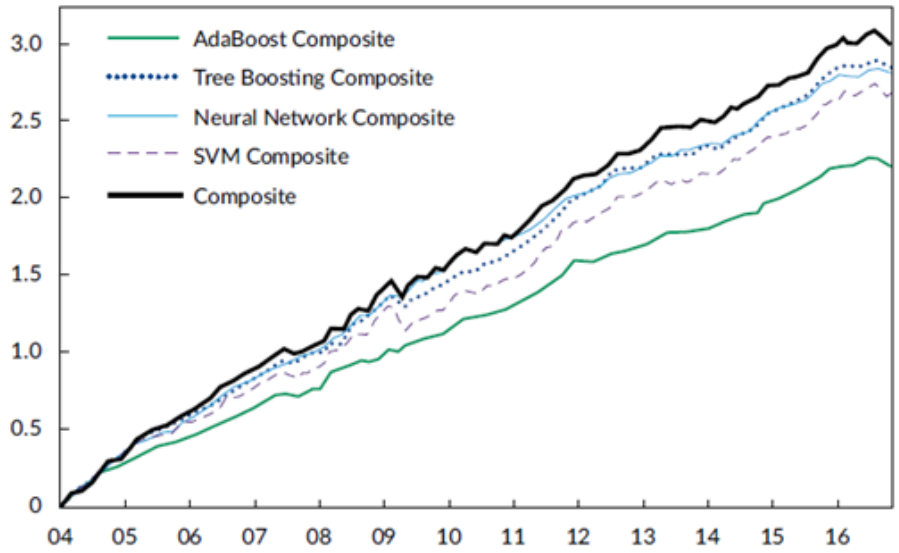


Figure 35: Machine Learning for Stock Selection

In conclusion, the paper discussed two primary ways to reduce the risk of overfitting—feature engineering and forecast combinations. Feature engineering can increase the signal-to-noise ratio by correctly framing the problem and transforming the data to produce cleaner signals.Forecast combinations reduce noise by focusing on relationships that are robust to different forecasting techniques (MLAs) and training windows. It demonstrated that, properly applied, MLAs can use a wide variety of company characteristics to forecast stock returns without overfitting. With sensible feature engineering and forecast combinations, MLAs can produce results that dramatically exceed those derived from simple linear techniques such as OLS. These MLA results are robust to various risk adjustments and work well both in the US market and in other developed markets.

9.11 Factor Investing in Currency Markets

In "Factor Investing in Currency Markets: Does it make sense?," Baku, Fortes, Herve suggest a factor model as a Fama-French equivalent but for the currency markets. This is a three-factor model that relies on carry, momentum and value factors. For each currency, they estimate the sensitivity with respect to

each risk factor, the importance of common risk factors, when specific risk does matter, etc. They also connect statistical figures with monetary policies and regimes, illustrating the high interconnectedness of market risk factors and economic risk factors. The primary goal of building an APT model for currencies is to have a framework for analyzing and comparing the behavior of currency returns. Section 2 of the paper is dedicated to the economics of foreign exchange rates. They introduce the concept of real exchange rate, which is central for understanding the different theories of exchange rate determination. Then, they focus on interest rate, purchasing power parities and identifying the statistical properties of currency returns also helps to define the market risk factors, which are presented in section 3. These risk factors are built using the same approach in terms of portfolio composition and rebalancing. Section 4 presents the cross-section and time-series analysis of each currency. We can then estimate a time-varying APT-based model in order to understand the dynamics of currency markets. The results of this dynamic model can be used to manage a currency portfolio. This is why section 5 considers hedging and overlay management.We plan to implement this paper for residual extraction when applying our original paper to a basket of FX futures instead of SP500.

10 Bibliography

- *Deep Learning Statistical Arbitrage* by Jorge Guijarro-Ordonez, Markus Pelger, and Greg Zanotti (2022).
- *Ensembling and Dynamic Asset Selection for Risk-Controlled Statistical Arbitrage* by Salvatore Carta, Sergio Consola, Alessandro Sebastian Podda, Diego Reforgiato Recupero, and Maria Madalina Stanciu (2021).
- *A general approach for risk controlled trading based on machine learning and statistical arbitrage* by Salvatore Carta, Diego Reforgiato Recupero, Roberto Saia, and Maria Madalina Stanciu (2020).
- *Statistical arbitrage in the US equities market* by Avellaneda, M. and J.-H. Lee (2010).
- *Algorithmic trading of co-integrated assets* by Cartea, A. and S. Jaimungal (2016).
- *Deep learning in asset pricing* by Chen, L., M. Pelger, and J. Zhu (2022).
- *Deep learning with long short-term memory networks for financial market predictions* by Thomas Fischer and Chris Krauss (2018).
- *Empirical Asset Pricing via Machine* by Shihao Gu, Bryan Kelly, and Dacheng Xiu (2018).
- *Alpha Go Everywhere: Machine Learning and International Stock Returns* by Darwin Choi, Wenxi Jiang, and Chao Zhang (2022).
- *Machine Learning and Factor-Based Portfolio Optimization* by Thomas Conlona, John Cotterb, and Iason Kynigakisc (2021).
- *Deep Learning for Portfolio Optimization* by Zihao Zhang, Stefan Zohren, and Stephen Roberts (2021).
- *Learning to trade via direct reinforcement* by John Moody and Matthew Saffell (2021).
- *Performance functions and reinforcement learning for trading systems and portfolios* by John Moody, Lizhong Wu, Yuansong Liao, and Matthew Saffell (1998).
- *Performance functions and reinforcement learning for trading systems and portfolios. Journal of Forecasting* by Zihao Zhang, Stefan Zohren, and Roberts Stephen (2020).
- *AlphaPortfolio: Direct Construction Through Deep Reinforcement Learning and Interpretable AI* by Lin William Cong, Ke Tang, Jingyuan Wang, and Yang Zhang (2022).
- *Asset Allocation via Machine Learning and Applications to Equity Portfolio Management* by Qing Yang, Zhenning Hong, Ruyan Tian, Tingting Ye, Liangliang Zhang (2020).
- *Investable and Interpretable Machine Learning for Equities* by Yimou Li, Zachary Simon, and David Turkington (2022).

- *Machine learning versus economic restrictions: Evidence from stock return predictability* by Avramov, D., S. Cheng, and L. Metzker (2022).
- *Asset pricing with panel tree under global split criteria* by Cong, L. W., G. Feng, J. He, and X. He (2022).