FX Trading via Recurrent Reinforcement Learning

Carl Gold
Computation and Neural Systems
California Institute of Technology, 139-74
Pasadena, CA 91125
Email carlg@caltech.edu

January 12, 2003

Abstract:
This study investigates high frequency currency trading with neural networks trained via Recurrent Reinforcement Learning (RRL). We compare the performance of single layer networks with networks having a hidden layer, and examine the impact of the fixed system parameters on performance. In general, we conclude that the trading systems may be effective, but the performance varies widely for different currency markets and this variability cannot be explained by simple statistics of the markets. Also we find that the single layer network outperforms the two layer network in this application.

1 INTRODUCTION

Moody and Wu introduced Recurrent Reinforcement Learning for neural network trading systems in 1996 [1], and Moody and Saffell first published results for using such trading systems to trade in a currency market in 1999 [3]. The goal of this study is to extend the results of [3] by giving detailed consideration to the impact of the fixed parameters of the trading system on performance, and by testing on a larger number of currency markets.

Section 2.1 introduces the use of neural networks for trading systems, while sections 2.2 and 2.3 review the performance and training algorithms developed in [1] and [3]. Section 2.4 details the application of these methods to trading FX markets with a bid/ask spread, while section 3.1 begins to discuss the test data and experimental methods used. Finally, sections 3.2, 3.3, and 3.4 compare results for different markets and for variations of the network and training algorithm parameters respectively.

2 TRADING WITH NEURAL NETWORKS

2.1 Neural Network Trading Functions

We begin by reviewing the use of recurrent neural networks to make trading decisions on a single price series, following the presentation given in [4] with additional details where appropriate. The input to the neural network is only the recent price history and the previous position taken. Because the previous output is fed back to the network as part of the input, the neural networks are called "recurrent". The output of the network $F \in [-1, 1]$ at time t is the position (long/short) to take at that time. Neutral positions are not allowed so the trader is always in the market, also known as a "reversal system". For a single layer neural network (also known as a perceptron) the trading function is

$$F_t = \mathrm{sign}(\sum_{\mathrm{i}=0}^{\mathrm{M}} \mathrm{w_i r_{t-i}} + \mathrm{w_{M+1} F_{t-1}} + \mathrm{v})$$

where $\vec{w}$ and v are the weights and threshold of the neural network, and r_t is the "price returns" at time t. For trading where a fixed amount is invested in every trade ("Trading Returns") , the price returns are given by

$$r_t = p_t - p_{t-1} \tag{1}$$

In practice the sign function is replaced with a tanh function so that derivatives can be taken with respect to the decision function for training as described in section 2.3. The tanh function is then thresholded to produce the output.

A more complex and in theory more powerful trading rule can be made from a neural network with two layers. The second layer of neurons is also known as the "hidden" layer. In this case the trading rule is:

$$\begin{aligned} F_t &= \mathrm{sign}(\sum_{\mathrm{j}=0}^{\mathrm{N}} \mathrm{w'_j x_j} + \mathrm{v'}) \\ x_j &= \tanh(\sum_{i=0}^{M} W_{i,j} r_{t-i} + W_{M+1,j} F_{t-1} + v_j) \end{aligned}$$

where $\vec{x}$ are the output of the neurons in the first layer, and $\mathbf{W}$, $\vec{v}$, $\vec{w}'$, and v' are the weights and thresholds for the first and second layers of the neural network respectively.

2.2 Returns and Performance

For FX trading it is standard to invest a fixed amount in each trade. Consequently the profit at time T is give by:

$$P_T = \sum_{t=1}^{T} R_t \tag{2}$$

$$R_t = \mu(F_{t-1}r_t - \delta|F_t - F_{t-1}|) \quad (3)$$

where μ is the number of shares traded, and δ is the transaction cost rate per share traded. For test purposes we invest $\mu = \frac{1}{p_t}$ shares per trade so the results in different currency markets are easily comparable as percentage gains and losses.

We use the Sharpe Ratio to evaluate the performance of the system for training. The Sharpe Ratio is given by:

$$S = \frac{\text{Average}(R_t)}{\text{Standard Deviation}(R_t)} \quad (4)$$

The standard deviation in the denominator penalizes variability in the returns.

Calculating the exact Sharpe Ratio at every point in time results in an $O(T^2)$ algorithm, so in order to evaluate the trader's performance we use the Differential Sharpe Ratio [4]. The Differential Sharpe Ratio is derived by considering a moving average version of the simple Sharpe Ratio (4) :

$$\begin{aligned} \hat{S}(T) &= \frac{A_t}{B_t} \quad (5) \\ A_t &= A_{t-1} + \eta(R_t - A_{t-1}) = A_{t-1} + \eta\Delta A_t \\ B_t &= B_{t-1} + \eta(R_t^2 - B_{t-1}) = B_{t-1} + \eta\Delta B_t \end{aligned}$$

where A_t and B_t are exponential moving estimates of the first and second moments of R_t respectively. The Differential Sharpe Ratio is derived by expanding the moving average to first order in the adaptation parameter η, and using the first derivative term as the instantaneous performance measure:

$$D_t = \frac{d\hat{S}_t}{d\eta}|_{\eta=0} = \frac{B_{t-1}\Delta A_t - \frac{1}{2}A_{t-1}\Delta B_t}{(B_{t-1} - A_{t-1}^2)^{\frac{3}{2}}}$$

The Differential Sharpe Ratio provides a convenient assessment of the trader's performance for use in Recurrent Reinforcement Learning, described in the next section.

2.3 Recurrent Reinforcement Learning

The goal of Recurrent Reinforcement Learning is to update the weights in a recurrent neural network trader via gradient ascent in the performance function:

$$w_t = w_{t-1} + \rho\frac{dU_t}{dw_t} = w_{t-1} + \Delta w \quad (6)$$

where w_t is any weight or threshold of the network at time t, U_t is some measure of the trading systems performance, and ρ is an adjustable learning rate.

We also tested using the "weight decay" variant of the gradient ascent learning algorithm. Using weight decay, (6) becomes:

$$w_t = w_{t-1} + \Delta w - \nu w_{t-1} = w_{t-1}(1 - \nu) + \Delta w$$

where ν is the co-efficient of weight decay. In theory, weight decay improves neural network performance because smaller weights will have less tendency to over-fit the noise in the data. [5]

Due to the path dependence of trading, the exact calculation of Δw after t trading periods is:

$$\Delta w_t = \rho\sum_{t'=1}^{t}\frac{dU_{t'}}{dR_{t'}}\{\frac{dR_{t'}}{dF_{t'}}\frac{dF_{t'}}{dw_{t'}} + \frac{dR_{t'}}{dF_{t'-1}}\frac{dF_{t'-1}}{dw_{t'-1}}\} \quad (7)$$

Note that due to the transaction cost for switching position, the return at time t is a function of both the current position and the previous position. As described in [4] we can approximate the exact batch learning procedure described by (7) with an approximate on-line update:

$$\Delta w_t = \rho\frac{dU_t}{dw_t} \approx \rho\frac{dU_t}{dR_t}\{\frac{dR_t}{dF_t}\frac{dF_t}{dw_t} + \frac{dR_t}{dF_{t-1}}\frac{dF_{t-1}}{dw_{t-1}}\} \quad (8)$$

For the Differential Sharpe Ratio the derivative of the performance function with respect to the returns is:

$$\frac{dU_t}{dR_t} = \frac{dD_T}{dR_t} = \frac{B_{t-1} - A_{t-1}R_t}{(B_{t-1} - A_{t-1}^2)^{\frac{3}{2}}}$$

For trading returns, the derivatives of the return function are:

$$\begin{aligned} \frac{dR_t}{dF_t} &= -\mu\,\delta\,\text{sign}(\text{F}_\text{t} - \text{F}_{\text{t}-1}) \\ \frac{dR_t}{dF_{t-1}} &= r_t + \mu\,\delta\,\text{sign}(\text{F}_\text{t} - \text{F}_{\text{t}-1}) \end{aligned}$$

The neural network trading function is recurrent so the derivatives $\frac{dF_t}{dw}$ for on-line training are calculated in a manner similar to back-propagation through time [4] [6]:

$$\frac{dF_t}{dw_t} \approx \frac{\partial F_t}{\partial w_t} + \frac{\partial F_t}{\partial F_{t-1}}\frac{dF_{t-1}}{dw_{t-1}}$$

The derivative of the output function with respect to any weight in the neural network can be calculated trivially for the single layer network, and by using a standard back-propagation algorithm for the two layer neural network. (see e.g. [5]) Thus all of the derivatives needed for the weight update given by (6) and (8) are readily available.

The algorithm used for training and trading is as follows: train the neural network in an initial training period of length L_{train}. The trades and performance during the training period are used to update the network weights but are then discarded so that they do not contribute to the final performance. The training period may be repeated for any number of epochs, n_e. The training period is then followed by an out of sample trading period of length L_{trade}. The trades made during the trading period are the actual trades for the period, and also update of the network weights continues during the trading period. After the trading period, the start of the training period is advanced by L_{trade} and the process is repeated for the entire sequence of data.

2.4 Bid/Ask Trading with RRL

As in [3], when RRL is used for a currency series with bid/ask prices the mid price is used to calculate returns and the the bid/ask spread is accounted for as the transaction cost of trading. That is, for a bid/ask price series the price returns input to the trader in (1) are calculated in terms of the mid-price, $p_t^m = \frac{p_t^a + p_t^b}{2}$, where p_t^a and p_t^b are the bid and ask price at time t respectively, and an equivalent transaction cost rate is applied in (3) to reflect the loss from position changes in bid/ask trading. For trading returns the equivalent transaction cost is simply the spread divided by two:

$$\delta(t) = \frac{p_t^a - p_t^b}{2} \tag{9}$$

where the factor of $\frac{1}{2}$ reflects the fact that the transaction cost is applied for a single change in position, while in reality the bid/ask spread is lost for a change in position from neutral to long and back to neutral (or neutral-short-neutral.) Because currency trading is typically commission free no further transaction cost is applied. For all experiments trading and performance is calculated in this way, but the final profit was calculated both by the approximate method described by using (2), (3) and (9), and also by the exact method of applying all trades at the exact bid and ask prices. The disagreement between the two methods for calculating profits was found to be insignificant.

3 EXPERIMENTS WITH CURRENCY TRADING

3.1 Data and Methods

The RRL traders were tested on the High Frequency Data in Finance (HFDF) 1996 Currency Market price series [1]. The series give prices for 25 different markets, including both major and minor currencies. The HFDF data set includes half hourly bid and ask prices for the entire year, a total of 17568 samples (note that 1996 was a leap year).

The markets were divided into a tuning set consisting of 10 markets on which the fixed parameters of the algorithm were tuned to maximize profits and the Sharpe Ratio, and a test set consisting of the remaining 15 markets in which out of sample performance was evaluated. The major currency markets were split equally into the tuning and test sets, but otherwise the two sets were created at random.

Parameters of the algorithm include the number of neurons in two layers of the network, M and N, the learning rate, ρ, the coefficient of weight decay, ν, the size of the training and test windows, L_{train} and L_{trade}, and the number of epochs of training, n_e. The large number of parameters made it quite impossible to systematically test all but a small number of parameter combinations. The parameters were tuned by systematically varying each parameter while holding the other parameters fixed. After all parameters had been tuned, the previous values would be re-checked to see if their optimum value changed due to the change in the other parameters. (This technique is commonly known as the "greedy graduate student" algorithm.)

In order to assure that the inputs to the neural networks were all in a reasonable range regardless of the magnitude of the prices in the different markets, all price returns were normalized before being input to the neural networks. In order to achieve this, the mean and variance of the price returns was calculated over the first training period (before beginning training) and then all price returns were normalized to zero mean and unit variance with respect to these values before being input to the neural network. Also the data was filtered to remove non-continuous price changes (i.e. "outliers") from the data.

3.2 Comparison of Results for Different Markets

Tables 1 and 2 gives the profit and Sharpe Ratio achieved for each of the currency markets in the tuning set and test set respectively. The results shown are for the final parameter values chosen to optimize performance on the tuning data sets. Overall, both the one layer and the two layer neural networks trade profitably in most of the currency markets. However, the final profit level varies considerably across the different markets, from -80% to 120%. Sharpe Ratios range from being small or even negative in some markets, to surprisingly high Sharpe Ratios of 7 or 8 in other markets.

It turned out that the performance in the currency markets chosen for testing was rather better than on the markets used for parameter tuning. This suggests that in the future this work would benefit from a cross-validation approach to determining the optimal parameters. The variability in the final results for each market is low considering that the random initialization of the network weights means that successive trials will never lead to identical position sequences, and that the final results are path dependent.

Overall it is apparent that some of these results seem rather too good to be true. The explanation for the improbably high performance is most likely the simple price model used in the simulations: a single price quote is used at each half hour and the trader is guaranteed to transact at that price. In reality, FX price quotes are noisy and the tick to tick price returns actually have a *negative* correlation. [2] Consequently many of the prices at which the neural networks trade in simulation are probably not trade-able in real time and performance in real trading can be expected to be worse than the results shown here. Performance with a less forgiving pricing model is currently under study.

Figure 1 shows single trials of a 1 layer neural network trading in the Pound-Dollar (GBP-USD), Dollar-Swiss Franc (USD-CHF) and Dollar-Finnish Markka (USD-FIM) markets. Only in the GBP-USD market is the trader successful throughout almost the entire year. The USD-CHF market shows an example of a market in which the trader makes profits and losses with approximately equal frequency. The USD-FIM is an example of a market where the trader loses money much more than it makes money, most likely because in this market the price movement is

[1] Olsen & Associates HFDF96 Data Set, obtainable by contacting http//:www.olsen.ch

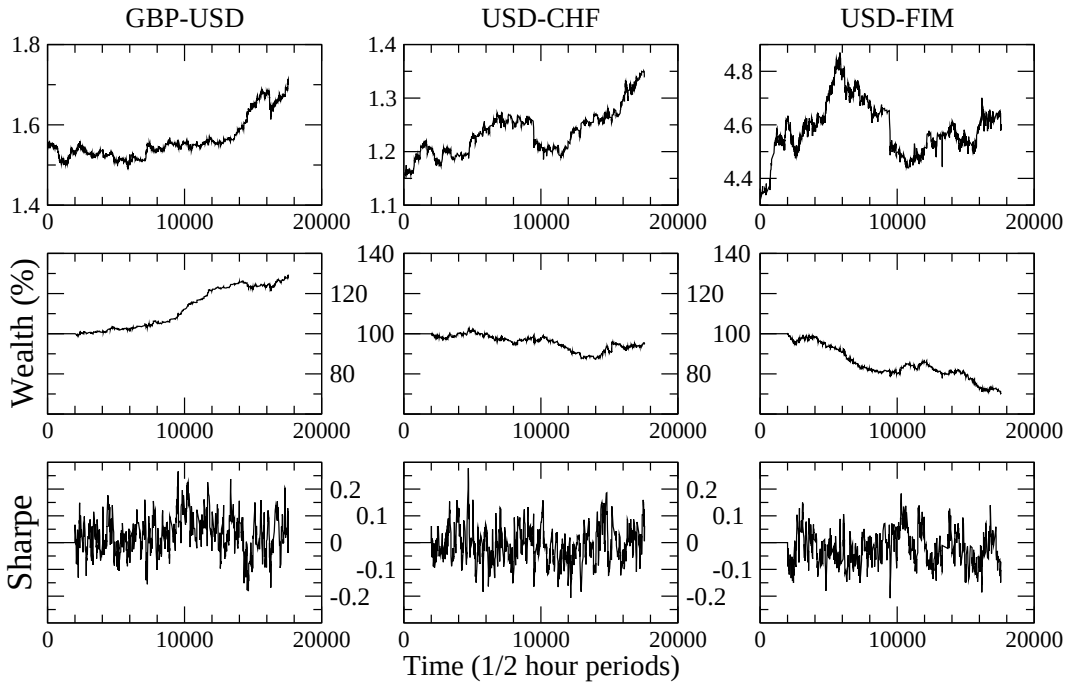

Figure 1: Examples of single trials of a 1 layer neural network trading in the Pound-Dollar (GBP-USD), Dollar-Swiss Franc (USD-CHF) and Dollar-Finnish Markka (USD-FIM) Markets. The Sharpe Ratio shown in the bottom plot is the moving average Sharpe (5) with $\eta = .01$.

small compared to the spread (see below). What is worth noting here is that success and failure for the traders is not absolute, even at a very coarse time scale: In the markets where the trader is successful there are sizeable periods without profits, and in the markets where the trader loses money overall there are periods where the trader does make a profit. The positions taken are not shown - in these example the average holding time ranged from 4 to 8 hours and at this scale individual trades would not be visible.

Basic statistics were calculated for the price series in the tuning markets in an attempt to determine what factors may influence the ability of the neural networks to profit. Statistics calculated from the moments of the price returns, such as the mean, variance, skew and kurtosis all showed no correlation with the ability of the neural networks to profit. However, a measure that did partly explain the profitability in the different markets was the average ratio of the absolute price movement to the spread over small windows of time. As noted above, the neural network traders hold a position for approximately five hours only. Because the bid/ask spread is lost when changing positions it is only reasonable to expect that if the movement of the prices is small compared to the spread then it will not be possible to trade profitably. This intuitive idea can by easily quantified with the Movement/Spread ratio, M/S, given by:

$$\frac{M}{S} = \frac{1}{T-\tau} \sum_{t=1}^{T-\tau} \frac{|\Delta p^m(t,\tau)|}{\bar{s}(t,\tau)} \tag{10}$$

where τ is the window size for which the ratio is calculated, $\bar{s}(t,\tau)$ is the average spread over a window of length τ beginning at time t, and $\Delta p^m(t,\tau)$ is the change in the mid price over same window, i.e. $p^m(t+\tau) - p^m(t)$.

Tables 1 and 2 show the average movement of the mid-price divided by the bid/ask spread for windows of $\tau = 10$, i.e. 5 hours, in each currency series (in the column labeled "M/S"). For the tuning data set the ratio is calculated for the entire data series, while for the test data set the ratio is only calculated for the first training period (in order to preserve the integrity of the data as a true out of sample performance test.) Looking at the tuning set, neither the single nor the two layer neural networks make a profit on any currency where this ratio is close to or below 1. Consequently, markets where the M/S ratio was below 1.5 were deemed "un-tradeable" and were ruled out for consideration of the parameter tuning and for final results in the test set. The results for the un-tradeable markets are shown in tables 1 and 2 for purpose of comparison. It is a reasonable question for further research to determine whether these markets might be tradeable at a lower frequency so that the movement of the prices would compensate for the spread within a number of inputs that the neural network can adapt to. Attempts to adapt traders to low movement markets by using a large number of price return inputs at the same data frequency failed.

However, while a low M/S ratio makes profit unlikely, a high M/S ratio by no means guarantees profitability. For example, the USD-CHF market has one of the highest M/S ratios calculated, yet the neural networks lost money in this market for all combinations of parameters that were attempted. Linear regression was performed for the M/S ratio

Table 1: Tuning Market Results: Final fixed parameters for 1 layer Neural Network: $M = 4$, $\rho = .01$, $\nu = 0$, $L_{train} = 2000$, $L_{trade} = 500$, $n_e = 4$; Final fixed parameters for 2 layer Neural Network: $M = 4$, $N = 16$, $\rho = .15$, $\nu = 1e-5$, $L_{train} = 2000$, $L_{trade} = 300$, $n_e = 5$. Averages and Standard Deviations are calculated for 50 trials of each type of neural network in each currency market. The Sharpe Ratio is the Annualized Sharpe Ratio, profits are the exact profits described in section 2.4, and M/S is the movement/spread ratio described in (10).

		1 layer NN		2 layer NN	
Market	M/S	Profit (%)	Sharpe	Profit (%)	Sharpe
AUD-USD	1.96	17.7 ± 0.8	2.2 ±0.09	18.1 ±2.0	2.3 ±0.26
DEM-ESP	1.17	-7.5 ±11.0	-1.7±2.37	-4.7 ±4.4	-1.1±1.05
DEM-FRF	2.05	31.3 ± 1.3	6.2 ± 0.1	38.9 ±0.8	7.7 ±0.16
DEM-JPY	2.47	17.2 ± 1.2	2.1 ±0.15	10.5 ±1.9	1.3 ±0.23
GBP-USD	2.29	22.6 ± 0.3	3.2 ±0.05	25.3 ±1.1	3.6 ±0.16
USD-CHF	2.68	-4.2 ± 0.6	-0.4±0.06	-15.7±2.1	-1.5±0.21
USD-FIM	0.90	-24.9± 0.3	-2.4±0.15	-65.5±3.2	-6.1±0.30
USD-FRF	2.81	49.3 ± 1.5	5.9 ±0.04	40.1 ±2.0	4.8 ±0.24
USD-NLG	2.43	22.1 ± 0.7	2.4 ±0.09	7.1 ±3.4	0.8 ±0.37
USD-ZAR	1.16	-82.1± 4.4	-8.1±0.41	-76.8±7.4	-8.0±0.74
Average, all markets		4.2 ± 2.2	0.9 ±0.35	-2.3 ±2.8	0.4 ±0.37
Average, M/S > 1.5		22.3 ±0.74	3.1 ±0.08	17.8 ±2.7	2.7 ±0.23

vs. the Sharpe Ratios for the all the markets - the correlation co-efficient was .17 for the single layer neural network and .09 for the two layer neural network. Precisely what characteristics of a currency market make it possible to trade profitably with an RRL trained neural network is one of the most important issues for further research in this area.

Table 3 gives some simple per trade statistics for the 1 layer neural network in the tuning markets, including the holding time and profit per trade. Note that the M/S ratio has a negative correlation to the mean holding time - the price series with low M/S ratios tend to have a longer mean holding time. Linear regression of the holding time to the M/S ratio on all markets gave a correlation co-efficient of -.48. This is consistent with the results in [1] and [4] which showed that RRL training adapts traders to higher transaction cost by reducing the trading frequency. In the case of FX trading a lower M/S ratio means that that the spread is a relatively higher equivalent transaction cost and we should expect trade frequency to be reduced.

3.3 Effect of Network Parameters

The parameters that must be chosen for the neural networks are the number of layers in the network and the number of neurons in each layer. Note that the number of neurons in the input layer is the choice for the number of price return inputs given to the network (minus one for the recurrent input.) Examining tables 1 and 2, one of the most striking results of this study is that a neural network with a single layer outperforms a neural network with two layers. This may seem like a surprise because 2 layer networks are generally a more powerful learning model; a single layer neural network can only make decisions that are linearly separable in the space of the inputs [5]. However, the high level of noise in financial data may make the learning ability of the two layer network a liability if it memorizes noise in the input data. On the other hand, this result seems to imply that reasonably good trading decisions for FX markets are in some sense linearly separable in the space of the recent price returns and therefore not as complex as we may believe.

Figure 2 shows the effect of the number of price return inputs on the performance of a single layer neural network trader. The results shown for three currency markets display the difficulty of choosing fixed parameters for neural network currency traders.

While the trader performs best in the Australian Dollar - US Dollar (AUD-USD) market with fewer inputs, its performance in the - Dollar - Dutch Guilder (USD-NLG) market is best with a larger number of inputs. In the GBP-USD market performance is best at fewer inputs, but the worst performance is for traders with an intermediate number of inputs. This seems to defy common sense, yet the the result is a genuine quirk of the market. The final number of price inputs that was chosen to optimize the average performance in the tuning markets was 4. This results is also rather surprising because it means that the neural network traders only need the price returns from the most recent two hours in order to make effective trading decisions.

Comparing the profit with the Sharpe Ratio for the currency markets in figure 2 shows that there does not appear to be a tradeoff between profit and stability of returns when choosing the optimal values for the fixed parameters of the model. The number of inputs that has the highest profit has the highest Sharpe Ratio as well. This was true for nearly all of the fixed parameters and currency markets.

For the two layer neural networks the dependence on the number of price return inputs was similar, and the final value chosen for optimal performance in the tuning markets was

Table 2: Test Markets Results: Fixed parameters for the Neural Networks and column headings are the same as described in table 1.

		1 layer NN		2 layer NN	
Market	M/S	Profit (%)	Sharpe	Profit (%)	Sharpe
CAD-USD	1.6	7.6 ±0.5	1.9 ±0.13	-1.3 ±1.0	-0.3±0.25
DEM-FIM	0.84	12.1 ±0.9	1.8 ±0.14	23.7 ±1.3	3.6 ±0.19
DEM-ITL	1.97	68.5 ±1.5	8.0 ±0.19	71.9 ±2.9	8.4 ±0.35
DEM-SEK	1.92	48.8 ±0.6	5.9 ±0.08	42.4 ±2.4	5.1 ±0.31
GBP-DEM	1.64	-20.7 ±0.7	-3.1 ±0.10	-25.7±2.7	-3.8±0.41
USD-BEF	2.89	55.5 ±4.4	4.4 ±0.35	45.5 ±4.9	3.6 ±0.38
USD-DEM	3.39	14.4 ±1.3	1.9 ±0.17	12.3 ±2.8	1.6 ±0.36
USD-DKK	2.17	6.6 ±0.7	0.8 ±0.09	-8.3 ±1.9	-1.0±0.22
USD-ESP	1.25	9.4 ±7.1	0.7 ±0.46	-6.1 ±7.1	-0.4±0.46
USD-ITL	1.74	118.9±2.2	12.0±0.24	88.3 ±1.8	9.0 ±0.19
USD-JPY	2.69	13.2 ±0.9	1.7 ±0.11	8.3 ±3.6	1.1 ±0.44
USD-MYR	1.50	-1.9 ±0.6	-0.6 ±0.20	-0.7 ±1.4	-0.3±0.48
USD-SEK	1.58	35.9 ±1.2	3.2 ±0.11	17.2 ±2.9	1.5 ±0.27
USD-SGD	0.82	-1.7 ±0.3	-0.4 ±0.06	-8.0 ±0.7	-1.8±0.17
USD-XEU	2.31	58.4 ±0.5	7.3 ±0.06	46.6 ±2.6	5.9 ±0.34
Average, All Markets		28.3 ±1.6	3.0 ±0.16	20.4 ±2.7	2.1 ±0.36
Average, M/S > 1.5		37.0 ±1.3	4.0 ±0.15	27.0 ±2.7	2.8 ±0.37

also 4. The dependence of the result on the number of neurons in the hidden layer was less significant than on the number of price return inputs. For the more profitable markets in the tuning set (GBP-USD, DEM-FRF, USD-FRF) there was no significant dependence on the number of hidden units. For the less profitable markets in the tuning set (AUD-USD, USD-JPY, USD-NLG) there was a slight preference for more units in the hidden layer and in the end the number of hidden units that optimized overall performance in the tuning markets was 16. While this seems to suggests that the less profitable markets are in some sense more complex and that is why addition hidden layer neurons improve performance, the fact that the single layer neural network outperforms the two layer neural network even in these markets makes this explanation seem untenable.

3.4 Effect of Training Parameters

The parameters of the training algorithm include the learning rate and the co-efficient of weight decay, ρ and ν, the length of the training and trading windows, L_{train} and L_{trade}, and the number of epochs for training, n_e. In general choosing optimal values for these parameters suffer from the same difficulties as in choosing the network parameters - optimal values for one market may be sub-optimal for another. Figure 4 illustrates this in the case of the size of the training window for the two layer neural network trader. While trading in the AUD-USD market performance is best with a longer training window, trading USD-FRF the performance is best with a shorter training window. Trading the DEM-FRF has the best performance at an intermediate value. Note that the USD-FRF performance with respect to training window size has the unusual property that as the profit declines the Sharpe Ratio increases slightly. This is the only case noted where the Sharpe Ratio was not strongly correlated with the profit.

While the parameters defining the neural network were relatively independent of each other (i.e. the optimal number of price return inputs did not impact the optimal number of hidden layer neurons or whether the neural network performed better with one or two layers) the parameters of the training algorithm showed a complex interdependence. Figure 3 illustrates this in the case of the number of training epochs n_e and the learning rate ρ for a two layer neural network in the USD-FRF market. For higher learning rates, fewer training epochs is best, while for lower learning rates more training epochs are needed. The optimal results overall were found with a balance between the two as simply using a high learning rate with a single training epoch generally gave worse performance than an intermediate learning rate and number of training epochs.

In the case of weight decay it is worth noting that a small amount of weight decay (on the order of $\nu = 1e-5$) gave some benefit to the two layer neural networks. However, weight decay never helped the single layer neural network for any combination of parameters tested. This result is not surprising since weight decay is theoretically a technique for simplifying the rule learned by the neural network and preventing the neural network from memorizing noise in the data - the single layer network is already about as simple a learning rule as possible so it is not surprising that further simplification gives no benefit. From a different perspective, the performance of 2 layer neural networks generally suffers when large weights result in the hidden layer neurons operating far from the linear range of the $\tanh$ function. Because the 1 layer neural network uses a only a single thresholded

Table 3: Statistics of trades for 1 Layer Neural Network in markets used for tuning: $\bar{H}$ = average holding time in hours, $\bar{P}$ = % profit per trade. Statistics are given for long positions, short positions, winning positions, and losing positions. $\bar{P}_{\%}$ = Overall percent of trades which are profitable.

	H	H_{long}	H_{short}	H_{win}	H_{lose}	P	P_{long}	P_{short}	P_{win}	P_{lose}	$P_{\%}$
AUD-USD	5.9	6.9	4.9	4.4	8.2	0.015	0.017	0.012	0.121	-0.146	60.1
DEM-ESP	7.1	7.8	6.3	5.4	9.1	-0.006	-0.006	-0.007	0.055	-0.076	53.2
DEM-FRF	3.1	2.3	3.9	2.2	4.7	0.015	0.016	0.013	0.057	-0.052	61.2
DEM-JPY	3.8	3.5	4.2	2.9	5.3	0.010	0.013	0.007	0.110	-0.143	60.3
GBP-USD	3.7	4.1	3.3	2.6	5.6	0.012	0.019	0.005	0.081	-0.115	64.9
USD-CHF	4.1	3.7	4.5	2.7	6.1	-0.002	0.003	-0.008	0.111	-0.162	58.3
USD-FRF	3.8	3.5	4.2	2.8	5.8	0.027	0.029	0.026	0.111	-0.126	64.7
USD-FIM	7.8	8.8	6.7	5.5	10.4	-0.026	0.004	-0.056	0.173	-0.261	54.1
USD-NLG	4.8	6.8	2.8	4.1	5.8	0.015	0.021	0.009	0.137	-0.150	57.6
USD-ZAR	11.0	13.4	8.6	9.0	12.2	-0.122	-0.104	-0.140	0.162	-0.296	38.1

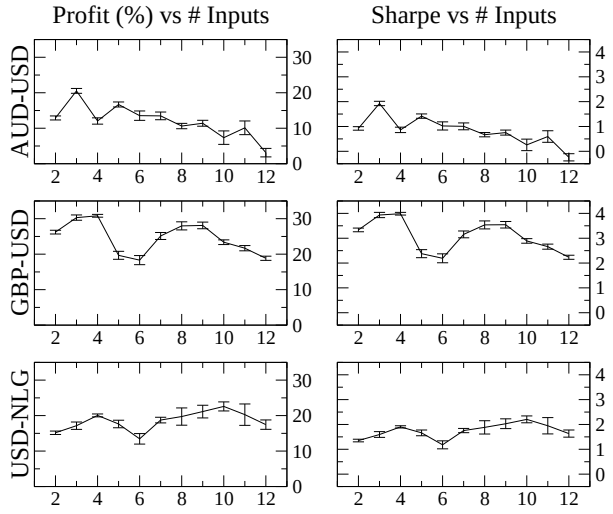

Figure 2: The effect of the number of price return inputs on the profit and Sharpe Ratio for single layer neural networks for three markets in the tuning data set. In all plots the x axis shows the number of inputs, M, and the y axis shows the profit (left column) or the annualized Sharpe Ratio (right column.) The neural networks also have an additional recurrent input not counted in this plot. All other system parameters are fixed at values given in table 1. Results shown are averages and standard deviation for 25 trials.

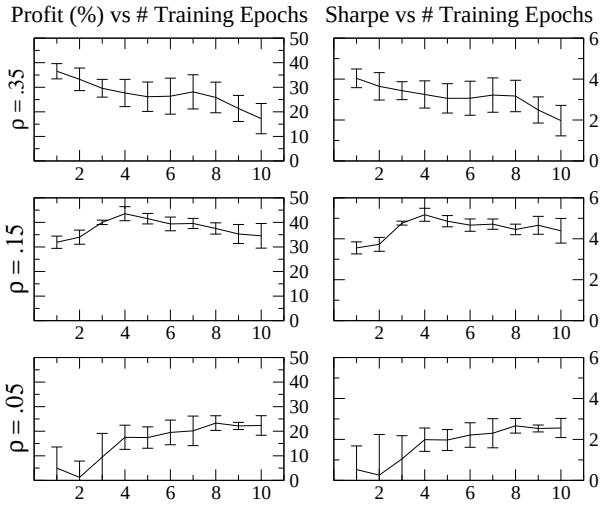

Figure 3: The relationship between Training Epochs and Learning Rate for a two layer neural network in the USD-FRF market. In all plots the x axis shows the number of training epochs, n_e, and the y axis shows the profit (left column) or the annualized Sharpe Ratio (right column.) All other system parameters are fixed at values given in table 1. Results shown are averages and standard deviation for 25 trials.

tanh unit it is in fact a simple linear function and the function is completely unchanged when all the weights are multiplied by an arbitrary constant.

4 CONCLUSIONS

The results presented here suggest that neural networks trained with Recurrent Reinforcement Learning can make effective traders in currency markets with a bid/ask spread. However, further testing with a more realistic and less forgiving model of transaction prices is needed. Initial experiments suggest that performance is substantially decreased and the dependence on the fixed parameters is altered when the the traders cannot automatically transact at any price which appears in the quote series. These experiments are now underway and will be reported when complete.

Regardless of the price model used, the RRL method seems to suffer from a problem that is common to gradient ascent training of neural networks: there are a large number of fixed parameters that can only be tuned by trial and error. Despite extensive experiments we cannot claim that we have found the optimal fixed parameters for currency trading in general as the possible combinations of parameters is very large and many of the parameters have a complex interdependence. A problem that is specific to the currency trading

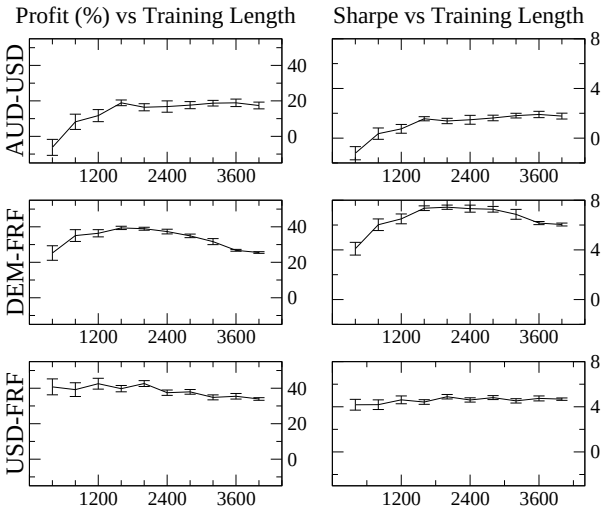

Figure 4: The effect of the size of the training window on the performance of a 2 layer neural network trader. In all plots the x axis shows the size of the training window, L_{train}, and the y axis shows the profit (left column) or the annualized Sharpe Ratio (right column.) All other system parameters are fixed at values given in table 1. Results shown are averages and standard deviation for 25 trials.

applications is that performance depends heavily on characteristics of the currency markets which are understood only poorly at this time. We can rule out trading in sluggish markets with a low ratio of absolute price movement to bid/ask spread, but this criteria does not have predictive value for the performance in markets with adequate price movement.

These conclusions point to a few avenues for further research. Probably the most important need is a more in depth analysis of the properties of the different currency markets that lead to the widely varying performance of the neural network traders. Another interesting question is how the performance may benefit from giving the traders data other than the recent price series, such as interest rates or other information which has an impact on currency markets. Finally, a more open ended goal is to achieve a greater theoretical understanding of how and why Recurrent Reinforcement Learning works that may answer questions like why some markets are tradeable and others not, if we can improve the performance of the neural networks further, or adapt the principle of the RRL method to other learning models such as Radial Basis Functions or Support Vector Machines that do not rely on gradient ascent for parameter tuning.

ACKNOWLEDGEMENTS

I would like to thank Yaser Abu-Mostafa, John Moody, and Matthew Saffell for their direction, feedback and insight throughout this work.

REFERENCES

[1] J Moody, L Wu, Optimization of trading systems and portfolios. In *Neural Networks in the Capital Markets (NNCM*96) Conference Record*, Caltech, Pasadeana, 1996

[2] J Moody, L Wu, High Frequency Foreign Exchange Rates: Price Behavior Analysis and "True Price" Models. In *Nonlinear Modeling of High Frequency Financial Time Series*, C Dunis and B Zhou editors, Chap. 2, Wiley & Sons, 1998

[3] J Moody, M Saffell, Minimizing downside risk via stochastic dynamic programming. In *Computational Finance 1999*, Andrew W. Lo Yaser S. Abu-Mostafa, Blake LeBaraon and Andreas S. Weigend, Eds. 000, pp. 403-415, MIT Press

[4] J Moody, M Saffell, Learning to Trade via Direct Reinforcement, *IEEE Transactions on Neural Networks*, Vol 12, No 4, July 2001

[5] C Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, 1995

[6] P Werbos, Backpropagation Through Time: What It Does and How to Do It, *Proceedings of the IEEE*, Vol 78, No 10, Octoboer 1990