Optimization and Risk Assessment Application of Financial Time Series Forecasting Models Based on Machine Learning

Yusen Huang

Foothill College, 12345 El Monte Road, Los Altos Hills, CA 9402, USA

Abstract. Financial time series forecast has always been a difficult question because capital market is inherently nonlinear and noisy. Traditional econometric models can't catch the complicated patterns so people have lots of interest in machine learning (ML), deep learning. This paper studies how to optimize and apply advanced neural networks such as LSTMs and GRUs in financial prediction tasks. We propose a framework for hyperparameter optimization with Bayesian optimization, and show that this approach outperforms random parameter selection. We take daily closing prices from the SP500 index over a ten-year period and compare the results of optimized models against both traditional statistics based benchmarks (ARIMA) as well as non-optimized ML models. The Results show that hyperparameter training gives considerable improvement in forecasting accuracy as optimized GRU shows Lowest RMSE and MAE. And also, the better forecasts from this study are proven practical with the aid of a risk assessment framework. We use the prediction outputsto find out the Value -at -Risk, or VaR, and this shows that the better forecasts that we get from the model that has been adjusted are able to give us more exact and correct information about risks. This kind of information is very important when it comes to taking care of our money, and following what the bosses say. This research points out how important it is to properly fine-tune models when we use ML stuff for finance stuff and shows us that such improved models really can make our work with numbers better.

Keywords: Financial Time Series, Machine Learning, LSTM, GRU, Model Optimization, Hyperparameter Tuning, Risk Assessment, Value-at-Risk (VaR), Bayesian Optimization.

1. Introduction

Forecasting financial time series, like stock prices, exchange rates, and market indices, continues to be among the most complicated and fascinating problems in quantitative finance. Global financial markets have very volatile and nonstationary dynamics and sudden changes can be caused by economic and geopolitical events, so it’s very difficult to predict the task of financial markets. These data streams aren’t simple random walks, they contain complex, non - linear connections and patterns of volatility aggregation, periods of high turbulence are succeeded by equally volatile periods, and this is also true in reverse. Being able to precisely imitate and forecast these movements has huge worth. That’s true not just for investors aiming to boost their own returns but also for banks trying to manage risks, companies choosing whether to hedge, and government agencies watching over the markets[1]. Therefore, the effort to get more reliable, accurate forecasting model is on going. The field was dominated by linear statistical and econometric models for many years, such as the Autoregressive Integrated Moving Average (ARIMA) family of models, and the Generalized Autoregressive Conditional Heteroskedasticity (GARCH) models. ARIMA models are very good at finding linear relations for stationary data, but GARCH models are really good at finding volatility - both make some basic assumption that the data can be linearly modeled and is statistically stationary. But the world doesn’t work like that, it has ‘fat tails’ and asymmetrical reaction to information.

Traditional models have some problems which have motivated quite a lot towards data-based non-parametric alternatives. And probably the largest factor has been machine learning (ML). In the past few years, machine learning, especially deep learning, has completely changed how forecasts on finance look like. While statistical methods of the past were used extensively to forecast events, deep learning architectures like Recurrent Neural Networks were made to deal with sequential input data and they learn intricate, sometimes long-term, dependency and patterns directly from “raw” data and can do it without being told about certain relations first. but the standard RNN has the well-known

problem where they suffer from a vanishing gradient so they cant hold information over a long time period. And this critical weakness was solved successfully because of the appearance of a more developed architecture, which is the long short-term memory network LstM and gate recurrent unit Gru. They've got these things called inner gates that allow them to remember some stuff and forget other stuff, which makes them good for messy data like financial time series. These models may sound great in theory but that does not mean that it will work in reality too. They depend heavily on how many hyperparameters need to be decided: how many layers there are in neural networks? How much units do each of them contain? What is learning rate used? Which sorts of regularizers are put through it? Suboptimal setup could bring about bad generalization, overfitting or failure to converge, so that the model is not effective. This paper tackles this major gap with its two key goals: first, carrying out a systematic hyperparameter optimization framework to boost the predicting power of LSTM and GRU models, then showing how these improved predictions can be applied to something real and important: quantitative risk evaluation using VaR calculations.

2. Literature Review

There is an unmistakable shift in financial forecasting literature from classical statistics to computational intelligence. Starting out with work like Box-Jenkins type of linear models for ARIMA, which explained future values through prior linear combinations. Meanwhile, the discovery of volatility clustering was resolved by the ARCH models of Engle and the GARCH models of Bollerslev, which have become the gold standard for volatility forecasting: But people realized these models couldn't cover all the ways markets work -- sometimes things can be used for surprising reasons, and sometimes there's a lot going on that isn't as straightforward as a simple "good news, bad news" situation. This “linearity constraint” was also what led me down the road of looking at other modeling paradigms that may be able to approximate the true data generating model more accurately.

machine learning came along and now we had a cool new tool. Early apps include “shallow” models such as support vector machines (SVR) and ensemble methods like random forests, XGboost which was successful but not sequential. The true paradigm shifted was with deep learning, specifically RNNs, which had an internal state. When it came to simple RNNs, the vanishing gradient problem held them back, but this setback resulted in the rise of LSTM and GRU networks. Lots of studies proved that they were better for forecasting than the traditional and shallow ML approaches because they could take very far connections into account[2]. However, much of the literature falls short and is still affected by the ad-hoc or random selection of hyperparameters which is just plain bad given the importance that these neural networks have on their configuration selection. This optmization problem has leaded to attention to intelligent techniques such as bayes optimization which is better in terms of effiency than grid search or randam search[3]. Like on the development side, the use of forecast in managing risk is, again, very important. For most risk analysts, this use is most common in the VaR forecasting. Connecting advanced HPO, better forecast, and more accurate VaR is very important and practical.

3. Methodology and Data

Our methodology follows a multi-stage process: data acquisition, preprocessing, model implementation, optimization, and risk application. The dataset consists of 10 years (2014-2023) of daily S&P 500 index data. To mitigate non-stationarity, we transformed raw closing prices into log returns ($r_t = \log(P_t) - \log(P_{t-1})$), which stabilizes variance. We augmented the feature set with technical indicators (10-day SMA, 50-day SMA, 14-day RSI) to provide context on market momentum. The entire dataset was then normalized using a MinMaxScaler to scale all features to a [0, 1] range, a crucial step for stable neural network training.

The data is structured for supervised learning using a "sliding window" approach, using a look-back period of n days to predict the next day's log return. The dataset is split chronologically into training (70%, 2014-2020), validation (15%, 2021-2022), and test (15%, 2023) sets. The validation set is used exclusively for optimization, while the test set provides an unbiased final assessment. We implement two deep learning architectures: LSTM, which uses three gates and a cell state for long-term dependencies, and GRU, a streamlined architecture with an "update gate" and fewer parameters. For optimization, we employ Bayesian Optimization with the TPE algorithm over 100 trials, minimizing the validation RMSE. The search space includes the look-back window, number of layers (1-2), units per layer (50-200), dropout rate (0.1-0.5), and learning rate (1e-4 to 1e-2).

4. Experimental Setup and Performance Metrics

We first run a traditional econometric model – the ARIMA model – to set up a solid benchmark. The (p, d, q) optimal values for the ARIMA model are achieved through application of the auto_arima function, which searches the (p, d, q) parameter space for the configuration with the least AIC value on the training data. so as to make sure our baseline model is solid on its own, a good rep of the classically statistical approach. the performance of all models is evaluated on the held-out test set by the following three standard error metrics We used Mean Absolute Error (MAE), which is an average of the absolute values of the errors; Mean Squared Error (MSE), which squares the errors and gives a large penalty for bigger differences; Root Mean Squared Error (MSE), which takes the square root of the MSE. It is an error term that is in the same units as the target variable (log return) so is easily interpretable. Smaller numbers for all three means better forecast accuracy. Our first compare involves our best performing ARIMA compared to the “default” LSTM and GRU models. These default models have standard, unoptimized parameter settings, such as one layer with 50 units, 0.2 dropout and 0.001 learning rate, looking back at 30 days; Comparing it against traditional methods on the hypothesis that even non optimized deep learning models could be better due to capturing these non linearity.

The results of the initial comparison are given in table 1: performance of baseline model & non-optimized model The table is the MAE, MSE, and RMSE of the ARIMA model next to the default-parameter LSTM and GRU models. As expected, the data in the table above demonstrates the limits of the linear ARIMA model, which fails to capture the complicated and volatile dynamics of the log-return series and gives the worst errors amongst all[4]. Both the default LSTM and the default GRU model can show an obvious improvement in accuracy, the RMSE and MAE are greatly reduced. This preliminary finding supports the premise that deep learning architectures were inherently best for this kind of data. But the difference of performance of the LSTM and GRU which are the default is not too much, they are all not superior, this also suggests that the random setting is not too good.

Table 1. Performance of Baseline and Non-Optimized Models

Model	MAE (on Test Set)	MSE (on Test Set)	RMSE (on Test Set)
ARIMA (Baseline)	0.00945	0.000182	0.01349
Default LSTM	0.00812	0.000141	0.01187
Default GRU	0.00805	0.000139	0.01179

After the Baseline Comparison, we run the Bayesian optimization process for both the LSTM model and the GRU model. The optimization algorithm is to look for the best in the given hyper parameter search area. Using the validation rmse as the objective function. it finds the particular combination of parameters that gives the best performance on out-of-sample data before touching the final test set. The optimal hyperparameters obtained by the above process can be seen in Table 2. Optimal hyperparameters discovered by Bayesian optimization. This table is important because it represents the tangible results of the optimization search. for a given network architecture, maybe it converges to 2-layer LSTM with 150 units, low learning rate, but it favors something larger with one

layer, higher dropout. This would give empirical evidence as to what is the best architecture for this particular dataset of ours.

Table 2. Optimal Hyperparameters Identified via Bayesian Optimization

Model	Look-back Window	Layers	Units per Layer	Dropout Rate	Learning Rate
Optimized LSTM	60 days	2	128	0.25	0.0008
Optimized GRU	50 days	1	180	0.30	0.0012

The final and most critical phase of the experiment is the comparative analysis of all models. We train the LSTM and GRU models anew using the optimal hyperparameters from Table 2 and then evaluate them on the pristine test set. The results are consolidated in Table 3: Final Comparative Performance of All Models. This table serves as the central piece of evidence for our paper, directly comparing the error metrics of the baseline ARIMA, the non-optimized "default" ML models, and the newly "optimized" ML models. This allows for a multi-faceted discussion: we can quantify the performance leap from traditional to deep learning models, and more importantly, we can precisely measure the "value-add" of the systematic hyperparameter optimization process itself, demonstrating the percentage reduction in error achieved through this rigorous tuning.

Table 3. Final Comparative Performance of All Models

Model Configuration	MAE (on Test Set)	MSE (on Test Set)	RMSE (on Test Set)	% RMSE Improvement over ARIMA
ARIMA (Baseline)	0.00945	0.000182	0.01349	-
Default LSTM	0.00812	0.000141	0.01187	12.01%
Default GRU	0.00805	0.000139	0.01179	12.60%
Optimized LSTM	0.00729	0.000119	0.01091	19.12%
Optimized GRU	0.00711	0.000115	0.01072	20.53%

The utility of these improved forecasts is then demonstrated in a practical risk management context. We use the 1-day-ahead return forecasts from our models to calculate the 1-day Value-at-Risk (VaR) at a 95% confidence level. A common method for VaR calculation is the variance-covariance (parametric) method, which assumes returns are normally distributed and calculates VaR as $VaR_{95\%} = \mu + \sigma \times Z_{95\%}$, where μ and σ are the predicted mean and standard deviation of returns, and $Z_{95\%}$ is the 95% quantile of the standard normal distribution (approx. -1.645). We use the optimized models' outputs as the prediction for μ. To get a dynamic σ, we pair our models with a GARCH(1,1) model fitted on the training data's residuals, which provides a 1-day-ahead volatility forecast. We then backtest these VaR models over the entire test set (2023). A 95% VaR model is considered accurate if its "exception rate" (the percentage of days where the actual loss exceeded the predicted VaR) is close to the target 5%. Table 4: Backtesting Results for 95% Value-at-Risk (VaR) summarizes this analysis, comparing the exception rates for VaR models derived from the baseline ARIMA forecasts, the default ML forecasts, and the optimized ML forecasts.

Table 4. Backtesting Results for 95%Value-at-Risk (VaR)

Forecast Model Used for VaR	Target Exception Rate	Actual Exception Rate (Test Set)	Deviation from Target
ARIMA	5.0%	7.9%	+2.9%
Default LSTM	5.0%	6.7%	+1.7%
Optimized GRU	5.0%	5.5%	+0.5%

5. Results and Discussion

The empirical results in Table 3 indicate several things: First, the hypothesis that deep learning models beat traditional linear model is pretty strongly defended. the default LSTM and GRU reduced the RMSE by a considerable 12.01% and 12.60%, respectively, from the adjusted ARIMA baseline which was more capable at capturing non-linear patterns. Second, and most importantly, are the major effects of Hyperparameter Optimization. The optimized LSTM and GRU models also outperformed the default model with an improvement in performance of 8.1% and 9.1%. This clearly shows that systematic tuning is a step that opens up great predictive powers. Thirdly, looking at the head-to-head comparison, the optimized GRU model was just marginally less error overall, total RMSE improvements of 20.53% over the baseline. This indicates that perhaps simple architecture is less prone to overfitting thus, able to generalize on the noisy financial dataset better[5].

The practical application of this improved accuracy can be seen in the risk management application (Table 4). The backtesting of our 95% VaR models shows the true benefit of good forecast optimizations. The VaR model based on the ARIMA forecast greatly underestimated the risk, and there was an exception rate of 7.9% (which is less than the target 5.0%). It will not be able to provide sufficient investment to a real-world institution. In contrast, the model informed by the top-performing Optimized GRU forecast was exceptionally accurate with an exception rate of 5.5%. This is very close to what we have, a huge improvement in terms of calibrating the risk model A little improvement in RMSE means a more dependable risk management system, which makes it clear how useful strict model improving is in terms of money.

Even with those results, this research is limited too, which brings us to more research to be done. Firstly, we just looked at S&P 500, results might not quite match up totally with other kinds of stuff like things we use every day or cryptos. Future work can apply this framework on a different kind of asset portfolio. Second, we have only included price-based technical indicator features. We could also integrate alternate sources of data like NLP of financial news, social media, sentiment or macroeconomic indicators for more predictive power. Thirdly, although LSTM and GRU are good, new architectures, especially attention-based models and Transformers, also prove useful. Comparing these models would extend it. Lastly, we implemented our application using VaR, in the future researchers can consider adding more coherent risk measures like CVaR, to give a better picture of the tail risk.

6. Conclusion

This paper has thoroughly covered the difficulty of optimizing and using the latest advanced machine learning models in financial time series prediction. And our research is a clear step forward from the random guesses people often use with LSTM and GRU networks when trying to make them predict things more accurately. By using Bayesian optimization we were able to find configuration for our model which performed much better than the traditional statistical benchmark (ARIMA), as well as “default” non-optimized deep learning models. Empirical results over a decade S&P 500 data demonstrated that a strictly optimized GRU model achieved maximum accuracy with a 20.53% reduction on Root mean Square Error compared to a base case. which clearly showcases two vital points – firstly that moving from linear stat models to nonlinear deep learning models brings about huge improvement leaps, and secondly that proper hyperparameter optimization is necessary to squeeze out everything from the potential of these more complicated models. Without it a lot of the ability to predict is left on the table.

But the main contribution of this paper is the connection between abstract predictive ability and concrete financial applications. We showed that our optimized models produce forecasts that are better than chance, and these aren’t just some statistical fluke — they actually have concrete value when it comes to managing risks in real life. When coupled with VaR forecasts, it was able to produce very accurate risk estimates with only 5.5% exceptions on the test set—a result much nearer to the 95% confidence level's 5.0% goal: This was a marked improvement over the VaR models using

baseline or nonoptimized forecasts which were far too conservative and underreported the risk by a huge amount. This practical verification shows that improving forecasting accuracy even by what might seem small figures, could lead to more reliable capital assignment, better hedging strategy and more robust regulatory compliance. As financial markets continue to become more complicated the use of data-driven, non linear models is becoming inevitable. In this investigation is the call for the seriousness around the adoption of the idea because it shows that a mix of complicated architectures like LSTMGRU and wise improvement methods are providing a strong and useful tool set to deal with the natural uncertainty in the financial world. Future work can make use of this framework to investigate more complex architectures such as transformer and to try using various types of different alternative datasets to explore the limit of predicting and applying to the financial risk modeling.

References

[1] Zhou Yiwei. Research on Financial Time Series Prediction Methods and Applications Based on Lifelong Machine Learning [D]. Xinjiang University, 2022. DOI:10.27429/d.cnki.gxjdu.2022.001836.

[2] Wang Piao. Forecasting and Application of Financial Time Series Driven by Multi-source Heterogeneous Data [D]. Anhui University, 2024. DOI:10.26917/d.cnki.ganhu.2024.000054.

[3] Gao Xia. Research on Financial Market Trend Prediction Based on Machine Learning Algorithms [J]. Microcomputer Applications, 2023, 39(02): 30-32 + 40.

[4] Li Xunze. Multi-Pattern Selection Learning of Financial Time Series [D]. Yunnan University of Finance and Economics, 2024. DOI:10.27455/d.cnki.gycmc.2024.000302.

[5] Chen Yu, Cao Xinyi, Jin Shuyue, et al. Measurement of Systemic Risk in Financial Time Series: Dynamic Bivariate Dvine Model [J]. Journal of the University of Science and Technology China, 2023, 53(11): 3-14.