Investments: Machine Learning for Finance

Viacheslav Buchkov[◇]

***Abstract*—The paper explores in details the deep learning modelling in market risk premium prediction task. We find that the MLP predictor produces stronger and more robust results than the stronger models. We train the model to recognize the momentum versus reversal patterns. We find the improvement in performance, based on the likelihood steepness, which might be explored further in details.**

I. Introduction

Financial markets represent a complex multi-agent game with high uncertainty about the future and constantly changing dynamics due to the changing agents behavior.

In the recent years we have seen a significant rise of application of machine learning methods, adapted from other domains, to the investment portfolio construction [1]. However, due to constantly changing environment, structural shifts and high number of constantly evolving market participants, the usual applications of machine learning in finance fail to hold in out-of-sample performance, and especially in the construction of practical trading strategy [2].

Therefore, financial applications require a significant reformulation of the machine learning problem setting to actually produce a robust out-of-sample estimator. This includes careful selection of features and reformulation of the usual machine learning setting.

Furthermore, the recent papers in financial machine learning have shown an existence of so-called "Virtue of Complexity" [3], [4]. This idea argues a quite interesting paradox in financial risk premia prediction - the more complex the Empirical Risk Minimizer, the better the predictive quality of the model. Authors use Radial Basis Function approximation (produced by finite Fourier transforms) to test increasing model complexity versus the out-of-sample predictive quality. Authors find that the Sharpe ratio and OOS prediction monotonically increase with the elevated model complexity, given by the number of Fourier transformations.

One may note that a Fourier transform, combined with the Ridge regression in [3] setting is actually a way to sample from Gaussian Process with an RBF kernel. Furthermore, let's remember the fact that a shallow Neural Net can be considered as a Gaussian Process with infinite width and some complex kernel (Theorem 2 in [5]) and the fact that any complex (but L-smooth) non-linear function on a compact domain of function values can be represented by the weighted sum of some positive-definite kernels ("Representer Theorem", [6]). Therefore, such a construction from [3] serves as a great motivation for this paper, where we aim to predict the future stock market risk premia.

Even though recently the results of the "Virtue of Complexity" have been challenged ([7]), the motivation for complex non-linear predictors for financial risk premia prediction still holds - just it raises the additional attention to the details and robustness of the modeled result. The argument of [7] is based on incorrect exclusion of the intercept from the modeling, which is potentially a deliberate modeling choice that has the grounds to it. For instance, one may argue that markets are driven only by dynamic risk premia without any constant drift, that is why there should be no backdoor for the model to allocate a noise with unknown noise in the regression setting.

Finally, this paper focuses on prediction of the overlapping long-term (1 year) market risk premium prediction. The reason for that is widely motivated by financial literature, as one may be more certain in robustness of long-term prediction. The reason for that is the short-term noise, which comes from agents constantly readjusting their beliefs from new incoming information, while the long-term changes in market are rather driven by economy cycles and readjustment of fundamental trends, which can be potentially extracted from macroeconomic and flow features.

II. Objective

A. Why prediction helps in investments?

The construction of a trading strategy can be viewed mathematically through a Reinforcement Learning formalism. In financial portfolio management we work in finite-space Markov Decision Process (MDP) setting with $\mathcal{M} := \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \zeta \rangle$ ([8]), where $\mathcal{S}$ denotes the states (features that describe current market environment), $\mathcal{A}$ the set of actions (in our case, the set of weights $w = (w_1, w_2, ..., w_n) \in \mathbb{R}^n$ for all available assets in our tradeable universe with some weights being zero), $\mathcal{P}$ the state transition probability matrix (which is - contrary to the usual RL settings, independent of taken actions), $\mathcal{R}$ the reward function, which is determined by our utility function, and ζ the discount factor that controls the algorithm convergence and is different from Time Value of Money discounting.

One may note a distinct feature of the financial markets setting, where in the usual approaches to modeling (except for special cases like market-making) our chosen actions does not affect the state visitation distribution. In other words, the chosen portfolio weights does not affect, in which state the economy will transition, agents perception of the future will

◇ Master of Science UZH ETH in Quantitative Finance

change etc. It allows to simplify the usual transition by noting that the state transition probability is independent of actions

$$P(s_{t+1}|s_t, a_t) = P(s_{t+1}|s_t), \quad \forall \quad a_t \in \mathcal{A}$$

Therefore, our investment portfolio decision-making reduces to sequential search for optimal action in the series of independent decisions, also known as *contextual bandits* problem. The reason for this is the fact that at each portfolio construction date one can close all of the existing positions in the portfolio and reconstruct it from scratch, while previous portfolio affects only the amount of transaction costs beared, which is (usually) considered to be exogenous to the model.

Such a reduction allows us to consider investment portfolio construction MDP as a series of individual independent optimizations ([9]), which can be viewed as simply an approach to find the optimal action (vector w) in each rebalancing period.

The decision-making process can also be viewed as a continuous-time decision-making problem. Then at each $t + \epsilon \notin T$, where T is the set of rebalancing dates $T = \{t_1, , ..., t_n\}$ we just make a decision $w_{t+\epsilon} = w_t$ due to transaction costs versus new information trade-off. I.e., our RL agent thinks that a new filtration $\mathcal{F}_{t+\epsilon}$ is not "different enough" from the past filtration $\mathcal{F}_t$, so transaction costs beared due to rebalancing is larger than the payoff from the new information gained. In practice such a trade-off is usually implemented as a heuristic, however, one can model it directly ([10] or derivatives hedging modeling that can be reformulated for investment process RL - [11]). In this paper I focus on the heuristic approach of fixed rebalancing dates, as this particular modeling is out of the paper's scope.

Furthermore, after establishing a fixed scheme independent optimizations, one may look at the idea behind our investment process task. We work in the setting, where we usually want to maximize a return-to-risk ratio. For the numerator of such a reward one may note an interesting idea. Recall that the return of the portfolio in some period of holding is described as a dot product of our action w_t and vector of future holding period compounded cumulative simple returns for each asset $R_{t+1} \in \mathcal{R}^k$ for k assets in the tradeable universe. Thus our decision-making is based on the expectation of the future return of the portfolio (subject to the risk taken):

$$\mathbb{E}[< w_t, R_{t+1} > |\mathcal{F}_\sqcup]$$

One may notice that we can apply the Cauchy-Schwarz inequality and get a decomposition

$$\mathbb{E}[< w_t, R_{t+1} > |\mathcal{F}_\sqcup] \leq \mathbb{E}[||w_t|| \; ||R_{t+1}|| \mid \mathcal{F}_\sqcup]$$

By independence of a filtration at time t from the future realized return (by filtration construction), one get

$$\mathbb{E}[< w_t, R_{t+1} > |\mathcal{F}_\sqcup] \leq \mathbb{E}[||w_t|| \mid \mathcal{F}_\sqcup] \; \mathbb{E}[||R_{t+1}|| \mid \mathcal{F}_\sqcup]$$

$$\mathbb{E}[PnL_{t+1}|\mathcal{F}_\sqcup] \leq ||w_t|| \; \mathbb{E}[||R_{t+1}|| \mid \mathcal{F}_\sqcup]$$

Finally, we can conclude that our strategy PnL is upper bounded by this exact norm. In other words, in return-maximization problem our expected strategy PnL maximization problem implies putting the higher weights onto the assets with the higher expected return $\mathbb{E}[||R_{t+1}|| \mid \mathcal{F}_\sqcup]$. And this is exactly what shapes our decision-making process in investment portfolio construction.

Additionally, note that in a more practically relevant optimization, where our goal is maximizing PnL with controlled risk (for instance, in Markowitz portfolio selection setting, [12]), the same argument goes through. In the interest of space I do not provide the derivation in this paper.

Therefore, our decision making process, which is exactly determining the vector of weights at time t, given all available information at time t, simplifies to a supervised learning task of determining $\mathbb{E}[||R_{t+1}|| \mid \mathcal{F}_\sqcup]$. This is exactly the usual setting of supervised learning prediction, which can be stated as minimizing some Empirical Risk of a Empirical Risk Minimizer:

$$f(X_t) = \mathbb{E}[||R_{t+1}|| \mid \mathcal{F}_\sqcup]$$

, where $f : \mathbb{R}^p \to \mathbb{R}$ is our Empirical Risk Minimizer and $X_{t-n:t} \in \mathbb{R}^{n \times p}$ is our set of features that quantify the filtration.

Thus, all of the investment decision-making process in this simplified setting turns out to be based on solving sequentially at each rebalancing period an optimization of the form

$f_t = \underset{f \in \mathcal{F}}{argmin} \; l(f_t(X_{t-h:t}), R_{t-h:t})$

, where h is a lookback window size (how much data we take into account).

Finally, we may note that such an empirical risk can be approximated under additive noise of R_t as a sum of empirical risk values at each object in the sample.

$f_t \approx \underset{f \in \mathcal{F}}{argmin} \; \sum_{i=t-h}^{t} l(f_t(X_i), R_i)$

Therefore, we can conclude that the investment portfolio construction decision-making process in a simple setting depends on the prediction of the future realized returns for each asset in the tradeable universe, which spans the objective for this paper - develop a robust estimator f to base the trading strategy for US Stock Market Index on exactly the prediction of this Empirical Risk Minimizer.

B. How does predictive quality translates into Sharpe Ratio?

Furthermore, if we look at the results of [13], the predictive power in a simple setting can be translated into the future realized Sharpe ratio, which is a common goal in investment portfolio construction.

Suppose that the stock excess return is given by

$$R^e_{t+1} = \mu + x_t + \epsilon_{t+1}$$

, where μ is some unconditional risk premium, while $x_t \sim \mathcal{N}(0, \sigma_x^2)$ is our predictor variable and random shock $\epsilon_{t+1} \sim\sim \mathcal{N}(0, \sigma_\epsilon^2)$

Then a mean-variance investor chooses the optimal weights ([12]) as:

- If x_t is unobserved: $w_t = \frac{1}{\gamma}\frac{\mu}{\sigma_x^2+\sigma_\epsilon^2}$
- If x_t is predicted perfectly: $w_t = \frac{1}{\gamma}\frac{\mu+x_t}{\sigma_\epsilon^2}$

Then under R^2 measure of predictive quality, which is given by

$$R^2 = \frac{var(x_t)}{var(R^e_{t+1})} = \frac{\sigma_x^2}{\sigma_x^2+\sigma_\epsilon^2}$$

Thus, our improvement of in portfolio's expected excess return is simplified to

$$\frac{\mathbb{E}\ Ret_{forecast} - \mathbb{E}\ Ret_{noisy}}{\mathbb{E}\ Ret_{noisy} - Ret_{risk-free}} = \frac{R^2}{1-R^2}\frac{1+SR^2}{SR^2} \sim \frac{R^2}{SR^2}$$

, where SR is the Sharpe ratio of the strategy without return predictor x_t and the last proportionality works under small R^2 and SR^2, which is a setup that is usually satisfied in finance due to absence of known strong predictor of future excess return.

Therefore, one may conclude that there is a proportionality of the excess return improvement from the R^2 score of our predictive model, which supports the need of Empirical Risk Minimizers in the trading strategies construction.

III. Network Architectures

A. Gaussian Process

As was outlined in I, a shallow Neural Network with infinite width can be viewed as a Gaussian Process with some complex kernel [5]. Therefore, we start with a baseline of Gaussian Process Regression with the known kernels from existing literature.

Gaussian Process Regression (GPR) for risk premium δ prediction is a Bayesian non-parametric method that models the mapping $s_t \mapsto y_t^*$ by placing a Gaussian prior over functions and updating it to a posterior after observing the data. Using a kernel $k(s, s')$, GPR yields a predictive mean

$$\hat{\delta}(s) \ = \ k(s, S)\left[K(S, S) + \sigma_n^2 I\right]^{-1} y^*,$$

which we take as the risk premium predictive estimate at each rebalancing date. This approach flexibly captures complex, nonlinear relationships and provides a natural uncertainty quantification through its posterior variance.

We consider the GP Regression as an important baseline for our study, as by placing a hyperprior over hyperparameters, it allows for tuning those on the training dataset. While still ensuring OOS construction, such an approach is alternative for Time Series Cross Validation and can be seen as another way to estimate the optimal hyperparameters (by estimating the prior standard deviation on the TSCV folds).

In our case, we employ the hyperprior for tuning, allowing the model to restart the optimizer 3 times at each refitting date (rebalancing date). The number of restarts is chosen for the computational speed, while we still find robustness of this hyperparameter in our experiments.

I work with the 2 types of kernels:

- $DotProduct(x, x') =< x, x' >$, also known as "Linear Kernel". The GPR under this kernel is exactly a Ridge Regression, meaning that such a baseline should produce the results close to the usual frequentist Ridge Regression from [9], but potentially gain a robustness of hyperparameters due to hyperprior.
- $RBF(x, x') = \exp(-\frac{||x-x'||_2^2}{2h})$ (also known as Gaussian Kernel). This kernel represents a complex non-linear relationships and is aimed to test the increase of complexity on the modeling results.

The motivation for the Gaussian Process assumption is not only the usual tractability of estimation that equips us with the closed-out solution, but something more powerful.

The Gaussian Distribution represents a distribution with the maximum entropy across all of the distributions with known mean and volatility [14]. As these two parameters are the ones that we usually aim to predict in finance due to lack of data and poor representability of the other moments of the distributions (such as skewness and kurtosis), this is exactly the setting that we work in.

The maximum entropy principle (also known as Laplace principle) [14] is an important feature in the Bayesian view of predictive task. It means that one should choose the most expressive distribution possible, i.e. the distribution that covers as much of the variation as possible, which is exactly the entropy-formulated task. By a distribution with maximum entropy we basically make as least as possible prior assumptions on the likelihood behavior and act in the "agnostic way". Such an approach is extremely useful in financial applications setting, as it allows to flexibly model the feature $\rightarrow$ risk premium mapping.

B. Feed-Forward Neural Nets

The next step of modeling is using a more expressive predictor. Therefore, instead of choosing the kernel as a hyperparameter, we basically aim to learn the kernel representation by a stacked non-linearity.

Deep learning covers a family of neural network architectures; here we adopt a simple multi-layer perceptron (MLP) to learn the mapping $s_t \mapsto y_t^*$. Here we use a multi-layer perceptron (MLP) with L hidden layers of size W and ReLU activations, followed by a linear output. Formally:

$$h^{(1)} = \max\big(0,\, W^{(1)} s_t + b^{(1)}\big)$$
$$h^{(i)} = \max\big(0,\, W^{(i)} h^{(i-1)} + b^{(i)}\big)\ \forall\ i \in \{2, ..., L\}$$
$$\hat{\delta}_t = W^{(L)} h^{(L)} + b^{(L)}.$$

Such a stack of non-linear transformations have been shown to be expressive enough to model any complex function on the closed and continuous range of function values with L at some fixed W for a given ϵ error [15] (proof is based on the Sigmoid activation as opposed of above example under ReLU).

This very expressive model can be very well expected to perform quite well in the stationary domains. However, in financial applications the extensive expressiveness might turn out to be a pitfall rather than a virtue.

Fig. 1. Autoencoder Architecture, Source: Deepak Birla

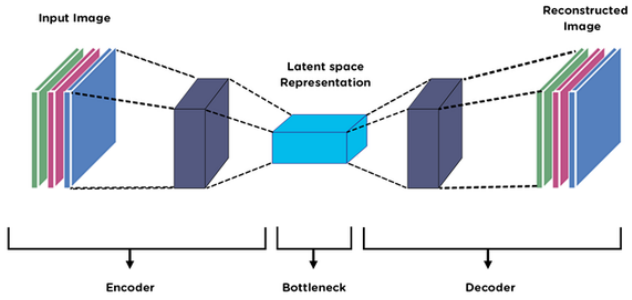
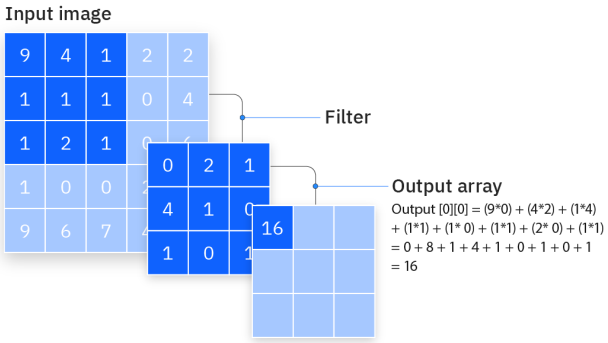


Fig. 2. CNN Kernels, Source: IBM



C. Autoencoders

The next approach in deep learning is known as Autoencoders. This model aims to transform a given features X_t into some latent space, where it is expressive enough to model in a linear fashion. Basically, one can view an autoencoder as a highly non-linear feature transformation mechanism that maps our initial noisy market description metrics into some meaningful "trading rules", but in a continuous manner.

Such an approach can we found in the III-B formulation. Basically, a backbone - all the non-linear layers instead of the last one - is exactly the autoencoder that maps the input features into the final modeling linear layer that actually produces a prediction.

Due to quite expressive power under W neurons, we hope that each such neuron will be able to reconstruct some pattern that is predictive of the risk premium pattern recognition task (meaning Empirical Risk minimization) due to the backpropagation of the signal through all the layers to adjust the weights everywhere - not only at the final layer that actually maps the autoencoder output to the risk premium estimation.

Usually, the autoencoder is formulated as a reconstruction task. It means that instead of predictive Empirical Risk we now aim to map the X_t to the same X_t, but trying to learn some "latent space" in the middle, which is actually of interest here, as can be viewed in 1

One may note a very important design feature that comes from the 1. One aims to map $f : X_t \to X_t$. Therefore, it is crucial to construct the AE function as a bottleneck, meaning that the consecutive layers should have lower W than the original dimension d in $X_t \in \mathbb{R}^{n \times d}$. The reason for that is simple - otherwise while our reconstruction loss will be perfectly zero, but the actual task of interest, learning latent representation, will not be completed, because our model will learn to reconstruct the original X_t by an identity, providing no additional value in terms of dimensionality reduction and feature extraction. Thus, to avoid that, one should ideally keep $W < d$ in such a construction.

D. Convolutional Networks

The next approach, which was initially developed for the Computer Vision (CV) tasks, but found the application across different domain, including finance (for instance, the order-book features extraction in higher frequency settings than the ones considered in this paper).

The idea of Convolutional Neural Nets (CNNs) is to extract *local patterns in the data*. We use a shifting kernel that captures some part of the data, moving over (either overlapping or non-overlapping) intervals, extracting a separate feature from each. Then a Pooling layer is applied, meaning that each of such features is stack with all another features and transformed into some meaningful representation by a non-linearity, which is different from III-B ReLU, but still serves the same purpose.

In the CV tasks, we motivate such a construction by local patterns in the picture. Instead of independently fitting the inputting pixels, we want to introduce a notion of positional dependence in our model to adjust the feature representation to where the initial input is located in the picture, as is illustrated by 2.

The same logic might be helpful in the financial Time Series setting, where one aims to incorporate the local patterns of returns behavior.

E. Recurrent Networks

However, another powerful approach for financial Time Series modeling is Recursive Neural Nets (RNNs) architectures. The aim of those is completely different to the CNNs. Here we aim to recognize not *local patterns*, but rather support the memory in Time Series, trying to make each sequential point be aware of the past behavior. Such model is trained with Backpropagation-Through-Time that accounts for signal attribution to the sequential data processing [16].

However, RNNs suffer from the vanishing or exploding gradients problem. Essentially, this issue is given by construction - as our model processes the data sequential in Backpropagation-Through-Time updates the gradient appears

as a multiplication of gradients for each of the sequential neurons due to chain rule [16]. This means that one should introduce a separate special reservoir for long-term memory.

During backpropagation, one uses the following rule:

$$\frac{\partial L}{\partial h_t} = W_{hh}^T \frac{\partial L}{\partial h_{t-1}}$$

Meaning that the Spectral Conditions for Stability become [17]:

- $\rho(W_{hh}) > 1$, gradients explode
- $\rho(W_{hh}) < 1$, gradients vanish
- $\rho(W_{hh}) \approx 1$ stable propagation

And this idea is exactly what inspired the famous Long-Short Term Memory (LSTM) network [18]. The model aims to separate the short-term memory from a special long-term memory cell that serves as a special "storage" for long-term patters. The LSTM updates are given by [18]:

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f)$$

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)$$

$$\tilde{c}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c)$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o)$$

$$h_t = o_t \circ \tanh(c_t)$$

F. Regularization

Finally, one should note that in usual applications, but especially in Finance, require a well-behaved out-of-sample predictions. Intuitively, it means that we aim at finding a high, but plateau region in complex likelihood landscape. If one finds a region with higher, but more peaky region, the model falls into a potential danger of dropping severely in likelihood, when the data slightly changes.

And a good approach for such a goal is introducing some degree of conservativeness into the model by applying *regularization* - a method to prevent (reasonably) the model from matching the extremes of the training data.

The main methods that are widely used are L1-norm and L2-norm on the weights. In those methods we introduce a restriction on the maximum value of the norm of vector weights. As the restriction can be reformulated into the penalization (adding a penalty into the Emprirical Risk) by Lagrangian, we end up with the following optimization objectives:

- L1-Penalization: $l(f(X_{t-h:t}), R_{t-h:t}) + \lambda||\theta||_1^2$
- L2-Penalization: $l(f(X_{t-h:t}), R_{t-h:t}) + \lambda||\theta||_2^2$

If the Empirical Risk is a convex function, the former provides a convex and sparse objective, while L2 - convex and smooth.

In the deep learning setting we, however, lose the convexity in the Empirical Risk, as if even our Loss function is convex (like Mean Squared Error), the composition of it with the highly non-convex Neural Network is not convex. Therefore, in deep learning setting such a regularization might be helpful, but we lose the theoretical guarantees of convergence.

Moreover, the L1-regularization is rarely applicable in deep learning case. As the L1-norm is not a differentiable function, we need to apply the coordinate descent optimization instead of the gradient descent, which is computationally very expensive due to randomization in the subgradient search that arises from set formulation instead of the definite value of the gradient. Therefore, training the Deep Neural Networks for multiple epochs becomes a very hard task. As opposed to this, L2-penalization provides a differentiable and smooth penalty in the loss function, allowing for gradient-based methods to work virtually for the same time as a no-penalty methods.

Another approach, tailored to Deep Neural Networks, is a Dropout [19]. The idea is to introduce a stochasticity in the model fitting process by randomly switching-off p neurons by a Uniform choice. The p serves as a hyperparameter for the model.

This switching-off is usually motivated by "disallowing the neurons to communicate constantly". But in the statistical sense, it works as slight perturbations to our likelihood landscape, ensuring the more plateau-like optimum region.

Finally, the dropout might serve as a great way to model the Bayesian epistemic uncertainty of the DNN [20]. This perturbation in the training regime can be still left in the inference mode. Therefore, by sampling B times during inference (as opposed to $B = 1$ in the usual setting), one can obtain not only the prediction (which is the average of the B values), but also standard deviation approximation. This may serve as a good indicator of the steepness of the likelihood, providing us with the estimation of our predictor's variation in out-of-sample setting.

IV. Experiments

A. Target Construction

Following the motivation of conditional risk premium prediction from [3], we work in a setting of monthly market return predictions with monthly rebalancing, which represents a meaningful baseline for trade-off between transaction costs, predictability and number of available datapoints.

Mainly, our idea is to convert our prediction problem into a more common financial problem by using the stylized market fact of stocks switching between "momentum" and "reversal" regimes. Therefore, we transform our target as a binary classification task of whether the market is now in momentum regime (previous monthly return had the same sign as the current market return) or in reversal regime (targets

Ticker	Transformations
CAPE	Log-Transform to obtain $\log(Price) - \log(Earnings)$.
STP	1st order differencing.
M2	Log-Transform + 1st order differencing.
_OIL	Untransformed.

TABLE I
SUMMARY OF FEATURE TRANSFORMATIONS USED.

differ). This idea introduces a prior knowledge into the model construction and shows promising results, compared to the usual regression setting.

Finally, our dataset is constructed as a rolling window of monthly size, where we aim to predict (and trade on) the next month sign of return. The prediction is obtained as an inverse transform that uses the sign of the previous month returns.

Finally, the position is rotated between long and short, which is given by the label prediction of the respective estimator with decision threshold $thr = 0.5$, as in long vs short setting (i.e., when zero holding is disallowed) the False Positive and False Negative errors are symmetric.

B. Choice of signals

Summary below outlines the chosen meaningful financial features that should represent - from financial logic perspective - the state of the economy and the market participants expectations

- **CAPE**: Cyclically-adjusted Price-to-Earnings Ratio. Being an important estimator for the level of valuations in the market, but adjusted for economic cycles. Aims to show the model the potential valuation-based reversal signal.
- **STP**: The term spread of the US Treasury Yield Curve. It has been known as a great predictor for the upcoming recession. While it is not a great indicator for overall market timing, especially in the recent years, it might be very well indicative of the market participants' perception of the current economic situation, which is crucial for momentum / reversal trade-off.
- **M2**: The indicator of M2 supply of money is a direct representation of the current monetary policy, which produces the signal of the economy's state from regulator's perspective.
- **_OIL**: Oil has been known as a good indicator for overall economic activity, which might be predictive not of the future economy's state, but of market agents' prediction upon those.

However, as these values are given as levels, one needs to transform those into more well-behaved values. The summary of transformations is given in Table I.

Furthermore, the features are transformed by the Standard Scaler to zero-one distribution, using the training data only, which controls from the lookahead bias.

Parameter	Grid
Hidden Size	$\{8, 16, 32, 64, 128\}$
Num Layers	$\{2, 4, 6\}$
Lr	$\{10^{-6}, 10^{-3}, 10^{-2}, 10^{-1}\}$

TABLE II
TIME SERIES CROSS-VALIDATION GRID.

C. Choice of network type and parameters

The chosen set of models and the reasoning is given below:

1) **Logistic Regression**. It is a weak learner (high bias and low variance) estimator, which is used as a pure baseline in our modeling procedure.
2) **MLP**. The well-balanced trade-off between
3) **LSTM**. As markets may exhibit long-term sequential patterns.
4) **Gaussian Process Classifier with RBF kernel**. As an expressive learner with uncertainty quantification that comes for free, it might represent a meaningful baseline for the models that learn the kernels automatically.
5) **Transformer**. A very expressive learner with positional encoding and attention weights, where the causal window is applied to prevent the look-ahead bias that comes from attention-weighting.
6) **MLP with Dropout** $p = 0.2$. A regularized version of the MLP with fixed dropout hyperparameter.
7) **MLP with Uncertainty Weighting**. An MLP with SVI through Dropout on Inference, where uncertainty is calculated through $B = 100$ restarts.

The models hyperparameters are tuned at each retraining date separately, using the Time Series Cross-Validation of the grid, summarized in Table II. All other hyperparameters are kept fixed, which is motivated solely by the modeling setup.

We train the Deep Learning models by minimizing mean-squared error via stochastic gradient descent with a cosine-annealing schedule.

D. Results

In Tables III and IV we present the final results. Contrary to the expectations, the well-balanced predictor of MLP is a more robust strategy fitting scheme than the more complex baselines. Moreover, the predictive quality does not change under dropout regularization, meaning that the initial architecture is robust enough.

Moreover, contrary to the initial idea, Feature Ablation method in 5 shows that the model is basing its predictions on the Oil Prices and Term Spread only.

Finally, we employ the scheme of trading to introduce the position sizing, based on the estimation of the likleihood steepness, which improves the final results.

Fig. 3. MLP Strategy Performance

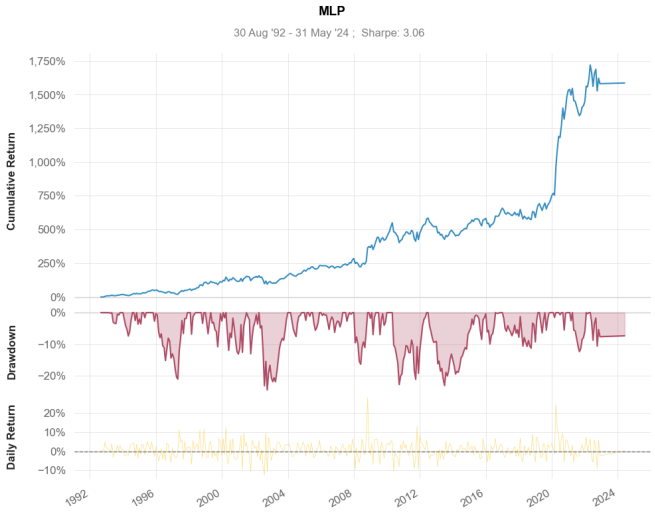


Fig. 4. MLP Strategy Drawdowns

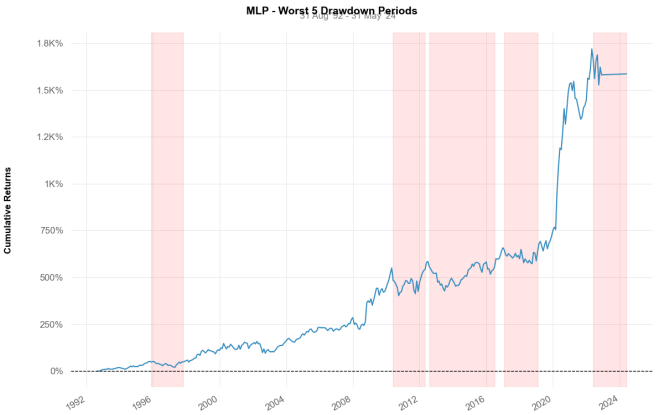


Fig. 5. MLP Strategy Feature Importance

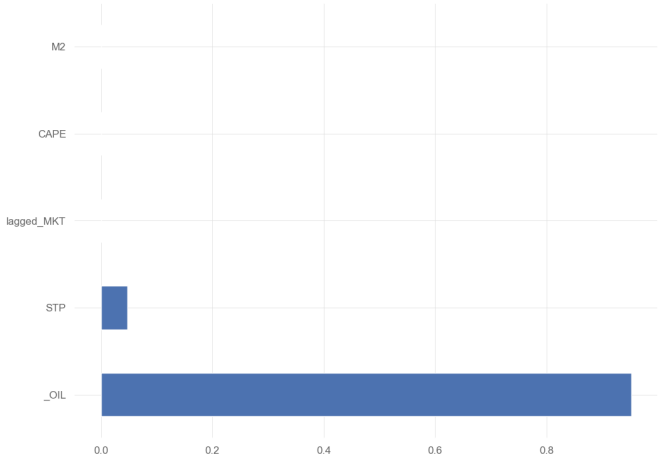


TABLE III
STRATEGIES PERFORMANCE, PART 1 (08/1992-05/2024)

BuyHold	MLP	LSTM	GPC
Sharpe Ratio			
0.47	0.45	-0.15	0.24
Alpha vs Mkt Loading			
N/A	4.85 (0.03*)	-2.48 (0.63)	-0.00 (0.76)
Information Ratio			
N/A	0.33	-0.16	-0.01

TABLE IV
STRATEGIES PERFORMANCE, PART 2 (08/1992-05/2024)

LR	T	MLP-D	MLP-U
Sharpe Ratio			
0.21	-0.15	0.45	0.42
Alpha vs Mkt Loading			
-2.18 (0.42)	-2.47 (0.63)	4.85 (0.03*)	5.10 (0.05*)
Information Ratio			
-0.21	-0.16	0.33	0.36

V. CONCLUSION

The paper explored in details the deep learning modelling in market risk premium prediction task. We find that the MLP predictor produces stronger and more robust results than the stronger models.

We find the improvement in performance, based on the likelihood steepness, which might be explored further in details.

REFERENCES

[1] M. L. De Prado, *Advances in Financial Machine Learning*. Wiley, 2018.

[2] M. López de Prado, "The 10 reasons most machine learning funds fail," *Journal of Portfolio Management*, 2018.

Fig. 6. MLP Strategy Performance vs Benchmark

[3] B. Kelly, S. Malamud, and K. Zhou, "The virtue of complexity in return prediction," *The Journal of Finance*, vol. 79, no. 1, pp. 459–503, 2024.

[4] AQR Capital Management, "Alternative thinking 2024 — issue 4: The Virtue of Complexity in the Cross-Section of Stocks," *AQR: Alternative Thinking*, November 2024, accessed: June 26, 2025.

[5] A. Jacot, F. Gabriel, and C. Hongler, "Neural tangent kernel: Convergence and generalization in neural networks," *Advances in neural information processing systems*, vol. 31, 2018.

[6] B. Schölkopf, R. Herbrich, and A. J. Smola, "A generalized representer theorem," in *International conference on computational learning theory*. Springer, 2001, pp. 416–426.

[7] D. Buncic, "Simplified: A closer look at the virtue of complexity in return prediction," *SSRN*, 2025.

[8] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 2018.

[9] G. De Nard and D. Kostovic, "Ai shrinkage: a data-driven approach for risk-optimized portfolios," *Working paper series*, 2025.

[10] K. Ishikawa and K. Nakata, "Online trading models with deep reinforcement learning in the forex market considering transaction costs," 2021. [Online]. Available: https://arxiv.org/abs/2106.03035

[11] H. Buehler, M. Phillip, and B. Wood, "Deep bellman hedging," *SSRN*, 2022.

[12] H. Markowitz, "Portfolio selection," *The Journal of Finance*, vol. 7, pp. 77–91, 1952.

[13] J. Y. Campbell and S. B. Thompson, "Predicting excess stock returns out of sample: Can anything beat the historical average?" *The Review of Financial Studies*, vol. 21, no. 4, pp. 1509–1531, 11 2007. [Online]. Available: https://doi.org/10.1093/rfs/hhm055

[14] A. Krause, pAI Fall 2024 Lecture Notes.

[15] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0893608089900208

[16] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[17] N. Antulov-Fantulin, mLFCS Spring 2025 Lecture Slides.

[18] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, pp. 1735–1780, 11 1997.

[19] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

[20] D. P. Kingma, T. Salimans, and M. Welling, "Variational dropout and the local reparameterization trick," 2015. [Online]. Available: https://arxiv.org/abs/1506.02557