Deep Learning based Global Tactical Asset Allocation

Gaurav Chakravorty [*1], Ankit Awasthi [†1] and Brandon Da Silva [‡2]

[1]Qplum
[2]OPTrust

October 19, 2018

Abstract

We show how one can use deep neural networks with macro-economic data in conjunction with price-volume data in a walk-forward setting to do tactical asset allocation. Low cost publicly traded ETFs corresponding to major asset classes (equities, fixed income, real estate) and geographies (US, Ex-US Developed, Emerging) are used as proxies for asset classes and for back-testing performance. We take dropout as a Bayesian approximation to obtain prediction uncertainty and show it often deviates significantly from other measures of uncertainty such as volatility. We propose two very different ways of portfolio construction - one based on expected returns and uncertainty and the other which obtains allocations as part of the neural network and optimizes a custom utility function such as portfolio Sharpe Ratio. We also find that adding a layer of error correction helps reduce drawdown significantly during the 2008 financial crisis. Finally, we compare results to risk parity and show that the above deep learning strategies trained in totally walkforward manner have comparable performance.

Keywords— tactical asset allocation, deep learning, walkforward, quantitative portfolio management

1 Qplum is a global investment management firm.
2 OPTrust invests and manages one of Canada's largest pension funds and administers the OPSEU Pension Plan, a defined benefit plan with over 92,000 members and retirees.
Qplum or OPTrust may or may not apply similar investment techniques or methods of analysis as described herein. The views expressed here are those of the authors and not necessarily those of Qplum or OPTrust. We would like to thank Dr Michael Steele for his valuable comments and research assistance. Please refer to important disclosures at the end of this document.

[*]gchak@qplum.co
[†]ankit@qplum.co
[‡]bdasilva@optrust.com

1 Introduction

While deep learning has had remarkable progress and is the state of the art in various domains such as computer vision, natural language understanding, and speech recognition, it is safe to say that investment management is still largely not driven by deep neural networks or machine learning in general. One of the major reasons for this is lack of data available in finance as we note later in our results. Moreover, owing to limited training data, poor generalization in out-of-sample data and fundamentally low signal to noise ratio in financial time series data, deep learning methods have not vastly outperformed other methods as witnessed in other domains. Traditionally quantitative asset managers come up work with a large corpus of hand crafted signals and over time, some of the signals might lose value in terms of predicting expected returns and new signals have to be created. Deep learning models trained in a walk-forward manner try to do this in a systematic manner where the model periodically adapts its weights to learn features as more data becomes available. So instead of focusing on the model parameters, we iterate over the policy through which we learn these weights. In this work, we show that the walk-forward learning of deep neural network models can be accomplished effectively with careful choice of certain hyper-parameters which determine how fast the model reacts to new data. Because of low signal to noise ratio in financial time series data, point estimates of expected returns typically have significant errors. Therefore, it is prudent to work in a Bayesian setting where expected returns are used in conjunction with uncertainty estimates.

Most of the deep learning research in finance till date has focused on security selection or predicting returns for single stocks [Bao et al.(2017)Bao, Yue, and Rao], or a cross-section of stocks [Abe and Nakayama(2018)]. [Alberg and Lipton(2017)] focuses on predicting fundamental data for stocks and show that stock selection using the predicted fundamentals leads to improved performance. While [Liew and Mayster(2017)] looks at predicting ETFs, they only work with price-volume features. Partly as a result of features used, they found that the sweet spot of their prediction accuracy was lower than what we notice in our experiments.

In their pioneering work, [Brinson et al.(1995)Brinson, Hood, and Beebower] pointed out that asset allocation explained most of the variance in pension funds' performance. Traditionally active investors have focused on security selection to outperform the broad index and as a result financial markets may be macro inefficient due to insufficient "smart money" being available to arbitrage mispricing effects away [Blitz and van Vliet(2008)].

Tactical asset allocation refers to investment decisions which lead to allocation changes across asset classes, geographies, sectors or industries with respect to a given static strategic allocation. We propose two different ways of portfolio construction. In the first approach we use expected returns and uncertainty estimates to obtain allocation to different asset classes. The position sizing in our case is inspired from Kelly position sizing logic [Kelly(1956)], [Breiman(1961)], [Chakravorty(2017)], and we add a layer of error correction. In the second approach, we take a holistic portfolio construction approach and use the utility function - in this case portfolio Sharpe Ratio, and directly train the feed-forward neural network with the utility function as the optimization objective. The benefits of doing this are manifold. Firstly, better prediction of expected returns does not directly translate to better utility function especially when performance metrics are risk-adjusted or anchored to a certain benchmark. Secondly, the walk-forward learning procedure employed here naturally takes into consideration transaction and slippage costs resulting from excessive turnover. In this way, the network penalizes any behavior in the allocation that hurts the utility function and these can be easily controlled using appropriate slippage or transaction costs. Lastly, although in this work we have simply added a penalty for concentrated portfolios, the same framework can easily be extended to work with a range of objectives. As [Louton et al.(2015)Louton, McCarthy, Rush, Saraoglu, and Sosa] argue, pension investors should have a lever on how tactical their plan allocations be and should be allowed to change based on considerations such as manager skills and risk-return tradeoff. This holistic way of asset allocation can accommodate investor preferences such as the amount of active risk one wants over the strategic allocation while maintaining a tactical tilt based on the model's predictions. While the utility function is single metric, it can be used to incorporate diverse investor preferences such as loss aversion or active risk constraints [Singhal and Chakravorty(2018)].

Macroeconomic data has been shown to be useful in forecasting economic growth (refer to [Estrella and Hardouvelis(1991)]) and could be useful detecting business cycle changes and predicting recession [Estrella and Mishkin(1996)]. [Kliesen(2018)] note that the yield curve helps

ETF	Description	Expense Ratio	Asset Class
BND	Vanguard Total Bond Market ETF	0.05	US Fixed Income
BNDX	Vanguard Total International Bond ETF	0.11	International Fixed Income
VWOB	Vanguard Emerging Markets Govt Bond ETF	0.32	Emerging Fixed Income
VTI	Vanguard Total Stock Market ETF	0.04	US Equities
VGK	Vanguard FTSE Europe ETF	0.10	Europe Equities
VWO	Vanguard FTSE Emerging Markets ETF	0.14	Emerging Equities
VNQ	Vanguard Real Estate ETF	0.12	US Real Estate
VNQI	Vanguard Global Ex-US Real Estate ETF	0.14	Ex-US Real Estate

Table 1: List of ETFs used, with a short description, expense ration and asset class

predict recession two to size quarters ahead when compared to other financial and macroeconomic indicators. [Smalter Hall(2017)] shows that deep neural networks outperform benchmark models for predicting civilian unemployment unto 4 quarters in future. The rest of the article is laid out as follows. In the next section, we discuss the securities that are used, the selection criteria and why ETFs are an efficient investment vehicle for the investment strategy we build. We also discuss the different features derived from end of day price-volume and macroeconomic data which are used as inputs to the deep learning models. Next, we discuss the model specification of the neural network architectures we experimented with, followed by the specifics of the walk-forward training procedure. Then, position sizing using the expected returns or utility function is discussed. This is followed by backtesting assumption. Finally, we discuss the results of our experiments and report the findings.

2 Data

2.1 Traded Securities

Since we are focusing on tactical asset allocation and not security selection, ETFs provide a low cost vehicle for expressing investment decisions across a large cross section of securities without undertaking the execution costs and risks involved. The selection of ETFs listed in table 1 was done by taking into account the following factors to make sure that our reported results are investable and realizable without significant costs.

- *Coverage* : ETFs provide coverage across a large cross-section of securities and alleviate the need to explicitly execute the basket of securities that represents.
- *Expense* : The selection of ETFs was done to make sure that the expense does not lead to a significant drag in performance. Average expense of the ETFs used in the portfolio is less than 13 cents.
- *Liquidity* : Active investment strategies require trading in and out of positions frequently. Higher liquidity leads to lower execution slippage.

2.2 Input Features

We have 46 input features - 14 based on macro-economic data and 32 based on price volume data.

- *Macroeconomic* : Listed in table 2. The choice of which economic indicators was mainly driven by two ideas - access to clean data and relevance for forecasting returns and detecting business cycle changes. It is important to note that the economic data releases are subject to revisions and restatements and naive use of these datasets can lead to significant look-ahead bias. Point in time estimates should be collected to ensure that the learning algorithm is not looking ahead.
- *Price-Volume* : We use features which capture historical price action over different horizons. In particular, we have 4 features per security, and these correspond to lagged returns over [260, 130, 60, 30] days respectively. The choice of the historical returns duration is done partly to eliminate short term noise from the daily return numbers and partly to keep turnover of the resulting investment strategy low.

10 Year - 2 Year Yield Spread	JPY	S&P500
2 Year - 3Month Yield Spread	10 Year Yield	US Inflation
Copper / Gold Ratio	Gold	US Non-farm Payroll
S&P 500/ Dow Jones Ratio	Copper	US GDP Growth
S&P 500/ Russell 2000 Ratio	Oil	

Table 2: List of macro-economic indicators used

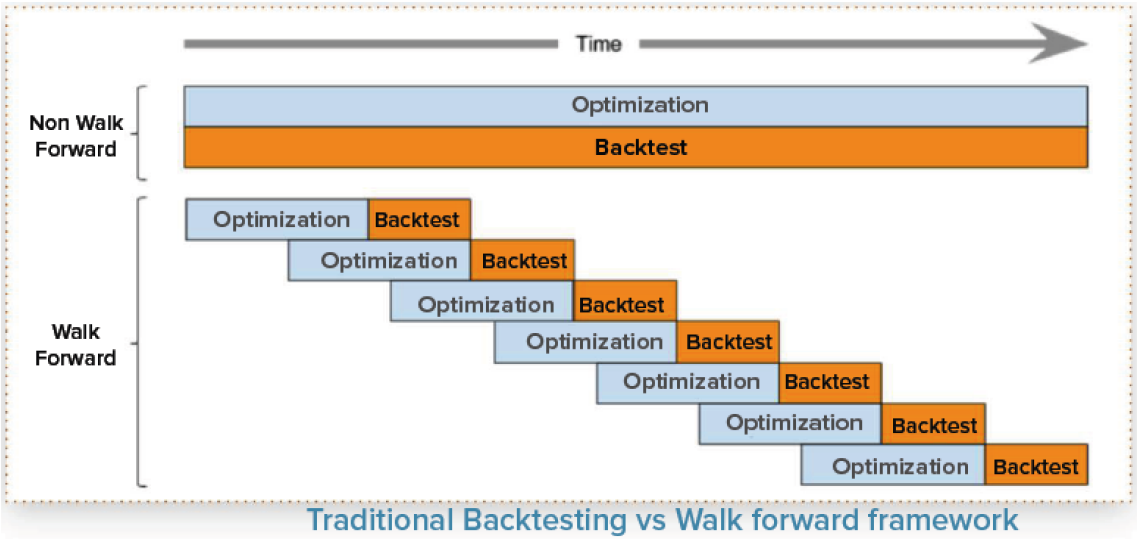


Figure 1: Walk-forward Optimization: In the optimization process, any point in the future does not have a bearing on past returns. This reduces the chances of over-fitting significantly. Instead of finding the best model, walk-forward optimization focuses on finding the best scheme of finding models that perform well in out of sample data.

3 Model Specification

We use feed-forward neural network architecture with the layer size getting smaller as you go deeper in the network (dropout was applied to each layer)

- The weight initialization mentioned in [Glorot and Bengio(2010)] was used for each layer
- We used inverted dropout just in case we wanted to allow for changes in dropout rate across training and testing instances
- To train the model we used mean squared error along with L2-norm regularization on all the weights as our loss function
- Adam optimizer was used to train the model
- The target variables are future returns over 130 days

4 Training Procedure

Training is done in a walk-forward manner, which basically means that at the time of training the parameters are learned without having any look-ahead bias. In other words, weights learned on a certain day are not affected by any data on that day or after that day. Figure 1 depicts how this works in practice. We have defined following hyper-parameters for the walk-forward training procedure.

- Initial number of days required for training - This was fixed at 500 days.
- Number of epochs for initial training.

Hyper-parameter	Final Values
Expected Returns Model Architecture	[Inputs, 72, 48, 32, Outputs]
Dropout probability	0.50
Softmax Layer Architecture	[Inputs, 4, Outputs]
# of Monte Carlo simulations	1000
Initial Training Data	500 days
Initial Training Epochs	100
Update Frequency	30
Model Update Epochs	10
Expected Returns Model Prediction Horizon	130 days

Table 3: List of all the hyperparameters with the final values used. For some hyperparameters like dropout probability or softmax layer architecture we did not search a lot. Wherever possible, we have used model accuracy to optimize hyperparameters.

- Number of epochs to train the model as it starts walking forward. This controls the extent to which one wants the network to adapt to new data as we walk forward in time.
- Update frequency - in how many days do we update our model to incorporate new data.
- We use batch-gradient descent over an expanding window as we walk-forward in time. In other words, every time we retrain the model, we use all the historical data available at that point in time
- The weights at each training interval are retained as the initial weights for the next training interval.

We use historical data since 1995. Data till the end of 2013 is used for training and validation purposes and the remaining as the test data that is only used to evaluate the results once the investment strategy has been fixed. The hyperparameters are optimized based on the walk-forward error only and not strategy performance so as to limit overfitting to the strategy performance. Moreover, strategy performance could potentially be affected by other hyperparameter settings. [1]. Table 3 shows the list of hyperparameters that are optimized and the final values obtained. An exponentially decaying learning rate with a minimum learning rate after our decay window expires is used. Every time we retrain our model we calculate a new mean and standard deviation for our data to normalize it using a z-score (applied column-wise). We use these same means and standard deviations to normalize all new data points that come in until the next time we have to retrain our model, at which time we will calculate a new mean and standard deviation to normalize our data.

At the time of prediction we compute two quantities - expected future returns over the prediction horizon and uncertainties of the prediction. Dropout is a popular regularization technique for training deep neural networks. Dropout is used in the training stage to improve generalization performance [Srivastava et al.(2014)Srivastava, Hinton, Krizhevsky, Sutskever, and Salakhutdinov]. In this work, Dropout has also been used as a Bayesian approximation to derive model uncertainty along with point estimates [Gal and Ghahramani(2016)].

- We use Monte Carlo dropout over M simulations for inference. As it happens while training, each simulations leads to certain connections in the neural network getting dropped randomly leading to different estimates. [2]
- Expected future return is defined as the mean of the predictions across simulations.
- Expected uncertainty of prediction is defined as the standard deviation of predictions across simulations. Let's call this uncertainty measure, dropout uncertainty.

In addition to the dropout uncertainty computed directly from the model, we contrast it with realized volatility as measure of uncertainty.

[1] While intuitively, a model with better accuracy should also lead to better strategy, in practice, this is often not the case because performance might be dictated by brief periods of high volatility and prediction accuracy is an aggregate metric

[2] M was set to 1000 for all simulations

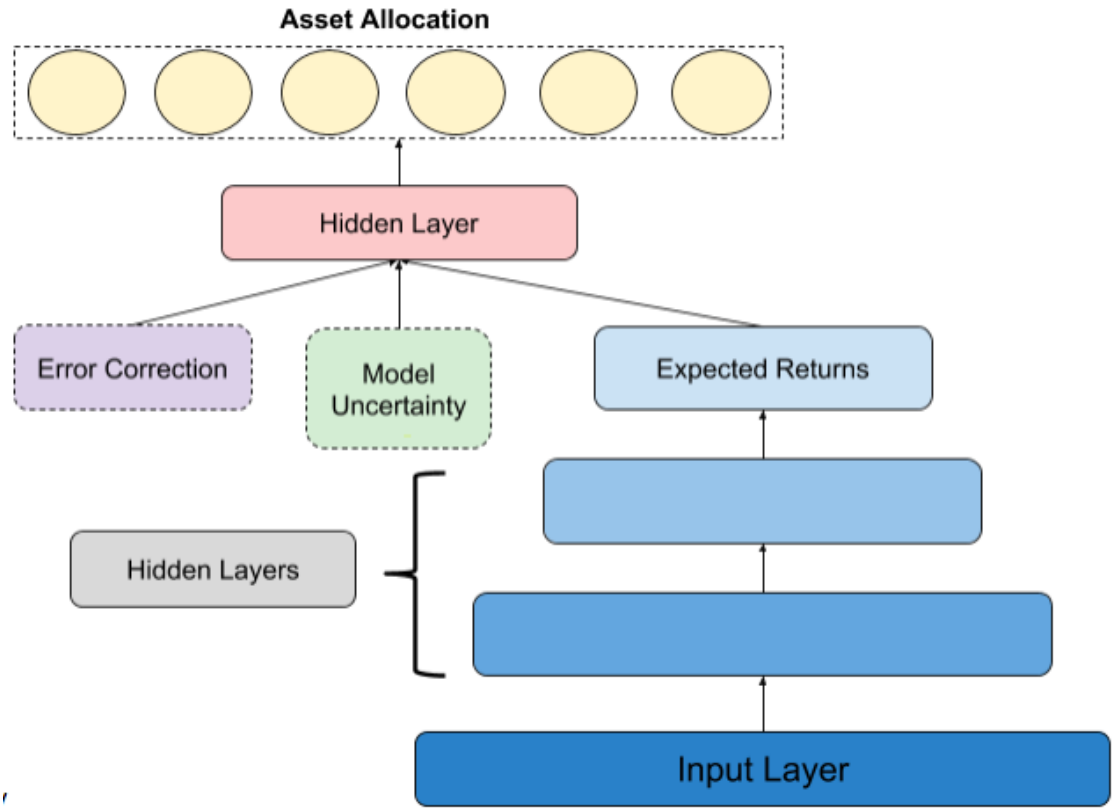


Figure 2: The architecture for the utility function based portfolio construction. The feed-forward network in blue is used to output expected returns which is then in the first method of portfolio construction. The uncertainty estimates in green are either obtained from Monte Carlo simulations using dropout approximation or they are simply obtained from the volatility of respective asset classes. The purple block of error correction are the prediction errors made by the expected returns model in blue. Expected returns, uncertainty estimates and errors are fed into a small 1-layer neural network where the allocations are obtained through a soft-max layer on top.

5 Backtesting

We use daily adjusted prices for backtesting. Both the risk parity benchmarks were implemented with the same parameters which are highlighted below

- Threshold based rebalancing is with a threshold of 5%.
- Transaction costs of 10bps is used for all backtests unless stated otherwise
- Slippage costs are set at 10% worse than the open price, where 10% is measure in terms of the difference between daily open and daily close

It is important to note that in utility based portfolio construction backtested results need to be computed quite often since that is directly used an objective function, therefore while computing backtested results for optimization purposes we simply assume daily rebalancing since that leads to faster computation of the objective function.

6 Portfolio Construction

6.1 Based on Expected Returns Model

Once we have expected returns and uncertainty estimates, we use these to tactically change the weights of an existing allocation within specified limits. The existing allocation could be dollar-

based like a market-cap based allocation or a risk based allocation like risk parity. We take risk parity as the underlying allocation. It is important to note that the following position sizing logic is agnostic to the underlying allocation and can therefore be used with any static strategic allocation.

- We first divide securities into two buckets - high conviction and low conviction. The securities are placed in the high conviction bucket if uncertainty is less than a certain threshold and placed in the low conviction bucket otherwise.
- For securities in the high conviction bucket, we overweight the predictions in the high-conviction bucket using the following factor, where i is a particular ETF[3]

$$F_i = 1 + \tanh\left(\frac{\mu_i}{\sigma_i}\right)^2$$

- The high conviction weights for each of the underlying assets are equal to their default underlying allocations multiplied by their respective factor intensity score (F_i)
- The cumulative weight of the high conviction bucket is then equal to the minimum of the sum of all the high conviction weights and 100 percent
- Once we know the high conviction bucket weight (HW), we can calculate the low conviction bucket weight as LW = 1 - HW

We add a final layer of error correction to boost performance. We calculate an exponentially weighted mean absolute error (denoted by δ) for predictions on each of the underlying assets we're trading. This allows us to discount predictions that haven't been correct for a period of time, and trust predictions that have been right for a period of time. If the errors have been too high recently (above a threshold), this asset will receive an allocation of 0. As a result, our final allocation within the high conviction bucket is:

$$A_i = HW \times \frac{\left(\frac{\mu_i}{\sigma_i}\right)}{\delta_i}$$

The final allocation in the low conviction bucket is the risk-parity allocation for the remaining assets.

6.2 Based on Utility Function

In this portfolio construction, we add a small dense neural network on top of our mean predictions, uncertainty, and exponentially weighted mean absolute error with soft-max layer at the top for portfolio allocations. The architecture is as shown in Figure 2. The soft-max function ensures that all portfolio allocations sum up to 1. The objective function for training this network is simply the utility function. The utility function we use in this case is simply the Sharpe Ratio of portfolio allocations inclusive of all transaction and slippage costs. In order to make sure that the portfolio is not very concentrated, we add term for entropy across portfolio allocations which favors allocations whose weights are spread out rather than being concentrated on a few securities.

7 Results

One of the most interesting aspects of our neural network architecture is the Bayesian approximation. Intuitively, the model uncertainty in expected returns should be a function of expected volatility, since higher volatility would mean higher inherent uncertainty in expected returns. Table 4 shows the correlation of model uncertainties, derived from Monte Carlo simulations using the dropout at the time of inference, with an estimate of the expected uncertainty. The uncertainty estimates seem to be more correlated for volatile asset classes. The low magnitude of the correlation values suggests that these uncertainty estimates might be independent and a plausible explanation could be that the model uncertainty captures the uncertainty in modeling future expected returns as captured in the weights of the neural network whereas the volatility uncertainty captures the inherent volatility of expected future returns irrespective of the model used.

[3]This caps the overweight to a factor of 2x

Security	Correlation between DU and VU
BND	0.08
BNDX	-0.16
VWOB	-0.17
VTI	0.36
VGK	0.30
VWO	0.15
VNQ	0.50
VNQI	0.54

Table 4:
Correlation between uncertainties obtained from the model using dropout approximation and uncertainty obtained using volatility in different asset classes.

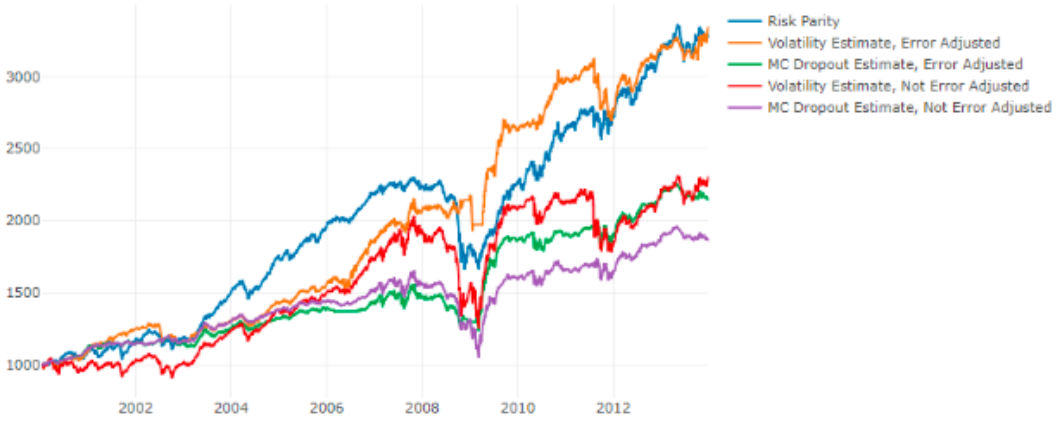


Figure 3: Performance of different portfolio construction methods based on expected returns and its comparison with risk parity. Volatility Estimate refers to the position sizing where volatility is used as the uncertainty estimate whereas MC dropout refers to the position sizing where variance over Monte Carlo simulations is used as volatility estimate. Error Adjusted vs not Error Adjusted depends on where error correction is applied respectively

	CAGR	Worst Drawdown	Sharpe Ratio	Sortino Ratio
Risk Parity	8.94179	27.7729	0.865561	1.20827
Volatility Estimate, Error Adjusted	9.05692	13.9433	0.984973	1.4137
MC Dropout Estimate, Error Adjusted	5.64951	20.9921	0.515271	0.749653
Volatility Estimate, Not Error Adjusted	6.18614	38.2948	0.39649	0.530413
MC Dropout Estimate, Not Error Adjusted	4.5963	36.6101	0.350224	0.487988

Figure 4: Table showing performance metrics of different position sizing heuristics. The position sizing logic with volatility as the uncertainty estimate and error correction has the best performance which other heuristics don't perform as well. Error correction helps reduce maximum drawdown significantly.

Figure 5: The first figure shows mean exponentially weighted MAE (mean absolute error) across all assets (scaled between 0 and 1). The second figure shows asset allocation plot after error correction. The third figure show asset allocation using expected returns and volatility based uncertainty estimate. 2008 financial crisis is particularly interesting - allocation to safe assets increases. Moreover, model accuracy gets increasingly worse and error correction kicks in and moves the allocation to cash thereby reducing the maximum drawdown experienced by the portfolio.

Figure 6: While error correction helps in reducing drawdown by allocating to cash, the same does not happen with naive estimates of expected returns. We contrast the performance against an expected return model where expected return is mean of all historical returns with different thresholds for moving to cash. It shows that deviation from expected returns is especially informative in case of our feed-forward neural network model

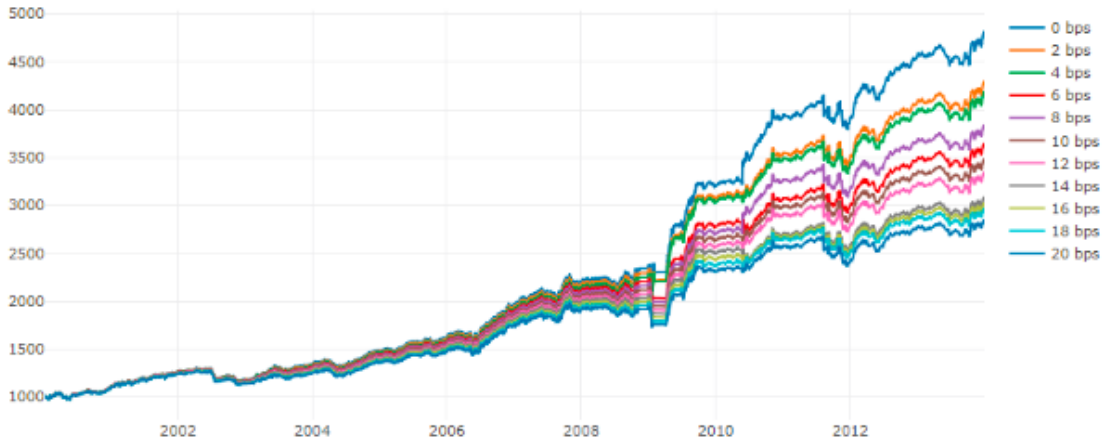


Figure 7: Performance across different transaction cost assumptions for position sizing using expected returns model. Although we have worked with conservative slippage assumptions of 10 bps, we show that the strategy is not very sensitive to the transaction costs - one of the main concerns for large asset allocators

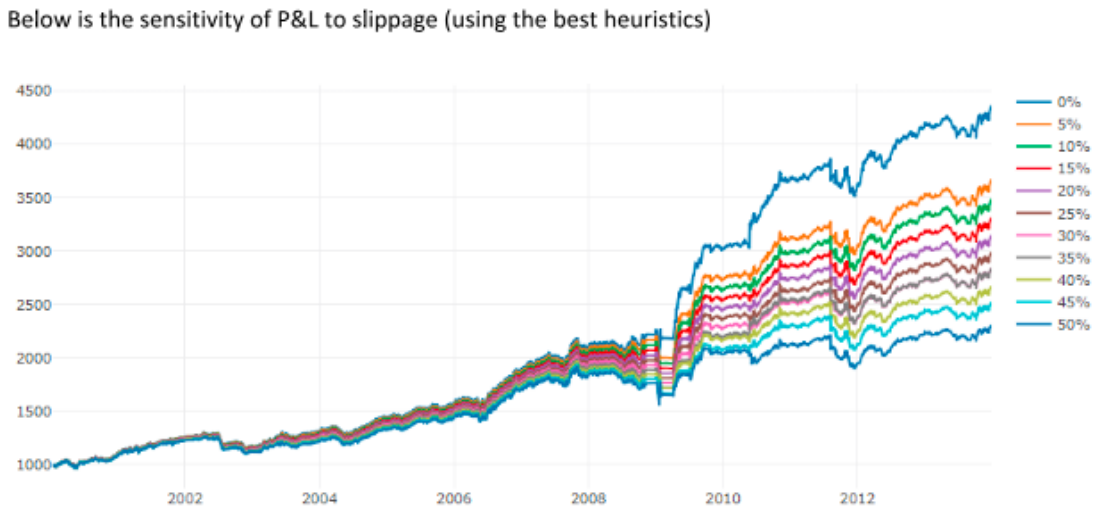


Figure 8: Performance across different slippage assumptions for position sizing based on expected returns model. Although we have worked with a fairly conservative slippage assumptions of 10% from open price (10% of the difference between day's open and day's close against the side of the order), the strategy fairly insensitive to these assumptions

Figure 3 shows the performance of different overlays on top of the risk parity strategy. The volatility based uncertainty estimated with error correction has the best performance. The max drawdown is significantly lower than risk parity as shown in figure 5, it is mostly due to error correction. It might appear that the error correction might in itself be causing this gains, but as shown in figure 6 naive estimates of expected returns don't show similar benefits with error correction.

Since we propose an active trading strategy that trades frequently, it is important to take into account the effect of transaction costs and execution assumptions. Figures 7 and 8 show performance under different assumptions of transaction costs and slippage.

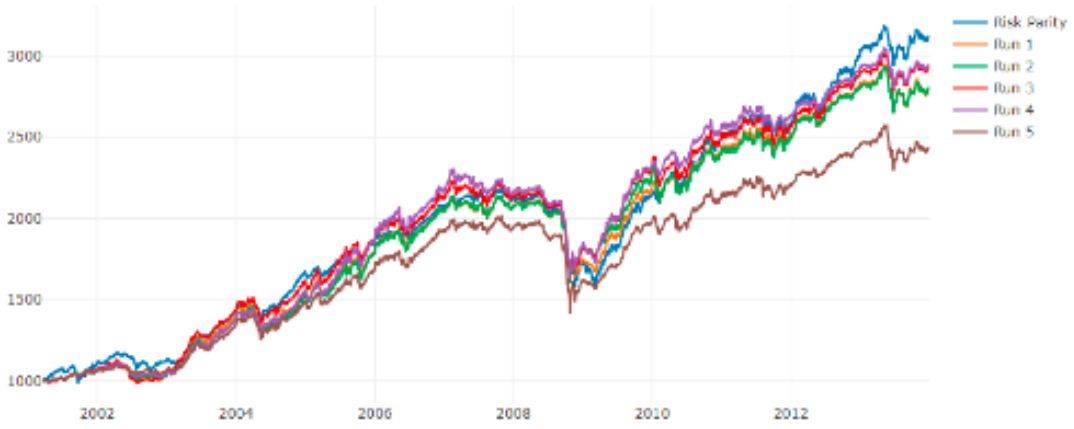
Figures 9 and 10 show the performance of utility function based portfolio construction using volatility and dropout variance as uncertainty estimates respectively. The results are in general better than risk parity. Moreover, it is interesting to note that dropout variance as an uncertainty estimate does significantly better in terms net returns while having somewhat worse Sharpe Ratio. Another important observations was the variance in performance across different runs of the neural network. Although, the aggregate prediction accuracy of the expected returns model did not change much, the resulting strategy's performance changed significantly across different runs. [4]

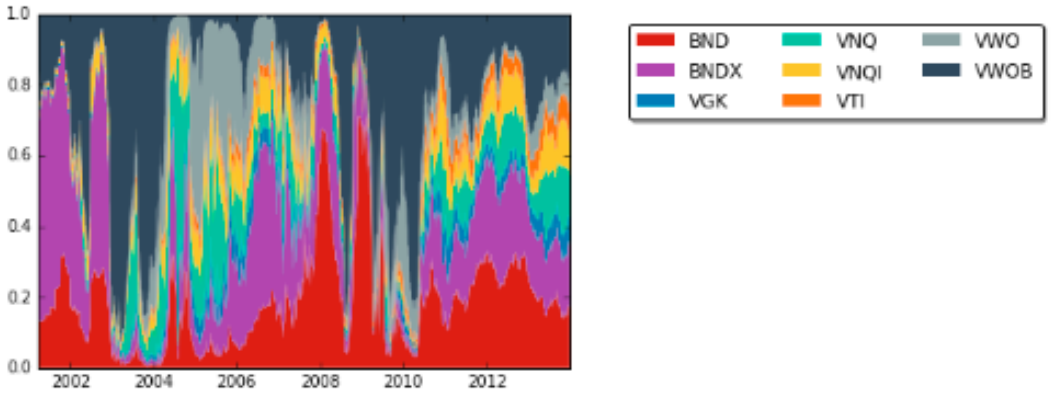
Finally, the figure 11 shows the performance in hold out data. It is important to note that while all the training/validation was done in a walk-forward manner in the period till 2014, this data was not used at all and all hyperparameters of the training model and strategy were fixed before running the strategy on this period. One important observation is that in this period, although there are some significant equity market drops, there is no recession. Major macro indicators such as Unemployment Rate or yield curve inversion were not particularly useful in detecting the equity market drops as they were in the period prior to 2014. The strategy with position sizing is based on the expected returns model with error correction does much worse because of cash allocation due to error correction. The performance of the utility function based allocations however hold continue to perform well in this period.

8 Conclusion

Following are some key takeaways of this research

[4]We tried to reduce the number of parameters in the model to have lower variance runs, but that resulted in worse median performance. Moreover it did not help much in terms of reducing the variance





	CAGR	Worst Drawdown	Sharpe Ratio	Sortino Ratio
Run 1	8.4136	23.8453	0.947307	1.29837
Run 2	8.44053	25.1687	0.891185	1.21639
Run 3	8.80409	26.9733	0.986297	1.35218
Run 4	8.85831	29.1525	0.984984	1.35987
Run 5	7.24692	29.8627	0.795867	1.07304

Figure 9: Results of the portfolio construction using utility function where we take volatility as uncertainty estimate. The first figure plots the returns of 5 different runs of the walkforward model learning procedure against risk parity. The variance comes from different initialization settings. The second figure shows the asset allocation for one run and the last figure tabulates the results of the all the runs. The utility based portfolio construction leads to much better performance compared to risk parity.

	CAGR	Worst Drawdown	Sharpe Ratio	Sortino Ratio
Run 1	9.32689	30.6769	0.740289	1.02741
Run 2	10.6056	29.8779	0.869735	1.22753
Run 3	10.345	28.8834	0.841006	1.18981
Run 4	9.92306	26.2566	0.844648	1.18264
Run 5	10.4827	31.9655	0.870996	1.22579

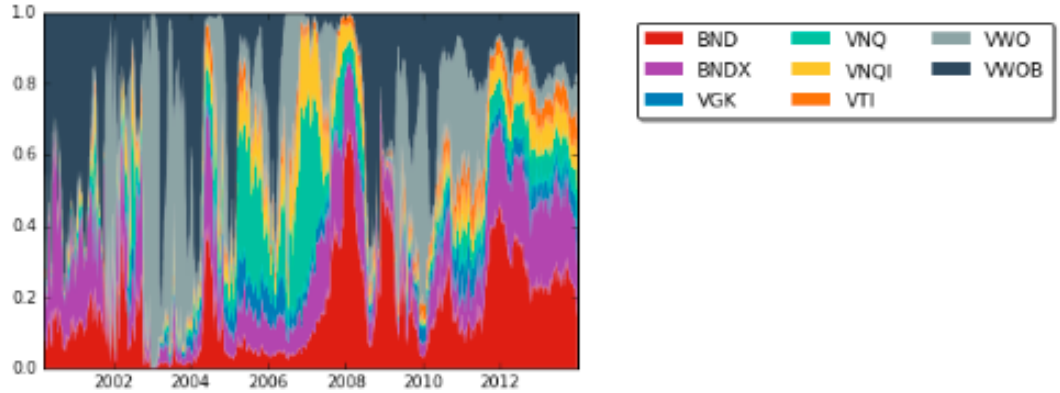


Figure 10: Results of the portfolio construction using utility function where we take Monte Carlo dropout variance as the uncertainty estimate. The first figure plots the returns of 5 different runs of the walkforward model learning procedure against risk parity. The variance comes from different initialization settings. The second figure tabulates the performance across these runs and the last figure shows the asset allocation for a sample run. The utility based portfolio construction leads to much better performance compared to risk parity. Moreover, using MC dropout variance as uncertainty leads to higher returns, although with lower Sharpe Ratio when compared to volatility based uncertainty estimate

Below we show the performance from January 1, 2014 – Present:

	CAGR	Worst Drawdown	Sharpe Ratio	Sortino Ratio
Machine Learning - Heuristics	1.96522	8.33759	0.370985	0.497467
Machine Learning - Softmax (MC)	4.33007	17.2495	0.435836	0.580331
Machine Learning - Softmax (Vol)	5.18963	12.9113	0.617004	0.8212
Risk Parity	4.63405	9.33065	0.756432	1.03652

Figure 11: The results for the period post 2014. Note that while all the simulations presented are done using walk-forward optimization, certain hyper-parameters were fit using the period up till 2014. Performance of the first method of portfolio construction which is based on certain heuristics has worse performance in this period as compared to others partly because of error correction which leads to some cash allocation

- We showed a walkforward implementation of a feed-forward neural network that uses macro-economic indicators and price-volume features to do tactical asset allocation.
- We propose an approach of deriving allocation directly as part of the neural network and optimizing the weights so as to maximize a custom utility function.
- We show that a deep neural network trained in a walk-forward manner with no look-ahead bias shows comparable performance to thoroughly researched and hand-crafted strategies such as risk parity.

9 Future Work

We have identified three main directions for future work.

- *Use of hypothetical data* : Lack of data makes walk-forward training particularly infeasible. One of the major issues we saw in the current implementation was significant variance in performance across different random seed settings. More data would help with convergence. Data augmentation has had remarkable success in some other settings for deep neural networks. We expect either generating simulated data, or using high frequency data as a proxy should help learning better features for the price-volume data which can then be used in conjunction with macro-data to get improved performance.
- *More macro indicators* : While we have used a number macroeconomic data, including more economic indicators could be useful. Moreover, most economic data releases are typically more dense than just single numbers . For instance, US Non-farm payroll numbers come out along with a host of other numbers such as wage growth and participation rate.
- *Other measures for uncertainty* : Given the low signal to noise ratio nature of financial data sets, we strongly believe that model derived uncertainty estimates are important. While in this work model uncertainty did not produce desired results, further work is needed to ascertain the reason for the same. Alternative ways of addressing model uncertainty like variational methods also need to be explored.

References

[Bao et al.(2017)Bao, Yue, and Rao] Wei Bao, Jun Yue, and Yulei Rao. A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PLOS ONE*, 12(7):1–24, 07 2017. doi: 10.1371/journal.pone.0180944. URL https://doi.org/10.1371/journal.pone.0180944.

[Abe and Nakayama(2018)] M. Abe and H. Nakayama. Deep Learning for Forecasting Stock Returns in the Cross-Section. *ArXiv e-prints*, jan 2018. URL https://arxiv.org/abs/1801.01777.

[Alberg and Lipton(2017)] J. Alberg and Z. C. Lipton. Improving Factor-Based Quantitative Investing by Forecasting Company Fundamentals. *ArXiv e-prints*, November 2017.

[Liew and Mayster(2017)] Jim Kyung-Soo Liew and Boris Mayster. Forecasting etfs with machine learning algorithms. *The Journal of Alternative Investments*, 20(3):58–78, 2017. ISSN 1520-3255. doi: 10.3905/jai.2018.20.3.058. URL http://jai.iijournals.com/content/20/3/58.

[Brinson et al.(1995)Brinson, Hood, and Beebower] Gary P. Brinson, L. Randolph Hood, and Gilbert L. Beebower. Determinants of portfolio performance. *Financial Analysts Journal*, 51(1):133–138, 1995. doi: 10.2469/faj.v51.n1.1869. URL https://doi.org/10.2469/faj.v51.n1.1869.

[Blitz and van Vliet(2008)] D.C. Blitz and P. van Vliet. Global Tactical Cross-Asset Allocation: Applying Value and Momentum Across Asset Classes. ERIM Report Series Research in Management ERS-2008-033-F&A, Erasmus Research Institute of Management (ERIM), ERIM is the joint research institute of the Rotterdam School of Management, Erasmus University and the Erasmus School of Economics (ESE) at Erasmus University Rotterdam, June 2008. URL https://ideas.repec.org/p/ems/eureri/12598.html.

[Kelly(1956)] J. L. Kelly. A new interpretation of information rate. *The Bell System Technical Journal*, 35(4):917–926, July 1956. ISSN 0005-8580. doi: 10.1002/j.1538-7305.1956.tb03809.x. URL https://www.princeton.edu/~wbialek/rome/refs/kelly_56.pdf.

[Breiman(1961)] L. Breiman. Optimal gambling systems for favorable games. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*, pages 65–78, Berkeley, Calif., 1961. University of California Press. URL https://projecteuclid.org/euclid.bsmsp/1200512159.

[Chakravorty(2017)] Gaurav Chakravorty. Choosing between mean variance, kelly and risk parity in asset allocation. *Qplum Technical Articles*, December 2017. URL https://www.qplum.co/documents/december-2017-market-commentary.

[Louton et al.(2015)Louton, McCarthy, Rush, Saraoglu, and Sosa] David Louton, Joseph McCarthy, Stephen Rush, Hakan Saraoglu, and Ognjen Sosa. Tactical asset allocation for us pension investors: How tactical should the plan be? *Journal of Asset Management*, 16(7):427–436, Dec 2015. ISSN 1479-179X. doi: 10.1057/jam.2015.26. URL https://doi.org/10.1057/jam.2015.26.

[Singhal and Chakravorty(2018)] Mansi Singhal and Gaurav Chakravorty. Investment objective is not only about maximizing returns. *Qplum Technical Articles*, Jan 2018. URL https://www.qplum.co/documents/january-2018-market-commentary.

[Estrella and Hardouvelis(1991)] Arturo Estrella and Gikas A. Hardouvelis. The term structure as a predictor of real economic activity. *The Journal of Finance*, 46(2):555–576, 1991. ISSN 00221082, 15406261. URL http://www.jstor.org/stable/2328836.

[Estrella and Mishkin(1996)] Arturo Estrella and Frederic S. Mishkin. The yield curve as a predictor of u.s. recessions. *New York Federal Reserve Research Articles*, 1996. URL https://www.newyorkfed.org/research/current_issues/ci2-7.html.

[Kliesen(2018)] Kevin L. Kliesen. Recession signals: The yield curve vs. unemployment rate troughs. *Federal Reserve Bank of St Louis Research Articles*, June 2018. URL https://research.stlouisfed.org/publications/economic-synopses/2018/06/01/recession-signals-the-yield-curve-vs-unemployment-rate-troughs/.

[Smalter Hall(2017)] Cook, Thomas-R Smalter Hall, Aaron. Macroeconomic Indicator Forecasting with Deep Neural Networks. *Federal Reserve Bank of Kansas City Working Paper*, September 2017.

[Glorot and Bengio(2010)] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010. URL http://proceedings.mlr.press/v9/glorot10a/glorot10a.pdf.

[Srivastava et al.(2014)Srivastava, Hinton, Krizhevsky, Sutskever, and Salakhutdinov] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, January 2014. ISSN 1532-4435. URL http://portal.acm.org/citation.cfm?id=2670313.

[Gal and Ghahramani(2016)] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 1050–1059. JMLR.org, 2016. URL http://dl.acm.org/citation.cfm?id=3045390.3045502.

Disclosures

All investments carry risk. This material is intended only for institutional investors, investment professionals, market counterparts or intermediate customers and may not be reproduced or otherwise disseminated in whole or in part without prior written consent.

This document has been provided to you solely for information purposes and does not constitute an offer or solicitation of an offer or any advice or recommendation to purchase any securities or other financial instruments and may not be construed as such. It is not an offer or a solicitation for the sale of a security nor shall there be any sale of a security in any jurisdiction where such offer, solicitation or sale would be unlawful. The factual information set forth herein has been obtained or derived from sources believed to be reliable but it is not necessarily all-inclusive and is not guaranteed as to its accuracy and is not to be regarded as a representation or warranty, express or implied, as to the information, accuracy or completeness, nor should the attached information serve as the basis of any investment decision. Past performance is not indicative of future performance.

This presentation may contain hypothetical performance results. Hypothetical performance results, while imminently useful in understanding the merit of the methodology, have many inherent limitations, some of which are described below. No representation is being made that any account will or is likely to achieve profits or losses similar to those shown. In fact, there are frequently sharp differences between hypothetical performance results and the actual results subsequently achieved by any particular trading program.

One of the limitations of hypothetical performance results is that they are generally prepared with the benefit of hindsight. In addition, hypothetical trading does not involve financial risk, and no hypothetical trading record can completely account for the impact of financial risk in actual trading. For example, the ability to withstand losses or adhere to a particular trading program in spite of trading losses are material points which can also adversely affect actual trading results. There are numerous other factors related to the markets in general or to the implementation of any specific trading program which cannot be fully accounted for in the preparation of hypothetical performance results and all of which can adversely affect actual trading results.

Investing in futures, derivatives or foreign exchange markets is highly speculative and involves substantial investment, liquidity and other risks. CTA managed accounts and hedge funds can be leveraged and their performance results can be volatile. Past performance of issuers, financial instruments and markets may not be indicative of future results, and there is no guarantee that targeted performance will be achieved.

Please visit our website for full disclaimer and terms of use.