# Bayesian Neural Networks for Quantifying Uncertainty and Predicting Stock Fluctuations: A Comparison of LSTMs, GRUs, and Transformers

**Kishore Basu** **Timothy Regis** **Toufic Ayoub**

## Abstract

We present a suite of Bayesian neural network models for predicting fluctuations in intraday stock market data. The models include long short-term memory networks (LSTMs) with and without attention, gated recurrent units (GRUs), and transformers. Our goal is to provide a practical and profitable solution for investors by quantifying uncertainty in model predictions without the need to construct ensembles of networks. We evaluate our models by measuring the error and uncertainty of model predictions, allowing investors to choose a model that complements their own risk profile. Additionally, we develop a payoff algorithm that uses our trained Bayesian neural networks to decide when to buy or sell a given equity. Our experimental results demonstrate that GRUs and LSTMs outperform attention-based models such as transformers in terms of both error and uncertainty. We test the predictive power of our model on popular stocks such as Apple, Microsoft, and Amazon, highlighting the usefulness of our approach in the context of intraday trading.

## 1 Introduction

The field of quantitative finance has witnessed the application of diverse machine learning methodologies. Recent developments have leveraged long short-term memory (LSTM) networks, which integrate current and past information to forecast the price signal (i.e., buy or sell) of a particular equity. Nevertheless, previous research has inadequately addressed the need for quantifying the uncertainty of such predictions. To this end, ensemble models comprising well-designed neural networks have been proposed to improve intraday stock forecasting [2]; however, these models remain computationally expensive and impractical in time-sensitive contexts like intraday trading. In this work, we augment the existing literature by introducing a Bayesian layer that models parameters as distributions rather than point estimates, thereby enabling us to estimate uncertainty without the need to train multiple neural networks [1]. We apply this technique to a range of models, including LSTMs, GRUs, and Transformers. Moreover, we evaluate these models on a novel collection of stock data, where we tailor our approach to specific companies instead of the entire market.

## 2 Related Works

The prediction of stock prices has been a widely studied field for several decades, with extensive research conducted by economists supporting or opposing the efficient market hypothesis. The inspiration for this paper comes from one of these many studies [2]. In this work, the authors proposed an ensemble of LSTM networks to estimate the probability of stock prices moving up or down in the next 5 minutes. They applied several techniques to summarize the predictions into a single point estimate, including equally weighted ensembles and selection of the prediction with the highest AUC score. Their results demonstrated consistent AUC scores above 0.5, providing evidence

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

against the efficient market hypothesis. This work is relevant to our study as we build upon the use of LSTM networks in stock market prediction, while also incorporating other neural network architectures and a Bayesian approach to uncertainty quantification.

In this paper, we take a different approach to the problem of uncertainty quantification in our predictions by using Bayesian neural networks instead of ensembles of networks. We also do not focus on classification, and instead build regression models predicting the stock prices themselves. Recently, there has been a growth of research in Bayesian neural networks as researchers combat pressing issues such as interpretability and uncertainty quantification that neural networks face. A standard neural network minimizes a predefined error function (typically related to the likelihood) using gradient-based methods, resulting in point estimates for our parameters that are our best approximation to the 'true' parameter.

The field of Bayesian statistics offers a framework for contextualizing uncertainty as a function of both prior beliefs and observed data. By modeling weights as distributions, Bayesian neural networks introduce variability into model parameters, and thus into the predictions made by the model. In situations where data is limited, Bayesian neural networks have demonstrated a particular effectiveness due to the way in which they update the posterior distribution of weights as more data becomes available, resulting in a closer approximation of the maximum likelihood estimate of weights [1, 5]. Bayesian neural networks can also be viewed as an ensemble, where each member is parameterized by different values of $\boldsymbol{\theta}$. This approach utilizes Bayes' rule to update the parameters based on the observed data, as given by $p(\boldsymbol{\theta}|D) \propto p(D|\boldsymbol{\theta})p(\boldsymbol{\theta})$, where $p(\boldsymbol{\theta})$, $p(D|\boldsymbol{\theta})$, and $p(\boldsymbol{\theta}|D)$ represent the prior, likelihood, and posterior distributions of the data $D$ and parameters $\boldsymbol{\theta}$, respectively [4]. Consequently, the cost function of a Bayesian neural network is modified to minimize the variational free energy, which is defined as

$$\mathrm{KL}[q_{\boldsymbol{\theta}}(\boldsymbol{\theta})||p(\boldsymbol{\theta}|D)] - \mathbb{E}q_{\boldsymbol{\theta}}[\log p(D|\boldsymbol{\theta})] \tag{1}$$

where KL denotes the Kullback-Leibler divergence between an estimate of the posterior $q_{\boldsymbol{\theta}(\boldsymbol{\theta})}$ and the posterior $p(\boldsymbol{\theta}|D)$, while $\mathbb{E}q_{\boldsymbol{\theta}}[\log p(D|\boldsymbol{\theta})]$ represents the expected log likelihood when weights are sampled from the estimate $q_{\boldsymbol{\theta}(\boldsymbol{\theta})}$. This is in contrast to standard neural networks, where the focus is typically on minimizing the latter term [4].

Our goal is to use Bayesian networks to predict the evolution of high-frequency stock market data, given past history, in comparison to the aforementioned paper that used ensembles of LSTM's for high-frequency stock market prediction [2]. The biggest criticism of ensemble methods in the context of intraday trading is its training time. It takes more than 150 minutes to train a single ensemble, so training a network lacks the temporal precision needed for intraday trading [2]. By employing a Bayesian framework in place of the ensemble, we aim to make significant improvements in training time and make the model viable in real world settings. We hypothesize that Bayesian methods will provide us with accurate predictions while also yielding a measure of uncertainty, with the most accurate (and lowest uncertainty) predictions coming from Transformer models.

## 3 Methods

To investigate our hypothesis, we constructed five different neural network models: a long short-term memory network (LSTM), an LSTM with Attention, a gated recurrent unit (GRU) [3], a multi-headed encoder-only transformer [6], and a decoder-only transformer with masking [6]. Each of these models were trained with $10000$ data points, with a train-test partition of $80\%$. For the transformer models, we ran into many RAM out-of-memory issues. Therefore, we did not experiment with larger dataset sizes. For each model, we added an additional Bayesian layer, with $100$ neurons. This allows the model to capture uncertainties in the data, however there is no general consensus on the amount of Bayesian layers/neurons to include to best capture uncertainty [5]. Each inference of a Bayesian neural network samples $\boldsymbol{\theta} \sim p(\boldsymbol{\theta}|D)$, and then passes these sampled parameters through the neural network, which functions similarly to a likelihood function. This technique can be seen in Algorithm 1.

This method allows us to make several predictions about the future evolution of a stock and estimate the posterior predictive distribution. The posterior predictive distribution is a statistical representation of the predicted values based on the available data. To better understand the reliability of our model's predictions, we can create $95\%$ credible intervals, which show the range of values that are considered plausible given the data.

An example of how credible intervals can illustrate the uncertainty around model predictions is shown in Figure 1. We compared three different models (LSTM, GRU, and Decoder Only Transformer) on the AAPL stock. In the training region, all three models had high confidence in their predictions, resulting in narrow intervals. However, as we move towards the validation region, the models' confidence decreases, resulting in wider intervals. This shows that our models' predictions are more uncertain when we extrapolate beyond the available data. By considering credible intervals, we can gain a better understanding of the limitations of our models and make more informed decisions.

To evaluate our model's performance, we need to consider both the accuracy of the predictions and the level of uncertainty surrounding them. We can compare the error in the validation set to the standard deviation across multiple inferences in the Bayesian network to get a sense of the model's reliability. A good model should have both low error and low uncertainty, indicating that it can make accurate predictions with high confidence. On the other hand, a model with high error but low uncertainty would be overconfident and potentially lead to poor investment decisions. Conversely, a model with low error but high uncertainty would be too conservative and might miss profitable opportunities.

To further assess the usefulness of our models for investment purposes, we developed a payoff algorithm that takes into account the model's predictions and the associated uncertainty. The algorithm is presented in Algorithm 2. The user specifies a cooldown period (represented as the lag time) between transactions, and the model estimates the average stock price and standard deviation at the end of this lag. Based on this information, the algorithm generates a Gaussian variate and compares it to the current stock price. If the generated price is higher than the current price and a trade has not already been initiated, the algorithm buys the stock. If a trade has already been initiated and the generated price is lower than the current price, the algorithm sells the stock. If neither of these conditions is met, the algorithm chooses to hold or refrain from buying the equity until the next time point.

By using the model's predictions and uncertainty estimates, the payoff algorithm can make more informed investment decisions. It takes into account both the potential profitability and the level of uncertainty, ensuring that the investor is not taking excessive risks. This approach can help investors make more informed decisions and increase their chances of achieving their financial goals.

## 4 Experiments

In our experiment, we focused on three popular tech companies: Apple (AAPL), Microsoft (MSFT), and Amazon (AMZN). To evaluate the performance of our models, we used two metrics: pointwise absolute error and pointwise standard deviation across inferences. The results for AAPL, AMZN, and MSFT are shown in Fig. 2, Fig. 3, and Fig. 4, respectively.

We conducted a study to compare different deep learning models for predicting stock prices of Apple (AAPL), Microsoft (MSFT), and Amazon (AMZN). Among the models tested, the Gated Recurrent Unit (GRU) had the lowest average error and consistently low standard deviation for AAPL stock, while the Long Short-Term Memory (LSTM) model had higher uncertainty and error on average than the GRU model. Furthermore, the LSTM outperformed the models with attention, encoder-only transformer, and decoder-only transformer on both metrics. The encoder-only transformer showed high uncertainty in its predictions and a relatively high error. On the other hand, the decoder-only transformer, which uses masked attention, achieved a lower error and standard deviation. This trend was consistent across AAPL and MSFT stocks, except for AMZN stock, where the decoder-only transformer had both low error and low uncertainty, likely due to the difficulty of fitting a model to AMZN stock. However, this should be investigated further in future work.

Our findings revealed an intriguing observation with regards to the performance of our models on AAPL stock. Specifically, for most of the equities we tested, we noticed a consistent trend whereby the incorporation of attention mechanism led to an increase in both the error and standard deviation, as compared to the models that did not utilize attention. This observation raises questions about the effectiveness of attention in forecasting fluctuations in AAPL stock when compared to other models.

In Fig. 5, we present the distributions of the total profit generated from Algorithm 2 across all stocks and models that we trained. Our primary goal is to maximize profit, and thus we analyze the results to identify the models that perform the best. We observe that both the GRU and LSTM models generate significantly higher profits than the LSTM with Attention or Transformer models, which consistently

holds across all tested stocks. These findings indicate that the inclusion of attention results in lower profits for our model. We note that this trend is true for both the LSTM and Attention model and the decoder-only model (which utilizes masked attention), as well as the encoder-only model (which does not). Moreover, the addition of attention leads to a substantial increase in the number of parameters in our model. Therefore, models with attention may require more data to generate equally accurate predictions as models without attention, as we have verified through our experiments. Notably, when we reduced the dataset size, all three models with attention showed a larger error than the original model trained on 8000 data points.

We present the results of our buy and sell signals algorithm applied to MSFT stock in Fig. 6 with varying lag times of $5$, $25$, and $60$. Our model displays the ability to identify uptrends in the stock and make profitable purchases, as evidenced by its decision to buy right before the large increase in share price at around the $1000$th time point in the validation set for all three lag times. However, there are instances where the model produces erroneous results. For example, with a lag time of $60$, the model decides to sell at around the $70$th point in the validation set, resulting in missed profits, as the stock continued to drop. Similar analyses were conducted for AAPL and AMZN stocks, shown in Fig. 8 and Fig. 7, respectively. Despite the model's occasional errors, these results demonstrate the potential profitability of our algorithm in intraday trading.

## 5 Conclusion

In this work, we aimed to devise trading strategies that can operate under uncertainty estimated by Bayesian neural networks. By adopting this approach, hedge funds can overcome several challenges associated with training computationally intensive neural network ensembles to estimate uncertainty. Contrary to our initial hypothesis, we observed that transformer models performed suboptimally compared to models that did not incorporate an attention mechanism. We quantified this observation by measuring the increased error and standard deviation of transformer-based models. Additionally, we proposed a payoff algorithm that leverages the predicted uncertainty in model prediction and the expected future behavior of a given equity to execute trades. We noticed a consistent trend across Apple, Amazon, and Microsoft (AAPL, AMZN, and MSFT, respectively) equities, wherein attention-based models including transformer models, demonstrated inferior performance compared to models that lacked attention, such as long short-term memory networks and gated recurrence units.

The obtained findings presented a surprising outcome, particularly given the recent advancements in GPT and the widespread utilization of attention-based models. Nonetheless, it is crucial to note that the inability to train our models on extensive datasets and explore the associated implications is a significant limitation. Despite the availability of supplementary data, our computational resources were insufficient to undertake this large-scale endeavor. Notably, the training of transformer architectures necessitates a substantial amount of data, considerably more than the other architectures analyzed in this study. Therefore, we conjecture that the relatively lower predictive performance of the trained transformer models is, to some extent, a consequence of this constraint. Future work must investigate the effects of additional compute resources and larger datasets to train the models and further elucidate the observed outcomes.

Intraday trading presents an attractive opportunity for investors due to its potentially lucrative nature. However, in addition to net profits, the associated costs of executing trades should also be considered when evaluating the trading scheme. These costs can vary widely depending on the broker, and may be a fixed fee, or a percentage of the total trade value. In one of our simulations, executing trades based on the models' buy and sell signals netted a profit of $\$150$! However, it executed over 600 trades to achieve this. While our approach demonstrated a net positive profit, the high number of trades required to achieve it may not be practical in real-world trading scenarios. Intraday trading is an attractive but complex field, with highly volatile assets and significant risks. Sophisticated Bayesian models are not immune to these risks, but can help traders make better decisions and modulate their level of risk exposure. By accurately predicting fluctuations in intraday stock market data and quantifying the underlying uncertainty in model predictions, our suite of Bayesian neural network models offers a practical and profitable solution for investors seeking to mitigate risk and achieve net positive profits.

NOTE: Appendix is shown AFTER references.

## References


[1] Charles Blundell et al. "Weight Uncertainty in Neural Networks". In: *32nd International Conference on Machine Learning, ICML 2015* 2 (May 2015), pp. 1613–1622. DOI: 10.48550/arxiv.1505.05424. URL: https://arxiv.org/abs/1505.05424v2.

[2] Svetlana Borovkova and Ioannis Tsiamas. "An ensemble of LSTM neural networks for high-frequency stock market classification". In: *Journal of Forecasting* 38 (6 Sept. 2019), pp. 600–619. ISSN: 1099131X. DOI: 10.1002/FOR.2585.

[3] Kyunghyun Cho et al. "Learning phrase representations using rnn encoder-decoder for statistical machine translation". In: *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference* (June 2014), pp. 1724–1734. DOI: 10.48550/arxiv.1406.1078.

[4] Runhai Feng, Dario Grana, and Niels Balling. "Variational inference in Bayesian neural network for well-log prediction". In: *Geophysics* 86 (3 May 2021), pp. M91–M99. ISSN: 0016-8033. DOI: 10.1190/GEO2020-0609.1. URL: http://pubs.geoscienceworld.org/geophysics/article-pdf/86/3/M91/5381291/geo-2020-0609.1.pdf.

[5] Laurent Valentin Jospin et al. "Hands-on Bayesian Neural Networks-A Tutorial for Deep Learning Users". In: *IEEE Computational Intelligence Magazine* 17 (2 July 2020), pp. 29–48. DOI: 10.1109/MCI.2022.3155327.

[6] Ashish Vaswani et al. "Attention Is All You Need". In: *Advances in Neural Information Processing Systems* 2017-December (June 2017), pp. 5999–6009. ISSN: 10495258. URL: https://arxiv.org/abs/1706.03762v5.

# 6 Appendix

**Algorithm 1** Inference Procedure for a BNN, $\Phi(\boldsymbol{x})$

**Require:** $N > 0$
  $p(\boldsymbol{\theta}|\boldsymbol{D}) \leftarrow \frac{p(D_y|D_x,\boldsymbol{\theta})p(\boldsymbol{\theta})}{\int_{\boldsymbol{\theta}} p(D_y|D_x,\boldsymbol{\theta'})p(\boldsymbol{\theta'})d\boldsymbol{\theta'}}$
  **for** $i < N$ **do**
    $\boldsymbol{\theta}_i \leftarrow p(\boldsymbol{\theta}|D)$
    $\boldsymbol{y_i} \leftarrow \Phi_{\boldsymbol{\theta}_i}(\boldsymbol{x})$
  **end for**
  **return** $Y = \{\boldsymbol{y_i}|i \in [0, N)\}, \{\boldsymbol{\theta}_i|i \in [0, N)\}$

**Algorithm 2** Profit-Seeking Algorithm

**Require:** $n_i > 0, n_v > 0, \text{lag} > 0$ ▷ $n_v, n_i$ is length of validation set, number of Bayesian inferences
**Require:** Pointwise mean, $\overline{\boldsymbol{y}} = \mathbb{E}(\Phi(\boldsymbol{x}))$, and standard deviation, $\sigma_y(\boldsymbol{x}) = \sigma(\Phi(\boldsymbol{x}))$ across $n_i$ Bayesian inferences.
  $p \leftarrow 0$ ▷ Total profit
  $n_t \leftarrow 0$ ▷ Number of Transactions
  inTrade $\leftarrow$ `FALSE` ▷ Have we recently bought
  $i \leftarrow 1$
  curr $\leftarrow y[0]$
  **while** $i < n_v$ **do**
    $\mu \leftarrow \frac{1}{\text{lag}} \sum_{j=i}^{i+\text{lag}} \overline{\boldsymbol{y}}[j]$
    $\sigma \leftarrow \frac{1}{\text{lag}} \sum_{j=i}^{i+\text{lag}} \sigma_y[j]$
    $z \leftarrow \mathcal{N}(\mu, \sigma)$
    **if** $z \geq$ curr & not inTrade **then**
      purchasePrice $\leftarrow y[i]$
      inTrade $\leftarrow$ `TRUE`
    **else**
      **if** $z <$ curr & inTrade **then**
        sellPrice $\leftarrow y[i]$
        $p \leftarrow p +$ sellPrice $-$ purchasePrice
        $n_t \leftarrow n_t + 1$
        inTrade $\leftarrow$ `FALSE`
      **end if**
    **end if**
    curr $\leftarrow \overline{\boldsymbol{y}}[i]$
    $i \leftarrow i +$ lag
  **end while**
  **return** $p, p/n_t$


Figure 1: Bayesian neural network predictions on Apple (AAPL) stock. Results shown for an LSTM model (a), GRU (b), and decoder-only transformer with masked attention (c). True trend of AAPL stock is shown in red. Training and validation regions are separated by a vertical red line, with the training region on the left, and the validation region on the right. Mean predicted output across $n_i = 30$ Bayesian inferences is shown in black in the validation region. Since the mean prediction is close to the true stock price, there is a significant degree of overlap between the red trend and black trend in the validation region.

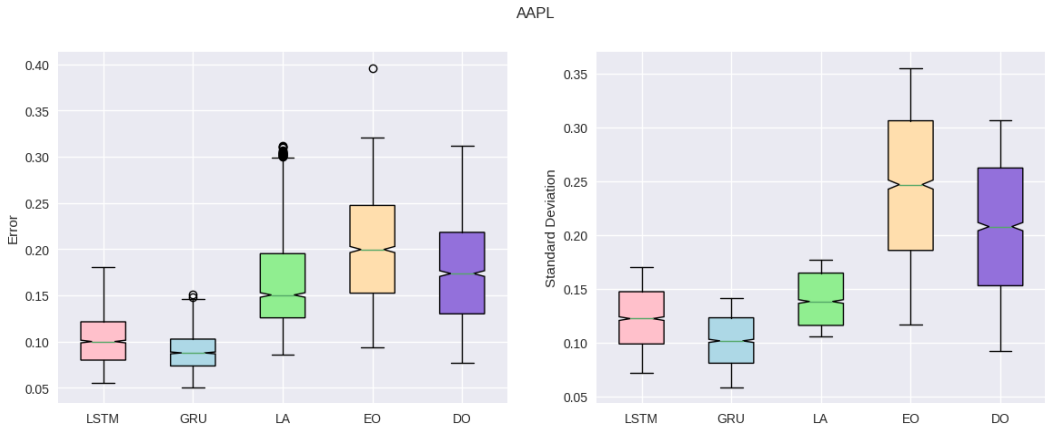


Figure 2: Distribution of pointwise error (absolute difference) and standard deviation of inferences in (a) and (b), respectively in the AAPL validation set. LSTM refers to a long short-term memory network, GRU a gated recurrence unit, LA refers to an LSTM with an Attention layer, EO refers to an encoder-only transformer, and DO refers to an decoder-only transformer.

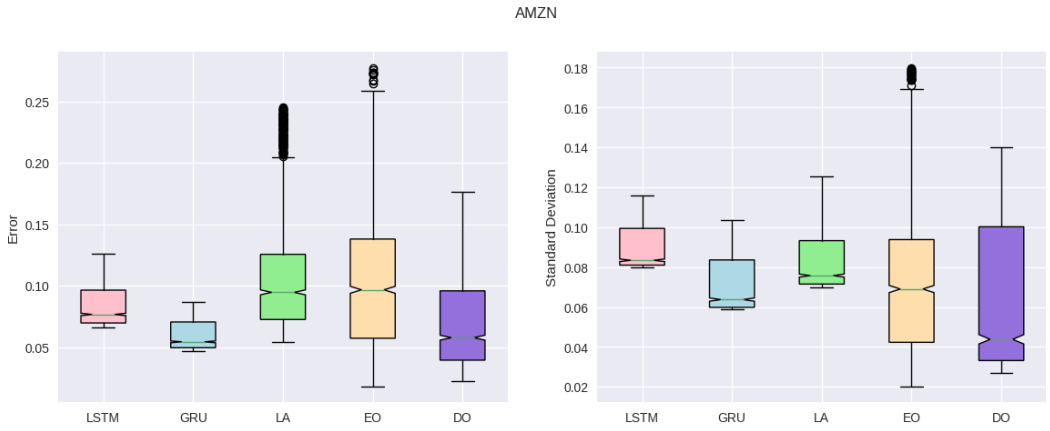


Figure 3: Distribution of pointwise error (absolute difference) and standard deviation of inferences in (a) and (b), respectively in the AMZN validation set. LSTM refers to a long short-term memory network, GRU a gated recurrence unit, LA refers to an LSTM with an Attention layer, EO refers to an encoder-only transformer, and DO refers to an decoder-only transformer.

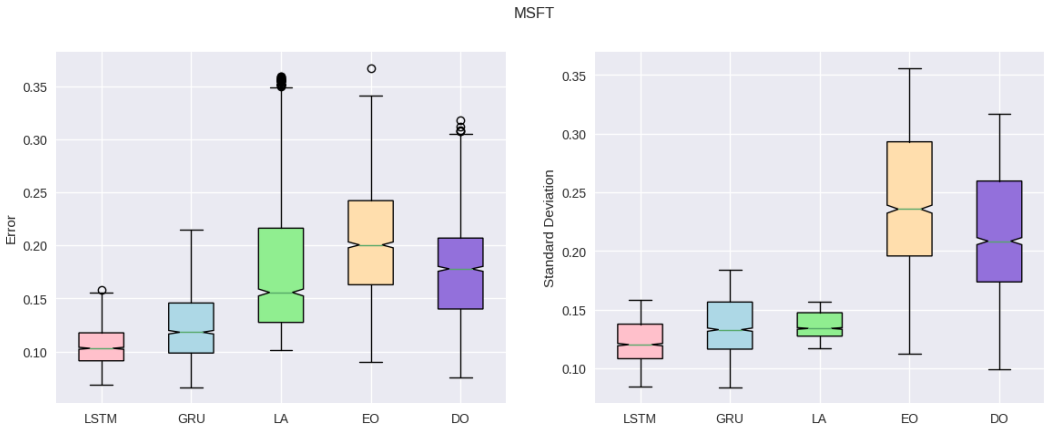


Figure 4: Distribution of pointwise error (absolute difference) and standard deviation of inferences in (a) and (b), respectively in the MSFT validation set. LSTM refers to a long short-term memory network, GRU a gated recurrence unit, LA refers to an LSTM with an Attention layer, EO refers to an encoder-only transformer, and DO refers to an decoder-only transformer.

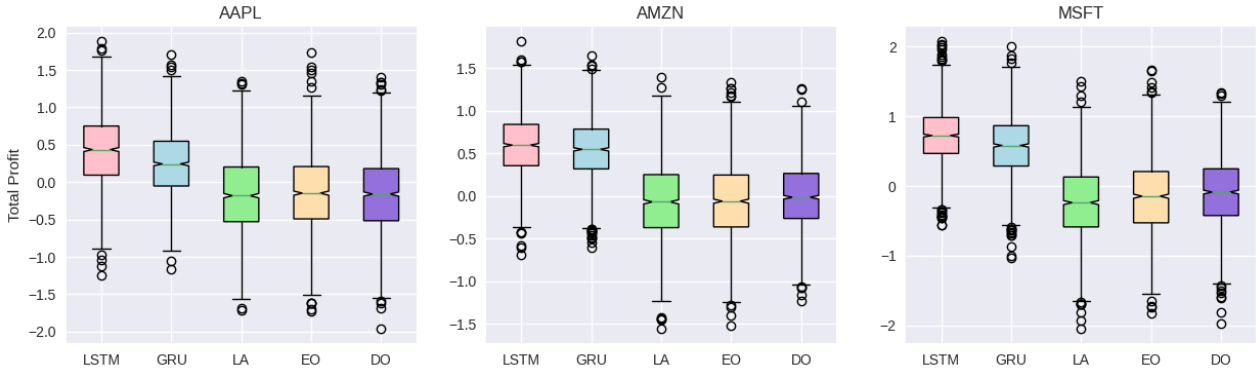


Figure 5: Distribution of profit, $p$, as computed from 1500 runs of Algorithm 2. Long short-term memory networks without attention are denoted by LSTM (respectively LA for LSTMs with an Attention layer), gated recurrence units denoted GRU, encoder-only transormers denoted EO, and decoder-only transformers denoted DO. Results were obtained for AAPL (a), AMZN (b), and MSFT (c) stock. Bayesian models were ran with $n_i = 30$ inferences.

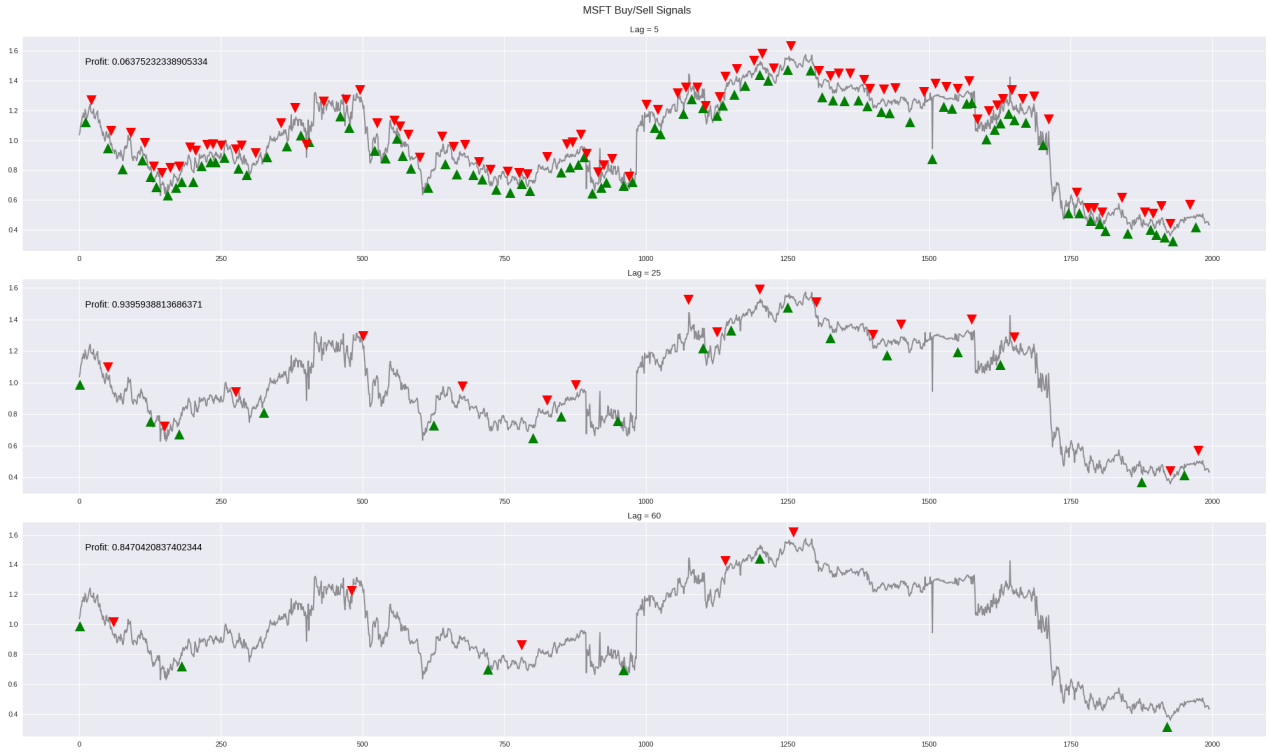


Figure 6: Buy (green) and sell (red) signals, as determined by Algorithm 2 in the validation region of Microsoft (MSFT) stock. Lag times of 5, 25, and 60 are shown in (a), (b), and (c), respectively. Total of $n_i = 30$ Bayesian inferences.

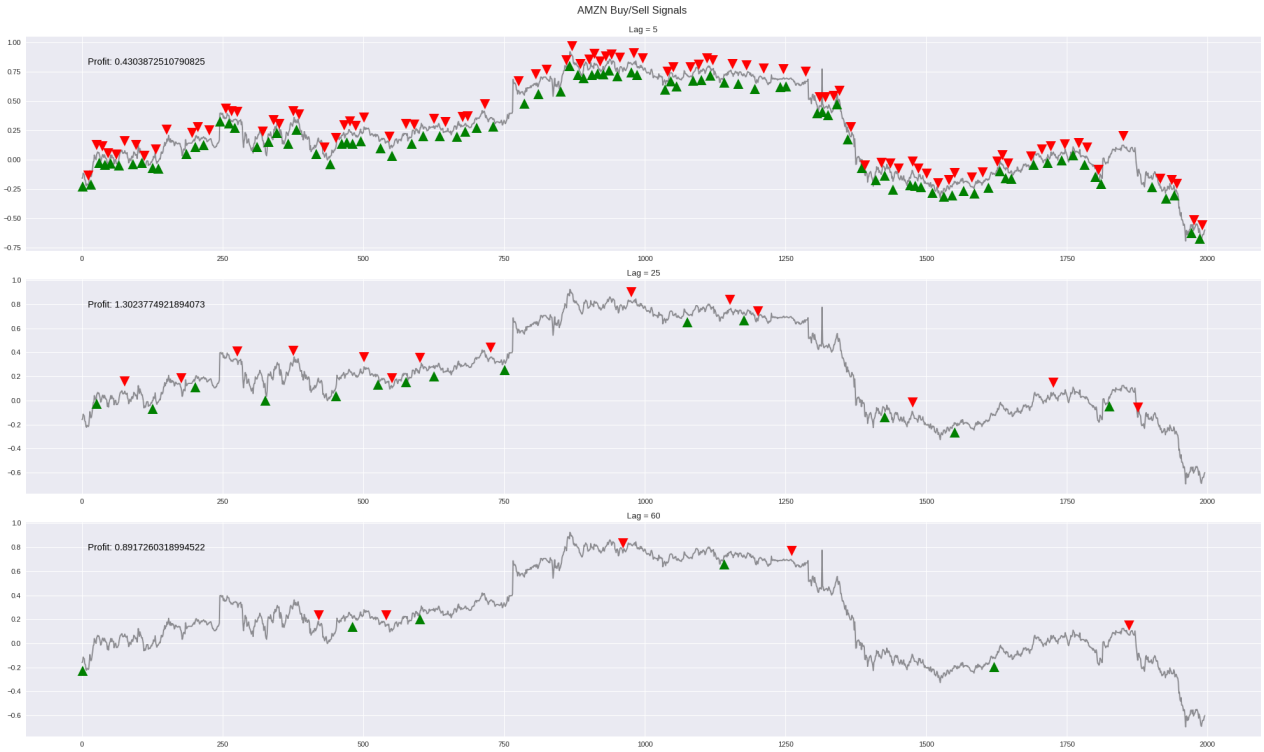


Figure 7: Buy (green) and sell (red) signals, as determined by Algorithm 2 in the validation region of Amazon (AMZN) stock. Lag times of 5, 25, and 60 are shown in (a), (b), and (c), respectively. Total of $n_i = 30$ Bayesian inferences.

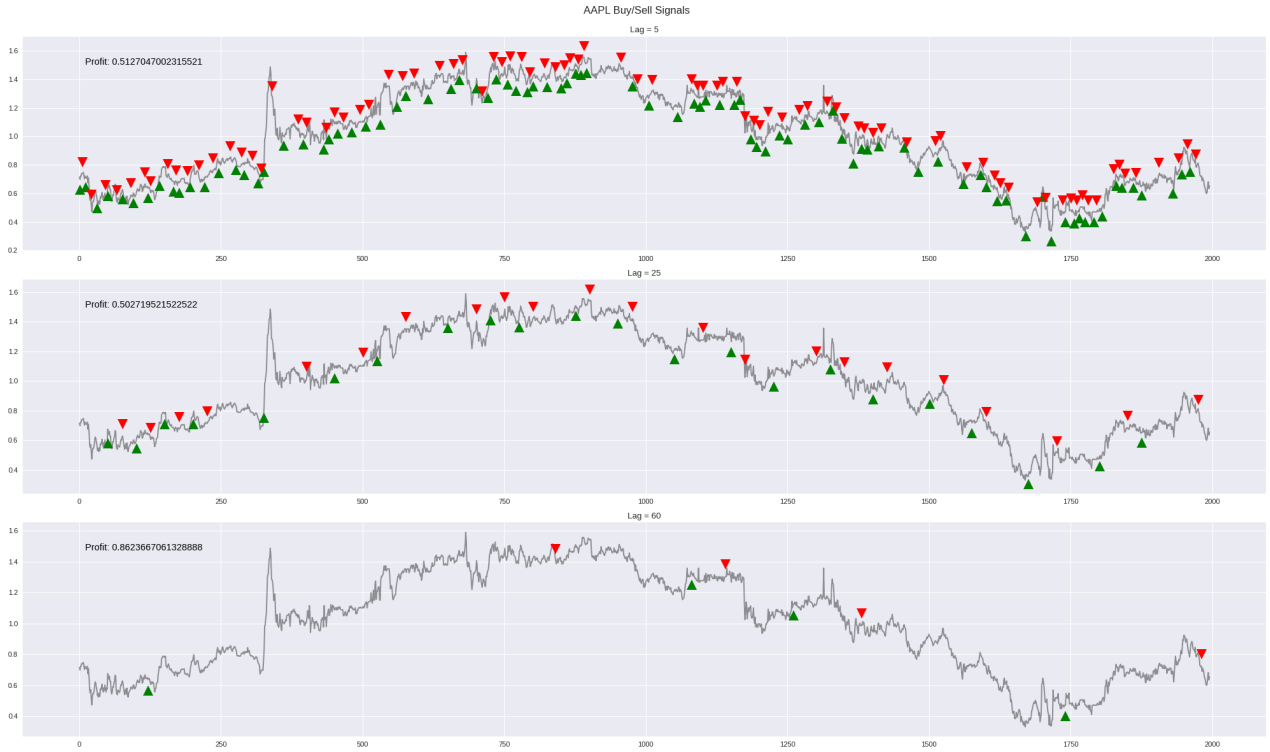


Figure 8: Buy (green) and sell (red) signals, as determined by Algorithm 2 in the validation region of Apple (AAPL) stock. Lag times of 5, 25, and 60 are shown in (a), (b), and (c), respectively. Total of $n_i = 30$ Bayesian inferences.