Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Economics and Management
Dept. of Industrial Economics and Technology Management

Elias Søvik Gunnarsson
Håkon Ramon Isern

# Volatility Prediction and Uncertainty Quantification using Bayesian Neural Networks

TIØ4900 - Financial Engineering, Master's Thesis

Master's thesis in Industrial Economics and Technology Management
Supervisor: Sjur Westgaard
Co-supervisor: Morten Risstad
June 2023

NTNU
Norwegian University of Science and Technology

Elias Søvik Gunnarsson
Håkon Ramon Isern

# Volatility Prediction and Uncertainty Quantification using Bayesian Neural Networks

TIØ4900 - Financial Engineering, Master's Thesis

Master's thesis in Industrial Economics and Technology Management
Supervisor: Sjur Westgaard
Co-supervisor: Morten Risstad
June 2023

Norwegian University of Science and Technology
Faculty of Economics and Management
Dept. of Industrial Economics and Technology Management

NTNU
Norwegian University of
Science and Technology

# Preface

The following thesis concludes our Master of Science degree in Industrial Engineering and Technology at the Norwegian University of Science and Technology (NTNU). More specifically within the main profile area of Financial Engineering. The thesis is a result of independent work conducted by Elias Søvik Gunnarsson and Håkon Ramon Isern during the spring of 2023.

First and foremost, we would like to express our appreciation to our main supervisor, Professor Sjur Westgaard, for his positive attitude, constructive feedback, and relevant academic discussions within the field of financial risk management and Bayesian models. Furthermore, we would like to extend our thanks to Adjunct Associate Professor Morten Risstad for valuable feedback and insights during the period. We would also like to express gratitude towards Ph.D. Candidate Benjamin Vigdel for his assistance in the initial stages of the process where he has provided us with highly relevant literature which further shaped the direction of our work.

Trondheim, June 2023

# Abstract

This thesis explores the Bayesian regime for uncertainty quantification in deep learning and applies it to forecasting the CBOE VIX index. The study compares deterministic and Bayesian deep learning models, including hybrid models that combine Bayesian and LSTM approaches. Various techniques for implementing Bayesian inference, such as Bayes-By-Backprop and Monte Carlo dropout, are employed to assess their impact on forecasting performance. The effectiveness of different prior distributions for the hybrid model and the inclusion of exogenous data are also investigated.

The results reveal that the Autoregressive model remains challenging to beat for VIX forecasting, consistently outperforming other models. The hybrid Bayes-By-Backprop model with a normal prior and posterior distribution emerges as the second-best performer, demonstrating the regularization benefits of Bayesian modeling. Pure Bayesian models struggle to capture the complexities of the VIX, and Bayesian models in general are sensitive to turbulent and noisy data, hindering their performance in dynamic forecasting regimes.

The Bayesian models exhibit calibration issues and are sensitive to the choice of hyper-parameters, priors, and posteriors, making them challenging to implement reliably. The observed unimodal posterior in variational inference suggests the need for alternative methods to capture epistemic uncertainty more accurately.

Future work could explore the use of empirical trainable prior distributions, estimate aleatoric uncertainty, and employ alternative Bayesian inference methods like Markov Chain Monte Carlo. Additionally, incorporating a richer set of exogenous data, expanding forecasting horizons and methods, integrating explainable AI (XAI), and conducting practical evaluations are recommended to address the identified shortcomings and improve the integration of Bayesian deep learning into a framework for volatility forecasting.

# Sammendrag

I denne avhandlingen utforsker vi Bayesianske metoder for usikkerhetskvantifisering gjennom dyp læring. Vi anvender metodene for å predikere CBOE VIX-indeksen, og vi sammenligner disse modellene med tradisjonelle deterministiske metoder innenfor dyp læring, samt hybride varianter som inkorporerer begge metodene. Ulike teknikker for Bayesiansk inferens implementeres, deriblant Bayes-By-Backprop og Monte Carlo Dropout, og vi tester hvordan dette påvirker modellenes resultater.

Resultatene viser at en autoregressiv modell er vanskelig å slå når det kommer til prediksjon av VIX-indeksen. De hybride Bayes-By-Backprop-modellene med en normal à priori og à posteriori sannsynlighetsfordeling gir generelt de nest beste resultatene, som demonsterer regulariseringseffekten man oppnår med Bayesiansk modellering. Rene Bayesianske modeller sliter med å fange opp kompleksitetene i VIX-indeksen, og de Bayesianske modellene er sensitive til turbulent data med støy som påvirker deres predikeringsevne for dynamiske prediksjoner.

De Bayesianske modellene demonstrerer kalibreringsproblemer og er sensitive til valget av både hyperparametre, à priori samt à posteriori sannsynslighetsfordelinger. Dette gjør dem utfordrende å implementere som pålitelige prediksjonsmodeller. Den observerte unimodale à posteriori sannsynslighetsfordelingen man oppnår gjennom variasjonell inferens kan antyde et behov for alternative metoder dersom man ønsker å modellere epistemisk usikkerhet på et mer nøyaktig nivå.

Fremtidig arbeid kan utforske bruken av empiriske trenbare à priori sannsynlighetsfordelinger, inkludere estimering av aleatorisk usikkerhet og anvende alternative metoder for Bayesiansk inferens som for eksempel Markov Chain Monte Carlo. Videre kan man granske bruk av et rikere datasett samt anvendelsen av flere prediksjonshorisonter og -metoder. Integrering av forklarbar kunstig intelligens samt praktiske økonomiske evalueringer er et annet element som kan utprøves for å forbedre integrasjonen av Bayesiansk dyp læring inn i et rammeverk for volatilitetsprediksjon.

# Contents

# List of Figures

# List of Tables

# 1 Introduction

The rising use of artificial intelligence and machine learning techniques has fueled interest in employing them for volatility prediction and forecasting. Volatility is highly important for finance and economics stakeholders. Accurately forecasting volatility enables them to anticipate market shifts and make informed adjustments accordingly (Gunnarsson, Isern, Risstad, Vigdel, & Westgaard, 2023). Models developed using machine learning and deep learning are widely utilized for various types of inference and decision-making processes. Consequently, it is increasingly crucial to assess the reliability and effectiveness of artificial intelligence (AI) systems before their practical application. This is particularly relevant in the context of volatility prediction and forecasting. Although AI models offer valuable insights, it is essential to acknowledge that their predictions are susceptible to noise and model inference errors. By recognizing and accounting for these factors, using uncertainty quantification, stakeholders can ensure the accuracy and robustness of volatility forecasts, thereby enhancing their decision-making processes (Abdar et al., 2021).

There exists many approaches which aims to mitigate the issues with the aforementioned problems regarding model noise and model inference errors. Jospin, Laga, Boussaid, Buntine, and Bennamoun (2022) point out that the Bayesian regime provides several techniques to analyze and train uncertainty of deep learning models, which then results in stochastic deep learning models called Bayesian neural network. To our knowledge, there have been conducted very limited research about the use of Bayesian neural networks to quantify uncertainty in volatility prediction despite its many desired properties compared to conventional deep learning methods. Conventional neural networks provide no measures of uncertainty in their predictions. Additionally, Bayesian neural networks exhibits increased uncertainty when presented with data which is very different from training data, which is especially important for financial time series data, where it can be beneficial to be able to express uncertainties during market irregularities. Lastly, Bayesian neural networks are less prone to overfitting (Jospin et al., 2022).

Volatility forecasting, as we understand it today, is built upon the incorporation of stylized facts about asset returns into parameterized models. The autoregressive conditional heteroscedasticity (ARCH) model, originally introduced by Engle (1982) and further developed by Bollerslev (1986) into the Generalized ARCH (GARCH), plays a pivotal role. The GARCH model considers the conditional variance as dependent on the disturbance term and the conditional variances from the previous day. Over time, these models have been expanded to include asymmetric behavior and leverage effects, see Nelson (1991); Glosten, Jagannathan, and Runkle (1993). Engle, Ghysels, and Sohn (2013) introduced a family of univariate GARCH models using Mixed Data Sampling Frequency (MIDAS) filters, which allow for the inclusion of additional variables to capture both short- and long-term effects.

As volatility is inherently unobservable, a type of proxy or estimator must be used in order to assess the model's performance. In the previous mentioned literature, squared log-returns are commonly used as such estimator for assessment purposes. Furthermore, the availability of high-frequency data has revolutionized volatility estimation, particularly with the use of realized volatility (RV). Realized volatility is calculated using the sum of intraday squared returns for the given day, and is often considered a closer estimation of the true volatility. Unlike the conditional variance, realized volatility does not rely on a predetermined model-based relationship between time periods. This

feature allows for direct modeling using autoregressive models or more importantly for our cause, machine learning models.

In addition to realized volatility, other volatility measures, such as implied volatility, have prompted alternative methodologies for volatility forecasting, including the utilization of machine learning and artificial intelligence. For example, Malliaris and Salchenberger (1996) introduced a neural network-based approach to forecast implied volatility, adopting a similar methodology to the renowned CBOE Volatility Index (VIX). The VIX, which is calculated based on real-time prices of options on the S&P 500 index, serves as a reliable indicator of the expected volatility of the S&P 500 index over a 30-day period.

The availability of derivatives for the VIX can make it an interesting approach for economic evaluation compared to using realized volatility (RV). Especially, since the assessment of different volatility models is a challenging task, and would often require an additional practical evaluation instead of pure reliant of statistical error metrics to get a more robust overview. This further motivates the use of the VIX as a dependent variable for supervised machine learning techniques.

Therefore, the main objective of this thesis aims to predict the CBOE VIX, as well as quantify the uncertainty in the predictions using Bayesian neural networks. In light of this objective, the following research questions will guide our investigation. Firstly, we will explore the predictive abilities of Bayesian neural networks in comparison to conventional neural networks for the VIX. Specifically, we seek to determine whether Bayesian neural networks can achieve similar performance in predicting the VIX as their conventional counterparts. Secondly, we will examine the feasibility of different methods for approximating Bayesian inference in practical applications of neural networks for implied volatility prediction. Additionally, we aim to assess the usefulness of quantifying uncertainty in these applications and explore how it can enhance decision-making processes. Lastly, we will investigate how already tuned conventional recurrent neural networks can be extended to Bayesian neural networks to effectively quantify uncertainty. By adapting these networks to incorporate Bayesian principles, we aim to enhance the understanding and representation of uncertainty in the context of implied volatility prediction.

In this thesis we contribute pioneering work in the use of Bayesian inference to quantify uncertainty in implied volatility forecasting. We have implemented two different methods of variational inference for Bayesian inference in neural networks, which include Bayes-By-Backprop and Monte Carlo dropout. These were then combined with the conventional Long-Short-Term-Memory architecture in order for it to be extended to a hybrid Bayesian Neural Network, where we also examined the use of three different prior distributions. Furthermore, we investigated the effect of adding exogenous data. We find that the Autoregressive model remains hard to beat for forecasting the VIX, however, a hybrid Bayes-By-Backprop model with a normal distributed prior and posterior generally ranks the next best, suggesting regularization effects over conventional neural networks. By inspecting the reliability diagrams of the probabilistic models' quantified uncertainty, we found them to be uncalibrated. Furthermore, they are highly sensitive to hyperparameter choices, as well as prior and posterior distribution for their weights, resulting in often unstable predictions, making it difficult for utilization in end-to-end frameworks. The observation of mode collapses using the variational inference suspects difficulties in fully capturing the epistemic uncertainty.

We propose some areas for further research which address some of the observed shortcomings and caps within our implementations. These include looking into empirically trainable priors, as opposed to fixed choices. Additionally, our models are limited to capturing epistemic uncertainty, where looking into aleatoric uncertainty might give a broader overview of the entire predictive uncertainty. Furthermore, using other methods for Bayesian inference, such as Markov Chain Monte Carlo in a high performance computing environment to forecast volatility may give more accurate posterior distributions. Lastly, we would like to emphasise the need for practical economic evaluation as well as the use of explainable artificial intelligence (XAI) methods to be incorporated to the Bayesian deep learning volatility forecasting framework.

The rest of this thesis is organized as follows. Section 2 provides an insight to existing literature in the relevant fields of volatility prediction and uncertainty quantification in deep learning, section 3 provides an overview of the theoretical foundation for the applied methods. Section 4 explains the data and its procedures, while section 5 details how we implemented and evaluated our selection of models. In crefresults we report our results and provides a discussion in the context of our findings and research questions. Lastly, section 7 concludes our work.

# 2 Literature Review

This literature review aims to give insights in the two relevant topics of this thesis, namely volatility prediction using machine learning and uncertainty quantification in deep learning models. The first one focuses mainly on previous literature that have used an implied volatility index as an estimator for volatility, as well as selected literature concerning realized volatility prediction due to their methodological relevance. To our knowledge, there have been conducted very limited research about the use of Bayesian neural networks to quantify uncertainty in volatility prediction. Thus, we will focus on the recent advances in uncertainty quantification methods in deep learning methods, and in the context of time series in particular.

## 2.1 Volatility Prediction Using Machine Learning

Christensen, Siggaard, and Veliyev (2022) studied a broad range of machine learning models to predict realized volatility of several DIJA constituents. Although not predicting implied volatility indices, but rather realized volatility, their work can be considered a starting point for applying machine learning models to predict volatility considering the broad range of models used. They also emphasise on further possibilities for additional improvements in their methods by doing more extensive hyperparameter tuning. The machine learning models capable of capturing non-linearities were found to be superior, but the difference in performance between machine learning models and econometric auto-regressive benchmarks were not substantial.

When predicting the VIX, or other implied volatility indices, one common approach is to predict the direction. Vrontos, Galakis, and Vrontos (2021) conducted a comparison of various non-linear machine learning models and logistic regression models to predict the direction of the VIX index one-month-ahead. Through the analysis of multiple statistical evaluation measures, they concluded that the machine learning approach yielded superior results for out-of-sample forecasts, achieving accuracies around 60%.

In their research, Oliveira, Cortez, and Areal (2017) introduced the concept of using Twitter data with sentiment values to analyze its impact on forecasting the VIX and the annualized RV of the S&P 500, RSL, DJIA, and NDQ. They compared multiple machine learning (ML) models with each other, as well as an AR(5) benchmark model. Through a DM-test to evaluate the forecasting results, they discovered that incorporating sentiment and attention indicators did not lead to a statistically significant improvement in forecasting performance. Overall, they found it difficult to outperform the AR(5) benchmark. Bucci (2020) also found the econometric auto-regressive benchmarks to be performing promising when trying to predict monthly realized volatility of the S&P 500. However, with a Long-Short-Term-Memory model and a NARX-model they were able to achieve slightly better results. The methods included both an endogenous, as well as an exogenous approach including macroeconomical and financial variables.

Ghosh and Sanyal (2021) examined the predictability of market fear in India during the Covid pandemic through the India VIX. They used the Boruta Algorithm to select the variables among the exogenous data for their XGBoost, Random Forest, and Long-Short-Term-Memory models. Another more pure practical approach by Osterrieder, Kucharczyk, Rudolf, and Wittwer (2020) was to use LSTM-model to replicate the CBOE VIX and look for arbitrage opportunities. By training the Long-Short-Term-Memory model on S&P 500 options, they propose methods for utilizing the VIX derivatives, mode precisely the intradaily deviations between VIX and its futures, to find arbitrage opportunities.

## 2.2 Bayesian Neural Networks for Time Series

As previously mentioned, there exist limited literature that focuses on uncertainty quantification using Bayesian Neural Networks for volatility prediction, and even financial time series data. However, Abdar et al. (2021) found that the use of uncertainty quantification in recent literature about AI and machine learning has significantly increased over the last ten years. They provide to their knowledge the first comprehensive review paper regarding the use of uncertainty quantification methods in traditional machine learning and deep learning, as well as the main categories of important applications of uncertainty quantification. The latter confirms our observation regarding applications within financial time series, as the relevant applications highlighted in the review regarding time series consisted of medical research applications, weather forecasting, and airport flight delays. The last-mentioned being a Variational Long Short-Term Memory model proposed by Vandal, Livingston, Piho, and Zimmerman (2018), where their objective was prediction and uncertainty quantification of daily airport flight delays, where they also confirms the predictive uncertainty is well explained through calibration analysis of the predictive results.

By quantifying predictive uncertainty in financial asset forecasting, Back and Keith (2019) attempt to improve a systematic trading strategy by scaling positions with uncertainty that is quantified using Bayesian neural networks. To approximate the Bayesian inference, they compare Monte Carlo dropout methods with variational, and Markov Chain Monte Carlo methods. While admitting that recurrent neural networks might be more appropriate for such temporal data, they opted for multi layer perceptrons as the backbone for their Bayesian Neural Networks, possibly due to simplicity during implementations. The results suggest that dropout and variational inference provide good regularization, but their predictive uncertainties could not improve a systematic trading

strategy. The MCMC model provided better results, however, suffered from a high computational complexity. Overall the effectiveness of the uncertainty quantification's of the proposed models would require some more research to be conclusive.

Chandra and He (2021) provided another attempt to forecast stock prices, where they use novel Bayesian neural networks for multi-step-ahead stock price forecasting before and during COVID-19 using Langevin gradients with parallel tempering MCMC. They found more challenging to forecast during the high-volatility periods that followed from the pandemic. They suggests that their models could be improved by using other deep learning models such as the Long-Short-Term Memory Model, as well as a multivariate environment.

# 3 Theoretical Foundation

In this section we provide the theoretical foundations for our thesis. We give a brief summary of the workings behind each of our implemented models. Furthermore, we introduce the framework we used for optimizing the hyperparameters and performing feature selection.

## 3.1 Artificial Neural Networks

In deep learning models, a multilayer perceptron or a feed forward neural network are quintessential. They define a mapping $y = f(x;\theta)$ and learn the parameters $\theta$ that results in the best function approximation (Goodfellow, Bengio, & Courville, 2016). Let a MLP with $L$ layers be denoted as $l = 1, 2, \ldots, L$, and $N_l$ be the number of neurons in layer $l$. The output of a neuron $j$ in layer $l$ can be computed,

$$z_j^{(l)} = \sum_{i=1}^{N_{l-1}} w_{ji}^{(l)} a_i^{(l-1)} + b_j^{(l)}, \tag{1}$$

where $w_{ji}^{(l)}$ is the weight connecting neuron $i$ in layer $l-1$ to neuron $j$ in layer $l$, $a_i^{(l-1)}$ is the output of neuron $i$ in layer $l-1$, and $b_j^{(l)}$ is the bias term for neuron $j$ in layer $l$. The function $z_j^{(l)}$ represents the weighted sum of inputs to neuron $j$ in layer $l$. Furthermore, the output of a neuron is obtain through an activation function to the weighted sum.

## 3.2 Recurrent Neural Networks

Recurrent neural networks (RNNs) are neural networks that are specifically made for sequential or temporal data which includes loops that allows for information to persist in the network (Rumelhart, Hinton, & Williams, 1986). Such networks contains a memory that remembers previous inputs and outputs, such that at each time step $t$ in the temporal data, the RNN takes an input vector $\mathbf{x}^{(t)}$ and produces an output vector $\mathbf{y}^{(t)}$. The hidden state $\mathbf{h}^{(t)}$ represents the memory of the network at time step $t$ and can be represented as,

$$\mathbf{h}^{(t)} = \sigma(\mathbf{W}^{(hh)}\mathbf{h}^{(t-1)} + \mathbf{W}^{(xh)}\mathbf{x}^{(t)} + \mathbf{b}^{(h)}), \tag{2}$$

where $\sigma(\cdot)$ is the activation function, $\mathbf{W}^{(hh)}$ is the weight matrix for the recurrent connections, $\mathbf{W}^{(xh)}$ is the weight matrix for the connections from the input to the hidden layer, and $\mathbf{b}^{(h)}$ is the

bias vector for the hidden layer. The output vector $\mathbf{y}^{(t)}$ at time step $t$ is computed based on the hidden state:

$$\mathbf{y}^{(t)} = \sigma(\mathbf{W}^{(hy)}\mathbf{h}^{(t)} + \mathbf{b}^{(y)}) \tag{3}$$

where $\mathbf{W}^{(hy)}$ is the weight matrix for the connections from the hidden layer to the output layer, and $\mathbf{b}^{(y)}$ is the bias vector for the output layer.

The Long Short-Term Memory (LSTM) model is an extension of the basic recurrent neural network (RNN) that addresses problems during back-propagation for RNNs where the error signals flowing backwards in time either blows up or vanishes (Hochreiter & Schmidhuber, 1997). In a LSTM, the hidden state at each time step consists of the cell state $\mathbf{c}^{(t)}$ and the hidden state $\mathbf{h}^{(t)}$ (Goodfellow et al., 2016). The cell state takes care of the long-term memory of the network, while the hidden state takes care of short-term memory. The other components in the LSTM is the input gate $\mathbf{i}^{(t)}$, the forget gate $\mathbf{f}^{(t)}$, and the output gate $\mathbf{o}^{(t)}$. These control the flow of information from the input that should be added, how much information from previous states that should be forgotten, and how much of the cell state should be converted to the hidden state. Finally we have the candidate cell state, $\mathbf{g}^{(t)}$, which contains the new information that will be added to the cell state. The cell state is updated based on the previous cell state $\mathbf{c}^{(t-1)}$, the input $\mathbf{x}^{(t)}$, and the hidden state $\mathbf{h}^{(t-1)}$,

$$\mathbf{i}^{(t)} = \sigma(\mathbf{W}^{(xi)}\mathbf{x}^{(t)} + \mathbf{W}^{(hi)}\mathbf{h}^{(t-1)} + \mathbf{b}^{(i)}), \tag{4}$$

$$\mathbf{f}^{(t)} = \sigma(\mathbf{W}^{(xf)}\mathbf{x}^{(t)} + \mathbf{W}^{(hf)}\mathbf{h}^{(t-1)} + \mathbf{b}^{(f)}), \tag{5}$$

$$\mathbf{o}^{(t)} = \sigma(\mathbf{W}^{(xo)}\mathbf{x}^{(t)} + \mathbf{W}^{(ho)}\mathbf{h}^{(t-1)} + \mathbf{b}^{(o)}), \tag{6}$$

$$\mathbf{g}^{(t)} = \tanh(\mathbf{W}^{(xg)}\mathbf{x}^{(t)} + \mathbf{W}^{(hg)}\mathbf{h}^{(t-1)} + \mathbf{b}^{(g)}), \tag{7}$$

$$\mathbf{c}^{(t)} = \mathbf{f}^{(t)} \odot \mathbf{c}^{(t-1)} + \mathbf{i}^{(t)} \odot \mathbf{g}^{(t)}, \tag{8}$$

where $\sigma(\cdot)$ represents the sigmoid activation function, $\tanh(\cdot)$ denotes the hyperbolic tangent activation function, and $\mathbf{W}$ and $\mathbf{b}$ represent the weight matrices and bias vectors. The hidden state $\mathbf{h}^{(t)}$ is then computed based on the updated cell state,

$$\mathbf{h}^{(t)} = \mathbf{o}^{(t)} \odot \tanh(\mathbf{c}^{(t)}). \tag{9}$$

## 3.3 Uncertainty Quantification

A key concept within any modelling exercise relates to the inherent uncertainty surrounding the process. There are uncertainties related to the data itself, uncertainties regarding your data sample, and uncertainties with the model's ability to capture these effects. Quantifying these uncertainties may aid in making more sound inferences based on a model result. To better understand this quantification, it is necessary to distinguish between the two main categories of uncertainty: aleatoric and epistemic, see Figure 1.

Epistemic uncertainty represents the average variance of the predictive distribution across different inputs $x_i$ Valdenegro-Toro and Mori (2022). It quantifies the model's uncertainty due to

limited data which can be expressed,

$$\mathbb{E}_x[\text{Var}[P(y|x)|x]] = \frac{1}{N}\sum_{i=1}^{N}\text{Var}[P(y|x_i)]. \tag{10}$$

Aleatoric uncertainty represents the average of the expected value of the predictive distribution across different inputs $x_i$. It captures the inherent stochasticity or noise in the data generation process and can be expressed,

$$\text{Var}_x[\mathbb{E}[P(y|x)|x]] = \text{Var}\left[\frac{1}{N}\sum_{i=1}^{N}P(y|x_i)\right]. \tag{11}$$

The predictive variance can be expressed using the law of total variance:

$$\text{Var}[P(y|x)] = \mathbb{E}_x[\text{Var}[P(y|x)|x]] + \text{Var}_x[\mathbb{E}[P(y|x)|x]]. \tag{12}$$

Hence the decomposition of predictive variance becomes,

$$\text{Var}[P(y|x)] = \frac{1}{N}\sum_{i=1}^{N}\text{Var}[P(y|x_i)] + \text{Var}\left[\frac{1}{N}\sum_{i=1}^{N}P(y|x_i)\right]. \tag{13}$$

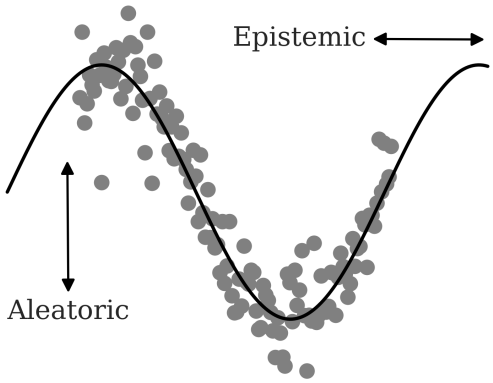



Figure 1: Aleatoric and epistemic uncertainty visualized. Aleatoric uncertainty is the inherent uncertainty in the data and is irreducible. Epistemic uncertainty is the uncertainty around the model in itself and the existing data, and can be reduced by giving the model more data. Based on the illustration from Abdar et al. (2021).

## 3.4 Bayesian Neural Networks

An approach to measure epistemic uncertainty involves the Bayesian paradigm, where probabilities represents beliefs about events. Given data $D$ consisting of inputs $x$ and labels $y$, we want to optimize some parameters $w$ of a function mapping the inputs to the labels. Using a Bayesian line of thinking, this function can be represented as the model likelihood $P(y|x, w)$. Next, using Bayes' theorem we can express the epistemic uncertainty as the posterior distribution of the parameters

given the data.

$$P(w|D) = \frac{P(D|w)P(w)}{P(D)} \tag{14}$$

In this specification, $P(D|w)$ is the likelihood of the data given the parameters. $P(w)$ represents the prior beliefs about the parameters' distribution. $P(w|D)$ is the posterior belief given the data, and represents the epistemic uncertainty about the data. Finally, $P(D)$ is the evidence or the marginal likelihood. To evaluate the evidence, we can utilize the fact that it is the marginalized joint distribution of the data and the weights. The term can then be expressed as the integral $\int_w P(D, w)dw$. To translate this into an output for the model, we can use marginalization to estimate the predictive distribution as

$$P(y|x, D) = \int_w P(y|x, w)P(w|D)dw \tag{15}$$

Equation 14 provides the foundation for uncertainty quantification using Bayesian inference. To incorporate this into the neural network paradigm, Bayesian Neural Networks (BNNs) were introduced in MacKay (1992); Neal (1995). A BNN is based on the foundations of a regular Neural Network with one crucial difference. The point estimates of the network parameters $w$ are replaced with probability distributions $P(w)$, as demonstrated in Figure 2. Hence, we can use Equation 14 to capture the epistemic uncertainty. This brings several benefits over a regular neural network, as highlighted by Jospin et al. (2022). First and foremost, it allows us to express the networks uncertainities around its predictions through the posterior distribution. Second, neural networks are susceptible to overfitting on training data. In BNNs, the prior can act as a regularization constraint, thereby helping addressing this issue. Furthermore, they are more data efficient, as the quantified uncertainty allows the model to express its the precariousness in its predictions.

While the mentioned features makes for a lucrative description of BNNs, there are some issues that needs addressing; how do we actually compute the posterior prediction, and how to we translate this into making predictions? Due to the large and complex nature of the network parameter space, an analytical evaluation of the marginal likelihood $P(D)$ is often intractable. An additional consequence of this comes to light when estimating predictive uncertainty, as we need to integrate over each possible posterior distribution - and therefore, the weight space - to evaluate it. To overcome these problems, we can resort to approximations. Making predictions can be resolved by resorting to Monte Carlo sampling of the posterior. By independently sampling parameters $w_i \sim P(w|D)$ from the posterior, we can approximate an output of the model. Tackling the first problem - that is, evaluating the posterior in the first place - requires approximating the posterior distribution using well-known Bayesian inference methods. In our thesis, we will mainly focus on two variants: Variational Inference (VI) and Monte Carlo Dropout (MC Dropout).

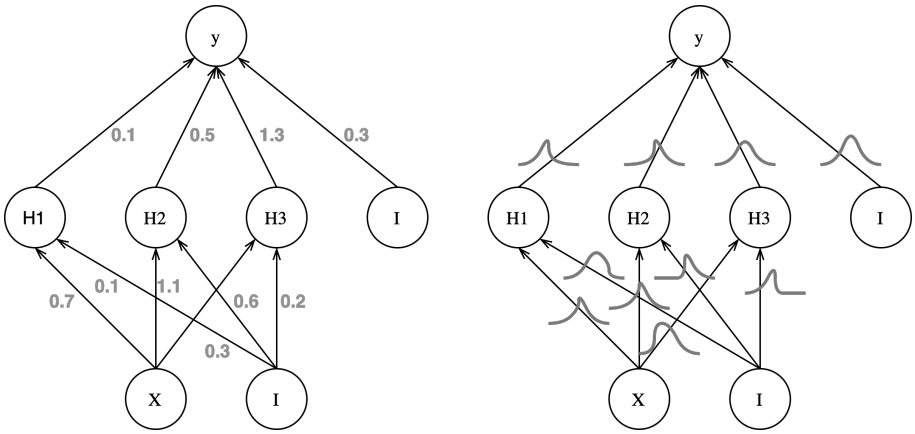



Figure 2: The left figure displays a regular neural network with point estimates over the weights, while the rightmost figure shows weights with probability distributions. Based on the illustration in Blundell et al. (2015)

## 3.5 Variational Inference

Variational inference is an approximate Bayesian inference method for finding a posterior distribution Goodfellow et al. (2016). It employs a variational surrogate posterior distribution parameterized by $\theta$, $q_\theta(w)$, in place of the true posterior $P(w|D)$. Then the task is restated into an optimization problem. Here, the goal is minimizing the dissimilarity between these two distributions. Under certain circumstances, it can derive precise approximations of the true posterior.

The dissimilarity is often measured by the Kullback-Leibler divergence. It is defined as the expectation of the logarithmic difference between two distributions $q$ and $P$,

$$D_{KL}(q||P) = \mathbb{E}_{x\sim q}[\log q(x) - \log P(x)] = \int_x q(x) \log \frac{q(x)}{P(x)}, \tag{16}$$

where $\mathbb{E}_{x\sim q}[\log q(x) - \log P(x)]$ denotes the expectation of $\log q(x) - \log P(x)$ with respect to $q(x)$. In our scenario we have a latent distribution over the function parameters $P(w|D)$ and a surrogate distribution $q_\theta(w)$. This allows us to redefine the divergence as

$$\begin{aligned} D_{KL}(q_\theta||P) &= \int_w q_\theta(w) \log \left( \frac{q_\theta(w)}{P(w|D)} \right) \\ &= \int_w q_\theta(w) \left[\log q_\theta(w) - \log P(w|D)\right] \\ &= \int_w q_\theta(w) \left[\log q_\theta(w) - \log P(D|w) - \log P(w) + \log P(D)\right] \\ &= \log P(D) - \int_w q_\theta(w) \left[\log P(D|w) + \log P(w) - \log q_\theta(w)\right] \\ &= \log P(D) + D_{KL}(q_\theta||P) - \mathbb{E}_{w\sim q_\theta}[\log P(D|w)] \end{aligned} \tag{17}$$

The final term in Equation 17 consists of three parts: the log-likelihood of the evidence, the log-likelihood of the data given the parameters $w$ and the KL-divergence between the surrogate and prior distribution. By definition, the KL divergence is always non-negative. Thus, we can provide a lower bound for the divergence by disregarding the log-likelihood of the evidence, thereby yielding an optimization objective. To minimize the KL divergence between the posterior and the surrogate we want to maximize the log-likelihood and KL divergence. This is called the Evidence Lower

Bound, or ELBO for short.

$$ELBO = \mathbb{E}_{w\sim q_\theta}[\log P(D|w)] - D_{KL}(q_\theta||P) \tag{18}$$

The ELBO offers a basis for adapting variational inference for deep learning. We need simply maximize the expected log-likelihood of the data given the model, and minimize the KL divergence between the surrogate and prior distribution. Together this provides a loss function for gradient descent in a neural network. The Bayes-By-Backprop algorithm of Blundell et al. (2015) details an implementation of this procedure, demonstrated in algorithm 1. First, the prior and surrogate posterior distribution types must be set. Then, let $\theta$ parameterize the surrogate posterior and sample weights from this distribution when making predictions. $\theta$ consists of two parts: the mean $\mu$ and the standard deviation $\sigma$. To ensure a non-negative $\sigma$, it is decomposed into parameter $\rho$ which is ran through the softplus function, $\log(1 + e^\rho)$.

For each step of gradient descent, we sample weights and make predictions. Using the additive inverse of ELBO as our loss function, we can back-propagate the recorded loss to update the parameter $\theta$, thereby learning the true posterior distribution. However, for many choices of prior and posterior distributions the KL divergence can be highly computationally expensive or even intractable. Furthermore, most gradient descent algorithms uses mini-batches to compute the gradients. As a consequence of this, the expectation of the likelihood $P(D|w)$ is computed over the mini-batches rather than the entire training data-set. To address these two issues, two corrections are made. First, to accommodate the choice of a greater set of distributions the divergence is approximated by drawing $I$ samples of weights $w_i$.

$$\sum_i^I \log q_\theta(w_i) - \log P(w_i) - \log P(D|w_i) \tag{19}$$

Then, to correct for the difference in cardinality, the KL divergence is weighted by the inverse of the length of the training examples $1/N$.

The resulting procedure is illustrated in Figure 3. The dissimilarity between the parameterized surrogate at step $i$ and the posterior is measured through Equation 18, the losses are backpropagated and a new parameterization is acquired where the surrogate encroaches on the true posterior.

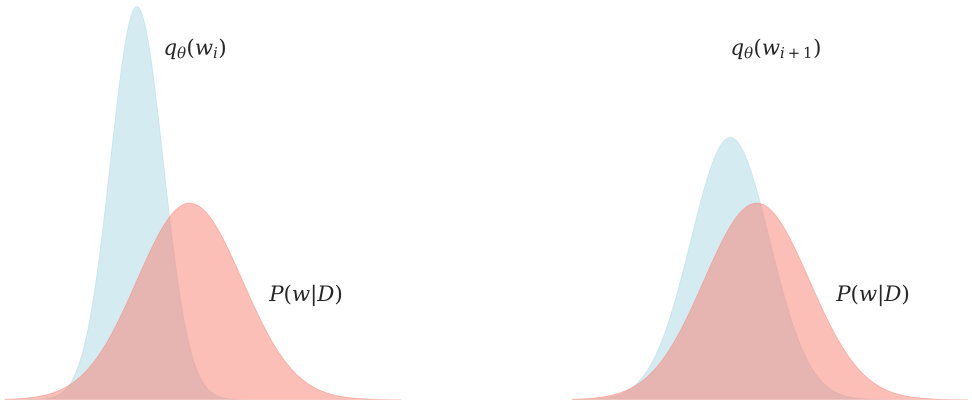



Figure 3: Updating surrogate posterior $q_\theta(w)$ to minimize dissimilarity to true posterior $p(w|D)$.

**Algorithm 1:** Bayes by Backprop

**Input:** Training data: $\mathbf{X}_{\text{train}}, \mathbf{Y}_{\text{train}}$
**Output:** Trained BNN model
Initialize model parameters;
Initialize variational posterior parameters;
**for** *epoch* $\leftarrow 1$ **to** *num_epochs* **do**
  **for** $(\mathbf{x}, \mathbf{y}) \in (\mathbf{X}_{train}, \mathbf{Y}_{train})$ **do**
    Sample a set of weights from the variational posterior;
    Compute the output prediction: $\mathbf{y}_{\text{pred}} = \text{model}(\mathbf{x})$;
    Compute the negative log-likelihood loss: $L_{\text{NLL}} = \text{negative\_log\_likelihood}(\mathbf{y}, \mathbf{y}_{\text{pred}})$;
    Compute the KL divergence loss: $L_{\text{KL}} = \text{kl\_divergence}()$;
    Compute the total loss: $L_{\text{total}} = L_{\text{NLL}} + L_{\text{KL}}$;
    Update the model and variational posterior parameters to minimize $L_{\text{total}}$;
  **end**
**end**
**Input:** Test data: $\mathbf{X}_{\text{test}}$
**Output:** Predictions and uncertainties
Initialize uncertainties: uncertainties = [];
**for** *sample* $\leftarrow 1$ **to** *num_samples* **do**
  Sample a set of weights from the variational posterior;
  **for** $\mathbf{x} \in \mathbf{X}_{test}$ **do**
    Compute the output prediction: $\mathbf{y}_{\text{pred}} = \text{model}(\mathbf{x})$;
    Append $\mathbf{y}_{\text{pred}}$ to uncertainties;
  **end**
**end**
Compute mean predictions: mean_predictions = mean(uncertainties);
Compute variance predictions: variance_predictions = variance(uncertainties);

## 3.6 Monte Carlo Dropout

Monte Carlo Dropout is another approach to variational inference, where the concept of dropout layers are forced upon the network to produce stochastic behavior. Gal and Ghahramani (2016) shows that a neural network with arbitrary depth and non-linearities, with dropout applied before every weight layer, is mathematical equivalent to an approximation to the probabilistic deep Gaussian process. The dropout objective minimises the Kullback-Leibler divergence between an approximate distribution and the posterior of a deep Gaussian process. The dropout layer samples binary variables for every input point and every network unit in each layer. Then, every binary variables takes value 1 with a probability $p$, i.e. the dropout rate, and a unit is dropped for a given input if its corresponding binary variable is 0. These values stays the same through the backpropagation pass. A deep Gaussian process can be approximated by placing a variational distribution over each component of a spectral decomposition of the Gaussian Process' covariance function, which further maps each layer of the deep Gaussian Process to a layer of explicitly represented hidden units.

The posterior distribution from the predictive probability of the deep Gaussian Process model is intractable, so we then use a distribution over matrices whose columns are randomly set to zero, to approximate the posterior, which is also called a variational distribution. Next, the Kullback-Leibler divergence between the approximate posterior and the posterior of the full deep Gaussian Process is minimized. Predictive mean and predictive uncertainty are then obtained by collecting an samples from stochastic forward passes in the neural network with dropout layers. Thus, the

process of obtaining model uncertainty in neural networks using Monte Carlo Dropout can be summarized in algorithm 2.

**Algorithm 2:** Monte Carlo Dropout

**Input:** Training data: $\mathbf{X}_{\text{train}}, \mathbf{Y}_{\text{train}}$
**Output:** Trained Monte Carlo Dropout LSTM model
Initialize model parameters;
**for** *epoch* $\leftarrow 1$ **to** *num_epochs* **do**
  **for** $(\mathbf{x}, \mathbf{y}) \in (\mathbf{X}_{train}, \mathbf{Y}_{train})$ **do**
    Enable Dropout in LSTM layers;
    Forward pass through LSTM layers: **output** = LSTM($\mathbf{x}$);
    Apply Dropout to **output**;
    Compute the output prediction: $\mathbf{y}_{\text{pred}}$ = model(**output**);
    Compute the mean squared error loss: $L_{\text{MSE}}$ = mean_squared_error($\mathbf{y}, \mathbf{y}_{\text{pred}}$);
    Update the model parameters to minimize $L_{\text{MSE}}$;
  **end**
**end**
**Input:** Test data: $\mathbf{X}_{\text{test}}$
**Output:** Predictions and uncertainties
Initialize uncertainties: uncertainties = [];
Enable Dropout in LSTM layers;
**for** *sample* $\leftarrow 1$ **to** *num_samples* **do**
  **for** $\mathbf{x} \in \mathbf{X}_{test}$ **do**
    Forward pass through LSTM layers: **output** = LSTM($\mathbf{x}$);
    Compute the output prediction: $\mathbf{y}_{\text{pred}}$ = model(**output**);
    Append $\mathbf{y}_{\text{pred}}$ to uncertainties;
  **end**
**end**
Compute mean predictions: mean_predictions = mean(uncertainties);
Compute variance predictions: variance_predictions = variance(uncertainties);

## 3.7 Autoregressive Models

An autoregressive (AR) model is a time series model where the current value of a variable is linearly dependent on its past values. The general form of an AR model of order p can be expressed

$$X_t = c + \phi_1 X_{t-1} + \phi_2 X_{t-2} + ... + \phi_p X_{t-p} + \epsilon_t, \tag{20}$$

where $X_t$ represents the value of the time series at time t, c is a constant term, $\phi_1, \phi_2, ..., \phi_p$ are the coefficients of the lagged terms, $X_{t-1}, X_{t-2}, ..., X_{t-p}$ are the lagged values of the time series, and $\epsilon_t$ is the error term or the random shock at time t, which represents the unexplained part of the model. Extending the $AR(p)$ to include exogenous data is the AutoRegressive eXogenous (ARX) model on the form

$$X_t = c + \phi_1 X_{t-1} + \phi_2 X_{t-2} + \ldots + \phi_p X_{t-p} + \beta_1 Z_{1,t-1} + \beta_2 Z_{2,t-1} + \ldots + \beta_m Z_{m,t-1} + \epsilon_t. \tag{21}$$

Here the additional $Z_{1,t-1}, Z_{2,t-1}, \ldots, Z_{m,t-1}$ are exogenous variables at time $t-1$.

## 3.8 Cross Validation and Hyperparameter Tuning

Common k-fold cross validation techniques assume that there is no relationship between each observation in the training data. However, this is not the case for time series. Appropriate validation techniques for temporal data, such as time series, are walk forward validation methods, namely expanding window cross validation and sliding window cross validation.

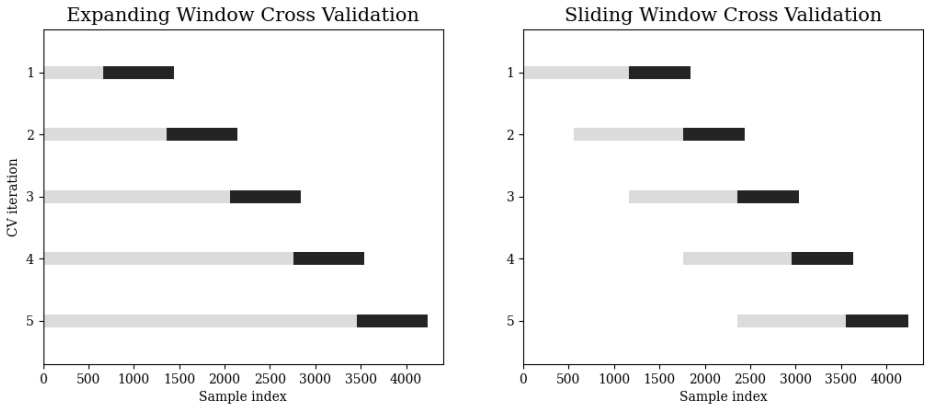



Figure 4: Expanding window- and sliding window cross validation. Grey indices indicate samples used for training the model, while black indices indicate samples used for validation.

There are several considerations to take while choosing between an expanding window-, and a sliding window approach. The first one being the frequency and the availability of historical data. With daily and intra-daily observations we often have a lot of available data compared to weekly and monthly data. Zang (2018) points out lot of available data will in an expanding window approach result in relatively large training sets within the cross validation iterations, which can negatively affect computational cost. Furthermore, sliding windows achieves a favourable balance between model accuracy and training time, and are more true to the environment such models would face in production, where it is retrained on the newest available data, and not necessarily the whole available history. Bergmeir and Benítez (2012) also point out benefits from this method compared to the common practice in which one reserve a part from the end of the time series for testing, stating it does not make full use of the available data.

A well-tuned model exhibits good average performance metrics across all folds, as well as a low standard deviation between the fold's intra-fold performance. In an exhaustive grid search, a fixed set of hyperparameters are evaluated for all possible combinations from a grid of hyperparameters. If we increase the number of hyperparameters or the number of options for each hyperparameter, the total number of combinations grows exponentially. Consequently, exploring every combination becomes unfeasible with our available computing resources.

Bayesian optimization is a technique used to optimize hyperparameters by constructing a probabilistic model of the objective function, and iteratively selecting the next set of hyperparameters to be evaluated based on the current model (Frazier, 2018). This can reduce the number of hyperparameter evaluations needed to achieve good performance, as it is much more likely to have found a good set of hyperparameters before every combination is evaluated than a grid search. For econometric models with a smaller search space, the same procedure can be used, but with a grid search for selecting parameters.

# 4 Data

In this section we will explain the VIX index and how it is estimated. We provide some insights into the nature of the VIX for the duration of our dataset, and elaborate on the motivation behind our choice of exogenous dataset. Furthermore, we describe the preprocessing steps we applied to the data and how we performed the feature selection.

## 4.1 VIX

The CBOE VIX, which is calculated based on real-time prices of options on the S&P 500 index, serves as a reliable indicator of the expected volatility of the S&P 500 index over a 30-day period. When it was first introduced in 1993, the model computed implied volatility using a linear combination of eight near-the-money options on the S&P 100. From 2003, VIX has been calculated in a model-free approach by assuming the underlying follows a non-jump diffusive process, and using the fair value of variance swaps as the basis for the calculations, the expected value of the risk neutral variance $V_t$ can be expressed as

$$V_t = \frac{2}{T}\left(rT - \left(\frac{S_0}{S_*}e^{rT} - 1\right) - \log\frac{S_*}{S_0} + e^{rT}\int_0^{S_*}\frac{1}{K^2}P(K)dK + e^{rT}\int_{S_*}^{\infty}\frac{1}{K^2}C(K)\,dK\right), \quad (22)$$

where $r$ is the risk-free interest rate. $T$ is the time of expiration of the listed contracts, $S_0$ is the current index price of the underlying and $S_*$ defines the boundary between puts and calls. Finally, K is the strike price of the option, while $P(K)$ and $C(K)$ is the current fair value of a put and call (Demeterfi, Derman, Kamal, & Zou, 1999). Chicago Board Options Exchange (2022) discretize this to compute the market's expectation of future volatility,

$$VIX = \left(\frac{2}{T}\sum_i \frac{\Delta K_i}{K_i^2}Q(K_i) - \frac{1}{T}\left[\frac{F}{K_0} - 1\right]^2\right)\cdot 100. \quad (23)$$

$T$ is the time to expiration in years and $F$ is the forward price implied by the option. $K_0$ is the strike price closest to the forward index $F$. By sequentially moving away from the $K_0$, $K_i$ is the strike price $i$ steps away, and $\Delta K_i$ is the interval between strike prices. $R$ is the risk-free interest rate, and $Q(K_i)$ denotes the midpoint between the bid-ask spread for every option with given strike $K_i$.

We included data from the VIX spanning the period 2003-2022 for a total of 5217 observations. The background for the decision to start the data in 2003 was based on two observations: the change of methodology for the VIX implemented in 2003, and the availability for some of the exogenous data. In Figure 5 the historic levels of the VIX are displayed, showcasing the extreme level of volatility for certain events. Most notably are the 2008 financial crisis and the early part of 2020 caused by the beginning of the Covid pandemic.

Figure 6 displays a histogram of the VIX with an overlaid estimated probability density function. It is both positively skewed and leptokurtic, much in part due to the drastic movements during times of financial distress. Furthermore, it is evident that the VIX is not normally distributed. Table 1 shows the results of an Adjusted Dickey-Fuller test for stationarity, where the null hypothesis of a unit root is strongly rejected at the 1% significance level. As such, we can infer

that the VIX is stationary according to this test. Figure 7 displays the autocorrelation function (ACF) and partial autocorrelation function (PACF) plots of the VIX. Figure 7a includes 100 lags and indicates that the VIX has long memory. The PACF plot in Figure 7b implies that the VIX is an AR process, as the partial autocorrelations quickly drops towards zero. Hence, the VIX appears to be an AR process with long memory.

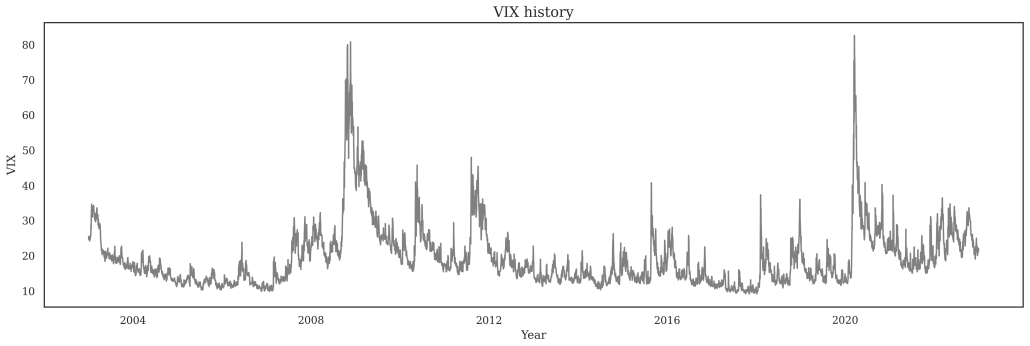



Figure 5: The history of the CBOE VIX in our dataset, ranging from January 2003 to the end of December 2022.

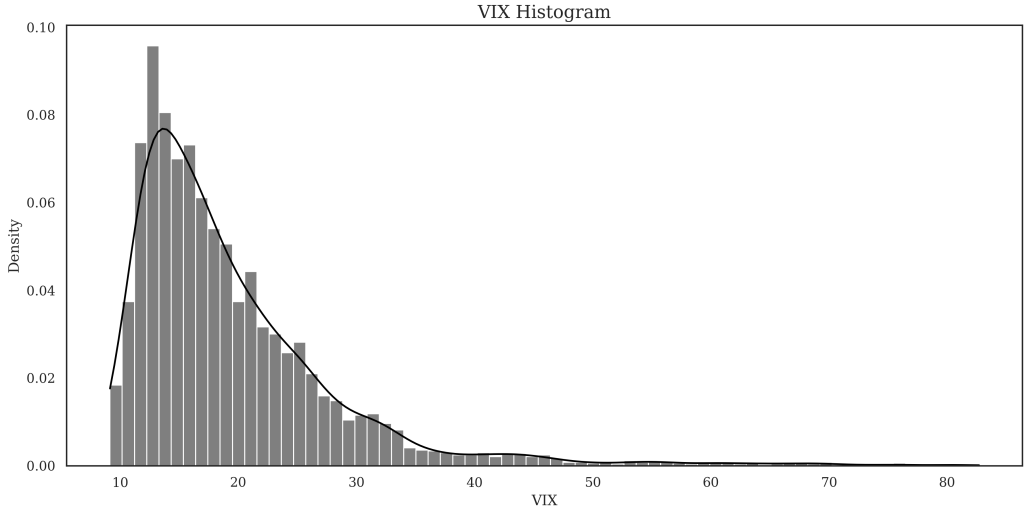
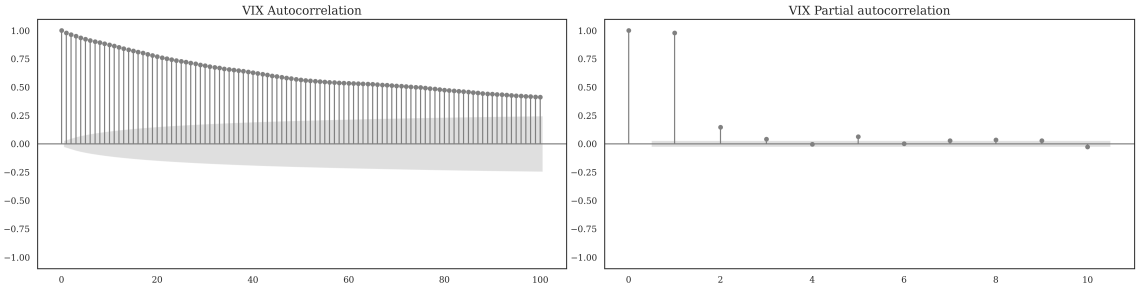



Figure 6: Histogram of VIX with superimposed estimated probability density function.

| Statistic | Value |
|---|---|
| ADF test statistic | -5.567438 |
| 1% | -3.432 |
| 5% | -2.862 |
| 10% | -2.567 |

Table 1: ADF test for the VIX time series. The resulting test statistic is reported in the first row, and the statistics for each significance level is displayed in the remaining rows. Apparent from the test statistic, the null hypothesis is rejected below a 1% significance level.




(a) Autocorrelation function.

(b) Partial autocorrelation function.

Figure 7: The autocorrelation (a) and partial autocorrelation function (b) of the VIX index. 100 lags are included in the ACF plot to illustrate the long memory of the series. The grey line encompasses two times the standard error at lag $k$ for the ACF, and a 95% confidence interval for figure (b).

## 4.2 Exogenous Data

The choice of exogenous data for predicting the VIX largely relies on previous work in forecasting volatility. This is due to the fact that investigating which exogenous data will benefit volatility prediction is not the main concern of this thesis. The data was chosen based on the work of Vrontos et al. (2021); Prasad, Bakhshi, and Seetharaman (2022), who both tried to forecast the direction of the VIX index. As we operate on a daily basis, any monthly and most weekly variables are excluded. Additionally, we included a set of technical indicators as proposed by Ghosh and Sanyal (2021), such as moving averages (MA), exponentially moving averages (EMA), Bollinger bands, and momentum. Furthermore, we included the relative strength index (RSI) and the average directional index (ADX) indicators. A overview of every variable in the initial dataset is shown in Table 35 in Appendix K.

Having a large set of features can be detrimental for several reasons. First, it makes it more difficult to draw inferences from the results about feature impacts when the feature space is very big. Second, too many unrelated features might end up obfuscating interaction for more relevant features, in turn leaving the model less apt at picking up effects from substantial features. This motivates the need for feature selection.

The technical indicators consists of 5- and 10-day moving averages of VIX, exponentially moving averages and momemtum, i.e. the daily observed value minus the 5- or 10-day prior value. Furthermore we have the Relative Strength Index, which measures the speed and change of price movement of the VIX, which is calculated as follows,

1. Calculate the average gain (AG) and average loss (AL) over the specified period
2. Calculate the relative Strength (RS) by dividing the average gain by the average loss
3. Calculate the RSI using the RS: $RSI = 100 - (100/(1 + RS))$.

Lastly, we have the trend strength indicator ADX, which is used to quantify the strength and direction of a trend in the price movement of an asset, and was calculated as shown in algorithm 3.

**Algorithm 3:** Calculation of ADX

**Data:** high, low, close, lookback
**Result:** plus_di, minus_di, adx_smooth
plus_dm = diff(high);
minus_dm = diff(low);
plus_dm[plus_dm < 0] = 0;
minus_dm[minus_dm > 0] = 0;
tr1 = high - low;
tr2 = abs(high - shift(close, 1));
tr3 = abs(low - shift(close, 1));
tr = max(tr1, tr2, tr3);
atr = rolling_mean(tr, lookback);
plus_di = 100 × (ewm_mean(plus_dm, lookback) / atr);
minus_di = abs(100 × (ewm_mean(minus_dm, lookback) / atr));
dx = (abs(plus_di - minus_di) / abs(plus_di + minus_di)) × 100;
adx = ((shift(dx, 1) × (lookback - 1)) + dx) / lookback;
adx_smooth = ewm_mean(adx, lookback);
**return** plus_di, minus_di, adx_smooth;

## 4.3 Preprocessing

The preprocessing procedure involved handling missing values, transforming non-stationary data and scaling the final dataset.

### Handling missing values

In total 7.6% of the days in the dataset contained missing for the daily features. The VIX index contained 176 missing values, constituting 3.4% of the total data. As this is a substantial amount of the total data window, we did not want to drop all records corresponding to the given days. The longest consecutive gaps were two days, and only two such gaps were present. Hence, we found forward filling the last value to be appropriate for our needs, while noting that this created a synthetic portion of our data.

For the remaining features, those on a daily basis exhibited similar characteristics as the VIX, with a mean percentage of missing values of 3% and a max of 6.7%. The longest gaps were never more than four days, with a majority of the features only having gaps of two days or less. The number of such gaps exceeding one day were mostly around two. Based on this observation, values were forward filled for the exogenous features. The same procedure was adopted for the weekly sentiment features, which contained no weekly missing values. To convert them into daily features the values were forward filled.

### Stationarity

An augmented Dickey-Fuller test was performed to check if the exogenous data was stationary. Any variables were the null hypothesis of a unit root was not rejected at a 1% significance level

were made stationary by using the log-returns. These alterations are generally not necessary for artificial neural networks, but the presence of non-stationarity in independent variables might affect an AR model. Hence, to ensure a consistent dataset was shared between all models these actions were undertaken. The resulting transformed features are reported with the suffix $\_R$ in the feature selected tables.

#### Scaling

Artificial Neural Networks can benefit from scaled inputs, especially with datasets containing several features with differences in the magnitude of values. Thus the dataset was scaled to a range between 0 and 1 to ensure stability in the learning process as well as faster convergence. There are several scaling and normalization techniques to consider, with the StandardScaler being a common choice. Daniel (2019) found MinMax to be performing similar as StandardScaler for time series data, and works better when the data distribution is not Gaussian. Additionally, the MinMaxScaler preserves the shape of the data. For a data set containing several features, it operates on each feature independently and transforms the expression through the following expression:

$$X_{\text{scaled}} = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}}. \tag{24}$$

We scaled the data using Scikit-learn's implementation(Pedregosa et al., 2011).

### 4.4 Feature Selection

In some applications, variables which have no correlation to the independent variable serve as pure noise and might introduce bias in the model and reduce the predictive performance. By applying feature selection techniques we can gain some insight into the process and can improve the computation requirement and prediction accuracy (Chandrashekar & Sahin, 2014). They classify variable elimination methods into filter and wrapper methods, where filter methods acts as preprocessing to rank the features to be selected to be applied to a predictor. In wrapper methods predictors are wrapped on a search algorithm which finds a subset which gives the highest predictor performance. One such wrapper method is the Boruta Algorithm as proposed by Kursa and Rudnicki (2010). The Boruta feature selection has several beneficial properties, one being an *all-relevant* problem rather than a *minimal optimal* problem, which means finding all relevant variables instead of only non-redundant ones, and is necessary when trying to understand mechanisms related to the subject of interest. Additionally, Boruta shares the same ideas which form the foundation of the Random Forest, where collecting results from ensembles of randomized samples, can reduce misleading information of random fluctuations and correlations (Breiman, 2001). In the context of feature selection, the randomness provides a clearer view of which variables are important and not (Kursa & Rudnicki, 2010). The fact that the Boruta algorithm is wrapped around the Random Forest algorithm is also a clear motivation from our side to use it as our feature selection algorithm, as Random Forest is a commonly used algorithm for financial time series, due to its capabilities in capturing nonlinear relationships (Gunnarsson et al., 2023). The steps of the Boruta algorithm are described in algorithm 4. We use a Python implementation of Boruta feature selection provided by Homola (2019) with Scikit-learn's Random Forest Regressor as the supervised learning estimator (Pedregosa et al., 2011). These were the hyper-parameters we used for the selection:

**Algorithm 4:** Feature Selection using Boruta Algorithm

**Data:** Original dataset $\boldsymbol{X} \in \mathbb{R}^{n \times m}$, target variable $\boldsymbol{y} \in \mathbb{R}^n$
**Result:** Final set of confirmed features $\boldsymbol{F}$
Initialize shadow dataset $\boldsymbol{X'} \in \mathbb{R}^{n \times m}$;
Initialize feature importance scores: $\text{imp}_i$, $\text{imp}'_i$ for each feature $i$;
Initialize final feature set $\boldsymbol{F} = \emptyset$;
**while** *Some features are not confirmed or rejected* **do**
  Permute feature values in $\boldsymbol{X}$ to obtain shadow dataset $\boldsymbol{X'}$;
  Train a random forest algorithm on $\boldsymbol{X}$ and calculate feature importance scores $\text{imp}_i$;
  **for** *Each feature* $i$ **do**
    Calculate maximum importance score $\text{imp}'_i$ in the shadow dataset;
    **if** $imp_i \geq imp'_i$ **then**
      Mark feature $i$ as confirmed;
      Add feature $i$ to $\boldsymbol{F}$;
    **end**
    **else**
      Mark feature $i$ as tentative;
    **end**
  **end**
  Train a random forest algorithm on the confirmed features $\boldsymbol{F}$;
  **for** *Each tentative feature* $i$ **do**
    Permute feature values in $\boldsymbol{X'}$ to obtain shadow dataset $\boldsymbol{X''}$;
    Calculate feature importance scores $\text{imp}'_i$ in $\boldsymbol{X''}$;
    **if** $imp_i \geq \max(imp'_i)$ **then**
      Mark feature $i$ as confirmed;
      Add feature $i$ to $\boldsymbol{F}$;
    **end**
    **else**
      Mark feature $i$ as rejected;
      Remove feature $i$ from consideration;
    **end**
  **end**
**end**
**return** $\boldsymbol{F}$;

- Number of estimators: 250
- Max depth: 5
- Max iterations: 250

To avoid data leakage, Boruta was applied separately to the training data within each fold. Then, we picked the features which were either tentative or confirmed in at least 3 or more folds. As a robustness check we performed an additional feature selection using an Elastic Net similarly to Vrontos et al. (2021) to compare the selected features.

# 5 Models and Model Selection

The following section describes how we implemented the models introduced in section 3. We will elaborate on the decisions behind how we optimized the hyperparameters and the related search space we explored for each set of models. Additionally, we detail the different Bayesian Neural

Network models we utilized, and we explain the motivation for including them. Finally, we report on how we applied the models on different sets of forecasting exercises.

## 5.1 Econometric Model

From the ACF and PACF plots of Figure 7, we deduced that the VIX appears to be an AR process with long memory. As pointed out in Brooks (2019), an autoregressive process has a geometrically decaying ACF, and the number of non-zero points of PACF equals the AR order. Hence, the Figure 7b suggests that the AR is of order 5. Based on these observations, the order of the AR models will be tuned with an upper limit of 5 lags. The AR and ARX models in this thesis were made with Statsmodels' Python implementations (Perktold et al., 2023).

## 5.2 Deterministic LSTM

To benchmark the BNN against a deterministic network, we implemented a traditional LSTM model. A summary of the hyperparameters we tuned can be found in Table 2. For all our deep learning models, we have used the LSTM-layer implementation from Tensorflow Keras as building blocks (Abadi et al., 2015). The Bayesian optimization procedure was applied using the Keras-Tuner module (O'Malley et al., 2019).

### Layers

One of the first considerations to make when using DNNs involves the choice of model architecture. Hornik, Stinchcombe, and White (1989) proved that a neural network with one hidden layer is a universal approximator. However, it does not guarantee that the given implementation will be able to converge to the desired function, as a number of choices with regards to network architecture, hyperparemeters and preprocessing will influence this ability.

For financial time series forecasting of volatility, implementations using LSTM does generally not have many hidden layers. Bucci (2020); Ghosh and Sanyal (2021); Osterrieder et al. (2020); Petrozziello et al. (2022) all used three layers or fewer for their models, often with no more than 50 units. Financial data such as the VIX is noisy data, which can make overly complex models overfit to spurious signals. This entails that the model performs well on the training data, but its performance drops when applied to unseen test data. Furthermore, neural networks are susceptible to overfitting due in part to their expressiveness. Creating too deep or too wide models - models with many layers or many neurons, respectively - may increase the risk of overfitting. While regularization approaches can be taken to limit this effect, we have opted to constrain the number of hidden LSTM layers between one and two.

### Number of neurons

Selecting the number of neurons within each layer can be a complicated task. Having to few neurons may lead to underfitting, while having too many can cause overfitting. As mentioned above, the number of neurons for LSTM models did generally not exceed 50 when forecasting

volatility-related data. Hence, to balance the exploration against a too encompassing search space, we constrained these to be within an interval of 16 and 64 for the first layer. For the final LSTM layer, we allowed the tuning process to include an increased number of neurons up to 128 units. If another hidden layer was present between these, it was allowed to have units within an interval of 32 and 128.

#### Regularization

Neural networks are powerful function approximators, but this comes at a cost; they can easily overfit to the training data. Srivastava, Hinton, Krizhevsky, Sutskever, and Salakhutdinov (2014) introduced the Dropout function as an easy alternative for preventing overfitting, as elaborated on in section 3. To give the deterministic network some regularization abilities, we included the option of a final Dropout layer after the last LSTM layer. For the tuning procedure, we allowed this rate to be in the interval of $r \in [0, 0.5]$ with a step size of 0.1.

#### Learning rate

The learning rate can be a crucial hyperparameter. It determines how quickly the network is able to learn from the gradient descent procedure. Having too low of learning rate will cause the network to learn more slowly, and raises the risk of getting caught in local optima. Similarly, too large of a learning rate may cause the back-propagation process to diverge. To have a more intuitive and limited search space, we let the learning rate hyperparameter be selected from three values: from 1e-4 up to 1e-2.

#### Optimizers

Optimizers in a deep learning method decides how the parameters are updated through gradient descent. We utilized the *Adam* optimizer of Kingma and Ba (2017). It is a popular choice for an optimizer, and is generally less sensitive to the choice of hyperparameters. Hence, to avoid expanding the search space we used this as the fixed optimizer of choice for all deep learning models.

| Hyperparameters | LSTM ENDOG/EXOG |
|---|---|
| Learning rate | [0.01, 0.001, 0.0001] |
| Input unit | [16-64], step = 16 |
| Use extra hidden layer | Bolean |
| Hidden units | [32-128], step = 32 |
| Last LSTM units | [16, 128], step = 16 |
| Dropout rate | [0, 0.5], step = 0.1 |

Table 2: Hyperparameter search space for LSTM models.

## 5.3 BNN

In implementing the BNN, there were two extra factors to consider: the choice of prior and posterior distribution. These can have severe effects on the resulting model, forcing special care to be taken when deciding on which distributions to use. To complicate the search space even more, these choices comes in addition to considerations surrounding the neural network architecture and optimization procedure. Thus, the main categories of models were directed at forecasting the VIX index 1-day ahead without exogenous data. This is the least complex forecasting environment; there is no exogenous data to cloud potential model performance. Overall, this makes it easier to infer results from how different models perform against each other, particularly with respect to the BNNs and their different specifications. Additionally, when considering each potential dimension of the forecasting exercise, tuning every model for all specification would be an arduous task beyond the intended scope of this thesis. Hence, to incorporate exogenous data into the mix we decided to limit this inclusion for the Normal BNN, and focus on comparing its performance against its endogenous counterpart in addition to the AR and LSTM models.

An important part of the Bayes-By-Backprop procedure involves the type of prior and posterior distribution. The prior is most often chosen to be a Gaussian prior (Fortuin, 2022; Jospin et al., 2022). While it might be an appropriate prior for some applications, there are some problems with using Gaussian priors. In particular, the cold posterior effect demonstrated by Wenzel et al. (2020), where a tempered posterior outperforms an un-tempered posterior, hints that Gaussian priors may be a misspecification. Furthermore, these kinds of networks can converge to a Gaussian process in the limit if number of units, and therefore disregard the need for a BNN in the first place. Thus, the choice of prior is highly complex and a difficult task in and of itself. To test the performance of different priors, we included models with different choice of distributions: a multivariate diagonal normal prior, a Laplacian prior and a Gaussian scale mixture prior. The latter was based on the implementation of Blundell et al. (2015), while the Laplacian implementation was chosen due to its heavier tails and potential for preventing the convergence into a Gaussian process. We tuned and tested each implementation separately to assess their performance in the forecasting exercise, both with regards to forecasting accuracy and uncertainty quantification.

To simplify the searching procedure, we opted to use a standard multivariate diagonal normal distribution as our posterior for every implementation, as was done in Blundell et al. (2015). For ensuring numerical stability, the standard deviation $\sigma$ in the posterior distribution is altered through three steps. Let $\rho$ be the raw estimated standard deviation. Then, the resulting $\sigma$ which parameterizes the surrogate posterior is obtained by the following equation:

$$\sigma = 0.001 + R \cdot Softplus(\log(e-1) + \rho) \tag{25}$$

First, we ensure that $\sigma$ does not obtain too small values by having the constant 0.001. Then, the input $\rho$ is added to $\log(e-1)$. This serves two purposes: it puts a minimum bound on the input of the $Softmax$ function for numerical stability, and as the initial values for $\rho$ are all zeros it makes the $Softmax$ return ones when initializing the model. Finally, a rescaling factor $R$ is multiplied with the output of the $Softmax$. This factor scales down $\sigma$ to disallow large fluctuations in the sampled weights, thereby causing more stable network behavior. We observed large differences in both the required training time and model predictions when this rescaling was not performed, and the obtained networks were highly sensitive to this parameter. Hence, we decided to include it into

the tuning procedure. The other factors of Equation 25 were not tuned to constrain the search space.

Training BNNs with Bayes-By-Backprop can take a long time. Due to the stochasticity of the sampled weights, training becomes noisy and many epochs are often required for the model to learn the parameters of the posterior. However, the networks can be simplified if we only use Bayesian layers in the last $n$ layers, as described in Jospin et al. (2022). In section 4, we noted that the VIX is a stationary time series, with long-term autocorrelation gradually decaying to zero. The LSTM network are able to exploit such long-term feature interactions and extract the information therein at a sophisticated level. If we allow the first few layers of a *hybrid* BNN to be LSTM, these can aid in catching time-dependent feature interactions in a quicker manner. In essence, the first few layers act as feature extractors, where they capture the time series aspect of the data and produces a condensed feature space for the uncertainty quantification process. Thus, while fewer Bayesian layers reduces some of the ability to fully capture the uncertainty, it can lead to more accurate predictions in a faster manner. To assess this effect, we implemented a pure BNN, hereafter referred to as a Pure BNN, in addition to the hybrid BNNs. In order to limit the size of our model environment, this was only done on a model with a normal prior. Further, we allowed the BNN models to have a larger search space for the learning rate by using a uniform distribution due to their more complex and stochastic nature.

To make predictions using the BNNs, we used 100 Monte Carlo samples. We found this to be an acceptable middle ground of adequate sampling against the added time requirements. Furthermore, we used one Monte Carlo sample of the weights during gradient descent to save on computing time. Both of these parameters could be included in the tuning procedure, but we decided to constrain the search space by pre-determining them. Finally, we decided against directly modelling the aleatoric uncertainty. During the early stages of our project we noticed a tendency of models to diverge or require much longer computing time when we accounted for aleatoric uncertainty. Thus, to reduce the computational complexity and simplify the tuning procedure we opted to focus on the epistemic uncertainty. Estimating only epistemic uncertainty in a pure epistemic model generally requires fewer computational resources compared to modeling both types.

To implement The Bayes-By-Backprop algorithm for the hybrid and pure BNNs, we utilized the probabilistic framework provided by Tensorflow through the Tensorflow Probability module (Abadi et al., 2015).

### Pure BNN

Similarly to the deterministic models, the BNN architecture had to be tuned. To limit the search space, the model could have up to three hidden layers. While testing BNNs on simpler toy problems, we noticed a tendency for deep pure BNNs to require substantially more training time. Thus, we concluded that three layers should be sufficiently expressive and offer an acceptable balance against the required tuning time. Further, the number of neurons were limited to be in an interval of between 8 and 64 units. As mentioned earlier, the prior and posterior distributions were both chosen to be diagonal multi-variate normal distributions. The ELU activation function of Clevert, Unterthiner, and Hochreiter (2016) were used for all layers but the last to aid with faster convergence. The final hyper-parameters were the prior $\sigma$, posterior rescale factor and learning rate. The

search space for this model can be found in Table 3.

| Hyperparameters | Pure Bayesian |
|---|---|
| Prior $\sigma$ | Uniform [0.1, 2] |
| Posterior rescaler | Uniform [0.0001, 0.1] |
| Input units | Uniform [8, 64] |
| First hidden | Boolean |
| First hidden units | Uniform [8, 64] |
| Second hidden | Boolean |
| Second hidden units | Uniform [8, 64] |
| Learning rate | Uniform [0.0001, 0.1] |

Table 3: Search space hyperparameters for Pure Bayesian model.

**Hybrid Bayes-By-Backprop**

To save on tuning time and reduce the search space, we used the pre-tuned LSTM architecture as the deterministic part of the hybrid BNNs. In doing so, we avoid having to tune all BNN parameters simultaneously as the LSTM aspect. While this simplifies the tuning process, we forego picking up on some other potential interactions between the deterministic and Bayesian architectures. Additionally, this may be viewed as an advantage for the hybrid models, as they essentially are tuned two times. However, due to the complexity of our problem and the sheer number of models to tune, we opted for this approach while keeping this factor in mind for any potential forecasting comparison.

We implemented three variants of the hybrid models, based on the type of prior distribution: a multivariate diagonal normal model (Normal BNN), a Laplace model (Laplace BNN) and a Gaussian scale mixture model (Scale Mixture BNN). The Normal BNN additionally included an exogenous variant. For each model, we used a maximum of two dense variational (DV) layers. These are the layers implementing the Bayes-By-Backprop algorithm. Having several DV layers after the deterministic part can potentially allow the model to capture more uncertainty. However, it increases the model's complexity, both with regard to the architecture and time needed to converge. For these reasons, we allowed searching process to include one potential hidden DV layer as a choice. This layer could have units drawn from a uniform interval between 2 and 64 units. Learning rate and posterior rescaling factor were picked similarly to the other models. Both the Laplace and multivariate normal models needed simply tuning the standard deviation of the prior. Their corresponding hyperparameters are described in Table 4. For the Scale Mixture BNN model, we followed the approach of Blundell et al. (2015), demonstrated in Equation 26. This involved having a mixture of two normal distributions, where both had zero mean but one had a standard deviation much larger than the other.

$$P(\boldsymbol{w}) = \prod_i \pi \mathcal{N}(w_i|0, \sigma_1) + (1 - \pi)\mathcal{N}(w_i|0, \sigma_2) \tag{26}$$

A visual demonstration of each prior distribution can be found in Figure 8. Note the heavier tails and higher peak of the Laplace and the Gaussian scale mixture. Since the Scale Mixture in the BNN operates in log-space, a sum of the distributions are shown here.

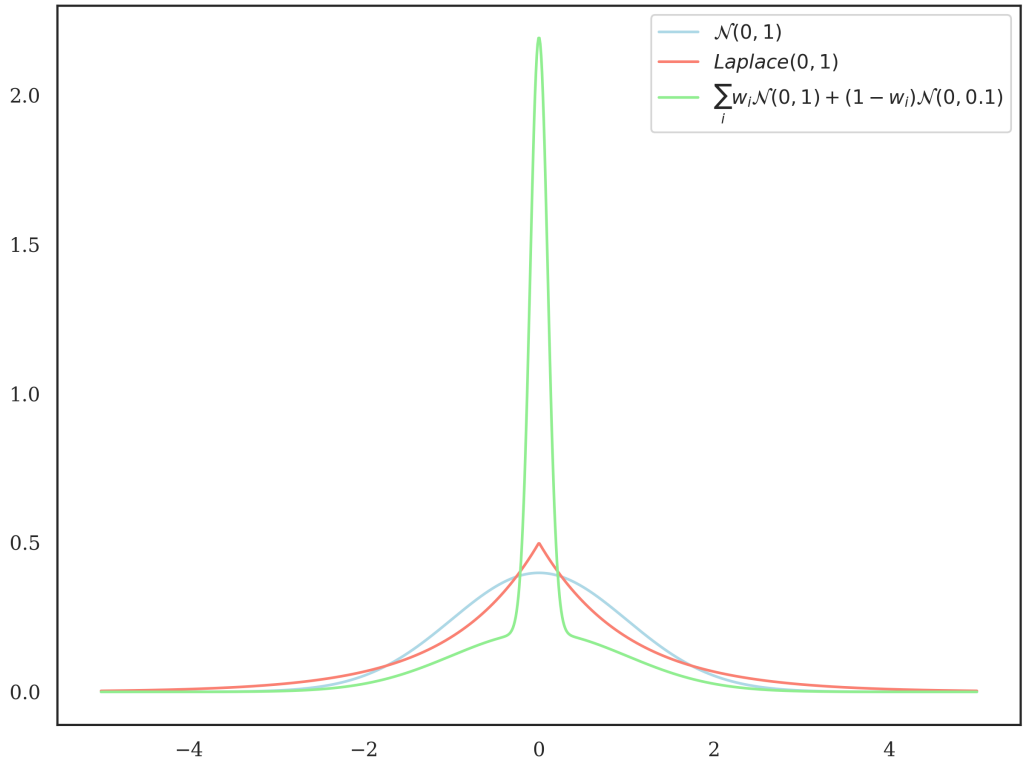



Figure 8: A visualization of the three prior distributions we implemented. The blue line displays a standard normal distribution, while the red line shows a Laplace distribution with zero mean and variance 1. The green line illustrates a mixture of two normal distributions, one standard normal and one with zero mean and standard deviation of 0.01.

In total, this entailed having to tune the mixing parameter between the distributions and the $\sigma$ for both. The hyper-parameters of the Scale Mixture model can be found in Table 5

| Hyperparameters | Laplace prior | Normal prior |
|---|---|---|
| Prior $\sigma$ | Uniform [0.1, 2] | Uniform [0.1, 2] |
| Posterior rescaler | Uniform [0.0001, 0.1] | Uniform [0.0001, 0.1] |
| Use hidden DV | Boolean | Boolean |
| Hidden DV units | Uniform [2, 64] | Uniform [2, 64] |
| Learning rate | Uniform [0.0001, 0.1] | Uniform [0.0001, 0.1] |

Table 4: Search space for hyperparameters in hybrid Bayesian neural networks with Laplace prior and Normal prior. DV is the dense variational layers responsible for making Bayesian inference.

| Hyperparameters | Scale mixture prior |
|---|---|
| Components weight | [0.25, 0.5, 0.75] |
| First $\sigma$ | [0.1, 0.25, 0.5, 0.75, 1] |
| Second $\sigma$ | [0.0001, 0.00025, 0.00075, 0.001] |
| Posterior rescaler | Uniform [0.0001, 0.1] |
| Use hidden DV | Boolean |
| Hidden DV units | Uniform [2, 64] |
| Learning rate | Uniform [0.0001, 0.1] |

Table 5: Search space for hyper parameters in hybrid Bayesian neural networks with Scale Mixture prior. DV is a Dense Variational Layer, and is the layers in the network responsible for performing variational inference.

### Monte Carlo Dropout

These models were mostly similar to the LSTM models, but with an additional dropout layer between the first and second layer. The learning rate was trained as usual. The hyperparameters are summarized in Table 6.

| Hyperparameters | MC Dropout |
|---|---|
| First dropout rate | [0.1, 0.6], step = 0.1 |
| Second dropout rate | [0.1, 0.6], step = 0.1 |
| Learning rate | Uniform [0.0001, 0.1] |

Table 6: Search space for hyper parameters in the MC Dropout models.

## 5.4 Forecasting and tuning procedure

To prevent data leakage, 70% of the initial data was put into a training set. This spanned a period of 3651 days, from 2003-01-02 into 2016-12-29. For the testing data, we wanted to have two test sets: one pre- and one during and post-Covid. These periods differ in the volatility of the VIX index, where the post-Covid era is characterized by more jumps and a higher VIX value. The split is visualized in Figure 9, demonstrating these effects. For the Walk-forward validation scheme,

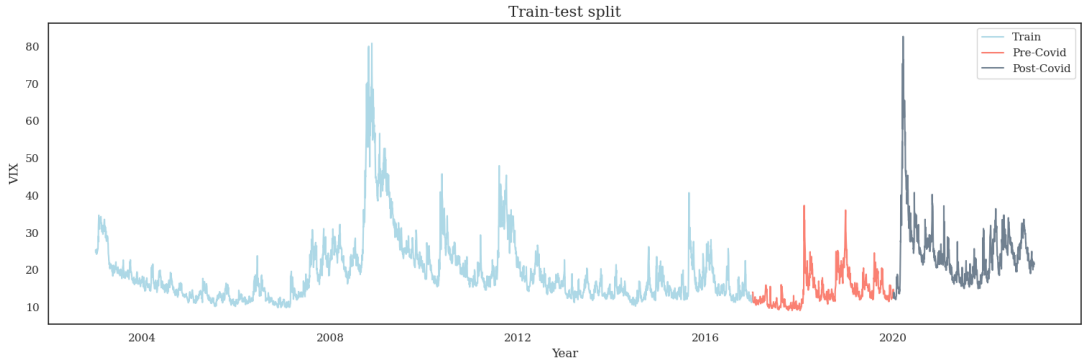



Figure 9: The train-test split. The blue line is the training data. The red line shows the first test set, and the dark grey line illustrates the second test set.

the training data was further split into five folds with a 3:1 ratio of training and validation data. The folds are visualized in Figure 10. Next, each model was optimized with respect to the choice of hyper-parameters. Due to the different nature of the econometric and ML models, the tuning procedure for each category of models were done differently.

### AR model

While there are not a single definite way to determine the number of lags $p$ in an AR model, the most common ways are to look at the autocorrelation function and the partial autocorrelation function of the time series, as well as using information criteria such as AIC and BIC. Bergmeir and Benítez (2012) state that there is a gap between evaluation of traditional forecasting procedures, and evaluation of machine learning techniques. Furthermore, they emphasize that a blocked form of cross validation lead to a model robust model selection, and describe this approach as "best of

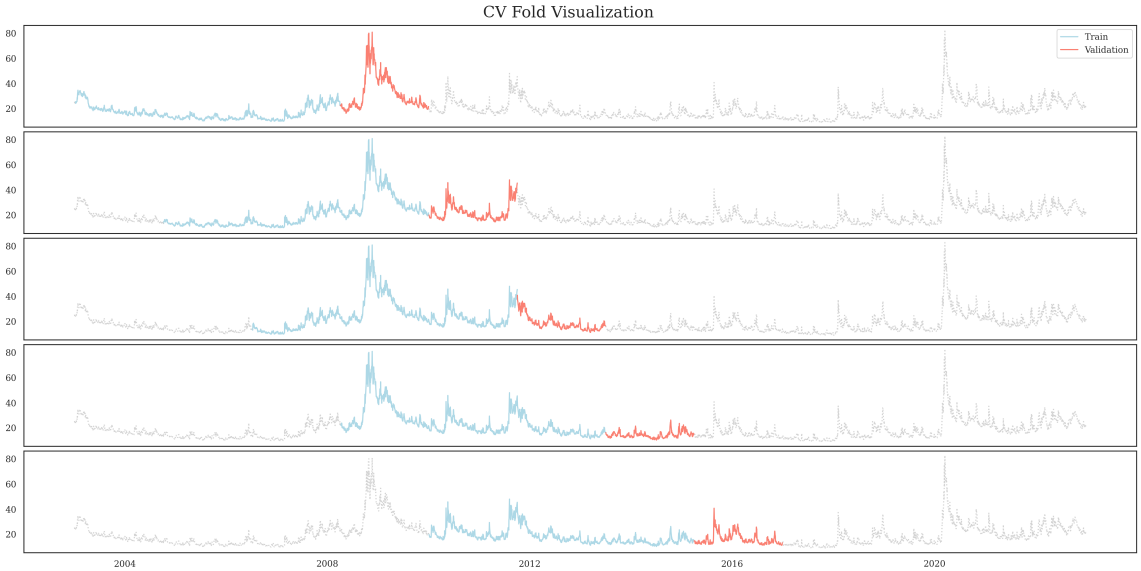



Figure 10: The five folds used for cross-validation. The blue line is the training data for each fold, while the red marks the validation data. The dotted grey line illustrates the entire dataset

both worlds". Therefore we have adapted the same cross validation procedure as which was done with our deep learning models, but evaluating every possible combinations of lags up to 5, and thereafter selecting based on a qualitative assessment of error metrics and information criteria.

### Machine learning models

For the ML models, the search space was significantly larger. We used the Bayesian tuner mentioned in section 3 for a total of 100 trials for each model. A trial consisted of initializing a model, fitting the model to the fold data, and finally evaluating the model according to pre-determined metrics. After repeating this for each fold, the evaluation metric was averaged to assess the model performance with the current hyper-parameters. Additionally, the standard deviation of the metric was recorded to check for the spread of the results. If the model in question was a BNN, the Monte Carlo expectation of 100 predictions samples were the basis for the error metric. For these models, two additional metrics related to uncertainty quantification were stored. The measured standard deviation to assess how much epistemic uncertainty varied, and its correlation to the residual value to assess if they exhibited a positive linear relationship.

When a tuning session was completed, the top ten best models were extracted for closer scrutiny. The tuner was focused on the mean squared error metric, but we wanted to ensure the best model's consistently outperformed the others with respect to more metrics. Performance within each fold were also investigated to check if the results were consistent amongst the folds. If the top performer appeared to consistently outperform the others, the hyper-parameters were set according to the given trial. We often observed close results amongst the top performers, but they were generally similar; due to the search space allowing for tiny increments, they could at times be identical except for a minor difference in the learning rate. Thus, in every case the best trial results were the picked for the final set of hyper-parameters. See Appendix J for information about evaluation metrics used.

**Forecasting procedure**

As mentioned in subsection 5.3, our main efforts were centered around forecasting the VIX index one-day-ahead without exogenous data. In financial data, the directional movement of an asset is often as interesting as the actual value itself. Hence, we included this as a separate forecasting exercise. This entailed making two realizations of 1-day forecasts: the future movements of the VIX through a regression problem, and the direction of the movement through a classification problem. For the latter, the models were trained on a binary target, with the positive class 1 indicating that the VIX will go up the next day, while the negative class 0 indicating a decrease or no change.

The overall forecasting procedure was done as follows:

1. Tune each model separately using cross-fold validation
2. Extract the best hyperparameters
3. Train the model on the training set
4. For each test set, iteratively forecast the next day of the VIX index, recording the residuals along the way
    - If the model in question was a BNN, 100 Monte Carlo samples of each prediction were drawn to obtain an estimate of the epistemic uncertainty.
5. Evaluate each model with the specified metrics
6. Perform a Diebold-Mariano test to check for equal predictive accuracy on the endogenous model to assess the statistical significance of the findings (Diebold & Mariano, 2002). As the ML models were trained on the mean squared error metric, it was chosen as the test criterion.

To inspect how the forecasting horizon affects our results, we conducted two additional forecasting exercises. First, we produced 5-day dynamic forecasts for a selected subset of models: the AR model, the LSTM model and the Normal hybrid BNN model. For a longer horizon, we produced a 20-day forecast using the same set of models, were they were tuned specifically for this purpose.

The 5-day dynamic forecasting exercise was implemented using the 1-day forecasting models. To produce the forecasts at time $t$, we first predicted $\hat{y}_{t+1}$. Then, we rolled the window one step forward and added the prediction to the input window. This process was repeated four times until a 5-day forecast $\hat{y}_{t+5}$ was obtained. As the exogenous models requires an estimate of external data to obtain all necessary predictors, they were excluded from this exercise.

For making 20-day forecasts, we included the exogenous variants of the machine learning models. The AR model used dynamic predictions, and the machine learning models were trained to explicitly forecast the VIX 20 days into the future. While this might introduce some advantage in favor of the machine learning models, we wanted to keep the VIX data on a daily basis without aggregating data into weekly or monthly variants. Hence, we found this approach to be the most fitting for the AR model.

For all forecasting exercises we used a 30-day window as input to the machine learning models. The window size corresponds to the number of lags included as input to the model. To investigate the impact of this parameter, we conducted additional forecasts using models that were trained with 5, 15, 45 and 60-day windows.

# 6 Results and Discussion

In this section we report the results from the feature selection, tuning procedure and the forecasting exercises. We will provide results for different forecasting horizons, were we mainly focus on 1-day forecasts. We detail the uncertainty quantification for a selected set of models, and include the findings from additional forecasting horizons. Finally, we recount criticisms concerning our implementation and describe an outline for future research into Bayesian deep learning and volatility forecasting.

## 6.1 Results From Feature Selection

The features in Table 7 were selected for 1-day ahead forecasting. Out of the non-technical 34 exogenous features, only three survived the Boruta feature selection. For 20-days-ahead, see Appendix B. Descriptive statistics of the selected features can be found in appendix C. An alternative feature selection using an Elastic Net-procedure for 1-day forecasts are reported in appendix D, generally yielding similar results as Table 7

| Variable name | Description | Type |
|---|---|---|
| VIX | CBOE VIX | Index |
| S&P 500_R | Standard and Poor's 500 | Index |
| BAMLC0A0CM_R | ICE BofA U.S. Corporate Index option-adjusted spread | Macro |
| BAMLC0A0CMEY_R | ICE BofA U.S. Corporate Index effective yield | Macro |
| MA_5 | VIX moving average 5 days | Technical |
| EMA_5 | VIX exponentially moving average 5 days | Technical |
| MOM_10 | VIX momentum 10 days | Technical |
| BOLUP_5 | VIX Bollinger upper band 5 days | Technical |

Table 7: Selected variables for one-day-ahead predictions. The _R suffix indicates that the data has been made stationary by taking the log-returns.

## 6.2 Results From Tuning Procedure

### AR

For the endogenous AR model, the outcomes were very close with respect to the number of lags. The best performance for the mean MSE and MAE was achieved with 1 and 5 lags. For the MSE the AR(5) model achieved the best results, and the AR(1) performed better with the MAE metric. However, when looking at the results side by side the AR(5) had a better overall ranking with both metrics, causing us to pick the AR(5) model. Similar observations were made with regard to the AIC and BIC information criteria.

The exogenous AR model gave more conclusive results. An AR(2) model provided the superior forecasts using all evaluation criteria, albeit by a small margin. Hence, we picked this model as the exogenous econometric model.

### LSTM

A survey of the top ten best models for both the endogenous and exogenous models showed a decent variation in model architecture, but the general trend portrayed a preference for selecting a higher number of neurons. The variations mainly came from how these were allocated between the first and second LSTM layer. For both models, the best model outperform the others with respect to all metrics, giving the results in Table 8.

| Hyperparameter | LSTM ENDOG | LSTM EXOG | LSTM ENDOG BIN |
|---|---|---|---|
| Learning rate | 0.0001 | 0.0001 | 0.0001 |
| Input unit | 64 | 32 | 16 |
| Use extra hidden layer | FALSE | FALSE | FALSE |
| Hidden units | 0 | 0 | 0 |
| Last LSTM units | 96 | 112 | 128 |
| Dropout rate | 0 | 0.1 | 0.1 |

Table 8: Tuning results LSTM 1-day-ahead.

### BNNs

For most of the cases we found that the best ranked model with respect to the MSE were amongst the better models for other metrics. The MC Dropout and the hybrid BNNs all shared an initial LSTM architecture, but they still varied in how conclusive their respective tuning results were. The Normal BNNs and Laplace Priors gave a clear indication of the preferred model architecture, while the Scale Mixture Gaussian BNN had a greater variety. Similarly, the Pure BNN proved to have the greatest level of diversity amongst the best performing models. As this model did not enjoy the benefits of tailor-made sequencing layers through the LSTM layers, these results can be expected.

The chosen set of hyperparameters for the MC Dropout model can be found in Table 9, while the Pure BNNs hyperparameterse are displayed in Table 10. Due to the similarity between their tuneable hyperparameters, the Normal endogenous and exogenous BNN and the Laplace BNN can be found together in Table 11, and the Scale Mixture models hyperparameters are reported in Table 12. In general, all models preferred to scale down the surrogate posterior distribution variance by a factor of 1e-2 to 1e-1. Furthermore, there is a tendency for the models to gravitate towards a learning rate between 1-e3 and 1e-4. All models appeared to have a prior standard deviation of around 1. However, the Laplacian model distinguishes itself by having a large prior standard deviation of 2 and a high rescale factor of 0.1. Hence, we can expect this model to give much more varied predictions than the others. A more detailed description of the hyperparameters can be found in appendix E.

| Hyperparameter | MC DROPOUT |
|---|---|
| First dropout rate | 0.4 |
| Second dropout rate | 0.5 |
| Learning rate | 0.00136 |

Table 9: Tuning result one-day-ahead Monte Carlo Dropout Model

| Hyperparameter | PURE BNN ENDOG |
|---|---|
| Learning rate | 0.0001 |
| Input units | 22 |
| Use first hidden | False |
| First hidden units | 0 |
| Use second hidden | TRUE |
| Second hidden units | 45 |
| Prior $\sigma$ | 1 |
| Posterior rescaler | 0.0239 |

Table 10: Hyperparameters pure Bayesian neural network using variational inference for one-day-ahead.

| Hyperparameter | NORMAL BNN ENDOG | NORMAL BNN EXOG | LAPLACE ENDOG |
|---|---|---|---|
| Learning rate | 0.001 | 0.001 | 0.0001 |
| Use extra hidden DVI layer | FALSE | FALSE | FALSE |
| Hidden DVI units | 0 | 0 | 0 |
| Prior $\sigma$ | 0.747 | 0.747 | 2 |
| Posterior rescaler | 0.0128 | 0.0128 | 0.1 |

Table 11: Tuning result one-day-ahead Normal BNN and Laplace BNN.

| Hyperparameter | SCALE MIXTURE ENDOG |
|---|---|
| Learning rate | 0.00265 |
| Use hidden DVI | FALSE |
| Hidden DVI units | 0 |
| Mixture weight | 0.5 |
| First component $\sigma$ | 0.75 |
| Second component $\sigma$ | 0.00075 |
| Posterior rescaler | 0.0849 |

Table 12: Hyperparameters for hybrid BNN using scale mixture priors for one-day-ahead.

## 6.3 One-day-ahead Predictions

The resulting forecasting performance for the entire test set is displayed in Table 15. As can be seen in the table, the endogenous and exogenous AR model outperformed all other models for every metric. This indicates that the AR model provides superior forecasts one-day-ahead compared to

the LSTM networks, in line with the results of Oliveira et al. (2017). The AR model is specifically developed to model autoregressive processes. As the VIX can be considered an autoregressive process, this supports its efficacy for modellings such phenomena. The next best model was the Normal Prior BNN model, narrowly beating the endogenous LSTM model. This was likely caused by the regularization provided by the BNN. However, opposite results can be observed for the exogenous variants, where the exogenous LSTM has better scores for all metrics but the MDA. We observe that both the LSTM and Normal BNN performs worse with exogenous data, but the latter appears to be impacted to a larger degree. This might stem from the inherent stochasticity of the hybrid BNNs and their general sensitivity to the modelling factors, causing them to react more variably to any potential spurious relationship between the dependent and independent variables.

Next, the other BNN models provided worse forecasts with respect to the metrics compared to the endogenous Normal prior. The Laplace BNN and Pure BNN yields the worst results, where the Laplace performance likely stems from its higher rescaling factor allowing for more variability in its predictions. A similar explanation can be surmised for the Pure BNN, as it incorporates more layers with uncertainty quantification. Further, the pure BNN model did not incorporate any LSTM layers. A visualization of a $\pm 2\sigma$ confidence band in appendix G supports this observation. The Pure BNN yielded worse results than its hybrid counterpart in the Normal BNN, motivating the use of hybrid BNNs for more effective learning.

The separated test sets give similar results as the overall test set. Pre-Covid results are presented in table Table 13. Again, the AR and Normal BNN models appears to perform the best, albeit with some differences to the overall test set. The MC dropout model beats the Normal BNN with regards to the RMSE and MSE. These metrics are both sensitive to outliers and larger errors, indicating that the MC Dropout model is less affected than the endogenous Normal BNN by larger jumps in the VIX for this period. This might be caused by the stochasticity of the Normal BNNs forecasts and the epistemic uncertainty. For more turbulent jumps, we can hypothesize that the BNN becomes more uncertain about its forecasts. Therefore, the sampled predictions may have a wider spread, resulting in less accurate mean predictions. Similarly, the MC dropout model may be less affected by these larger errors than the Normal BNN. However, the differences are marginal, and may be caused by small variations within the networks and errors caused by not performing sufficiently numerous Monte Carlo sampling of predictions.

During and after Covid, we can see similar results to the other test sets. In Table 14, the results for the test set pre-Covid are presented. In this regime, the endogenous LSTM outperforms the endogenous Normal BNN for the MAE, MAPE and Huber metrics. This is consistent with the trend observed for the first test set; the LSTM seems slightly more susceptible to larger errors. However, in this iteration the Normal BNN appears to be more affected by the generally turbulent times after Covid. One potential reason might again be the uncertainty. Next, the performance of the MC dropout model drops significantly compared to pre-Covid. A similar effect can be found for the rest of the BNNs with alternative priors, wherein their results are more degraded during the second test set compared to the other models. This observation supports the theory of BNNs general sensitivity to more turbulent data.

A noticeable aspect of the evaluation is each model's performance with and without exogenous data. Every model produced markedly worse results, most prevalent for the Normal BNN model.

Furthermore, the difference between the two testing periods were larger for the models with exogenous data. This hints at two potential features of our dataset. First, their inclusion appears to dilute the signal given by lagged values of the VIX itself for the machine learning models, thereby degrading the model's performance. Second, volatility is a dynamic phenomenon, therefore, it might be possible that the previous relationships between the exogenous variables and the VIX observed during the training have changed in this second more turbulent period, possibly with more noise. The exogenous AR model had better MSE and RMSE scores than its endogenous counterpart for the first test set, further suggesting this explanation.

The results of the Diebold-Mariano tests are presented in table Table 18 for each testing period, and generally coincides with the findings above. The endogenous AR, LSTM, Normal BNN and MC dropout consistently rejects the null hypothesis of equal predictive accuracy at the 5% level or less when comparing with a majority of the other models. Between these set of models', the results are less clear. The Normal BNN and LSTM do not significantly outperform the other, except for in the period pre-Covid. This corroborates the earlier results, where the Normal BNN appears to lose some of its predictive performance after Covid. A similar observation can be made for the MC dropout model. At the 5% significance level it outperforms the predictive accuracy of the endogenous LSTM pre-Covid, but the effect is reversed in the second test set. These findings may hint that inferences in Bayesian Neural Networks are more susceptible to outliers and periods of higher volatility.

As a final observation, we plot residual diagnostics of the endogenous AR(5) model in appendix F. These results show that residuals of the AR(5) are not normally distributed, likely caused by the large jumps in the VIX for the financial crisis of 2008 and other similar events.

| | MSE | MAE | MAPE | RMSE | Huber | R2 | MDA |
|---|---|---|---|---|---|---|---|
| AR ENDOG | 2.273 | 0.879 | **0.055** | 1.508 | **0.537** | 0.874 | 0.485 |
| AR EXOG | **2.265** | **0.877** | **0.055** | **1.505** | 0.538 | **0.875** | 0.481 |
| LSTM ENDOG | 2.54 | 0.972 | 0.062 | 1.594 | 0.605 | 0.859 | 0.449 |
| LSTM EXOG | 2.763 | 0.983 | 0.062 | 1.662 | 0.626 | 0.847 | 0.457 |
| NORMAL BNN ENDOG | 2.346 | 0.881 | **0.055** | 1.532 | **0.537** | 0.87 | 0.466 |
| NORMAL BNN EXOG | 3.062 | 1.078 | 0.069 | 1.75 | 0.698 | 0.83 | 0.455 |
| PURE NORMAL BNN ENDOG | 4.533 | 1.394 | 0.090 | 2.129 | 0.978 | 0.749 | 0.438 |
| LAPLACE BNN ENDOG | 4.217 | 1.312 | 0.084 | 2.054 | 0.91 | 0.766 | 0.423 |
| SCALEMIXTURE BNN ENDOG | 3.612 | 1.293 | 0.084 | 1.901 | 0.872 | 0.8 | 0.469 |
| MC DROPOUT ENDOG | 2.327 | 0.962 | 0.064 | 1.526 | 0.584 | 0.871 | **0.486** |

Table 13: Evaluation of forecasting performance of the specified models on the first test set. The best result amongst the models for each metric is highlighted in bold.

| | MSE | MAE | MAPE | RMSE | Huber | R2 | MDA |
|---|---|---|---|---|---|---|---|
| AR ENDOG | **6.491** | **1.498** | **0.055** | **2.548** | **1.097** | **0.912** | **0.475** |
| AR EXOG | 6.518 | 1.509 | **0.055** | 2.553 | 1.107 | **0.912** | 0.471 |
| LSTM ENDOG | 6.83 | 1.545 | 0.057 | 2.613 | 1.142 | 0.908 | 0.465 |
| LSTM EXOG | 8.25 | 1.748 | 0.064 | 2.872 | 1.337 | 0.888 | 0.461 |
| NORMAL BNN ENDOG | 6.822 | 1.556 | 0.057 | 2.612 | 1.157 | 0.908 | 0.467 |
| NORMAL BNN EXOG | 8.551 | 1.775 | 0.065 | 2.924 | 1.355 | 0.884 | 0.468 |
| PURE NORMAL BNN ENDOG | 13.064 | 2.265 | 0.085 | 3.614 | 1.826 | 0.823 | 0.456 |
| LAPLACE BNN ENDOG | 10.756 | 2.075 | 0.077 | 3.28 | 1.639 | 0.855 | 0.447 |
| SCALEMIXTURE BNN ENDOG | 9.216 | 2.035 | 0.077 | 3.036 | 1.584 | 0.875 | 0.471 |
| MC DROPOUT ENDOG | 8.026 | 1.587 | 0.057 | 2.833 | 1.183 | 0.891 | 0.465 |

Table 14: Evaluation of the models on the second test set. The best score for each metric is highlighted in bold text.

| | MSE | MAE | MAPE | RMSE | Huber | R2 | MDA |
|---|---|---|---|---|---|---|---|
| AR ENDOG | **4.387** | **1.189** | **0.055** | **2.094** | **0.818** | **0.941** | **0.479** |
| AR EXOG | 4.396 | 1.194 | **0.055** | 2.097 | 0.823 | **0.941** | 0.475 |
| LSTM ENDOG | 4.689 | 1.26 | 0.06 | 2.165 | 0.874 | 0.937 | 0.457 |
| LSTM EXOG | 5.512 | 1.366 | 0.063 | 2.348 | 0.982 | 0.926 | 0.459 |
| NORMAL BNN ENDOG | 4.588 | 1.219 | 0.056 | 2.142 | 0.848 | 0.938 | 0.466 |
| NORMAL BNN EXOG | 5.812 | 1.427 | 0.067 | 2.411 | 1.027 | 0.922 | 0.461 |
| PURE NORMAL BNN ENDOG | 8.807 | 1.831 | 0.088 | 2.968 | 1.402 | 0.882 | 0.447 |
| LAPLACE BNN ENDOG | 7.493 | 1.694 | 0.08 | 2.737 | 1.275 | 0.899 | 0.435 |
| SCALEMIXTURE BNN ENDOG | 6.42 | 1.665 | 0.081 | 2.534 | 1.229 | 0.914 | 0.469 |
| MC DROPOUT ENDOG | 5.182 | 1.275 | 0.06 | 2.276 | 0.884 | 0.93 | 0.475 |

Table 15: Evaluation metrics for the entire test set. The best score is highlighted in bold text for each metric.

| | AR ENDOG | LSTM ENDOG | NORMAL BNN ENDOG | NORMAL PURE BNN ENDOG | LAPLACE BNN ENDOG | SCALE MIXTURE BNN ENDOG | MC DROPOUT ENDOG |
|---|---|---|---|---|---|---|---|
| AR ENDOG | | -2.68 *** | -1.78 * | -6.84 *** | -6.42 *** | -9.1 *** | -1.17 |
| LSTM ENDOG | 2.68 *** | | 2.51 ** | -6.82 *** | -6.56 *** | -8.24 *** | 2.16 ** |
| NORMAL BNN ENDOG | 1.78 * | -2.51 ** | | -6.93 *** | -6.55 *** | -10.83 *** | 0.29 |
| NORMAL PURE BNN ENDOG | 6.84 *** | 6.82 *** | 6.93 *** | | 1.59 | 2.94 *** | 6.55 *** |
| LAPLACE BNN ENDOG | 6.42 *** | 6.56 *** | 6.55 *** | -1.59 | | 2.33 ** | 6.32 *** |
| SCALE MIXTURE BNN ENDOG | 9.1 *** | 8.24 *** | 10.83 *** | -2.94 *** | -2.33 ** | | 8.44 *** |
| MC DROPOUT ENDOG | 1.17 | -2.16 ** | -0.29 | -6.55 *** | -6.32 *** | -8.44 *** | |

Table 16: Diebold-Mariano test results for the first test set. A negative value indicates that the model in the row outperforms the model in the column. The stars shows whether the null hypothesis of equal predictive accuracy is rejected at the 10% (*), 5% (**) or 1% (***) level

| | AR ENDOG | LSTM ENDOG | NORMAL BNN ENDOG | PURE NORMAL BNN ENDOG | LAPLACE BNN ENDOG | SCALE MIXTURE BNN_ENDOG | MC_DROPOUT_ENDOG |
|---|---|---|---|---|---|---|---|
| AR ENDOG | | -1.76 * | -2.17 ** | -5.39 *** | -6.29 *** | -3.28 *** | -2.56 ** |
| LSTM ENDOG | 1.76 * | | 0.03 | -5.24 *** | -6.77 *** | -2.85 *** | -2.0 ** |
| NORMAL BNN ENDOG | 2.17 ** | -0.03 | | -4.91 *** | -5.51 *** | -3.22 *** | -1.74 * |
| PURE NORMAL BNN ENDOG | 5.39 *** | 5.24 *** | 4.91 *** | | 2.76 *** | 2.5 ** | 5.56 *** |
| LAPLACE BNN ENDOG | 6.29 *** | 6.77 *** | 5.51 *** | -2.76 *** | | 1.45 | 4.34 *** |
| SCALE MIXTURE BNN ENDOG | 3.28 *** | 2.85 *** | 3.22 *** | -2.5 ** | -1.45 | | 0.97 |
| MC DROPOUT ENDOG | 2.56 ** | 2.0 ** | 1.74 * | -5.56 *** | -4.34 *** | -0.97 | |

Table 17: Diebold-Mariano test results for the second test set. A negative value indicates that the model in the row outperforms the model in the column. The stars shows whether the null hypothesis of equal predictive accuracy is rejected at the 10% (*), 5% (**) or 1% (***) level

| | AR ENDOG | LSTM ENDOG | NORMAL BNN ENDOG | PURE NORMAL BNN ENDOG | LAPLACE BNN ENDOG | SCALE MIXTURE BNN ENDOG | MC DROPOUT ENDOG |
|---|---|---|---|---|---|---|---|
| AR ENDOG | | -2.8 *** | -2.55 ** | -6.96 *** | -8.33 *** | -4.81 *** | -2.64 *** |
| LSTM ENDOG | 2.8 *** | | 0.8 | -6.68 *** | -8.8 *** | -4.07 *** | -1.62 |
| NORMAL BNN ENDOG | 2.55 ** | -0.8 | | -6.41 *** | -7.52 *** | -4.86 *** | -1.71 * |
| PURE NORMAL BNN ENDOG | 6.96 *** | 6.68 *** | 6.41 *** | | 3.05 *** | 3.03 *** | 7.47 *** |
| LAPLACE BNN ENDOG | 8.33 *** | 8.8 *** | 7.52 *** | -3.05 *** | | 1.96 * | 6.63 *** |
| SCALE MIXTURE BNN ENDOG | 4.81 *** | 4.07 *** | 4.86 *** | -3.03 *** | -1.96 * | | 2.0 ** |
| MC DROPOUT ENDOG | 2.64 *** | 1.62 | 1.71 * | -7.47 *** | -6.63 *** | -2.0 ** | |

Table 18: Diebold-Mariano test results for the entire test set. A negative value indicates that the model in the row outperforms the model in the column. The stars shows whether the null hypothesis of equal predictive accuracy is rejected at the 10% (*), 5% (**) or 1% (***) level

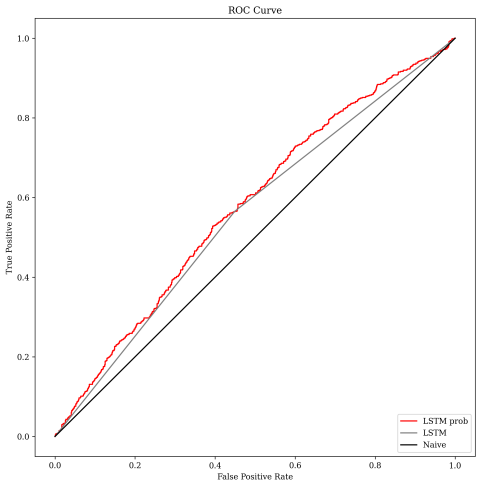
## 6.4 One-day-ahead Directional

The results from a one-day-ahead approach to predict the direction of the VIX can be seen in Table 19. We restrict the results to the conventional LSTM and the hybrid BNN with a normal prior, as the latter was the best-performing BNN model. We observe that both models have significantly higher accuracy compared to the mean directional accuracy of every regression model. This is however not surprising, as they are by design built to classify distinct classes using decision boundaries, which regression models do not in their aim to predict a continuous value.

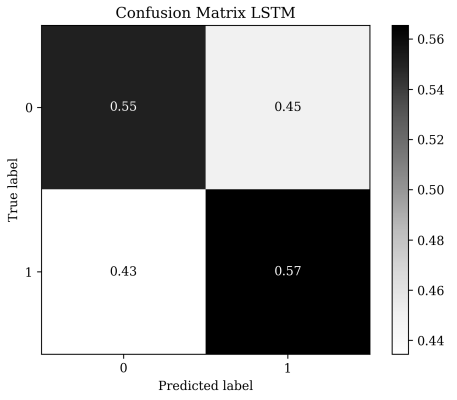
| Model | Accuracy | F1-score |
|---|---|---|
| LSTM | 0.557 | 0.524 |
| Normal BNN | 0.523 | 0.505 |

Table 19: Results on the entire test set for a conventional LSTM model and a hybrid BNN model with a normal prior trained on a binary directional target using previously lagged values and directions of the VIX.

From the ROC curves for the two models in Figure 11a and Figure 12a, we see that the conventional LSTM exhibits a larger area under the ROC curve, indicating higher predictive power than the hybrid BNN. We also see this from the balance of the confusion matrices in Figure 11b and Figure 12b. We have previously reported machine learning models being able to have accuracy and f1 score around 0.6, for the monthly direction of VIX (Vrontos et al., 2021). Despite there being some differences in the span of out of sample periods, we would expect our results to be lower. This is because monthly data, which they operated on, tends to have less noise and volatility, where daily stock market movements can be influenced by various intra-daily factors, making daily predictions more challenging. Next, monthly data give exogenous variables more time to be influential for the movements of the VIX, and potentially allows models to capture this long-term patterns. Lastly, daily data is more sensitive to sudden market events.

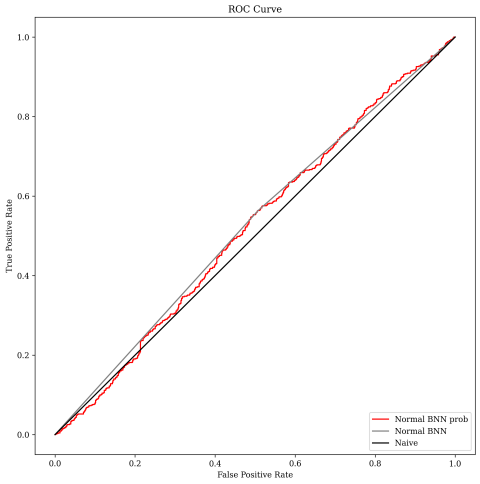



(a) ROC curve for the conventional LSTM model. The grey line indicates the optimal cutoff chosen by maximization of the Youden's J statistic on a validation set (Habibzadeh et al., 2016).

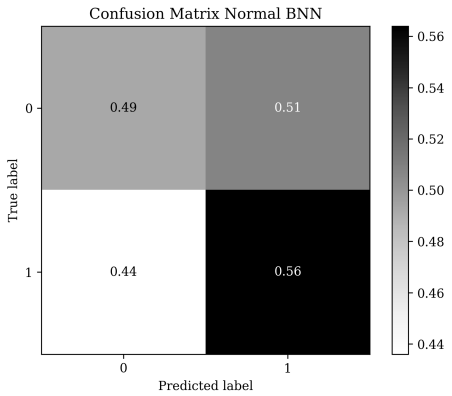



(b) Confusion matrix for the conventional LSTM model.

Figure 11




(a) ROC curve for the Normal BNN model. Similar as the conventional LSTM model, the cutoff on the grey line was selected using the Youden's J statistic (Habibzadeh et al., 2016).




(b) Confusion matrix for the Normal BNN model.

Figure 12

## 6.5 Uncertainty Quantification

In this section, we will assess a selection of the BNNs ability to quantify their underlying uncertainty pertaining to the resulting predictions. To limit the scope, we will focus on the models which performed the best; the endogenous hybrid Normal BNN model. The evaluation will involve the 1-day regression and directional forecasting exercises. Additionally, as the MC Dropout is an alternative method of performing Bayesian inference using neural network, this model will be included for comparison purposes for the regression problem. The evaluation of the uncertainty quantification of the regression forecast is performed by plotting a forecast with a confidence band obtained from the predictions, and visualizing the sharpness of the forecasts. Next, a reliability diagram is displayed for both the directional and regression forecasting exercise, which shows how well calibrated the models are in the context of being either overconfident, or underconfident

### One-day-ahead regression

An estimate of the epistemic uncertainty can be shown through the variability between the sampled prediction. As noted in section 5.4, we used 100 Monte Carlo samples to generate an ensemble of predictions. This allowed for extracting the standard deviation from the predicted values, giving an estimate of the spread between the predictions. Using the standard deviation, a visualization of the estimated epistemic uncertainty can therefore be done by plotting a confidence band around the mean predicted value. In Figure 13 and Figure 14, the mean forecast and a $\pm 2\sigma$ confidence band demonstrates the epistemic uncertainty obtained from the model. Figure 14 clearly displays a greater uncertainty around its predictions during and after the most extreme periods of the Covid pandemic. As the data gets more extreme, the model becomes more uncertain; a greater variability in the resulting predictions translates into greater epistemic uncertainty. During the time after Covid the uncertainty gradually decreases, reflecting that the model is more confident in its predictions. These results hints at an ability to display some quantification of the epistemic

uncertainty. Plotted forecasts of the other BNNs can be found in appendix G for reference.

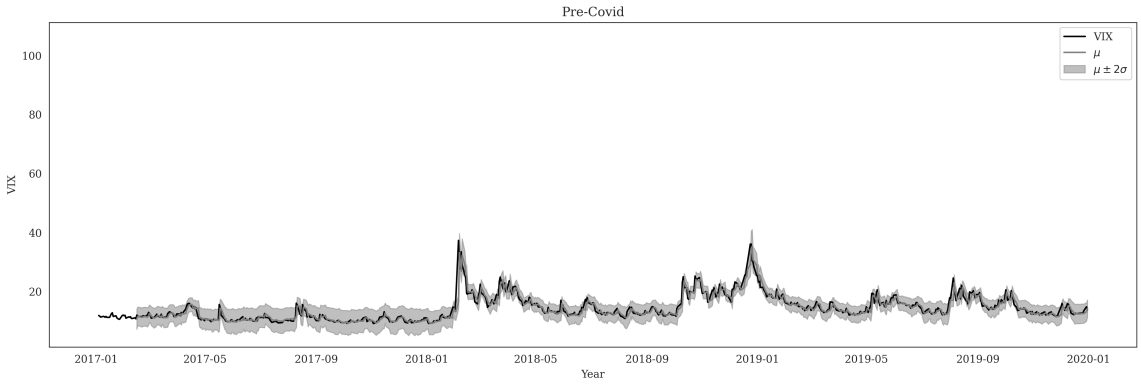



Figure 13: Pre-Covid test results from Normal BNN endogenous. A $\mu \pm 2\sigma$ confidence interval is displayed around the mean predictions.

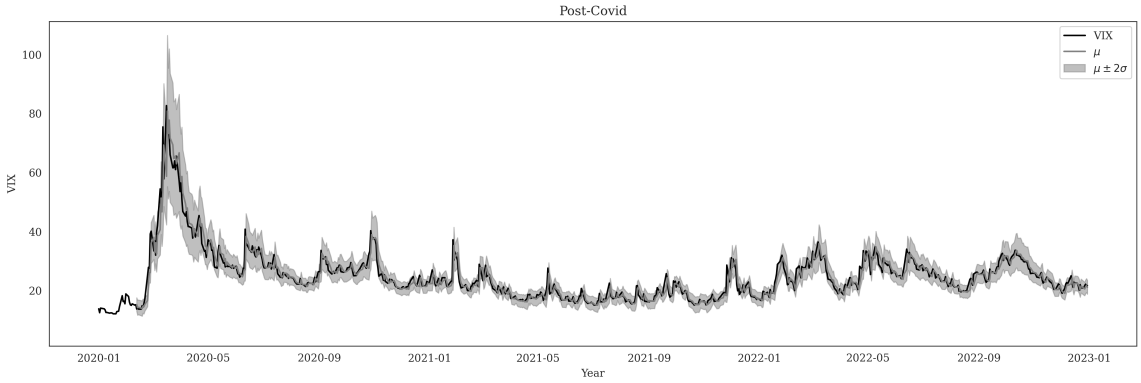
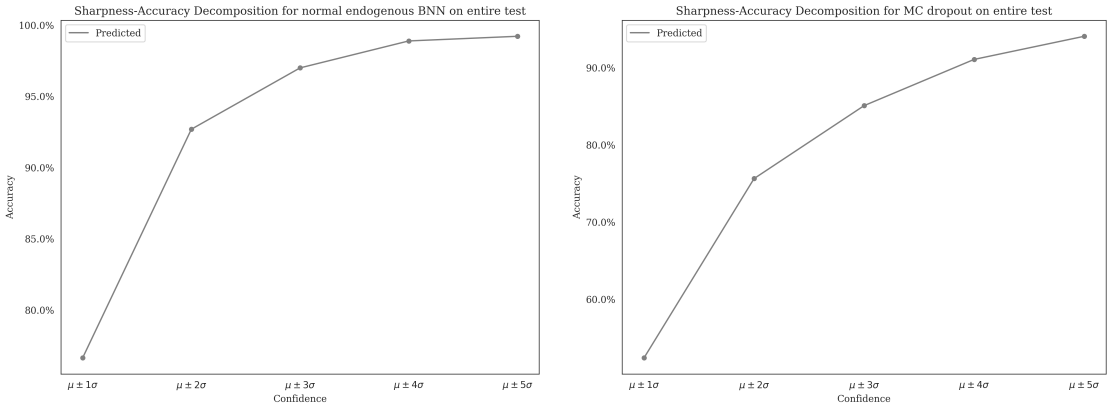



Figure 14: Post-Covid test results from the Normal BNN endogenous model. A $\mu \pm 2\sigma$ confidence interval is displayed around the mean predictions.

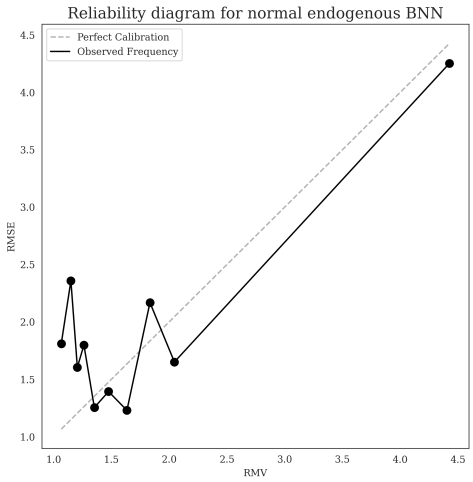
An important aspect of a forecaster is its ability to produce sharp predictions, i.e., the spread of the predicted distribution (Gneiting, Balabdaoui, & Raftery, 2007). Ideally, we want the model to produce sharp forecasts, while simultaneously exhibit the ability to signify when it is more uncertain. To measure the sharpness of the acquired forecasts, we can plot increments in the confidence bands against the number of true values which falls within the given confidence level. Figure 15a displays the resulting sharpness of the endogenous Normal BNN on the entire test. Circa 8% of the true values of the VIX falls outside of a $\mu \pm 2\sigma$-interval in the test set. For reference, Figure 15b displays the results for the MC Dropout model. This model is comparatively less sharp than the Normal BNN, caused by having significantly lower variability in its predictions.



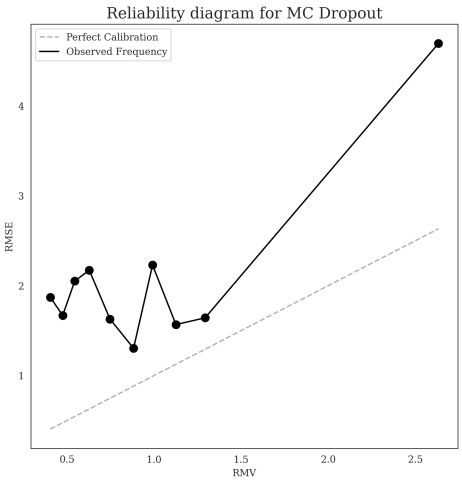

(a) (b)

Figure 15: The figures display the sharpness of the forecasts. The y-axis measures the percentage of the VIX which falls within the given confidence interval in terms of the entire test set.

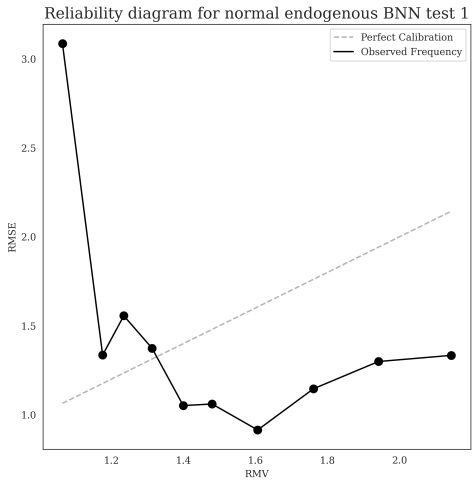
Another important aspect of evaluating uncertainty quantification concerns the calibration of the model. Following the procedure of Levi, Gispan, Giladi, and Fetaya (2022), a reliability diagram can be produced by binning the observed standard deviations into non-overlapping, increasing intervals. Then, the root mean variance in each bin is plotted against the root mean squared error of the corresponding forecasts. The resulting diagram for the endogenous Normal BNN is displayed in Figure 16a, contrasted against a perfect calibration line. For the first few bins, the model is underconfident. Then, as the standard deviation increases it grows generally more overconfident. Due to the large variance observed during Covid, there is a large gap between the last and next-to-last bin. This makes the model seem more calibrated than it is; inspecting the reliability diagram for the model on the first test in Figure 16c set confirms this. In general, the model fails to show an adequate degree of calibration. When comparing the Normal BNN against the MC Dropout model in Figure 16b, we can see that the latter is generally more underconfident. Additionally, it shows a greater deviation from the perfect calibration curve, indicating that the Normal BNN is somewhat better calibrated.




(a) The reliability diagram for the endogenous Normal BNN model on the entire test data.




(b) Reliability diagram for the MC Dropout model on the entire test data.




(c) Reliability diagram for the endogenous Normal BNN model on the first test set pre-Covid.

Figure 16: The figures displays a reliability diagram of the forecasted uncertainty for the endogenous Normal BNN and the MC Dropout model on the entire test data, and for the Normal BNN on the first test set. Each marked point corresponds to a binning of the standard deviations, wherein the root of the means of the variances are computed. The y-axis contains the RMSE of the forecasts belonging to each bin.

### One-day-ahead directional

In our binary classification models, the output from the last layer through the Sigmoid activation function corresponds to the probability of the example being the positive class. Consequently, this kind of output then already expresses uncertainty in it's predictions in the form of a probability. However, an effective uncertainty quantification cannot be guaranteed without assessing the model's uncertainty quantification. This can be done by examining reliability diagrams.

In Figure 17 we have the reliability diagrams from both the hybrid BNN with a normal prior and the conventional LSTM. There are several key observations to take from this diagram regarding their effectiveness in the uncertainty quantification. Most noticeably, the LSTM shows overconfidence, that is, it assigns higher probabilities to incorrect outcomes. Additionally it ex-

hibits a narrower prediction interval than what is justified by the data. This limited prediction spread also shows that the model assigns similar level of confidence or probability to different examples, thus, it will have a reduced ability to distinguish between the positive and negative class. Consequently, the model will have a restricted uncertainty quantification.

The Hybrid BNN on the other hand, exhibits both under-confidence and over-confidence. When a model is underconfident, it assigns lower probabilities to correct outcomes, or have wider prediction intervals than necessary, which further indicates that the model is not able to fully capture the true uncertainty in the data. We can see that for the lower predicted probabilities, the model is underconfident, and overconfident for the higher predicted probabilities. Compared to the conventional LSTM, the hybrid BNN have a larger prediction spread, allowing for more expressive uncertainty quantification. However, as one can see from the histogram, the number of examples falling in the high and low probabilities are low, which indicates that despite it's larger prediction spread, it still struggles to accurately quantify uncertainty. Abdar et al. (2021) state that Bayesian uncertainty estimations are often imprecise because of the use of approximate inference and model misspecifications. Such misspecification implies that the model is not able to capture the underlying data generation process. With that being said, the underlying data generation process of the VIX is highly complex.

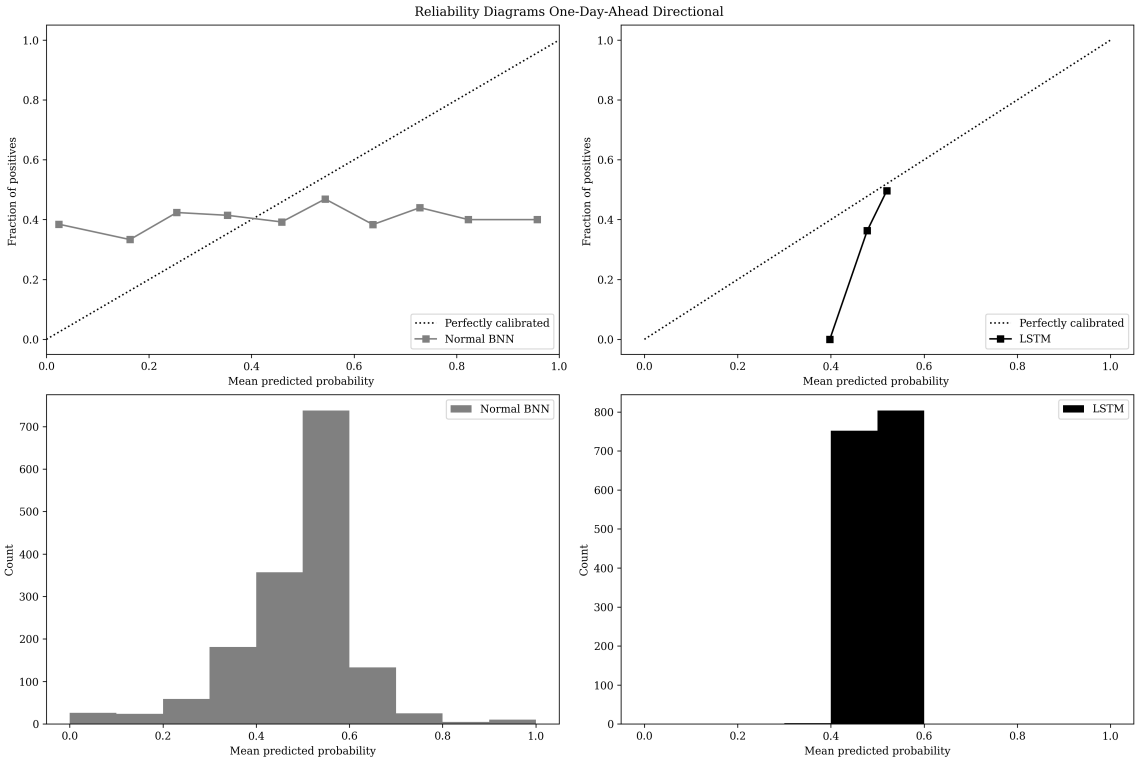



Figure 17: Reliability diagrams for directional models.

## 6.6 Additional Forecasting Horizons

The results from the additional forecasting horizons are presented below. These included a 5-day dynamic and a 20-day forecast evaluation, and models obtained using alternate window sizes. To limit the number of models, only results from the best-performing models from the 1-day forecasting exercise are included.

| | MSE | MAE | MAPE | RMSE | Huber | R2 | MDA |
|---|---|---|---|---|---|---|---|
| AR | **15.837** | **2.405** | **0.112** | **3.98** | **1.969** | **0.785** | **0.487** |
| LSTM | 20.725 | 2.648 | 0.117 | 4.552 | 2.218 | 0.719 | 0.456 |
| BNN | 34.686 | 3.708 | 0.155 | 5.889 | 3.257 | 0.529 | 0.467 |

Table 20: Results of performing 5-day ahead dynamic forecasting on the entire test set.

### 5-day dynamic forecasts

The resulting evaluation metrics for the 5-day dynamic forecasting exercise are reported in Table 20. Only the best performers on a 1-day basis are included: the AR, LSTM and Normal hybrid BNN models. Results from the first and second test set can be found in the appendix in Appendix H. The AR model outperformed all other models with respect to every metric. Most notably was the drastic drop in performance for the Normal BNN model, having more than two times the MSE compared to the AR. The reason behind the deteriorating BNN results can be better understood when investigating the post-Covid period. Similarly to the 1-day forecasting exercise, it appears to be more affected by the noisier period during and after Covid. However, in this implementation it is heavily amplified. These effects become clearer in Figure 18, where the period after Covid appears to make the network consistently forecast a value lower than the true VIX. We suspect the cause for the deteriorating performance are caused by BNNs sensitivity to the level of noise in the data. 1-day forecasts signified such an effect, albeit at a lower level. When the network has to make dynamic predictions, these disturbances may propagate through each forecast, causing increased errors. Additionally, despite the less accurate forecast the uncertainty of the network does not reflect the change in performance. These findings might indicate that the regularization effects provided by the BNN are overshadowed by the apparent sensitivity to noisy data.

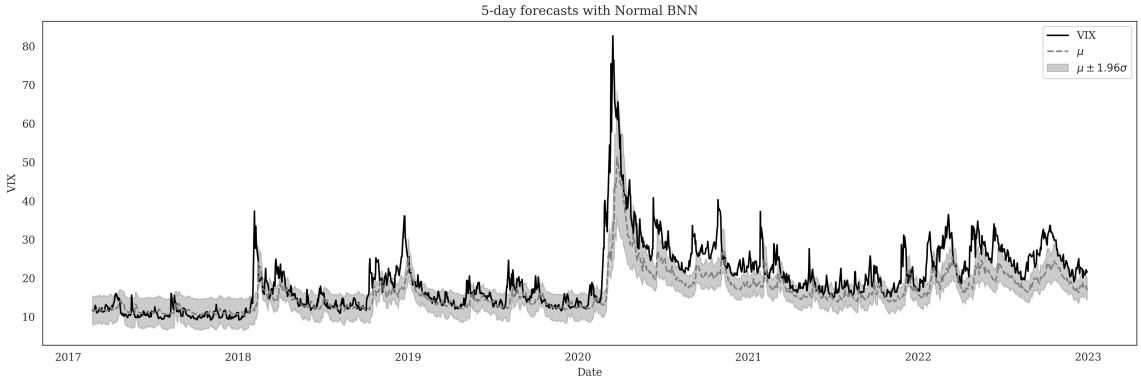



Figure 18: 5-day dynamic forecasts of the Normal BNN model against the VIX. A $\pm 1.96 \cdot \sigma$ confidence interval derived from an ensemble of predictions is plotted around the mean prediction.

### 20-day forecasts

The results of the 20-day forecasts on the entire test are shown in Table 21, while results for the two separate test sets can be found in section H. Again, the AR model showed the most efficacy for forecasting the VIX index, followed by the endogenous Normal BNN model. This aligns with the findings for the 1-day forecasts. However, in contrast to the other results the machine learning models performed better in the first test set. In particular, the exogenous LSTM

model outperformed all other models. The potential advantage caused by training to make 20-days predictions might be the reason behind this observation. Due to these models being more severely impacted by the second test set, they fell behind the AR model on the total test set.

| | MSE | MAE | MAPE | RMSE | Huber | R2 | MDA |
|---|---|---|---|---|---|---|---|
| AR | **49.38** | **4.139** | **0.19** | **7.027** | **3.665** | **0.33** | 0.498 |
| LSTM ENDOG | 53.162 | 4.253 | 0.193 | 7.291 | 3.783 | 0.279 | 0.477 |
| LSTM EXOG | 60.057 | 4.464 | 0.189 | 7.75 | 3.998 | 0.186 | **0.502** |
| Normal BNN ENDOG | 50.602 | 4.293 | 0.204 | 7.114 | 3.818 | 0.314 | 0.476 |
| Normal BNN EXOG | 58.602 | 4.611 | 0.214 | 7.655 | 4.143 | 0.205 | 0.485 |

Table 21: 20-day forecasts with the AR model, and the LSTM and normal hybrid BNN models with and without exogenous data. The results are reported on the entire test set.

#### Additional window sizes

The results using different window sizes are reported in appendix I. Overall, the findings for different window sizes aligns with the other results. For shorter windows, the endogenous LSTM proves to be better or at par with the endogenous Normal BNN. With window sizes of 45 and 60 days, the Normal BNN significantly outperforms the LSTM, likely due to the regularization effect. For the exogenous variants, a similar observations as with the 30-day windows can be made; the exogenous LSTM performs better than the Normal BNN.

## 6.7 Criticisms

There are numerous issues that comes with combining Bayesian Inference with deep learning. One problem with Variational Inference methods are their tendency to create unimodal posterior distributions, while in actuality the true posterior are often multimodular (Izmailov, Vikram, Hoffman, & Wilson, 2021; Jospin et al., 2022). Following the approach of Izmailov et al. (2021), Figure 19 visualizes the posterior and prior distribution of the endogenous Normal BNN model. The visualization was produced by sampling three sets of weights $w_{100}, w_{250}, w_{500}$ from the model. These were obtained during the training phase, where it had trained for 100, 250 and 500 epochs respectively. Then, a basis created through the three weights maps the weight landscape onto a 2-dimensional plane. Inspection of the log-posterior reveals a unimodular weight space. Furthermore, the similarities between the prior and posterior hints that the structure of the former is heavily influenced by the latter. The density of the sampled weights and their pairwise joint density is displayed in Figure 20 for reference, supporting the findings above.

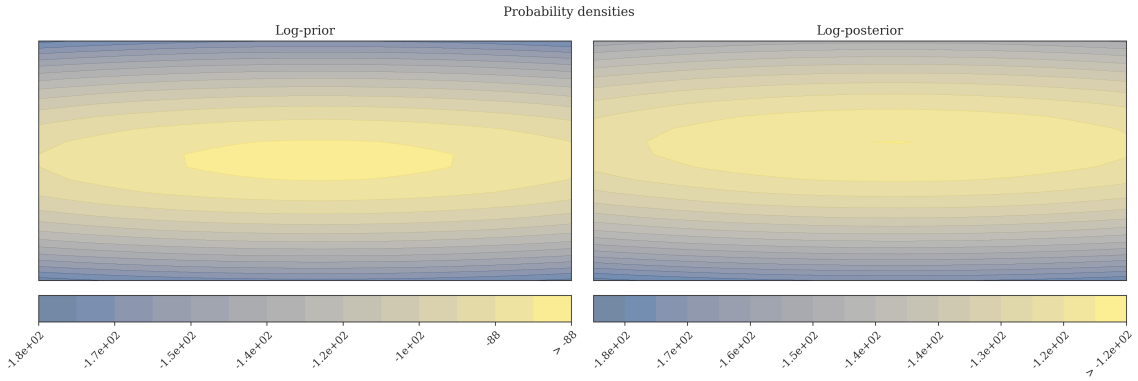



Figure 19: Log-probability densities of the prior (left) and posterior (right) created by constructing a basis of three weights sampled during different stages of training the endogenous Normal BNN model.

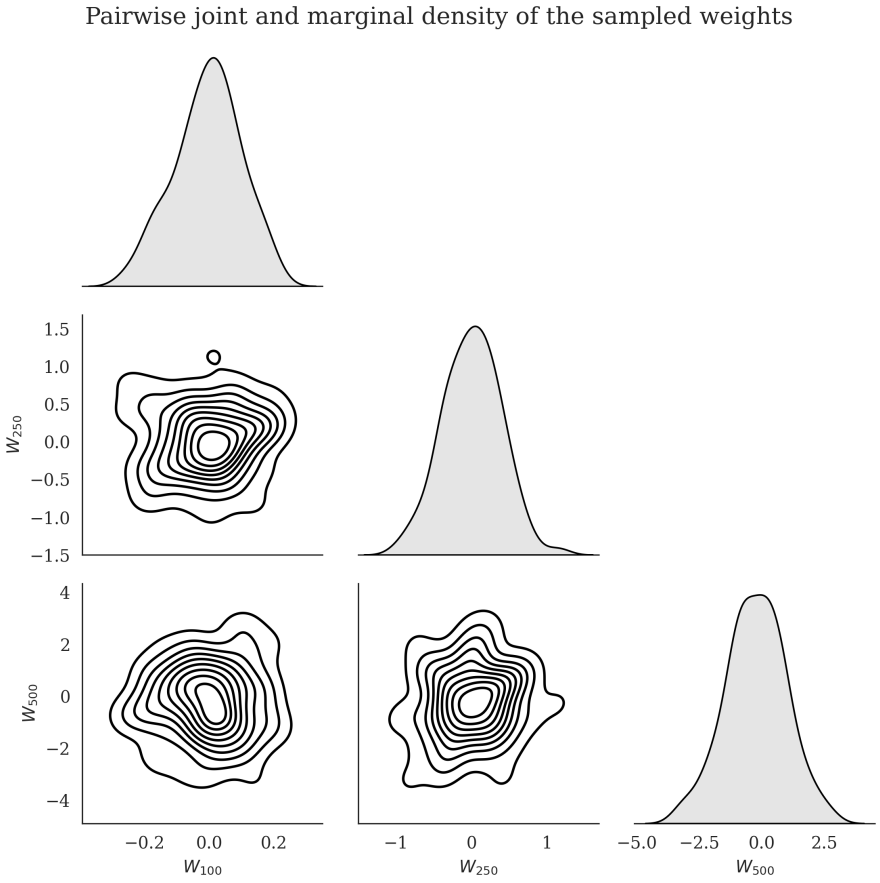



Figure 20: The pairwise joint density of each of the sampled weights are plotted on the lower diagonal. On the diagonal the marginal density of the sampled weight is plotted.

An issue we encountered in training BNNs was the instability in their performances. With some specifications of hyper-parameters, we could train the same architecture two times and achieve completely different outcomes. One cause of this may be due to not running more Monte Carlo samples in predictions or the gradient descent. Figure 21 shows how the parameterized surrogate posterior distribution changes over the training of the model. The figure uses one weight for the visualization, but a similar trend was observed in all other weights. While the mean changes in small increments around zero, the standard deviation continues to increase steadily for the duration of the training. The final value for the standard deviation was 1.7, thereby exceeding the

corresponding standard deviation of 0.7 in the prior distribution. As this parameter determines the spread of the posterior, a consequence of the rising variance is a higher degree of variability in the sampled weights. These effects might induce the need for performing more Monte Carlo samples on each iteration of gradient descent, as touched upon in section 5.3. However, utilizing more samples comes at the cost of increasing the computational requirements of the method.

Selecting prior and posterior distributions are a major issue with BNNs and variational inference. In particular, the choice of prior distribution is especially unintuitive; how do you incorporate prior belief about a network's weights? To make matters worse, this choice can significantly impact the estimation of the posterior distribution, as demonstrated in Figure 19. While we tried to compare different choice of prior distributions, the cardinality of the search space may have restricted a fuller exploration of their true effects. Specifically allowing for different rescaling factors between the models may have impacted the resulting model selection process in such a way that directly comparing the models became more difficult. The models with a higher rescaling factor allowed for greater variance in the surrogate distribution, and therefore had a greater magnitude of stochasticity clouding their performance. Furthermore, you have to decide on the surrogate posterior which introduces another sensitive element of opacity into the model building process. The sum of these parts can be a difficult and unclear modelling framework. In light of our indecisive results about the BNNs ability to model predictive uncertainty, this raises the question of whether these models are worth the extra degree of complexity. However, we used 100 trials when tuning each model. A potential reason for the varying results with respect to the prior may stem from the tuning procedure not being allowed to sufficiently explore the search space and being caught in local optima. The Laplace BNN had very similar hyperparameters amongst the top models, which could suggest that the tuner was caught in such an optima.

We compared a hybrid BNN against an LSTM for many of the forecasting exercises, where the hybrid model was tuned on a fixed LSTM architecture. The problems with this approach is twofold. It may introduce some advantage in favor of the hybrid model as it enjoys a pre-tuned architecture, while also hindering the ability to obtain any different beneficial LSTM architecture. Tuning both components of a hybrid models simultaneously may be the better approach. We did not have the time to tune such a model from scratch, and consequently cannot comment on the veracity of this method contra the other. For the purpose of a streamlined machine learning pipeline we believe that the better approach involves tuning the all elements of a hybrid model concurrently.

## 6.8 Future work

In our implementations we did not utilize trainable prior distributions. Blundell et al. (2015) recommend against using such methods, both on the grounds of its validity and its tendency to limit the optimization procedure for the posterior. While the use of a hybrid model can be viewed as a sort of trainable prior, the specific parameters of the prior distribution were not trainable. A simple extension would be to let the LSTM layers parameterize the prior distribution, and pretrain this on a separate batch of data. In Abdar et al. (2021), several other implementations of trainable priors are mentioned, laying the foundations for further research into the combination of such priors and forecasting financial time series data. Further, to simplify the tuning and evaluation procedure we opted to not model aleatoric uncertainty, and we did not model the correlation between weights. By having a distribution as the final layer of the model, the outputs of the

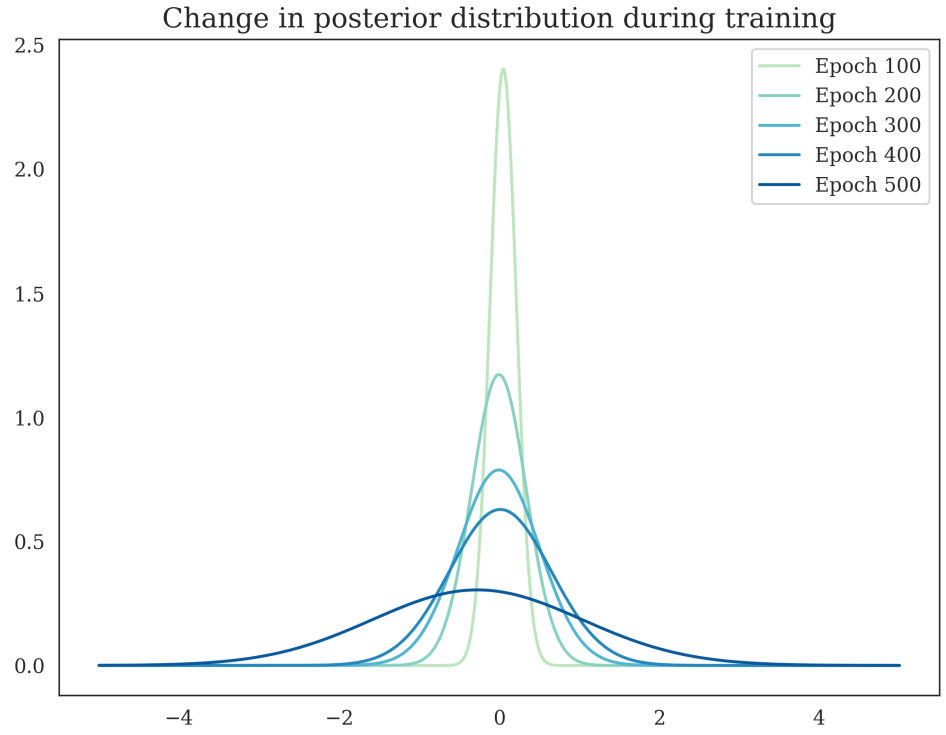



Figure 21: The figure shows the evolution of the parameterized normal surrogate distribution over the training period. To achieve the visualization the parameters for the first weight in the Dense Variational Layer is extracted.

second to last layer can parameterize this distribution. If combined with Dense Variational layers, this allows for modelling both the aleatoric and epistemic uncertainty. For the correlations, their inclusion increases the number of parameters substantially, causing us not to use them for our thesis. However, there may be some beneficial effects in including correlation between the weights, and future research could investigate this direction.

There are several methods for making Bayesian Inference. One popular alternative inference method is the Markov Chain Monte Carlo family of methods (Jospin et al., 2022). In our work we attempted to implement this, but due to the demanding computational requirements of such models and the already large selection of models to tune, we were not able to optimize the model to achieve convergence. Another promising avenue is the deep ensemble method for quantifying uncertainty, as introduced by Lakshminarayanan, Pritzel, and Blundell (2017). Abdar et al. (2021) recommend such approaches due to their ability to build on existing frameworks and methodologies within traditional deep learning, making the transition into uncertainty quantification less arduous. Furthermore, methods such as Laplacian approximations could also be implemented within a volatility forecasting framework.

Our main focus for this thesis lied in applying Bayesian Neural Networks to volatility forecasting. As such, to limit the scope of the project we did not focus on the selection of the exogenous data, and we generally did not include data which was not on a daily basis. Other applications of volatility forecasting has generally found exogenous data to assist in achieving better results, particularly on longer forecasting horizons (Christensen et al., 2022; Ghosh & Sanyal, 2021; Bucci,

2020). Hence, future research can focus on incorporating exogenous data to forecast volatility with Bayesian Neural Networks, and try to forecast the VIX for longer forecasting horizons. Another dimension to consider is using alternative forecasting regimes, namely rolling and recursive forecasts. These methods allows for inspecting model performances under different conditions in a more realistic setting, and ideally could be combined with practical applications. As such volatility models discussed in our work are difficult to evaluate using only statistical error measures, it is essential to evaluate them using practical financial evaluations to further assess the model's performances. Practical financial or economic evaluations include among other portfolio optimization simulations as well as simulating real world applications using trading strategies.

An important aspect of the modern machine learning literature concerns the issue of explainability. Deep learning models are notoriously known as black boxes, without much insight into the inferences made by the model. XAI has emerged as a popular methods for gaining an understanding into the inner workings of machine learning inferences (Vilone & Longo, 2020). While uncertainty quantification can be interpreted as a step towards more explainability, future research could try to combine elements of XAI with Bayesian Neural Networks for financial time series forecasting.

# 7 Conclusions

In this thesis, we explore the Bayesian regime for uncertainty quantification in deep learning to quantify uncertainty in model predictions of the CBOE VIX. More specifically, we utilize both deterministic and Bayesian deep learning models in an end-to-end framework to assess their efficacy in forecasting the VIX. We combine different approaches for implementing Bayesian inference in Bayesian neural networks, namely Bayes-By-Backprop and MC dropout, to investigate how this affects their performance. Furthermore, we check the potential of combining a LSTM with Bayesian deep learning through a hybrid model. We test three different prior distributions for the hybrid model to assess how it affects the model's performance, and we investigate the effect of including exogenous data into the model framework. To benchmark the machine learning models we apply an Autoregressive model. Finally, we test the models' performances in different forecasting regimes to fully explore their ability to forecast the VIX index, as well as the effectiveness of the uncertainty quantification.

We find that the Autoregressive model remains hard to beat for forecasting the VIX, outperforming all other models in nearly every specified forecasting exercise. A hybrid Bayes-By-Backprop model with a normal prior and posterior distribution generally ranks as the second-best model, suggesting that the regularization effect provided by the hybrid Bayesian Model can aid traditional deep learning methods in combating their tendency to overfit. Additionally, the pure Bayesian deep learning model struggles to adequately capture the intricacies of the VIX, further motivating the use of hybrid models. However, the Bayesian models appears to be sensitive to more turbulent and noisy data, and when applied to a dynamic forecasting regime they were not able to outperform their deterministic counterpart.

By inspecting a reliability diagram of the models' quantified predictive uncertainty, we have found that the Bayesian models generally appears to be uncalibrated. Furthermore, they are highly sensitive to the choice of hyper-parameters, priors and posteriors, and they can often yield unstable

performances making them difficult to implement in an end-to-end framework. Additionally, the mode collapse observed when using variational inference suggests that other methods are needed to fully capture the epistemic uncertainty.

We have identified some areas of research that could be explored in future work to address some of the observed shortcomings and gaps within our implementations. First, using empirical trainable prior distributions could overcome some of the difficulties in specifying these priors beforehand, and allow for more direct interactions between the deterministic and hybrid model. All of our models could be extended to provide an estimate of aleatoric uncertainty, thereby making them capture the predictive uncertainty in a more holistic manner. Furthermore, using other methods of Bayesian inference such as Markov Chain Monte Carlo in forecasting volatility may give more accurate posterior distribution estimates. We suggest that future applications of volatility forecasting incorporate a richer set of exogenous data, and apply them in a wider set of forecasting horizons and methods. Finally, we recommend that XAI and practical evaluations should be incorporated into the Bayesian deep learning volatility forecasting framework.

# References


Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., ... Zheng, X. (2015). *TensorFlow: Large-scale machine learning on heterogeneous systems.* Retrieved from `https://www.tensorflow.org/` (Software available from tensorflow.org)

Abdar, M., Pourpanah, F., Hussain, S., Rezazadegan, D., Liu, L., Ghavamzadeh, M., ... Nahavandi, S. (2021). A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion*, *76*, 243-297. Retrieved from `https://www.sciencedirect.com/science/article/pii/S1566253521001081` doi: https://doi.org/10.1016/j.inffus.2021.05.008

Back, A., & Keith, W. (2019). *Bayesian neural networks for financial asset forecasting.* KTH Royal Institute of Technology. Retrieved from `https://www.diva-portal.org/smash/get/diva2:1320142/FULLTEXT01.pdf`

Bergmeir, C., & Benítez, J. M. (2012). On the use of cross-validation for time series predictor evaluation. *Information Sciences*, *191*, 192-213. Retrieved from `https://www.sciencedirect.com/science/article/pii/S0020025511006773` (Data Mining for Software Trustworthiness) doi: https://doi.org/10.1016/j.ins.2011.12.028

Blundell, C., Cornebise, J., Kavukcuoglu, K., & Wierstra, D. (2015). *Weight uncertainty in neural networks.*

Bollerslev, T. (1986, April). Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, *31*(3), 307-327. Retrieved from `https://ideas.repec.org/a/eee/econom/v31y1986i3p307-327.html`

Breiman, L. (2001). *Machine Learning*, *45*(1), 5–32. Retrieved from `https://doi.org/10.1023/a:1010933404324` doi: 10.1023/a:1010933404324

Brooks, C. (2019). *Introductory econometrics for finance* (4th ed.). Cambridge University Press. doi: 10.1017/9781108524872

Bucci, A. (2020, 06). Realized Volatility Forecasting with Neural Networks. *Journal of Financial Econometrics*, *18*(3), 502-531. Retrieved from `https://doi.org/10.1093/jjfinec/nbaa008` doi: 10.1093/jjfinec/nbaa008

Chandra, R., & He, Y. (2021, 07). Bayesian neural networks for stock price forecasting before and during covid-19 pandemic. *PloS one*, *16*, e0253217. doi: 10.1371/journal.pone.0253217

Chandrashekar, G., & Sahin, F. (2014, jan). A survey on feature selection methods. *Computers & Electrical Engineering*, *40*(1), 16–28. Retrieved from `https://doi.org/10.1016j.compeleceng.2013.11.024` doi: 10.1016/j.compeleceng.2013.11.024

Chicago Board Options Exchange. (2022). Volatility index methodology: Cboe volatility index. Retrieved 2022-11-15, from `https://cdn.cboe.com/api/global/us_indices/governance/Volatility_Index_Methodology_Cboe_Volatility_Index.pdf`

Christensen, K., Siggaard, M., & Veliyev, B. (2022, jun). A machine learning approach to volatility forecasting. *Journal of Financial Econometrics*. Retrieved from `https://doi.org/10.1093/jjfinec/nbac020` doi: 10.1093/jjfinec/nbac020

Clevert, D.-A., Unterthiner, T., & Hochreiter, S. (2016). *Fast and accurate deep network learning by exponential linear units (elus).*

Daniel, F. (2019). *Financial time series data processing for machine learning.*

Demeterfi, K., Derman, E., Kamal, M., & Zou, J. (1999, may 31). A guide to volatility and variance swaps. *The Journal of Derivatives 1999-may 31 vol. 6 iss. 4*, *6*. Retrieved from `https://doi.org/10.3905/jod.1999.319129` doi: 10.3905/jod.1999.319129

Diebold, F. X., & Mariano, R. S. (2002). Comparing predictive accuracy. *Journal of Business Economic Statistics*, *20*(1), 134–144. Retrieved 2023-06-07, from `http://www.jstor.org/stable/1392155`

Engle, R. F. (1982, jul). Autoregressive conditional heteroscedasticity with estimates of the variance of united kingdom inflation. *Econometrica*, *50*(4), 987. Retrieved from `https://doi.org/10.2307/1912773` doi: 10.2307/1912773

Engle, R. F., Ghysels, E., & Sohn, B. (2013, jul). Stock market volatility and macroeconomic fundamentals. *Review of Economics and Statistics*, *95*(3), 776–797. Retrieved from `https://doi.org/10.1162/rest_a_00300` doi: 10.1162/rest_a_00300

Fortuin, V. (2022, 05). Priors in bayesian deep learning: A review. *International Statistical Review*, *90*. doi: 10.1111/insr.12502

Frazier, P. I. (2018). *A tutorial on bayesian optimization.*

Gal, Y., & Ghahramani, Z. (2016). *Dropout as a bayesian approximation: Representing model uncertainty in deep learning.*

Ghosh, I., & Sanyal, M. K. (2021). Introspecting predictability of market fear in indian context during covid-19 pandemic: An integrated approach of applied predictive modelling and explainable ai. *International Journal of Information Management Data Insights*, *1*(2), 100039. Retrieved from https://www.sciencedirect.com/science/article/pii/S266709682100032X doi: https://doi.org/10.1016/j.jjimei.2021.100039

Glosten, L. R., Jagannathan, R., & Runkle, D. E. (1993, dec). On the relation between the expected value and the volatility of the nominal excess return on stocks. *The Journal of Finance*, *48*(5), 1779–1801. Retrieved from https://doi.org/10.1111/j.1540-6261.1993.tb05128.x doi: 10.1111/j.1540-6261.1993.tb05128.x

Gneiting, T., Balabdaoui, F., & Raftery, A. E. (2007, 03). Probabilistic Forecasts, Calibration and Sharpness. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, *69*(2), 243-268. Retrieved from https://doi.org/10.1111/j.1467-9868.2007.00587.x doi: 10.1111/j.1467-9868.2007.00587.x

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning.* MIT Press. (http://www.deeplearningbook.org)

Gunnarsson, E. S., Isern, H. R., Risstad, M., Vigdel, B., & Westgaard, S. (2023). *Prediction of realized volatility and implied volatility indices using ai and machine learning: A review.* (Manuscript submitted for publication)

Habibzadeh, F., Habibzadeh, P., & Yadollahie, M. (2016). On determining the most appropriate test cut-off value: the case of tests with continuous results. *Biochem Med*, 297–307. Retrieved from https://doi.org/10.11613bm.2016.034 doi: 10.11613/bm.2016.034

Hochreiter, S., & Schmidhuber, J. (1997, 12). Long short-term memory. *Neural computation*, *9*, 1735-80. doi: 10.1162/neco.1997.9.8.1735

Homola, D. (2019). Retrieved from https://pypi.org/project/Boruta/

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, *2*(5), 359-366. Retrieved from https://www.sciencedirect.com/science/article/pii/0893608089900208 doi: https://doi.org/10.1016/0893-6080(89)90020-8

Izmailov, P., Vikram, S., Hoffman, M. D., & Wilson, A. G. (2021). *What are bayesian neural network posteriors really like?*

Jospin, L. V., Laga, H., Boussaid, F., Buntine, W., & Bennamoun, M. (2022). Hands-on bayesian neural networks—a tutorial for deep learning users. *IEEE Computational Intelligence Magazine*, *17*(2), 29-48. doi: 10.1109/MCI.2022.3155327

Kingma, D. P., & Ba, J. (2017). *Adam: A method for stochastic optimization.*

Kursa, M. B., & Rudnicki, W. R. (2010). Feature selection with the boruta package. *Journal of Statistical Software*, *36*(11), 1–13. Retrieved from https://www.jstatsoft.org/index.php/jss/article/view/v036i11 doi: 10.18637/jss.v036.i11

Lakshminarayanan, B., Pritzel, A., & Blundell, C. (2017). *Simple and scalable predictive uncertainty estimation using deep ensembles.*

Levi, D., Gispan, L., Giladi, N., & Fetaya, E. (2022). Evaluating and calibrating uncertainty prediction in regression tasks. *Sensors*, *22*(15). Retrieved from https://www.mdpi.com/1424-8220/22/15/5540 doi: 10.3390/s22155540

MacKay, D. J. C. (1992, 05). A Practical Bayesian Framework for Backpropagation Networks. *Neural Computation*, *4*(3), 448-472. Retrieved from https://doi.org/10.1162/neco.1992.4.3.448 doi: 10.1162/neco.1992.4.3.448

Malliaris, M., & Salchenberger, L. (1996). Using neural networks to forecast the sp 100 implied volatility. *Neurocomputing*, *10*(2), 183-195. Retrieved from https://www.sciencedirect.com/science/article/pii/0925231295000194 (Financial Applications, Part I) doi: https://doi.org/10.1016/0925-2312(95)00019-4

Neal, R. M. (1995). Bayesian Learning for Neural Networks.
(Ph. D. Thesis)

Nelson, D. B. (1991, mar). Conditional heteroskedasticity in asset returns: A new approach.

*Econometrica*, *59*(2), 347. Retrieved from https://doi.org/10.2307/2938260 doi: 10.2307/2938260

Oliveira, N., Cortez, P., & Areal, N. (2017, may). The impact of microblogging data for stock market prediction: Using twitter to predict returns, volatility, trading volume and survey sentiment indices. *Expert Systems with Applications*, *73*, 125–144. Retrieved from https://doi.org/10.1016/j.eswa.2016.12.036 doi: 10.1016/j.eswa.2016.12.036

O'Malley, T., Bursztein, E., Long, J., Chollet, F., Jin, H., Invernizzi, L., et al. (2019). *Keras Tuner*. https://github.com/keras-team/keras-tuner.

Osterrieder, J., Kucharczyk, D., Rudolf, S., & Wittwer, D. (2020, aug). Neural networks and arbitrage in the VIX. *Digit Finance*, *2*(1-2), 97–115. Retrieved from https://doi.org/10.1007/s42521-020-00026-y doi: 10.1007/s42521-020-00026-y

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Perktold, J., Seabold, S., Sheppard, K., ChadFulton, Shedden, K., jbrockmendel, ... Halchenko, Y. (2023, May). *statsmodels/statsmodels: Release 0.14.0*. Zenodo. Retrieved from https://doi.org/10.5281/zenodo.7899735 doi: 10.5281/zenodo.7899735

Petrozziello, A., Troiano, L., Serra, A., Jordanov, I., Storti, G., Tagliaferri, R., & Rocca, M. L. (2022, jul). Deep learning for volatility forecasting in asset management. *Soft Comput*, *26*(17), 8553–8574. Retrieved from https://doi.org/10.1007s00500-022-07161-1 doi: 10.1007/s00500-022-07161-1

Prasad, A., Bakhshi, P., & Seetharaman, A. (2022). The impact of the u.s. macroeconomic variables on the cboe vix index. *Journal of Risk and Financial Management*, *15*(3). Retrieved from https://www.mdpi.com/1911-8074/15/3/126 doi: 10.3390/jrfm15030126

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986, oct). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–536. Retrieved from https://doi.org/10.1038323533a0 doi: 10.1038/323533a0

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*(56), 1929–1958. Retrieved from http://jmlr.org/papers/v15/srivastava14a.html

Valdenegro-Toro, M., & Mori, D. S. (2022, jun). A deeper look into aleatoric and epistemic uncertainty disentanglement. In *2022 IEEE/CVF conference on computer vision and pattern recognition workshops (CVPRW)*. IEEE. Retrieved from https://doi.org/10.1109/cvprw56347.2022.00157 doi: 10.1109/cvprw56347.2022.00157

Vandal, T., Livingston, M., Piho, C., & Zimmerman, S. (2018, 24–25 Oct). Prediction and uncertainty quantification of daily airport flight delays. In C. Hardgrove, L. Dorard, & K. Thompson (Eds.), *Proceedings of the 4th international conference on predictive applications and apis* (Vol. 82, pp. 45–51). PMLR. Retrieved from https://proceedings.mlr.press/v82/vandal18a.html

Vilone, G., & Longo, L. (2020). *Explainable artificial intelligence: a systematic review.*

Vrontos, S. D., Galakis, J., & Vrontos, I. D. (2021). Implied volatility directional forecasting: a machine learning approach. *Quantitative Finance*, *21*(10), 1687-1706. Retrieved from https://doi.org/10.1080/14697688.2021.1905869 doi: 10.1080/14697688.2021.1905869

Wenzel, F., Roth, K., Veeling, B. S., Światkowski, J., Tran, L., Mandt, S., ... Nowozin, S. (2020). *How good is the bayes posterior in deep neural networks really?* doi: 10.48550/arXiv.2002.02405

Zang, R. (2018). Omphalos, uber's parallel and language-extensible time series backtesting tool. Retrieved from https://www.uber.com/en-NO/blog/omphalos/

# Appendix

## A Stationarity Tests Cross-Validation Folds

The following table shows the results of an Adjusted Dickey-Fuller test for stationarity performed on each of the cross-validation folds. We report the test statistic and corresponding p-value for each fold. Additionally, we provide the statistic signifying the given significance level in the three final rows.

| CV Fold | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| ADF Statistic | -3.370 | -3.299 | -3.118 | -5.0316 | -5.266 |
| p-value | 0.012 | 0.0149 | 0.025 | 0.00002 | 0.000006 |
| 1% | -3.433 | -3.433 | -3.433 | -3.433 | -3.433 |
| 5% | -2.863 | -2.863 | -2.863 | -2.863 | 2.863 |
| 10% | -2.567 | 2.567 | -2.567 | -2.567 | -2.567 |

Table 22: Adjusted Dickey-Fuller test for stationarity within each cross-validation fold. Each fold is stationary at a 5% significance level or less.

## B Results from Feature Selection

The table presents the features selected for a 20-day forecasting horizon using cross-valiation and the Boruta algorithm.

| Variable name | Description | Type |
|---|---|---|
| VIX | CBOE VIX | Index |
| S&P 500_R | Standard and Poor's 500 | Macro |
| T10Y3M_R | 10-Year Treasury Constant Maturity Minus 3-Month Treasury Constant Maturity | Macro |
| BAA10Y_R | Moody's Seasoned Baa Corporate Bond Yield Relative to Yield on 10-year Treasury Constant Maturity | Macro |
| DGS10_R | Market Yield on U.S. Treasury Securities at 10-Year Constant Maturity, Quoted on an Investment Basis | Macro |
| BAMLC0A1CAAA_R | ICE BofA AAA U.S Corporate INDEX option-adjusted spread | Macro |
| DFII5_R | Market Yield on U.S. Treasury Securities at 5-Year Constant Maturity, Quoted on an Investment Basis, Inflation-Indexed | Macro |
| T5YIE | 5-Year Breakeven Inflation Rate | Macro |
| EFFR_R | Effective Federal Funds Rate | Macro |
| Neutral | Neutral sentiment | Sentiment |
| Bullish 8-week Mov Avg | Bullish 8-week Mov Avg | Sentiment |
| MA_10 | VIX moving average 10 days | Techical |
| RSI_10 | VIX RSI 10 days | Techical |

Table 23: Selected variables for 20-days-ahead predictions. The _R suffix indicates that the variable was made stationary by transforming it into the log-return.

# C Descriptive Statistics of Feature Selected Data

We report descriptive statistics for each of the selected datasets. Table 24 contains the dataset for 1-day forecasts, while Table 25 contains the the 20-day forecasting data. For both tables, the _R suffix indicates features in which the Adjusted Dickey-Fuller test did not reject the null hypothesis of a unit root at a 1% significance level, and were therefore transformed by taking the log-returns.

| | Mean | Standard deviation | Minimum | 25% | 50% | 75% | Maximum |
|---|---|---|---|---|---|---|---|
| VIX | 19.363 | 8.788 | 9.14 | 13.44 | 16.9 | 22.5 | 82.69 |
| S&P500_R | 0.693 | 0.006 | 0.631 | 0.691 | 0.693 | 0.696 | 0.749 |
| BAMLC0A0CM_R | 0.693 | 0.006 | 0.654 | 0.69 | 0.693 | 0.695 | 0.816 |
| BAMLC0A0CMEY_R | 0.693 | 0.006 | 0.656 | 0.69 | 0.693 | 0.696 | 0.762 |
| MA_5 | 19.364 | 8.661 | 9.376 | 13.46 | 17.02 | 22.477 | 74.618 |
| EMA_5 | 19.364 | 8.618 | 9.467 | 13.524 | 16.977 | 22.475 | 72.788 |
| MOM_10 | -0.007 | 4.396 | -25.61 | -1.88 | -0.22 | 1.41 | 49.27 |
| BOLUP_5 | 21.686 | 10.4 | 9.663 | 14.727 | 18.942 | 25.149 | 94.253 |

Table 24: The table reports some descriptive statistics about the selected dataset for 1-day forecasts. Percentages are the respective quantiles, and the _R suffix are added to features which were made stationary by taking the log-returns.

| | Mean | Standard deviation | Minimum | 25% | 50% | 75% | Maximum |
|---|---|---|---|---|---|---|---|
| VIX | 19.363 | 8.788 | 9.14 | 13.44 | 16.9 | 22.5 | 82.69 |
| S&P500_R | 0.693 | 0.006 | 0.631 | 0.691 | 0.694 | 0.696 | 0.749 |
| T10Y3M_R | 0.693 | 0.019 | 0.46 | 0.687 | 0.693 | 0.7 | 0.993 |
| BAA10Y_R | 0.693 | 0.006 | 0.662 | 0.69 | 0.693 | 0.696 | 0.765 |
| DGS10_R | 0.693 | 0.013 | 0.548 | 0.687 | 0.693 | 0.699 | 0.879 |
| BAMLC0A1CAAA_R | 0.693 | 0.011 | 0.458 | 0.687 | 0.693 | 0.696 | 0.937 |
| DFII5_R | 0.693 | 0.024 | 0.482 | 0.686 | 0.693 | 0.7 | 1.072 |
| T5YIE | 1.9 | 0.589 | -2.24 | 1.59 | 1.91 | 2.31 | 3.59 |
| EFFR_R | 0.694 | 0.047 | 0.205 | 0.693 | 0.693 | 0.693 | 1.634 |
| Neutral | 0.292 | 0.075 | 0.077 | 0.237 | 0.29 | 0.341 | 0.529 |
| Bullish 8-week Mov Avg | 0.377 | 0.076 | 0.223 | 0.323 | 0.371 | 0.428 | 0.645 |
| MA_10 | 19.366 | 8.561 | 9.631 | 13.528 | 17.09 | 22.434 | 69.36 |
| RSI_10 | 47.828 | 16.822 | 0.0 | 36.736 | 48.4 | 59.417 | 97.619 |

Table 25: Descriptive statistics of the dataset for the 20-day forecasting horizon. The _R suffix indicates features which were made stationary by taking log-returns.

# D Alternative Feature Selection Using Elastic Net

To check the robustness of the feature selection using the Boruta algorithm we implemented an additional selection using Elastic Net for 1-day forecasts, along the lines of Vrontos et al. (2021). We report the results in Table 26. The Elastic Net was implemented using Scikit-Learn's ElasticNet (Pedregosa et al., 2011). We tuned two parameters using the cross-validation procedure: $\alpha$ and the $L_1 - ratio$. Then, we applied the network to the training set and recorded all parameters with coefficients different from zero. The results generally coincides with the ones found using the Boruta algorithm, albeit with some additional features. Most notably is the exclusion of the S&P 500 returns, which may indicate some non-linearity not captured by the Elastic Net.

| Variable name | Description | Type |
|---|---|---|
| VIX | CBOE VIX | Index |
| BAMLC0A4CBBBEY_R | ICE BofA BBB U.S. Corporate Index effective yield | Macro |
| BAMLC0A0CMEY_R | ICE BofA U.S. Corporate Index effective yield | Macro |
| USEPUINDXD | Economic Policy Uncertainty Index for United States | Index |
| T5YIE | 5-Year Breakeven Inflation Rate | Macro |
| SMB | Small Minus Big | Factors |
| Neutral | AAII Sentiment Survey Neutral | Sentiment |
| Bearish | AAII Sentiment Survey Bearish | Sentiment |
| MA_10 | VIX moving average 10 days | Technical |
| EMA_5 | VIX exponentially moving average 5 days | Technical |
| BOL_UP_10 | VIX Bollinger upper band 10 days | Technical |
| RSI_5 | VIX RSI 5 days | Technical |

Table 26: The selected features using an Elastic Net feature selection procedure. Features marked with the _R suffix are made stationary by taking the log-returns.

# E Tuning Results of the BNNs

In the following section we will provide more detailed descriptions of the results from the tuning procedure for each of the BNNs.

### MC Dropout

The MC dropout was almost identical to the LSTM networks, except for the addition of a Dropout layer between the two intermediate LSTM layers. Out of the ten best performing models, there were two main specifications of models present. Each category used a variation of the same hyper-parameters, only varying slightly with respect to the learning rate. Eight of these used a variation of what would be our selected hyper-parameters, reported in Table 9. The other two used a dropout of 0.1 for both layers. The chosen specification performed best with regard to the MSE and Huber metrics, and comparatively on the MAE errors. Thus, we opted to use the best ranked model as the blueprint for the hyper-parameters.

### Pure BNN

The Pure BNN model did not have the luxury of previous LSTM layers to aid in capturing time-series dependencies, making this an interesting exercise for juxtaposing against the hybrid models. Out of all tuning trials, it had one of the greater variations amongst the top ten results. These included every possible subset of layer architecture, each with a rich selection of neurons. The best performing model outperformed all others by a significant margin; the mean metric from the folds were over two times better than the second best model for both the MSE, MAE and Huber metric. The resulting hyper-parameters are reported in Table 10.

### Normal and Laplace BNNs

The Normal and Laplace prior BNNs all used the same set of tuning hyper-parameters, so their results are combined into Table 11. The results for the exogenous Normal BNN trials showed a clear preference for the same set of hyper-parameters. In the top ten results, the main differences can be found in small fluctuations in the prior $\sigma$, posterior rescaling factor and the learning rate. These generally differed by around 0.1. The best model beat all the others with a clear margin, making for an easy choice of final model.

Amongst the top ten performers of The Laplace models, the chosen hyper-parameters were practically identical. Their only difference laid in the learning rate, which differed by small amounts. Consequently, the results were mainly homogeneous. Of note is the difference between this and the Normal BNN models. It opted for a lower learning rate and higher $\sigma$ for the prior distribution. Furthermore, the posterior rescaling factor was significantly larger.

### Scale Mixture Gaussian

The scale mixture Gaussian model tuning trials produced quite varied results. Amongst the top ten trials, every possible specifications of the weight component was present. Generally, there appeared to be a preference for a higher standard deviation on the first distribution component, with the majority being between 0.5 and up. For the second distribtuion, the results were mainly focused around 0.00075, with some going as high as 0.001. None of the top ten models included the hidden dense variational inference layer, aligning with the findings from the other BNN tunings. The posterior rescaling factor lied within 0.08 and 0.03, and the learning rate was mainly centered around 0.001. The error metrics unanimously declared one model the sole winner by a clear margin, with the specifications reported in Table 12. It mainly differed with respect to the learning rate, where it was amongst the higher rates utilized by the best performers.

# F AR Residual Analysis

In figure Figure 22 we show a diagnostic plot of the residuals stemming from the endogenous AR model. These were obtained from the in-sample fitted residuals. From the QQ-plot in the lower left and the histogram in the upper right, we can deduce that the residuals are not normally distributed. This may suggest non-linearities are present in the VIX index, however they are likely caused by outliers due to periods of financial distress such as the 2008 financial crisis.

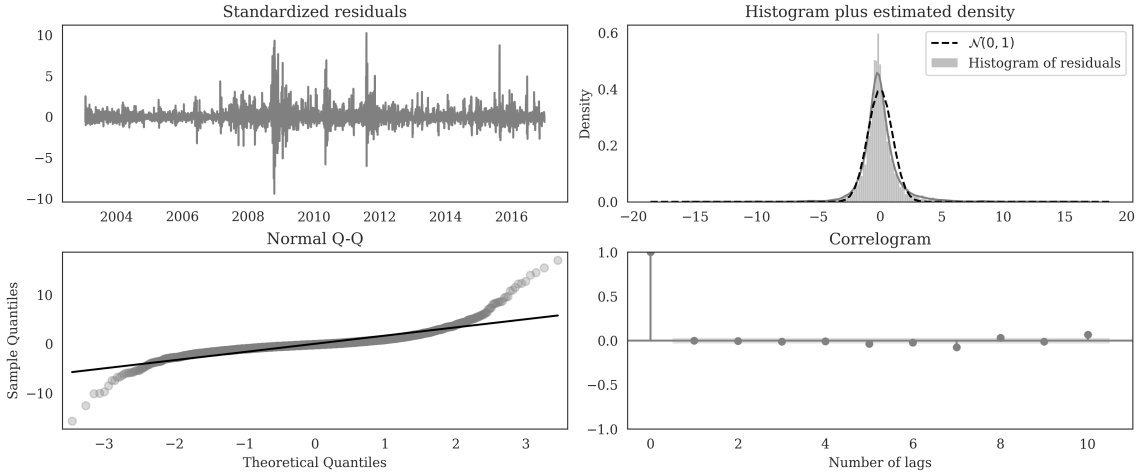
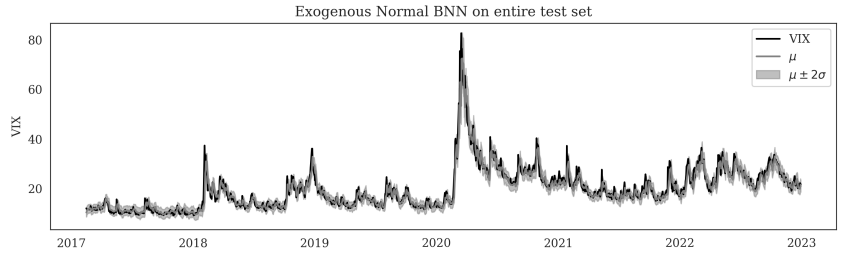


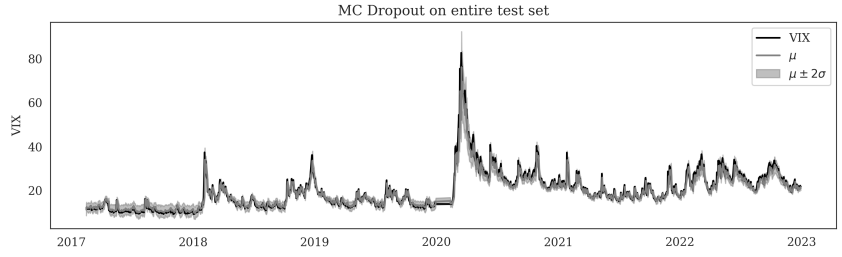

Figure 22: Residual analysis of the endogenous AR model. The upper left figure displays the standardized residuals over time, while the upper right figure shows a histogram of the residuals with a standard normal density plot for reference. The lower left figure contains a QQ-plot of the residuals against a standard normal reference line, while the correlogram plots the ACF of the residuals.

# G Forecasts and Confidence Bands of BNNs

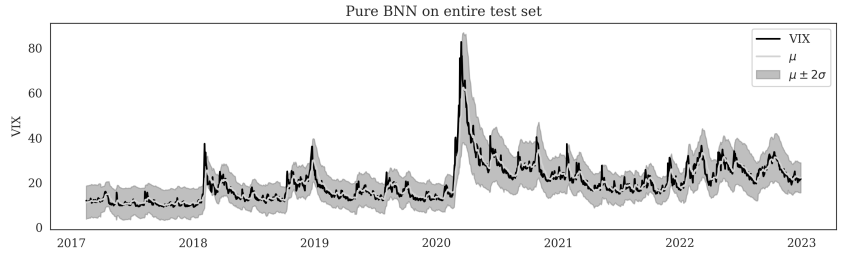
In this section we provide a visual demonstration of the 1-day forecasts for all BNN models in Figure 23. Each plot plots the VIX, the mean prediction and a $\pm 2$ standard deviation confidence band.



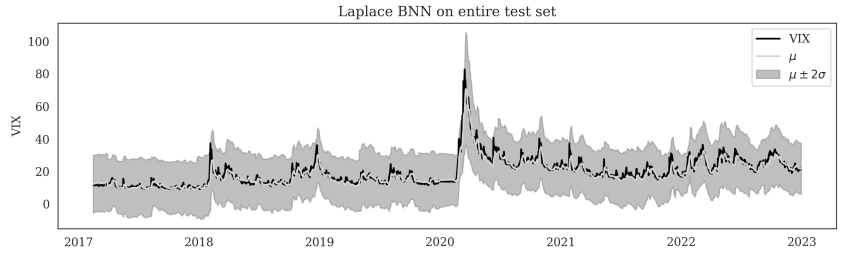

(a) The forecasts produced by the exogenous Normal BNN model.



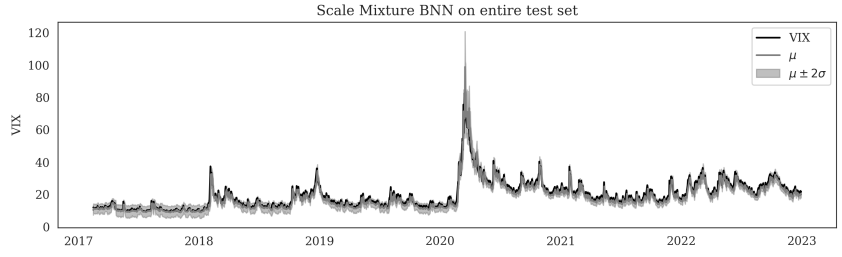

(b) Forecasts acquired from the MC Dropout model.




(c) Forecasts using the Pure BNN model.




(d) Forecasts produced by the Laplace BNN model.




(e) Forecasts from the Scale Mixture Gaussian BNN model.

Figure 23: The mean forecasts produced by all BNN models on the entire test set. Each model include a $\pm 2\sigma$ confidence band.

# H Results of 5-day Dynamic Forecasting Before, During and After Covid

The section expands upon the results of the 5-day dynamic forecasting by including evaluation on both test sets. Table 27 reports the results on the first test set, while Table 28 contains the second test set evaluation.

**Before Covid**

| | MSE | MAE | MAPE | RMSE | Huber | R2 | MDA |
|---|---|---|---|---|---|---|---|
| AR | **8.261** | **1.829** | 0.115 | **2.874** | **1.408** | **0.544** | **0.448** |
| LSTM | 9.447 | 1.866 | **0.113** | 3.074 | 1.452 | 0.479 | 0.44 |
| Normal BNN | 10.473 | 1.918 | 0.114 | 3.236 | 1.5 | 0.422 | 0.454 |

Table 27: Results of 5-day dynamic forecasting on the pre-Covid test set. The AR performed best in all metrics.

**During and after Covid**

| | MSE | MAE | MAPE | RMSE | Huber | R2 | MDA |
|---|---|---|---|---|---|---|---|
| AR | **23.045** | **2.953** | **0.109** | **4.8** | **2.503** | **0.695** | **0.524** |
| LSTM | 31.455 | 3.392 | 0.121 | 5.609 | 2.946 | 0.584 | 0.473 |
| Normal BNN | 57.723 | 5.41 | 0.194 | 7.598 | 4.929 | 0.237 | 0.48 |

Table 28: Results of 5-day dynamic forecasting on the post-Covid test set

# Results of 20-day Forecasts Before, During and After Covid

We present the additional evaluations of the 20-day forecasting exercise on the two test sets, where Table 29 contains the first test set and Table 30 contains the second.

**Before Covid**

| | MSE | MAE | MAPE | RMSE | Huber | R2 | MDA |
|---|---|---|---|---|---|---|---|
| AR | 16.918 | 3.011 | 0.198 | 4.113 | 2.541 | 0.076 | 0.503 |
| LSTM ENDOG | 16.844 | 3.005 | 0.199 | 4.104 | 2.539 | 0.08 | 0.479 |
| LSTM EXOG | **15.693** | **2.664** | **0.167** | **3.961** | **2.207** | **0.143** | **0.508** |
| Normal BNN ENDOG | 16.817 | 3.22 | 0.225 | 4.101 | 2.74 | 0.082 | 0.49 |
| Normal BNN EXOG | 18.574 | 3.362 | 0.237 | 4.31 | 2.888 | -0.014 | 0.488 |

Table 29: Results of 20-day Forecasts Before Covid.

**During and after Covid**

| | MSE | MAE | MAPE | RMSE | Huber | R2 | MDA |
|---|---|---|---|---|---|---|---|
| AR | **79.686** | **5.191** | **0.182** | **8.927** | **4.715** | **-0.053** | 0.494 |
| LSTM ENDOG | 87.069 | 5.417 | 0.188 | 9.331 | 4.945 | -0.151 | 0.474 |
| LSTM EXOG | 101.475 | 6.145 | 0.209 | 10.073 | 5.671 | -0.341 | **0.496** |
| Normal BNN ENDOG | 82.144 | 5.296 | 0.185 | 9.063 | 4.823 | -0.086 | 0.463 |
| Normal BNN EXOG | 95.971 | 5.777 | 0.193 | 9.796 | 5.313 | -0.269 | 0.483 |

Table 30: Results of 20-day forecasts during and after Covid. The negative R2 score implies that all models were unable to make sensible forecasts in this period.

# I Window Sizes

The following tables summarizes the evaluation of a selected subset of models using different window sizes as input. For clarity, a window size of 5 entails that the input included 5 lags of the data. We limit the reported models to the best performing BNN and the LSTM model, both with an endogenous and exogenous variant. In the intention of brevity we only include results on the entire test set. Table 31 reports the results using 5-day window sizes, Table 32 contains the results with a 15-day window, Table 33 reports for the 45-day window sizes and Table 34 holds the 60-day window size results.

## 5-day window

| | MSE | MAE | MAPE | RMSE | Huber | R2 | MDA |
|---|---|---|---|---|---|---|---|
| Normal BNN ENDOG | 5.288 | 1.426 | 0.069 | 2.300 | 1.014 | 0.928 | 0.469 |
| Normal BNN EXOG | 5.767 | 1.502 | 0.076 | 2.401 | 1.079 | 0.922 | 0.456 |
| LSTM ENDOG | **4.591** | **1.191** | **0.056** | **2.143** | **0.822** | **0.938** | 0.459 |
| LSTM EXOG | 5.619 | 1.377 | 0.064 | 2.37 | 0.99 | 0.924 | **0.471** |

Table 31: Results of using a 5-day input window into selected ML models on the combined test set.

## A 15-day window

| | MSE | MAE | MAPE | RMSE | Huber | R2 | MDA |
|---|---|---|---|---|---|---|---|
| Normal BNN ENDOG | **4.486** | 1.220 | 0.058 | **2.118** | 0.834 | **0.939** | 0.471 |
| Normal BNN EXOG | 5.8 | 1.463 | 0.072 | 2.408 | 1.049 | 0.921 | 0.465 |
| LSTM ENDOG | 4.587 | **1.171** | **0.054** | 2.142 | **0.806** | 0.938 | 0.461 |
| LSTM EXOG | 5.432 | 1.332 | 0.061 | 2.331 | 0.951 | 0.926 | **0.472** |

Table 32: Results of using 15-day window size on combined test set.

## B 45-day window

| | MSE | MAE | MAPE | RMSE | Huber | R2 | MDA |
|---|---|---|---|---|---|---|---|
| Normal BNN ENDOG | **4.417** | **1.198** | **0.055** | **2.102** | **0.828** | **0.940** | **0.471** |
| Normal BNN EXOG | 6.294 | 1.528 | 0.076 | 2.509 | 1.105 | 0.915 | 0.464 |
| LSTM ENDOG | 5.218 | 1.258 | 0.056 | 2.284 | 0.889 | 0.929 | 0.464 |
| LSTM EXOG | 5.214 | 1.336 | 0.062 | 2.283 | 0.952 | 0.929 | 0.469 |

Table 33: Results of using 45-day input window size on combined test set.

## C 60-day window

| | MSE | MAE | MAPE | RMSE | Huber | r2 | MDA |
|---|---|---|---|---|---|---|---|
| Normal BNN ENDOG | **4.879** | **1.247** | **0.056** | **2.209** | **0.878** | **0.934** | **0.472** |
| Normal BNN EXOG | 5.437 | 1.435 | 0.07 | 2.332 | 1.021 | 0.926 | 0.461 |
| LSTM ENDOG | 5.399 | 1.267 | 0.058 | 2.324 | 0.891 | 0.927 | 0.464 |
| LSTM EXOG | 5.433 | 1.375 | 0.064 | 2.331 | 0.986 | 0.926 | 0.466 |

Table 34: Results of using 60-day input window on combined test set

# J Evaluation Metrics

To evaluate performances and guide the tuning procedure, we decided to use the Mean Squared Error metric, defined in equation Equation 27

$$MSE = \frac{1}{T}\sum_{t=1}^{T}(\hat{y}_t - yt) \tag{27}$$

where $\hat{y}_t$ corresponds to the predicted value at time $t$ and $y_t$ the actual value. The metric measures the $L_2$ error of the forecasts, causing it to penalize large errors heavily. This makes it suitable for trying to minimize the total prediction accuracy. However, using it as the sole metric can be detrimental for several reasons. It is heavily affected by large errors, consequently making it sensitive to outliers. The VIX index contains some major shocks, e.g., the Covid and 2008 financial crisis. Adjusting the models based on this metric alone could therefore bias the selection towards those who aptly captured larger errors by chance, but performed generally worse over other periods. To combat this, we included a supplementary set of metrics less sensitive to outliers. These included the root mean squared error metric (RMSE), the mean absolute error (MAE) and the mean absolute percentage error (MAPE).

$$\begin{aligned} RMSE &= \sqrt{MSE} \\ MAE &= \frac{1}{T}\sum_{t=1}^{T}|\hat{y}_t - y_t| \\ MAPE &= \frac{100\%}{T}\sum_{t=1}^{T}\frac{|y_t - \hat{y}_t|}{y_t} \end{aligned} \tag{28}$$

Additionally, we included the Huber loss, a piece-wise function. For smaller errors it behaves similarly to the MSE error, allowing it to exploit the smoothness and sensitivity to errors. If the errors exceed a pre-defiend limit $\delta$, it changes to be more like the MAE for larger errors which prevents larger errors to dominate the performance metric.

$$L_\delta = \left\{ \begin{array}{ll} ||\frac{1}{2}(y-\hat{y})^2 & \text{for} |y-\hat{y}| \leq \delta \\ ||\delta(|y-\hat{y}| - \frac{1}{2}\delta) & \text{otherwise} \end{array} \right\} \tag{29}$$

Another consideration to make concerns the symmetricity of the error metrics. In financial data, it can often be of interest to assess the model directional accuracy; an error in the correct direction is less severe than an equivalent error in the wrong direction. Hence, we also included the *mean directional accuracy* (MDA) as a supplementary metric,

$$MDA = \frac{1}{T}\sum_{t=1}^{T} Sign(\hat{y}_t - y_t) \tag{30}$$

where $Sign$ is an indicator function for the directional value.

To illustrate the performance of a binary classification model, it is necessary to consider the trade-off between the true positive rate (TPR) and the false positive rate (FPR). This is called the ROC (Receiver Operating Characteristic) curve, and is generating by plotting these two rates against each other with different sets of thresholds. We have,

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{31}$$

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}, \tag{32}$$

where TP is true positives, FN is false negatives, FP is false positives, and TN is true negatives.

Furthermore, we have the F1 score, which is a measure of a classification's accuracy which balances both precision recall. We have precision,

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{33}$$

which measures how many of the positively predicted instances are actually true positives. Then, recall,

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{34}$$

is the true positive rate. F1 combines these two into a single metric, and is an important metric if both precision and recall are important for the prediction.

# K The Full Dataset

In Table 35 we present the entire dataset we made for our thesis.

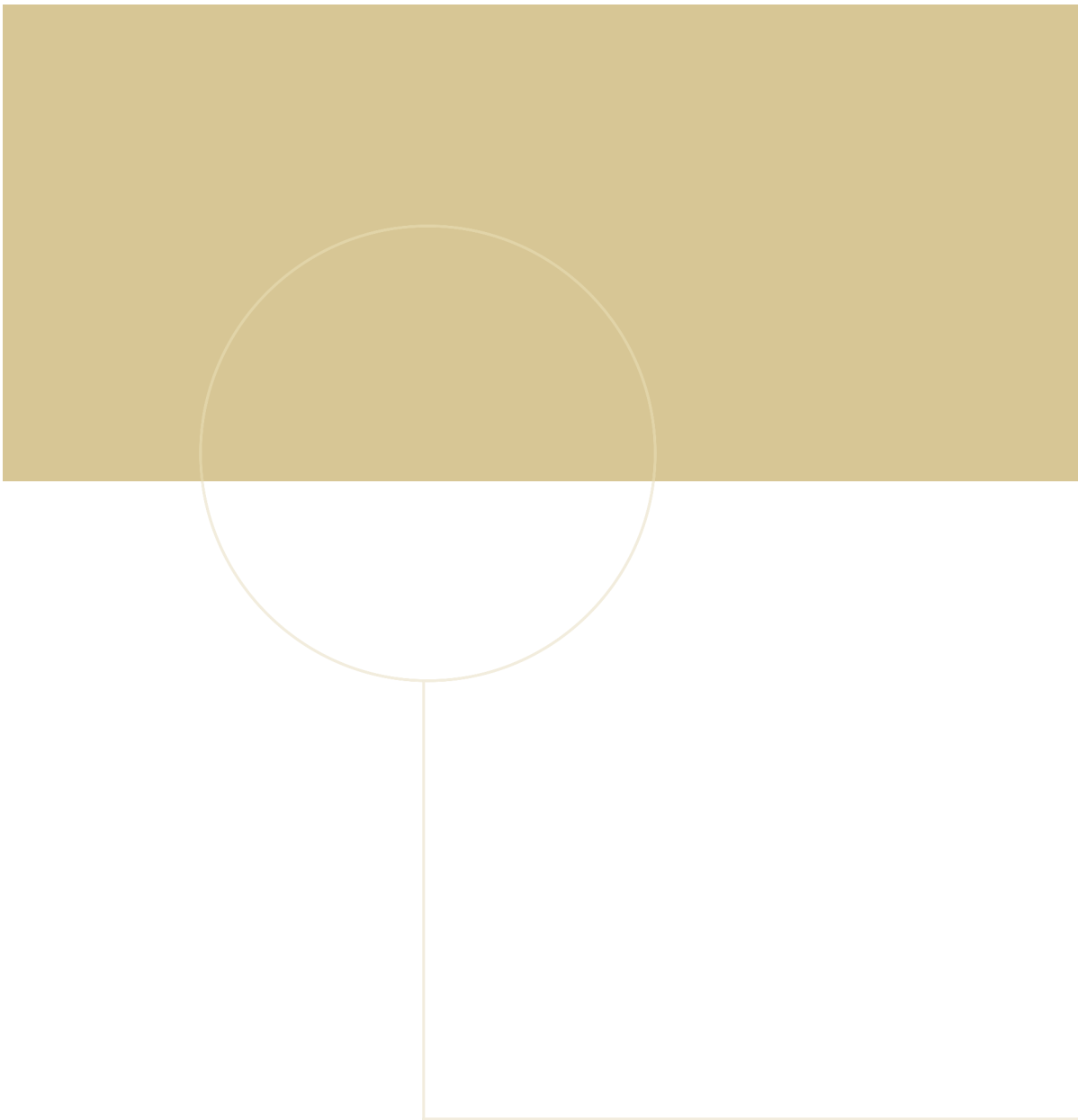
| Name | Paper source | Type | Data source | Freq |
|---|---|---|---|---|
| SMB (Small minus big | (Vrontos et al., 2021) | Factors | Fama-French | D |
| HML (High minus low) | (Vrontos et al., 2021) | Factors | Fama-French | D |
| MKT (Equity market returns) | (Vrontos et al., 2021) | Factors | Fama-French | D |
| MOM (Medium price Momentum) | (Vrontos et al., 2021) | Factors | Fama-French | D |
| STR (Short term reversal) | (Vrontos et al., 2021) | Factors | Fama-French | D |
| 10-Year Treasury Constant Maturity Minus 3-Month Treasury Constant Maturity | (Vrontos et al., 2021) | Macro | FRED | D |
| Moody's Seasoned Baa Corporate Bond Yield Relative to Yield on 10-Year Treasury Constant Maturity | (Vrontos et al., 2021) | Macro | FRED | D |
| Market Yield on U.S. Treasury Securities at 10-Year Constant Maturity, Quoted on an Investment Basis | (Vrontos et al., 2021) | Macro | FRED | D |
| Crude Oil Prices: West Texas Intermediate (WTI) - Cushing, Oklahoma | (Vrontos et al., 2021) | Macro | FRED | D |
| Nominal Broad U.S. Dollar Index | (Vrontos et al., 2021) | Macro | FRED | D |
| Bullish | (Vrontos et al., 2021) | Sentiment | AAII | W |
| Neutral | (Vrontos et al., 2021) | Sentiment | AAII | W |
| Bearish | (Vrontos et al., 2021) | Sentiment | AAII | W |
| Bullish 8-week mov avg | (Vrontos et al., 2021) | Sentiment | AAII | W |
| Bull-bear spread | (Vrontos et al., 2021) | Sentiment | AAII | W |

| | | | | |
|---|---|---|---|---|
| Economic Policy Uncertainty Index for United States | (Prasad et al., 2022) | Index | FRED | D |
| Equity market-related Economic Uncertainty Index | (Prasad et al., 2022) | Index | FRED | D |
| ICE BofA BBB U.S. Corporate Index option-adjusted spread | (Prasad et al., 2022) | Macro | FRED | D |
| ICE BofA AAA U.S. Corporate Index option-adjusted spread | (Prasad et al., 2022) | Macro | FRED | D |
| ICE BofA BBB U.S. Corporate Index effective yield | (Prasad et al., 2022) | Macro | FRED | D |
| ICE BofA AAA U.S. Corporate Index effective yield | (Prasad et al., 2022) | Macro | FRED | D |
| ICE BofA U.S. Corporate Index option-adjusted spread | (Prasad et al., 2022) | Macro | FRED | D |
| ICE BofA U.S. Corporate Index effective yield | (Prasad et al., 2022) | Macro | FRED | D |
| Market Yield on U.S. Treasury Securities at 5-Year Constant Maturity, Quoted on an Investment Basis | (Prasad et al., 2022) | Macro | FRED | D |
| Market Yield on U.S. Treasury Securities at 10-Year Constant Maturity, Quoted on an Investment Basis | (Prasad et al., 2022) | Macro | FRED | D |
| 5-Year Breakeven Inflation Rate | (Prasad et al., 2022) | Macro | FRED | D |
| 10-Year Breakeven Inflation Rate | (Prasad et al., 2022) | Macro | FRED | D |
| Effective Federal Funds Rate | (Prasad et al., 2022) | Macro | FRED | D |
| CBOE Crude Oil ETF Volatility Index | (Prasad et al., 2022) | Macro | FRED | D |
| Treasury yield curve rates (1-month maturity) | (Prasad et al., 2022) | Macro | QUANDL | D |
| Treasury bill rates (4-week bank discount rate) | (Prasad et al., 2022) | Macro | QUANDL | D |
| Gold fixing price (London a.m. and p.m. time) in London Bullion Market, based in U.S. dollars | (Prasad et al., 2022) | Financial | QUANDL | D |
| S&P 500 | | Financial | FRED | D |
| CBOE VIX | | Financial | FRED | D |
| VIX moving average 5 days | | Technical | | D |
| VIX moving average 10 days | | Technical | | D |
| VIX exponentially moving average 5 days | | Technical | | D |
| VIX exponentially moving average 10 days | | Technical | | D |
| VIX momentum 5 days | | Technical | | D |
| VIX momentum 10 days | | Technical | | D |
| VIX RSI 5 days | | Technical | | D |
| VIX RSI 10 days | | Technical | | D |
| VIX Bollinger upper band 5 days | | Technical | | D |
| VIX Bollinger upper band 10 days | | Technical | | D |
| VIX Bollinger lower band 5 days | | Technical | | D |
| VIX Bollinger lower band 10 days | | Technical | | D |
| VIX ADX 5 days | | Technical | | D |
| VIX ADX 10 days | | Technical | | D |

Table 35: The full dataset including the VIX, macroeconomic variables, factors, sentimental variables, as well as technical indicators.

# L Code

The following repository will contain all necessary code needed in order to reproduce our results: `https://github.com/Eliassg/bayesian-VIX`



NTNU

Norwegian University of
Science and Technology