Literature Review

1 Technical Analysis

Brief introduction to Technical Analysis and pattern matching.

2 Neural Networks

2.1 Neural Network Algorithms and Structures

An overview of the relevant algorithms, and early network structures.

2.2 Deep Learning

A brief overview of the developments in deep learning as applied to neural networks

2.3 Neural Network Variations

A brief overview of the variations of neural networks - e.g. perceptrons, RBM, feedforward, LTSM etc.

2.4 Applicatons of Neural Networks in Finance

A coverage of the relevant applications of neural networks in predicting prices.

3 Stacked Autoencoders

3.1 High Dimensional Data Reduction

As noted, machine learning techniques have been shown to be extremely effective at modelling non-linear inputs to outputs - neural networks have even been shown to be universal function approximators in this regard [28]. More traditional statistical models will typically process the available feature data to select the most significant features to be used in the model once its defined - evident in a processes such as subset selection [42]. Machine learning techniques are no different in this regard, and feature data will typically be transformed to smaller observations of more significance prior to being used as input to a model, such as the neural networks described above.

Financial data, in line with the complex and dynamic system that it represents, is often of a very high dimensional nature, which offers opportunities through more sophisticated analysis, but also introduces the curses of dimensionality [15]. The increased dimensionality can result in higher processing complexities when needing to do basic tasks such as estimating a covariance matrix (a commonplace necessity in finance), as well as increase the risk of incorrect assumptions based on spurious variable collinearity [19]. Noise accumulation in high dimensional data can create further problems, resulting in problems performing variable selection and ultimately having a large impact on classification and regression models [20].

Time series data can introduce its own set of challenges - there is often not enough data available to understand and predict the process [18], the time variable dependence creates complexity in how much past data to consider at any point, and the data is typically non-stationary [32]. Thus, high dimensional time series data (which many financial problems focus on), require careful consideration on how to handle their inputs and analysis.

Deep learning techniques are a natural choice in this context, and much research has been done to show their (varying) efficacy on time series data. The most successful of these models have been ones which modify deep learning techniques to incorporate the temporal aspect of the data (e.g. Conditional Restricted Boltzmann Machines or Recurrent Neural Networks), rather than static, and those which have performed feature selection processes rather than operating on the raw data (e.g. Auto-encoders) [32].

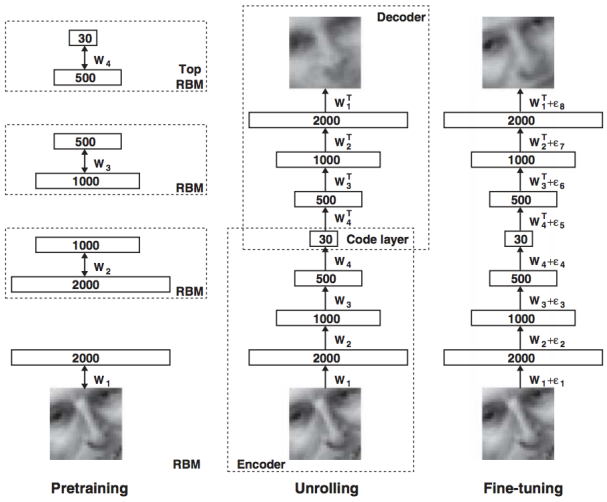

Fig. 1: The Autoencoder training steps [25]

Two of the seminal pieces of research that have lead to the resurgence in machine learning and deep learning were the algorithms for training deep belief networks [24], as well as the usage of stacked auto-encoders [41, 9].

3.2 Deep Belief Networks

Autoencoders were suggested by Hinton et. al as a method of transforming high dimensional data to lower dimensional input vectors, in order to alleviate some of the problems detailed above, and increase performance of deep belief networks [25].

One of the more prominent classical techniques for dimension reduction is principal components analysis (PCA), which uses linear algebra to find the directions of greatest variance, and represent the observation samples features along each of these directions, thus maximising the variational representation. Hinton et al. show that autoencoders are a nonlinear generalization of PCA. The structure and training algorithms of the autoencoder show it to be a specialised neural network - there is a multilayer encoder network which is able to transform to a lower dimension, and a symmetrical decoder network to recover the data from the code as represented in Figure 1. As with neural networks, the gradient weights can be trained through the feedforward and backpropagation algorithms.

The primary challenge presented here is the initial weighting of the networks - with large initial weights the autoencoder will often find a poor local minima, and with small initial weights the gradients are too small to effectively train deep layered networks. The critical suggestion by Hinton et al. was to used layered Restricted Boltzmann Machines (RBM) in order to initialise the weights. For each layer of the desired autoencoder, a RBM is formed and trained with the previous layer (or RBM) [26]. Once all the layers have been trained in this way, they are mirrored to form the decoder network. This then forms the initial weights to be fine tuned further, as per the Fine-tuning step in 1. They showed the deep autoencoder networks were significantly more effective than PCA or shallow autoencoders on multiple dataset types.

3.3 Stacked Denoising Autoencoders

The second important piece of work was the development of a denoising autoencoder (DAE), by Vincent et al. [48]. One of the problems identified in the DBN model (and those similar), is that if the encoder dimensions were too high, it is likely that the encoder would learn a trivial encoding - essentially creating a copy the input model. The one way of tackling this issue is to constrain the representation with bottlenecks and sparse autoencoder layers, which can be seen in figure 1.

Vincent et al. explore a very different approach to the problem, which was to develop an implementation of autoencoder which focused on partially corrupting the input, and so force the network to denoise it. The theory here is based on two ideas - the first, is that a higher dimensional representation should be robust to partial corruption of the input data; and the second is that the denoising process will force model focus to shift to extracting useful features from the input.

The algorithms and structures are largely the same as described for DBNs above, with the key difference being that the model is trained to reconstruct the original input, but only using a corrupted version of the input (where noise has been added to it), and so is forced to learn smarter feature mappings and extractions. The DAE suggested then is a stochastic variant of the autoencoder, which has the benefit of being able to implement higher dimensional representations without

risking training of a trivial identity mapping. Notably, in the Stacked Denoising Autoencoder (SDAE) formation, only the initial input is corrupted (as opposed to the input from layer to layer). It was shown that the SDAE model outperformed previous AE and DBN networks on numerous benchmark datasets [48] .

3.4 Pre-training

The methods described above follow a similar approach: greedy layer-wise unsupervised pre-training in order to determine initial weights, followed by supervised fine tuning to arrive at the final model. It is shown numerous times, that the pretraining process results in significant performance gains [48]. However it is not immediately apparent, given the nature of backpropagation algorithms and the like, why this is the case. Erhan et al. performed extensive empirical simulations in order to suggest an explanation to the mechanism of pretraining [17].

While their results were not entirely conclusive, they did lend themselves to a reasonable hypothesis: the unsupervised pre-training results in a form of regularization on the model - variance is minimized, and the bias introduced directs the model configuration towards a sample space that is effective for the unsupervised learning generalization optimisations.

3.5 Time Series Applications

The autoencoder papers reviewed so far in this section derive their results primarily from classification problems, and so do not necessarily account for the problems involved with time series as described in 3.1. Due to the inherent difficulties with predictions in the financial system, it can sometimes be unclear if the shortcoming in results is due to this system complexity or if the methodologies used are unsuited for the purpose. In light of this it is worth pointing out that Stacked Autoencoder (SAE) implementations have been shown to be effective in many time series systems.

Lv et al. implemented a deep learning SAE model using the methods described in 3.3 in order to predict traffic flow at various time intervals (15, 30, 45 and 60 minutes) - a problem not so structurally dissimilar from what will be presented in this paper [36]. They were able to show that the deep SAE was able to offer prediction results which were both objectively good and also persistently outperformed the comparison models used (backpropagation neural network, random walk forecast, support vector machine and a radial basis function neural network).

In a review of unsupervised feature learning and deep learning methods on time series, Langkvist et al. noted that the use of autoencoders, either as a technique in themselves, or as an auxiliary technique to models such as convolutional neural networks, were able to offer performance increases in areas such as video analysis, motion capture data and bacteria identification [32].

3.6 Financial Applications

There have of course also been successful applications of stacked autoencoders and deep learning models in finance as well. Takeuchi et al. performed some earlier work showing the use of autoencoders when applied to a momentum trading strategy. They implemented an RBM pre-trained DBN as per 3.2, and assessed the networks classification performance for ordinary shares on NYSE, AMEX and Nasdaq. This showed that using a DBN network resulted in significant performance increases compared to the standard momentum strategy [45].

Zhao et al. used SDAEs and combined them with the bootstrap aggregation ensemble method (bagging) in a study of predicting the crude oil price. They compared the proposed model to a variety of benchmarks, including standard SAE, bagged and standard feedforward networks and SVRs. The results indicated that the SAE models were more accurate, with the bagged SAE model performing the best, though at a significant increase in computational costs in comparison to standard SAE [55].

While much of the financial literature has focused on the use of RBM based models, Autoencoders and SAEs have recently been gaining popularity in performing feature reduction. Troiano et al. specifically investigate the use of different feature reduction models for trend prediction in finance [46]. In line with being primarily interested in the effect of feature reduction techniques, rather than the classification performance itself, only an SVM model was used to test results. Using various periods from historical SP 500 data, they were able to show that AE outperformed the RBM model significantly in numerous accuracy measures, and was able to do so at a fraction of the training time.

Bao et al. note that the research has been lacking with regards to whether SAEs should be used for financial prediction models or not [7]. They suggest a novel model which combines Wavelet Transformation, SAEs and a Long Short Term Memory (LSTM) network. Using data from several financial exchanges (considering a range of developed and undeveloped markets), they assess the models applicability to OHLC prediction. Comparing the model to configurations without the SAE layers, and a RNN model as benchmark, they showed that the inclusion of SAEs resulted in less volatility and greater

accuracy, which in turn offered higher profitabilities in a buy-and-hold trading strategy.

More novel autoencoder applications have also been attempted, with Hsu suggesting the use of a Recurrent Autoencoder for multidimensional time series prediction [29]. There is a clear pattern through the literature that the use of AEs and SAEs both by themselves and when used as an assisting technique result in more accurate prediction results and less computationally expensive training.

4 Data Segmentation

One of the aspects of time series not yet discussed is how the data might be segmented for analysis. Financial pattern matching, as discussed in 1, requires methodologies to decide the length of data to consider when determining whether a subset of data matches a pattern or not.

There are numerous classical approaches to this problem, which were widely applied in machine learning prior to the resurgence in deep learning. One of the earliest and more common approaches was the Sliding Window, where each the model input for each observation is padded by a predetermined number of observations that occurred both prior and after the one in question. This has the advantage of being fairly model agnostic, but fails to capture any correlations in the dependent variable values [12]. The Recurrent Sliding Window method aims to resolve this by including the same number of prior predictions made as part of the input. In this case, input would be $< x_{t-d}, ...x_t, , x_{t+d} > and < y_{t-d}, ..., y_t >$. This was shown to be significantly more effective than the plain sliding window method [6]. Notably, the sliding window approach can only be implemented in an offline model.

These methods can be adapted to take on online formats, as well as incorporate data reduction benefits through algorithms such as Piecewise Linear Approximation (PLA), which adapts a linear representation to the leading portion of the window. Some novel approaches (Feasible Space Window and its stepwise adaption) were suggested by Liu et al. in order to compensate for the computational requirements of reprocessing the entire window at each step for online models [34].

Window based methods represent a fairly static and unsophisticated approach to data segmentation. A suggested solution to this is to segment the data by dynamically identifying significant partition points, known as perceptual important points (PIP) [13]. The computational cost of identifying PIPs was initially rather large, making them unsuitable for quickly changing and dynamic environments such as finance. However, Zhou et al. suggested a novel approach which reduced computational costs through intelligent binary tree traversal. They were able to show this approach was effective in identifying traditional financial stock patterns with the use of a layered neural network [57]. In a similar fashion, input data turning points (TP) were shown to potentially offer lower error rates than PIPs by Yin et al. [52].

More recently, Wan et al. conducted a review on the effect of segmentation on financial time series pattern matching, comparing perceptual important points, piecewise aggregate approximations, piecewise linear approximations and turning point based methods [49]. They use several pattern matching algorithms (template-based, rule-based, hybrid, decision tree, and symbolic aggregate approximation) for each of the segmentation methods in order to determine their effect on a broader level. The analysis was performed on real stock market data, as well as synthetic data generated to display common patterns such as positive/negative head and shoulders (HS). Measuring accuracy, precision and recall, they showed that PIP based segmentation generally offered superior results to the others (though it is worth noting that there were various performance differences within the segmentation methods depending on the matching algorithm used).

5 Online Learning Algorithms and Gradient Descent

Most classic machine learning algorithms operate under the assumption that, for all intents and purposes, the full dataset has been collected and that the amount of training data for the model is both finite and immediately available. However, as the growth of information grows in an exponential fashion, there are numerous areas where the expected training data for the model will continue to grow. In these cases it would be disadvantageous to go through the full training and validation process again in order to incorporate the newly available data.

Online algorithms are designed to offset these issues by adjusting the batch training technique to rather repetitively draw on single samples from the data on which the models parameters can be adjusted. The benefit is that they are able to quickly process a large number of observations and readjust the model, though the downfall is that they are not always able to optimize the cost function to the same extent as offline batch algorithms [1].

Bottou and Cun argue that as the size of the dataset grows significantly, online algorithms advantages result in them outperforming offline models, despite any initial drawbacks [10]. Previous research had shown that online algorithms

typically perform as fast as batch algorithms during the search phase of parameter optimization, but that final phase convergence tended to fluctuate around the optima due to the noise present in single sample gradients [33, 11]. Bottou and Cun showed in fact, that it is more practical to consider the convergence towards the parameters of the optima, rather than the optima itself (as defined by the cost function) - the difference between the learning speed and optimization speed, respectively [10]. Theoretical and empirical findings were presented to show that a stochastic online gradient descent (SGD) algorithm **[referenceappendix]** was able to outperform the batch model for parameter estimation, and was able to asymptotically outperform in the number of samples processed in a time period. The stochastic aspect of the algorithm is related to random observations from batch sample groups being used as the gradient basis. Theoretically, this slows down the convergence, but speeds up the processing speed of each batch - a technique which has later been shown to be generally successful [44, 56].

The SGD algorithm has resulted in a fair amount of further research due to its applicability to machine learning and the online benefit, which can largely be group into two categories: improvements affecting gradient learning and convergence rates, and processing improvements through parallelization.

One of the earlier improvements to convergence rates was the Momentum algorithm, as developed by Tseng [47]. As noted, stochastic descent often introduces significant oscillation around an optima, which slows down convergence. Momentum reduces this by decreasing movement in directions of high curvature, and increasing increasing movement towards directions consistent with previous gradients (this is achieved through combining gradient movements in opposite directions) .

There have also been several attempts to introduce effective regularization into the SGD process. Bartlett et al. presented Adaptive Online Gradient Descent, which implements an adaptive step size through a λ penalty on the learning rate, which was shown to be nearly optimal in a strong sense [8]. Langford et al. demonstrated a variation named Truncated Gradient, which introduced an enforced weight sparsity parameter. The weight sparsity is able to achieve equitable effects to L_1 regularization (similar to Lasso Regression). They were able to show that implementation performed effective feature reduction, while having little effect on performance [31]. Other approaches, such as AdaGrad, aim to improve the robustness of gradient training by adjusting the updates to parameters according to frequency - e.g. larger updates to infrequent parameters, and smaller updates to frequent parameters [16, 53].

A parallel implementation of SGD largely rests on the idea of splitting up data to be processed in individual runs of the gradient descent. The results are periodically aggregated, and the new model parameters are distributed to processing nodes for further training [54]. Depending on the configuration, the variance within results can result in poor convergence rates [37]. Mahajan et al. suggest a novel implementation which improves the distribution impacts through the use of better approximating functions in the processing nodes, which in turn improves the efficacy of the algorithms convergence [37]. Povey et al. presented a Natural Gradient SGD, which improves the learning rates through the use of a factor matrix used on the new gradients. They were able to show empirical evidence of this improving performance issues introduced from the parameter averaging typically used in parallelization [40].

There have been some more recent improvements focusing on making the SGD algorithm more dynamic, including the Inconsistent SGD (ISGD) and AdaBatch adaptations. The idea behind ISGD is to treat the training on a particular batch as a stochastic process, and adjust it according to the expected loss identified. Gradient updates from small loss batches are relatively small compared to large loss batches, and so by focusing efforts here ISGD is able to optimize performance. Wang et al. performed careful testing of SGD vs. ISGD and found inconsistent training to consistently outperform in terms of convergence speed and results [50]. It is worth noting that due to its nature, ISGD can be effectively combined with other SGD improvements (e.g. Momentum, as per the authors experiments). AdaBatch, as presented by Devarakonda et al., introduces dynamism through the adjustment of batch sizes. Small batch sizes in SGD offer the benefit of faster convergence in fewer training epochs, however larger batch sizes are more computationally efficient due to their applicability to parallelization. The algorithm uses a monotonically increasing batch size, which is started small to gain traction in convergence, and later increased to allow the benefits of data parallelism. The effect, as shown, is to offer improvements in training performance of up to 6x, with less than 1% of accuracy when compared to the fixed batch baselines [14]. Both of these algorithms have been shown to be effective and applicable in the context of online neural network training.

6 Backtesting and Model Validation

Much of financial academic literature is currently facing a problem in terms of validation and verification of results. The primary method of going about these ends in the past has been to perform historical simulations, or backtests , in order to prove profitability of a trading strategy. The recent advances in both technology and the algorithms available to

construct these strategies has resulted in researchers being able to run so many iterations of a model or strategy configuration through these backtests, that its become increasingly difficult to control for spurious results, with some papers suggesting that most published research findings are false [30].

The standard way of implementing backtests is to split the data into two portions: an In Sample (IS) portion which is used to train the model, and an Out of Sample (OOS) portion which is used to test the model and validate results. The problem lies in that millions of different model configurations might be tested, and if more sophisticated test measures are not in place (i.e. not just the standard Neyman-Pearson hypothesis testing framework is implemented), then it is only a matter of time before a false positive result occurs which shows high performance both IS and OOS (i.e. overfitting). The nature of financial data, where there is a low signal-to-noise ratio in a dynamic and adaptive system, and where there is only one true data sequence, makes it difficult to resolve these issues effectively [3, 38].

Overfitting is not a novel issue, and has of course been tackled in various literature areas, including machine learning. However, in that context, the frameworks are often not suited to the buy/sell with random frequency structure of investment strategies. They also do not account for overfitting outside of the output parameters, or take into consideration the number of trials attempted. Other methods, such as hold-out, are arguably still faulty due to researcher knowledge while constructing models [43]. One of the downfalls of the typical IS-OOS set up in the financial context is also that the most recent (and relevant) data will not be able to be used for the model training.

There have been some suggestions to resolve the problem that is occuring in the literature as a result of this - some work suggesting new frameworks, which this section will cover, and others which focus on the review process or how data and replication procedures are made available [39]. While the points made with regard to the review process and so on are certainly important, they don't aid with more effective model training for the researcher up front, and so will not be covered here.

6.1 Testing Methodologies

Considering the issues laid out above, there has been much work to develop alternative approaches to backtesting. One of the common approaches to avoid backtest overfitting is the hold-out strategy, where a certain portion of the dataset is reserved for testing true OOS performance. Numerous problems have been pointed out with this approach, including that the data is often used regardless, or that awareness of the movements in the data may, consciously or otherwise, influence strategy and test design by the researchers [43]. For small samples, a hold-out strategy may be too short to be conclusive [51], and even for large samples it results in the most recent data (which is arguably the most pertinent) not being used for model selection [23, 3].

There has been work by several authors to try and lay out techniques to try and avert backtest overfitting. The Model Confidence Set (MCS), as developed by Hansen et al. [21], starts with a collection of models or configurations, and remove models iteratively according to a defined loss function. The confidence set is defined by the remaining models once a non-rejection takes place within the process, and these models are considered to be statistically similar within a certain confidence range. MCS is thus able to facilitate equitable model selection. However, Aparicio et al. [2], showed that while MCS is a potential strategy, in practice is is ineffective due to the inordinate requirement of signal-to-noise necessary to identify true superior models, as well as a lack of penalization over the number of trials attempted.

Bailey et a. [3] have developed a more robust approach to backtesting and how overfitting during strategy selection might be avoided. Their research defines backtest overfitting as having occurred when the strategy selection which maximizes IS performance systematically underperforms median OOS in comparison to the remaining configurations. They use this definition to develop a framework which measures the probability of such an event occuring, where the sample space is the combined pairs of IS and OOS performance of the available configurations. The probability of backtest overfitting (PBO) is then established as the likelihood of a configuration underperforming the median IS while outperforming IS.

Formulaically, the definition of backtest overfitting is given by

$$\sum_{n=1}^{N} E[\overline{r_n}|r \in \Omega_n^*]Prob[r \in \Omega_n^*] \leq N/2 \tag{1}$$

Where the search space Ω consists of the N ranked strategies, and their ranked IS performance r and OOS performance $\bar{r}$. This allows the PBO, using the bayesian formula, to be defined as

$$PBO = \sum_{n=1}^{N} Prob[\overline{r} < N/2|r \in \Omega_n^*]Prob[r \in \Omega_n^*] \tag{2}$$

Notably, the above definitions consider IS as the data made available to the strategy selection, rather than the models calibration (e.g. the full IS dataset, rather than, by was of example, the number of days used in a moving average). This allows the model-free and non-parametric nature of the definition.

They further developed the combinatorially symmetric cross-validation (CSCV) framework as a methodology to reliably estimate the probability used in PBO, which allows a concrete application of the concept. The CSCV framework does not require using the typical hold-out strategy (and thus avoids credibility issues), and is ultimately able to provide a bootstrapped distribution of OOS performance.

The methodology can be briefly summarised (skipping some details and nuances) as the following steps:

1 Generate a TxN performance series matrix, M, representing the profits and losses by the N trials over T time periods

2 Partition the M matrix into S submatrices

3 Generate the combination set C of all combinations of the S submatrices

4 For each combination in C:

   a Form the training set by joining the 2 combination sets, and testing set as the rest of the combinations (all in order)

   b Determine the ranked in-order IS and OOS performance for the sets

   c Determine n* as the best performing IS strategy

   d Determine the relative rank of the n* strategys OOS performance, where we should observe that $\bar{r}^*$ systematically outperforms OOS as well. Define logit $\lambda = \frac{\overline{w}_c}{(1-\overline{w}_c)}$, where a high value implies consistency between IS and OOS performance, and thus a low level of backtest overfitting

5 The λ values can then be collected and used to define the relative frequency at which λoccurs across all combination sets in C, signified by f(λ).

The CSCV framework and results thus allows the consideration of several notable statistics. First and foremost, the PBO may now be estimated using the CSCV method and using an integral over the f(λ) function as defined above which offers a rate at which the best IS strategies underperform the median of OOS trials. If $\varphi \approx 0$, it is evidence of no significant overfitting (inversely, $\varphi \approx 1$ would be a sign of probable overfitting). Critically then, a PBO measure may be used in a standard hypothesis test to determine if a model should be rejected or not. This can be extended, as shown by Bailey et al., to show the relationship between overfitting and performance degradation of a strategy. It becomes clear that with models overfitting to backtest data noise, there comes a point where seeking increased IS performance is detrimental to the goal of improving OOS performance.

The CSCV methodology provides several important benefits (some already mentioned) over traditional testing frameworks, including the usual K-fold cross validation used in machine learning. By recombining the slices of available data, both the training and testing sets are of equal size, which is particularly advantageous when comparing financial statistics such as the Sharpe Ratio (SR), which are susceptible to sample size. Additionally, the symmetry of the set combinations in CSCV ensure that performance degradation is only as a result of overfitting, and not arbitrary differences in data sets. It is pointed out that while CSCV and PBO should be used to evaluate the quality of a strategy, they should not be the function on which strategy selection relies, which in itself would result in overfitting.

6.2 Test Data Length

The CSCV methodology offers an important but highly generalised framework to assess models and backtest overfitting. It doesnt however indicate which metrics should be used to assess the IS and OOS performance, nor any indication on the amount of data needed to do so effectively. One of the noted limitations of the framework is that a high PBO indicates overfitting within the group of N strategies, which is not necessarily indicative that none of the strategies are skillful - it could be that all of them are. Also, as pointed out, it should not be used as an objective function to avoid overfitting, but rather as an evaluation tool. To this end it helps assess overfitting, but not necessarily avoid it.

A typical measure of evaluation used for financial models is the Sharpe Ratio (SR), which is the ratio of between average excess returns and the returns standard deviation - a measure of the return on risk. In the context of comparing models, SR is typically expressed annually to allow models with different frequencies to be compared. Lo et al. [35] show that annulaized SR can be expressed as

$$SR = \frac{\mu}{\sigma}\sqrt{q} \tag{3}$$

Using sample means and deviations, $\hat{\mu}$ and $\hat{\sigma}$, SR can be shown to converge as follows (as y $\to \infty$)

$$\hat{SR} \to \mathcal{N}[SR, \frac{1 + \frac{SR^2}{2q}}{y}] \tag{4}$$

Thus, when using SR estimations, which follow a Normal distribution, it is possible that where the true SR mean is zero we may still (with enough configurations attempted) find an SR measurement which optimises IS performance. This is shown by Bailey et al. [4], who propose the non-null probability of selecting an IS strategy with null expected performance OOS. Notably, typical methods such as hold-out once again fail, as the number of configurations attempted are not recorded. They add a further derivation, thich is the Minimum Backtest Length (MinBTL), ultimately showing that

$$MinBTL \approx (\frac{(1-\gamma)Z^{-1}[1-\frac{1}{N}] + \gamma Z^{-1}[1-\frac{1}{N}e^{-1}]}{\overline{E[max_N]}})^2 < \frac{2ln[N]}{\overline{E[max_N]}}^2 \tag{5}$$

The statistic highlights the relationships between: selecting a strategy with a higher IS SR than expected OOS, the number of strategies tested (N), and the number of years tested (y). The equation shows that as the number of strategies tested increases, the minimum back test length much also increase in order to contain the likelihood of overfitting to IS SR.

As shown extensively throughout ML literature, increased model complexity and number of parameters is one of the primary causes of overfitting. In context of the MinBTL formula, model complexity affects the number of configurations that are available and which may be tested, which in turn will increase likelihood of overfitting. A lack of consideration, or reporting, of the number of trials makes the potential for overfitting impossible to assess.

Bailey et al. expanded on this view with assessing the impact of presenting overfit models as correct. They were able to show that in lieu of any compensation effects (i.e. a series following a Gaussian random walk), there is no reason for overfitting to result in negative performance. However, where compensation effects apply (e.g. economic/investment cycles, bubble bursts, major corrections etc.), then the inclusion of memory in a strategy is likely to be detrimental to OOS performance if overfitting isnt controlled for [4].

6.3 Sharpe Ratio

The use of the Sharpe Ratio in financial backtesting is not just an arbitrary or persistent literature choice. The statistic offers two benefits: the effectively strategy-agnostic financial information contained, as well as being relatable to the t-statistic, and so simple to perform hypothesis testing. The SR ratio (estimate from sample as $\hat{SR}$) is defined as

$$SR = \frac{\mu}{\sigma} \tag{6}$$

The t-ratio is defined as

$$t - ratio = \frac{\hat{\mu}}{\hat{\sigma}/\sqrt{T}} \tag{7}$$

Evidently, the link here is trivial, as per formula (3) . As noted earlier though, the chances of overfitting with the SR ratio, even if true mean returns are zero, are relatively significant. One of the strategies employed to try and counteract this is to use a 'haircut', where the reported SR ratio is discounted by 50%.

The 50% however, is merely a rule of thumb, and Harvey et Lui [22] were able to report significant work showing that in a context of multiple testing, the haircut is nonlinear - the highest Sharpe ratios are moderately penalized, whereas the marginal Sharpe ratios were heavily penalized. While initially fairly sensible, Harvey et al raise valid concerns regarding the effect on option strategies, controlling for risk as well as, pertinently, what constitutes an appropriate level of significance testing. In light of this, they develop a p-value based statistic for multiple testing, the haircut adjusted sharpe ratio HSR, as well as expand upon work by Harvey et al. [27] to provide a distribution that can be used in a dependent multiple testing framework with an appropriate p-value adjustment.

This work is relevant, in that the HSR statistics proposed offer another framework in which investment strategies might be evaluated against each other. The primary difference in comparison to PBO and CSCV, is that where they offer a methodology for evaluating strategies within a group, HSR aims to adjust the statistical significance of each strategy such that the overall risk of spurious correlation is controlled for. A benefit of this method is that there is less chance of a relevant strategy being disregarded as a result of just poor peer performance. PBO however, does have the primary benefit of being metric-agnostic, where the HSR framework is largely based on using the Sharpe ratio (though it can be generalized to another statistic with a probabilistic interpretation). Additionally, PBO is generally more in line with

machine learning literature with the cross validation like approach on time series data.

It should be noted, that the literature detailing usage of the Sharpe ratio for strategy comparison is extensive, with numerous variations and methodologies offered [5]. However, the crux of this paper lies in whether an online neural network is able to make effective enough predictions that a strategy might use the predictions to be profitable. The subtlety here is that we will consider the usage of such forecasting *within* a strategy, rather than *as* a strategy itself. In line with this, statistics such as the Sharpe ratio will be used, but not form a critical consideration of the research here as the comparison of strategies used will be a secondary concern.

7 References

[1] Albers S. Online Algorithms: a survey. Mathematics Subject Classification (1991): 68W25, 68W40. Available at: https://link.springer.com/content/pdf/10.1007

[2] Aparicio, Diego and Lopez de Prado, Marcos, How Hard Is It to Pick the Right Model? (December 2017). Available at SSRN: https://ssrn.com/abstract=3044740 or http://dx.doi.org/10.2139/ssrn.3044740

[3] Bailey, David H. and Borwein, Jonathan and Lopez de Prado, Marcos and Zhu, Qiji Jim, The Probability of Backtest Overfitting (February 27, 2015). Journal of Computational Finance (Risk Journals), 2015, Forthcoming. Available at SSRN: https://ssrn.com/abstract=2326253 or http://dx.doi.org/10.2139/ssrn.2326253

[4] Bailey, David H. and Borwein, Jonathan and Lopez de Prado, Marcos and Zhu, Qiji Jim, Pseudo-Mathematics and Financial Charlatanism: The Effects of Backtest Overfitting on Out-of-Sample Performance (April 1, 2014). Notices of the American Mathematical Society, 61(5), May 2014, pp.458-471. Available at SSRN: https://ssrn.com/abstract=2308659 or http://dx.doi.org/10.2139/ssrn.2308659

[5] Bailey, David H. and Lopez de Prado, Marcos, The Deflated Sharpe Ratio: Correcting for Selection Bias, Backtest Overfitting and Non-Normality (July 31, 2014). Journal of Portfolio Management, 40 (5), pp. 94-107. 2014 (40th Anniversary Special Issue).. Available at SSRN: https://ssrn.com/abstract=2460551 or http://dx.doi.org/10.2139/ssrn.2460551

[6] G. Bakiri and T. G. Dietterich. Achieving high-accuracy text-to-speech with machine learning. In R. I. Damper, editor, Data Mining Techniques in Speech Synthesis. Chapman and Hall, New York, NY, 2002. 22

[7] Bao W, Yue J, Rao Y (2017) A deep learning framework for financial time series using stacked autoencoders and long-short term memory. PLoS ONE 12(7): e0180944. https://doi.org/10.1371/journal.pone.0180944

[8] Bartlett P., Hazan E. Adaptive Online Gradient Descent. Available at: http://papers.nips.cc/paper/3319-adaptive-online-gradient-descent.pdf

[9] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In Bernhard Scholkopf, John Platt, and Thomas Hoffman, editors, Advances in Neural Information Processing Systems 19 (NIPS06), pages 153160. MIT Press, 2007. Available at: http://papers.nips.cc/paper/3048-greedy-layer-wise-training-of-deep-networks.pdf

[10] Bottou L., Le Cun Y. Large Scale Online Learning. Available at: http://papers.nips.cc/paper/2365-large-scale-online-learning.pdf

[11] Bottou, L. and Murata, N. (2002). Stochastic Approximations and Efficient Learning. In Arbib, M. A., editor, The Handbook of Brain Theory and Neural Networks, Second edition,. The MIT Press, Cambridge,

[12] Chu C., Time series segmentation: A sliding window approach. Information Sciences, Volume 85, Issues 13, July 1995, Pages 147-173. Available at: https://www.sciencedirect.com/science/article/pii/002002559500021G!

[13] F. L. Chung, T. C. Fu, R. Luk, and V. Ng, Flexible time series pattern matching based on perceptually important points, in Proc. Int. Joint Conf. Artificial Intelligence Workshop Learning from Temporal and Spatial Data in International Joint Conference on Artificial Intelligence (IJCAI'01), Seattle, Washington, USA, 4-10 August, 2001, p. 1-7. Available at: http://hdl.handle.net/10397/48578

[14] Aditya Devarakonda, Maxim Naumov Michael Garland. ADABATCH: ADAPTIVE BATCH SIZES FOR TRAINING DEEP NEURAL NETWORKS. Available at: https://arxiv.org/pdf/1712.02029.pdf

[15] David L. Donoho, High-Dimensional Data Analysis: The Curses and Blessings of Dimensionality (August 8, 2000). Available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.329.3392rep=rep1type=pdf

[16] John Duchi, Elad Hazan and Yoram Singer. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. Journal of Machine Learning Research 12 (2011) 2121-2159. Available at: http://www.jmlr.org/papers/volume12/duchi11a/duchi11a.pdf

[17] Erghan D., Bengio Y., Courville A, Manzagol P., Vincent P. Why Does Unsupervised Pre-training Help Deep Learning?, Journal of Machine Learning Research 11 (2010) 625-660 . Available at: http://www.jmlr.org/papers/volume11/erhan10a/erhan10a.pdf

[18] E.F. Fama, The behavior of stock-market prices, J. Bus., 1 (1965), pp. 34-105. Available at: https://doi.org/10.1086/294743

[19] Jianqing Fan, Runze Li, Statistical Challenges with High Dimensionality: Feature Selection in Knowledge Discovery (7 Feb, 2006). Available at: https://arxiv.org/abs/math/0602133

[20] Fan, J., Fan, Y. (2008). High Dimensional Classification Using Features Annealed Independence Rules. Annals of Statistics, 36(6), 26052637. http://doi.org/10.1214/07-AOS504

[21] Hansen, Peter Reinhard and Lunde, Asger and Nason, James M., The Model Confidence Set (March 18, 2010). Available at SSRN: https://ssrn.com/abstract=522382 or http://dx.doi.org/10.2139/ssrn.522382

[22] Harvey, Campbell R. and Liu, Yan, Backtesting (July 28, 2015). Available at SSRN: https://ssrn.com/abstract=2345489 or http://dx.doi.org/10.2139/ssrn.2345489

[23] Hawkins, Douglas. (2004). The Problem of Overfitting. Journal of chemical information and computer sciences. 44. 1-12. 10.1021/ci0342472.

[24] Geoffrey E. Hinton, Simon Osindero, and Yee Whye Teh. A fast learning algorithm for deep belief nets. Neural Computation, 18:15271554, 2006. Available at: https://www.mitpressjournals.org/doi/abs/10.1162/neco.2006.18.7.1527

[25] G. E. Hinton, R. R. Salakhutdinov, Reducing the Dimensionality of Data with Neural Networks. Science 28 Jul 2006: Vol. 313, Issue 5786, pp. 504-507 DOI: 10.1126/science.1127647. Available at: http://science.sciencemag.org/content/313/5786/504/tab-pdf

[26] Geoffrey E. Hinton, Training Products of Experts by Minimizing Contrastive Divergence,Neural Computation Volume 14 — Issue 8 — August 2002 p.1771-1800 . Available at: https://www.mitpressjournals.org/doi/pdf/10.1162/089976602760128018

[27] Campbell R. Harvey Yan Liu Heqing Zhu, 2016. " and the Cross-Section of Expected Returns," Review of Financial Studies, vol 29(1), pages 5-68.

[28] K. Hornik, Multilayer feed-forward networks are universal approximators, Neural Networks, vol 2, 1989

[29] Hsu D. Time Series Compression Based on Adaptive Piecewise Recurrent Autoencoder (17 August 2017). Available at: https://arxiv.org/pdf/1707.07961.pdf

[30] Ioannidis JPA (2005) Why Most Published Research Findings Are False. PLoS Med 2(8): e124. https://doi.org/10.1371/journal.pmed.0020124

[31] John Langford, Lihong Li and Tong Zhang. Sparse Online Learning via Truncated Gradient. Journal of Machine Learning Research 10 (2009) 777-801 Submitted 6/08; Revised 11/08; Published 3/09. Available at: http://www.jmlr.org/papers/volume10/langford09a/langford09a.pdf

[32] Martin Lngkvist, Lars Karlsson, Amy Loutfi, A review of unsupervised feature learning and deep learning for time-series modeling, Applied Autonomous Sensor Systems, School of Science and Technology, rebro University, SE-701 82 rebro, Sweden. Available at: https://www.sciencedirect.com/science/article/pii/S0167865514000221bi005

[33] LeCun Y., Bottou L, Orr G, Muller K. Efficient Backprop. Available at: http://yann.lecun.com/exdb/publis/pdf/lecun-98b.pdf

[34] Liu X., Lin Z., Wang H. Novel Online Methods for Time Series Segmentation. IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 20, NO. 12, DECEMBER 2008. Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4445667tag=1

[35] Lo, Andrew W., The Statistics of Sharpe Ratios. Financial Analysts Journal, Vol. 58, No. 4, July/August 2002. Available at SSRN: https://ssrn.com/abstract=377260

[36] Yisheng Lv, Yanjie Duan, Wenwen Kang, Zhengxi Li, and Fei-Yue Wang. Traffic Flow Prediction With Big Data: A Deep Learning Approach. IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, VOL. 16, NO. 2, APRIL 2015. Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=arnumber=6894591tag=1

[37] Mahajan D., Keerthi S., Sundararajan S., Bottou L. A Parallel SGD method with Strong Convergence. Available at: https://arxiv.org/pdf/1311.0636.pdf

[38] McLean, R. David and Pontiff, Jeffrey, Does Academic Research Destroy Stock Return Predictability? (January 7, 2015). Journal of Finance, Forthcoming. Available at SSRN: https://ssrn.com/abstract=2156623 or http://dx.doi.org/10.2139/ssrn.2156623

[39] Lopez de Prado, Marcos, The Future of Empirical Finance (May 31, 2015). Journal of Portfolio Management, 41(4). Summer 2015. Forthcoming.. Available at SSRN: https://ssrn.com/abstract=2609734 or http://dx.doi.org/10.2139/ssrn.2609734

[40] Povey D., Zhang X., Khudanpur S. Parallel training of Deep Neural Networks with Natural Gradient and Parameter Averaging. Available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.745.6995rep=rep1type=pdf

[41] MarcAurelio Ranzato, Christopher Poultney, Sumit Chopra, and Yann LeCun. Efficient learning of sparse representations with an energy-based model. In B. Scholkopf, J. Platt, and T. Hoffman, editors, Advances in Neural Information Processing Systems 19 (NIPS06), pages 11371144. MIT Press, 2007. Aavilable at: http://papers.nips.cc/paper/3112-efficient-learning-of-sparse-representations-with-an-energy-based-model.pdf

[42] Robert L. Schaefer (2012) Subset Selection in Regression, Technometrics, 34:2, 229, DOI: 10.1080/00401706.1992.10484917

[43] Schorfheide, Frank, and Kenneth I. Wolpin. 2012. "On the Use of Holdout Samples for Model Selection." American Economic Review, 102 (3): 477-81.

[44] Shai Shalev-Shwartz, Yoram Singer, and Nathan Srebro. Pegasos: Primal Estimated sub-GrAdient SOlver for SVM. In Proceedings of the Twenty-Fourth International Conference on Machine Learning (ICML-07), 2007.

[45] Takeuchi L, Lee Y. Applying Deep Learning to Enhance Momentum Trading Strategies in Stocks. Technical Report, 2013. Available at: http://www.smallake.kr/wp-content/uploads/2017/04/TakeuchiLee-ApplyingDeepLearningToEnhanceMomentumTradingStrategiesInStocks.pdf

[46] Troiano L., Mejuto E., Kriplani P.On Feature Reduction using Deep Learning for Trend Prediction in Finance (11 Apr 2017). Available at: https://arxiv.org/abs/1704.03205.

[47] Tseng P., AN INCREMENTAL GRADIENT(-PROJECTION) METHOD WITH MOMENTUM TERM AND ADAPTIVE STEPSIZE RULE. SIAM J. OPTIM. Vol. 8, No. 2, pp. 506531, May 1998. Available at: https://pdfs.semanticscholar.org/1a29/6a1577478654a54a9f801f93f71b7d853c53.pdf

[48] Vincent P., Larochell H., Lajoie I., Bengio Y., Manzagol P., Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion. Journal of Machine Learning Research 11 (2010) 3371-3408. Available at: http://www.jmlr.org/papers/volume11/vincent10a/vincent10a.pdf

[49] Wan Y., Gong X., Si Y. Effect of segmentation on financial time series pattern matching. Applied Soft Computing Volume 38, January 2016, Pages 346-359. Available at: https://www.sciencedirect.com/science/article/pii/S1568494615006341

[50] Linnan Wang, Yi Yang, Renqiang Min, Srimat Chakradhar. Accelerating deep neural network training with inconsistent stochastic gradient descent. Neural Networks Volume 93, September 2017, Pages 219-229. Available at: https://www.sciencedirect.com/science/article/pii/S0893608017301399

[51] Weiss, S. M, Kulikowski, C. A. (1991). Computer systems that learn : classification and prediction methods from statistics, neural nets, machine learning, and expert systems. San Mateo (Calif.): Kaufmann.

[52] Yin J., Si Y., Gong Z. Financial Time Series Segmentation Based On Turning Points. Proceedings of 2011 International Conference on System Science and Engineering, Macau, China - June 2011. Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=arnumber=5961935

[53] Zeiler M. ADADELTA: An Adaptive Learning Rate Method. Available at: https://arxiv.org/abs/1212.5701

[54] Zinkevich M, Weimer M, Smola A, Li L. Parallelized Stochastic Gradient Descent. Available at: http://papers.nips.cc/paper/4006-parallelized-stochastic-gradient-descent.pdf

[55] Zhao Y., Li J., Yu L. A deep learning ensemble approach for crude oil price forecasting. Available at: https://doi.org/10.1016/j.eneco.2017.05.023

[56] Tong Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In Proceedings of the Twenty-First International Conference on Machine Learning (ICML-04), pages 919926, 2004

[57] Zhou B., Hu J. A Dynamic Pattern Recognition Approach Based on Neural Network for Stock Time-Series. 2009 World Congress on Nature Biologically Inspired Computing (NaBIC 2009). Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=arnumber=5393674