University of Zürich

Seminar: Applied Quantitative Finance

Deep learning for pricing: An (American) option to consider

Sven Spa (20-726-527)

Jasper Grootscholten (20-744-553)

Richard Breitschopf (20-743-407)

Supervisors: Prof. Dr. Markus Leippold, Dr. Nikola Vasiljevic and Jordy Rillaerts

Department of Banking and Finance

May 2021

Contents

1 Introduction

This seminar looks critically at the proposed methods given in the scientific paper 'Generative Bayesian neural network model for risk-neutral pricing of American index options' by Jang and Lee (2019).

Financial markets are driven by its investors. Through the dominating law of supply and demand regarding tradeable assets, deviations of asset prices are common. Especially larger companies are subject to such fluctuations. Therefore, financial derivatives were introduced in order to be able to handle risks stemming from the above mentioned. Finding a price of such an instrument has been a large area in quantitative finance, especially since the famous work of Black and Scholes (1973). Over the years, several methods have been proposed to price options, for European as well as American options.

In this paper we investigate deep learning techniques, specifically support vector machines and neural networks, as well as a new generative method, proposed in Jang and Lee (2019), for the purpose of pricing American put options. We compare the performance of these models with two more classical models, the Black-Scholes and Heston model. Our performance comparison is based on data for American put options on the S&P 100 index, over the period 2003-2012. We show that the generative Bayesian neural network performs best at the task of option pricing, especially in prediction.

The rest of this paper is structured as follows. The remainder of this section is devoted to discussing related literature on the topic of (American) option pricing. Next, in Section 2 we introduce and outline the models used. In Section 3 we give an overview of the data used, and discuss the performance of the different models. Lastly, in Section 4 we draw our conclusion, as well as a critical discussion of the model performance, limitations, differences with Jang and Lee (2019) and some recommendations for further research.

1.1 Literature Review

Ever since the seminal paper by Black and Scholes (1973), in which they derive a closed form pricing formula for European options, the pricing of financial derivatives has received increased attention in literature. More sophisticated models, which solve problematic assumptions/results from the Black-Scholes framework, arose. Popular classes of models include stochastic volatility models, which relax the assumption of constant volatility (see for example Heston (1993), Hagan et al. (2002)) and jump diffusion models, which allow for jumps to better match short-term behavior of empirical data (see Bates (1996), Kou (2002), Carr et al. (2003)). Most of these models also allow for a (semi)-closed form solution, which allows for efficient pricing of European options.

However, these models are centered around European options, whereas in practice American options are more frequently traded (Jang and Lee (2019)). American options can be exercised at any time before maturity, which makes pricing them more complicated than their European counterpart. Numerous solutions have been suggested, with three of the most popular approximation algorithms having being introduced by Cox et al. (1979), Barone-Adesi and Whaley (1987) and Longstaff and Schwartz (2001).

Alongside these classical models, there has been a vast amount of recent research in option pricing with deep learning techniques. In a recent literature review, Ruf and Wang (2020b) list 150 papers covering this topic, ranging from early papers like Hutchinson et al. (1994) to modern papers like Buehler et al. (2019). Similarly to classical models, the main focus of these papers is on European options, though Kelly et al. (1994), Chen and Lee (1999), Meissner and Kawano (2001), Pires and Marwala (2004), Pires and Marwala (2005) and Amornwattana et al. (2007) do consider American options. Deep learning models considered are Artificial Neural Networks (ANN) (Hutchinson et al. (1994), Lajbcygier and Connor (1997), Anders et al. (1998), Yao et al. (2000)), Bayesian Neural Networks (BNN) (Pires and Marwala (2005)), Support Vector Machines (SVM) (Wang (2011), Kazem et al. (2013)) and Gaussian Processes (GP) (Han and Lee (2008)).

Important take away messages from this literature are the following: Essential inputs for the networks are the stock and strike price, which can be combined using moneyness, and time to maturity. Another popular input is some measure of the volatility, like historical or implied. The most straightforward idea is to output the option price, but recent papers also consider directly modelling hedging strategies (Buehler et al. (2019)), predicting the implied volatility surface (Zheng et al. (2019)) or solving PDE's related to option prices (Chen and Wan (2021)). An important drawback to deep learning techniques is that it is data reliant. Deep in- or out of the money option data is scarce, meaning prediction results on this type of options are unreliable. Jang and Lee (2019) suggest solving this problem by generating additional artificial data from an arbitrage free model.

2 Methodology

In this chapter we formally introduce all models used in this paper to price options. We start with the more classical approaches and then move to deep learning techniques.

2.1 Black-Scholes model

By far the most famous option pricing model is the Nobel prize winning Black-Scholes model (Black and Scholes (1973)). It assumes the stock price follows the specification

$$\mathrm{d}S_t = \mu S_t \mathrm{d}t + \sigma S_t \mathrm{d}W_t,$$

where S_t is the stock price at time t, μ is the drift, σ is the volatility and W_t is a Brownian motion. From this specification Black and Scholes (1973) derived a closed form formula for the price of a European option (usually the call price is given, but in this paper we are concerned with put options)

$$\begin{aligned}
&P(S_t, K, \sigma, (T-t), r) = Ke^{-r(T-t)}\Phi(-d_2) - S_t\Phi(-d_1),\\
&d_1 = \frac{\log(S_t/K) + (r + \frac{\sigma^2}{2})(T-t)}{\sigma\sqrt{T-t}},\\
&d_2 = d_1 - \sigma\sqrt{T-t},
\end{aligned}$$

where r is the risk-free rate, and Φ is the CDF of a standard normal distribution. Note that the only unknown variable in this formula is the volatility σ.

Practitioners often work with the so called implied volatility σ_{IV}, which is the volatility one has to input into the Black-Scholes formula to obtain the market price. To be more concrete, σ_{IV} solves $P_{BS}(S_t, K, \sigma_{IV}, (T-t), r) = P_{market}(S_t, K, (T-t), r)$. This solution is not available in closed form, but the derivative of P_{BS} with respect to σ, which is called 'vega' is given by $\frac{\partial P_{BS}}{\partial \sigma} = S_t \phi(d_1)\sqrt{T-t}$. Using this derivative, we can apply Newton's method to find a simple numerical approximation.

To price American put options with the Black-Scholes model, we use an ad-hoc approach described in Jang and Lee (2019). This approach uses the linear regression

$$\sigma_{IV} = \beta_0 + \beta_1 K + \beta_2 (T-t) + \beta_3 (T-t)K + \varepsilon \quad \varepsilon \sim \mathcal{N}(0,1),$$

to estimate the implied volatility. These estimated implied volatilities can then be plugged back into the Black-Scholes formula to obtain the option prices $P_{BS}(S_t, K, \hat{\sigma}_{IV}, (T-t), r)$. Note that in this approach we ignore the fact that we are working with American instead of European options. Berkowitz (2009) has shown that this is not that much of a problem if the regression is estimated on American option data.

2.2 Heston model

The assumption of constant volatility in the Black-Scholes model is rather restrictive. The Heston model (Heston (1993)) is a stochastic volatility model for option pricing, where the asset price S_t and volatility ν_t are given by the following specification:

$$\begin{aligned} \mathrm{d}S_t &= \mu S_t \mathrm{d}t + \sqrt{\nu_t} S_t \mathrm{d}W_t^S, \\ \mathrm{d}\nu_t &= \kappa(\theta - \nu_t)\mathrm{d}t + \eta\sqrt{\nu_t}\mathrm{d}W_t^\nu, \end{aligned}$$

where W_t^S and W_t^ν are Brownian motions with correlation ρ and the parameters μ, κ, θ and η describe the mean return, mean reversion rate of the volatility, long turn average volatility and volatility of volatility respectively. In order to price American options using this model, the parameters have to be calibrated. The calibration and pricing is done in 3 steps, following AitSahlia et al. (2010), which are described in the coming subsections.

2.2.1 Structural parameter estimation

First the structural parameters μ, κ, θ and η are estimated. We note that the average rate of return is not significantly different from 0, hence we set $\mu = 0$ (as is commonly done, see AitSahlia et al. (2010)). Next we use the method of indirect inference by Gourieroux et al. (1993) to estimate $\Theta = (\kappa, \theta, \eta)$. This method involves using an auxiliary model that is simple to estimate, whose parameter set Γ is estimated on the actual data. Then the parameter set of the original model, which is harder to estimate, is found through moment matching.

To do the estimation, we need to discretize the Heston model, and introduce an auxiliary model. The Euler discretization of the Heston model is given by

$$\begin{aligned} r_t &= \sqrt{\nu_t \tau}\varepsilon_{1t}, \\ \nu_t &= \kappa\theta\tau + (1-\kappa\tau)\nu_t + \eta\sqrt{\nu_{t-\tau}}\varepsilon_{2t}, \end{aligned}$$

where $r_t = \log(\frac{S_t}{S_{t-1}})$ is the log return for the period $(t-\tau, t]$, with τ being the time discretization increment, and ε_{1t} and ε_{2t} are uncorrelated standard normal variables (estimation of the correlation ρ follows in step 2). This approach is the same as taken by

Engle and Lee (1996) and Zhang and Shu (2003).

We work with the GARCH(1,1) (Bollerslev et al. (1994)) model as an auxiliary process, given by

$$\begin{aligned} r_t &= \varepsilon_t, \\ h_t &= \omega + \alpha\varepsilon_{t-1}^2 + \beta h_{t-1}, \end{aligned}$$

where $\varepsilon_t \sim \mathcal{N}(0, h_t)$ and h_t is the conditional variance of r_t. In order to get an initial estimate $\Gamma^{(0)} = (\omega^{(0)}, \alpha^{(0)}, \beta^{(0)})$, we calibrate the GARCH model to actual market data. From those estimated parameters, we get an initial estimate $\Theta^{(0)} = (\kappa^{(0)}, \theta^{(0)}, \eta^{(0)})$, given by (Engle and Lee (1996))

$$\kappa^{(0)} = \frac{\omega^{(0)}}{\tau}, \quad \theta^{(0)} = \frac{(1 - \alpha^{(0)} - \beta^{(0)})}{\tau}, \quad \eta^{(0)} = \alpha^{(0)}\sqrt{(\xi - 1)\tau},$$

where ξ is the conditional kurtosis of the shocks in volatility estimated from the actual data[1].

Using $\Theta^{(0)}$, we simulate a sample of N observations (chosen equal to the length of the data), to which we again fit a GARCH(1,1) model, whose estimated parameters we denote by $\Gamma^{(1)}$. With these estimated, we can define the optimality criterion used to find our final parameter set $\Theta^{(1)}$

$$\Theta^{(1)} = \underset{\Theta}{\operatorname{argmin}}\{m^T(\Theta, \Gamma^{(1)})\Omega m(\Theta, \Gamma^{(1)})\}.$$

To define m and Ω, we first give the log-likelihood of the GARCH model $l_t(r(\Theta), \Gamma) = -\log(h_t) - \frac{r(\Theta)^2}{2h_t}$, where $r_t(\Theta)$ are returns simulated using the parameter set Θ and h_t is given through the parameters Γ. Then the maximum log-likelihood moment estimator m is given by

$$m(\Theta, \Gamma^{(1)})_{3x1} = \frac{1}{N}\sum_{t=1}^{N} \frac{\partial l_t(r(\Theta), \Gamma)}{\partial \Gamma}\bigg|_{\Gamma=\Gamma^{(1)}},$$

and the weight metric Ω is the matrix

$$\Omega = \left(\frac{1}{N}\sum_{t=1}^{N} \frac{\partial l_t(\tilde{r}_t, \Gamma)}{\partial \Gamma}\frac{\partial l_t(\tilde{r}_t, \Gamma)}{\partial \Gamma^T}\bigg|_{\Gamma=\Gamma^{(1)}}\right)^{-1},$$

where $\tilde{r}$ denotes the returns in the actual data.

We apply this approach to S&P 100 index data for the time period January 1, 2003 to December 31, 2012. We use 77 time discretizations, which corresponds to 5-minute intervals ($\tau = 1/77$). Moreover, we set the initial index level to the index level on the first day, and the initial volatility to 0.187 (as in AitSahlia et al. (2010)). The obtained parameter estimates are given in Table 1. Note that the final parameters $\Theta^{(1)}$ satisfy the Feller condition $2\kappa\theta > \eta^2$, which guarantee the continuous process v_t to be positive.

[1]This is done by setting $\xi = 3\frac{(\hat{\nu}-2)}{(\hat{\nu}-4)}$, where $\hat{\nu}$ is the estimated degrees of freedom of a GARCH(1,1) model of the actual data with student-t distribution

Table 1: Parameter estimates obtained through the indirect inference method

	ω	α	β
$\Gamma^{(0)}$	0.0227	0.1218	0.8599
$\Gamma^{(1)}$	0.0282	0.1048	0.2803
	κ	η	θ
$\Theta^{(0)}$	1.2424	0.0183	0.2773
$\Theta^{(1)}$	1.2011	0.0887	0.3213

2.2.2 Option price calibration

Next, we estimate the remaining parameters ρ and ν (ν is the spot volatility used at the time of pricing). This is done by calibrating the parameters to European option prices. The price of a European option can be calculated under the Heston model using the characteristic function, which is given in Heston (1993). These prices can be quickly evaluated using the Fast Fourier Transform, which is described in detail in Carr and Madan (1999).

We can apply this procedure for option i traded on day t to obtain the model option price $\hat{y}_i^t(\rho_t, \nu_t)$. We then define the error between the real market price y_i^t and the model option price $\hat{y}_i^t(\rho_t, \nu_t)$ via

$$\varepsilon_i^t = y_i^t - \hat{y}_i^t(\rho_t, \nu_t).$$

If we define n_t as the amount of options traded on day t, we can find estimates of $\hat{\rho}_t$ and $\hat{\nu}_t$ for day t by minimizing the sum of squared errors $\sum_{i=1}^{n_t}(\varepsilon_i^t)^2$. Finally, the parameter estimates are obtained by averaging over all trading days. Let T denote the total amount of trading days, such that the parameter estimates are given by $\hat{\rho} = \frac{1}{T}\sum_{t=1}^{T}\hat{\rho}_t$ and $\hat{\nu} = \frac{1}{T}\sum_{t=1}^{T}\hat{\nu}_t$.

Applying this procedure to European put options on the S&P 100, traded from January 1, 2003 to December 31, 2012, we obtain the parameter estimates given in Table 2. To make the optimization more efficient and to give reasonable parameter estimates, we imposed the bounds $\nu \in [0, 1]$ and $\rho \in [-0.999, 0.999]$ which are derived from the fact that they represent a volatility (which is typically small) and correlation respectively. As can be seen in Table 2, these bounds are hit in the optimization during certain days.

Table 2: Parameter estimates obtained through European put option price calibration

	Mean	Median	Std. dev	Min.	Max.
ρ	-0.2391	-0.1185	0.3558	-0.999	0.999
ν	0.0011	0.0006	0.0017	0.0000	0.0175

2.2.3 American option pricing

With all parameters estimated, we can now move to pricing American options. To do this, we apply the Least-Squares Monte Carlo algorithm described by Longstaff and Schwartz (2001). This approach assumes that the option can only be exercised at discrete time points, for which we use $m = 50$, so that $0 \leq t_1 \leq t_2 \leq ... \leq t_m = T$, where T is the expiration date. We start by sampling $N_S = 1000$ paths $\{S^n_{t_i}\}^{n=1,...,N_S}_{i=1,...,m}$ from the underlying model, which is the calibrated Heston model in this case. We then determine the option value $V(t_i)$ at time step t_i by comparing the cashflow obtained by exercising with the expected cashflow obtained from continuing, i.e. the expected value of the option after not exercising

$$V(t_i) = max[X(t_i), E[\sum_{j=i+1}^{m} e^{-r(t_j-t_i)}C(t_j : t_m)]], \quad i = 1, ..., m-1,$$

where $X(t_i) = (K - S_{t_i})^+$ is the value of exercising at time t_i, r is the risk-free interest rate and $C(t_j : t_m)$ are the cashflows obtained from time t_j to t_m by continuation. With initial condition $V(t_m) = (K - S_T)^+$, this gives a recursion that can be rolled backwards to find the option value at time 0 for path n. The final option value is computed by simply averaging the values over all N_S paths.

The key to use this algorithm is to approximate the continuation value $F(t_i) = E[\sum_{j=i+1}^{m} e^{-r(t_j-t_i)}C(t_j : t_m)]$. This is done by regressing the realized cash flows from continuation on a set of basis functions, for which we use the weighted Laguerre polynomials

$$L_n(S) = \exp(\frac{-S}{2})\frac{e^S}{n!}\frac{d^n}{dS^n}(S^n e^{-S}).$$

Thus the approximation is

$$F(t_i) \approx \hat{\alpha}_0 + \sum_{j=0}^{2} \hat{\alpha}_{j+1} L_j(S_{t_i}),$$

where $\hat{\alpha}_j, j = 0, ..., 3$ are the estimated coefficients from the regression

$$C(t_{i+1} : t_m) = \alpha_0 + \sum_{j=0}^{2} \alpha_{j+1} L_j(S_{t_i}).$$

As suggested in Longstaff and Schwartz (2001) we only do this approximation for in the money paths at time t_i to improve efficiency of the algorithm.

2.3 Common Machine Learning models

Having seen one of the most well known classical approaches to american option pricing in the previous part, we now shift our focus towards machine learning models and how they can be used for pricing problems.

2.3.1 Support Vector Regression

Before the rise of neural networks (which will be discussed in the next sections), the most popular class of machine learning models for classification and regression tasks were support vector machines (Vapnik (2013)). Support vector regression (SVR) tries to find the most 'flat' hyperplane that fits the data so that the error is at most ε. Given the option data $\{(x_i, y_i)\}_{i=1}^N$ (where x_i is the pair of moneyness and maturity and y_i the option price), this implies the following minimization problem, which includes slack variables to deal with infeasability (Jang and Lee (2019))

$$\begin{aligned}
\min\ & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*), \\
\text{s.t. } & y_i - (w^T\Psi(x_i) + b) \leq \varepsilon + \xi_i, \\
& (w^T\Psi(x_i) + b) - y_i \leq \varepsilon + \xi_i^*, \\
& \xi_i, \xi_i^* \geq 0,
\end{aligned}$$

where w, which determines the 'flatness', and b form the hyperplane, $C > 0$ is a constant that helps with regularization, Ψ is a nonlinear map from the input space to the feature space, which is implicit from the kernel discussed later, ξ_i and ξ_i^* are the slack variables and ε is the error tolerance.

To efficiently solve this, we use the Lagrangian dual problem given below, which due to the convexity of the primal problem (and some technical conditions) gives the same optimal solution (Awad and Khanna (2015)).

$$\begin{aligned}
\max\ & -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)\mathbf{k}(x_i, x_j) \\
& \quad - \varepsilon\sum_{i=1}^{N}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{N} y_i(\alpha_i - \alpha_i^*), \\
\text{s.t. } & \sum_{i=1}^{N}(\alpha_i - \alpha_i^*) = 0, \\
& \alpha_i, \alpha_i^* \in [0, C],
\end{aligned}$$

which is a quadratic program, with α and α^* being the Lagrangian multipliers. Here $\mathbf{k}(x, y) = \Psi(x)^T\Psi(y)$ is the kernel function, which can be used instead of specifying Ψ explicitly (this is often referred to as the kernel trick). We use the most popular radial basis function kernel, which allows the model to learn non-linear data, given by

$$\mathbf{k}(x, y) = exp(-\gamma\|x - y\|^2)$$

where $\gamma > 0$ is a parameter that measures to what degree the influence of a single datapoint reaches. Even though this duality helps the computational efficiency, the complexity of SVM's is between $O(N^2)$ and $O(N^3)$ (Bottou and Lin (2007)), depending on the chosen hyperparameters. Therefore, the SVM should be used for no more than a couple of 10.000 datapoints, which is a problem for our purposes further discussed in Section 3.

The three hyperparameters C, γ and ε are chosen through a joint grid search from which we obtain the values $C = 1$, $\gamma = 10000$ and $\varepsilon = 0.0001$.

2.3.2 Bayesian Neural Network

An Artificial Neural Network (henceforth, ANN) stems from a neuroscience argument. It is a collection of nodes (also noted as artificial neurons) which intends to model the neurons in the biological brain.

The structure of a feedforward ANN consists of an input layer, one or multiple hidden layers and an output layer which are all connected. Every hidden layer consists of nodes which represent a (non-)linear function (the activation function). The inputs of the nodes are the weighted output values of the previous layer (input values in case of the first hidden layer) plus a bias term. The weights play an important role for the ANN because they are trained along the way such that the weights describe the relative importance of the incoming value for every node.

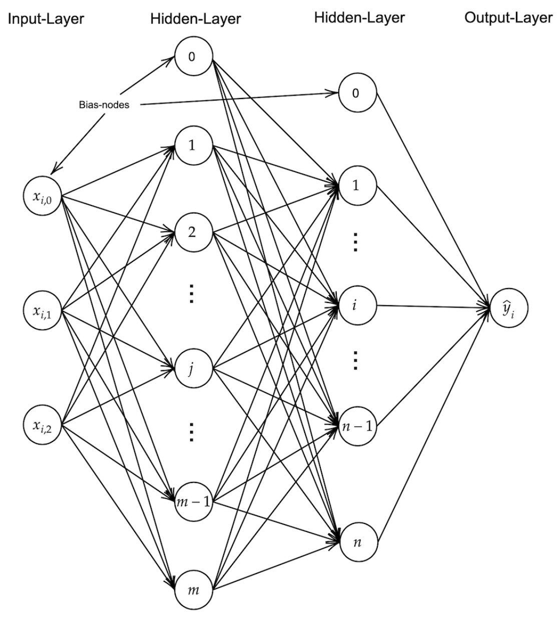

Figure 1: Graphical representation of an ANN with two hidden layers [2]

Mathematically, a representation of an ANN with two hidden layers as shown in Figure 1 is given by the following equation:

$$f(x,w) = h\left(\sum_{k=1}^{n} w_k^{(3)} \sigma_2\left(\sum_{j=1}^{m} w_{kj}^{(2)} \sigma_1\left(\sum_{i=1}^{l} w_{ji}^{(1)} x_i + w_{j0}^{(1)}\right) + w_{k0}^{(2)}\right) + w_0^{(3)}\right), \qquad (1)$$

where $w_{ij}^{(k)}$ is the weight between node i in hidden layer $k-1$ and node j in hidden layer k. To clarify our notation, $w_{ji}^{(1)}$ are the weights between the input layer and the first hidden layer, and $w_k^{(3)}$ are the weights from the last hidden layer to the output layer. x_i is the i^{th} input variable, $\sigma_k(.)$ is the activation function of the k^{th} hidden layer and $h(.)$ is the activation function of the output layer.

To train the ANN, the backpropagation algorithm is conducted, and optimization is done using the ADAM algorithm (Kingma and Ba (2014)) which is one of the most popular gradient descent methods, to train the weights of the ANN. This method is used to minimize the (pre-)specified loss function which depends on the true value and the predicted value

[2]Created with https://www.mathcha.io/editor

by the model.

In this research, a Bayesian regularized loss function is used, therefore, following Jang and Lee (2019), we refer to it as a Bayesian Neural Network (henceforth, BNN). The regularized loss function is specified as follows

$$\beta \sum_{i=1}^{N} (y_i - f(x_i, w))^2 + \alpha \sum_{j=1}^{N_w} w_j^2, \tag{2}$$

where y_i is the true output data, $f(x_i, w)$ the predicted data, N_w is the number of weights and w is the weights of the BNN. The loss function, as shown in Equation (2), is the Bayesian regularized loss function with regularization parameters α and β.

The Bayesian aspect of the loss function is derived by assuming that the option data $\{(x_i, y_i)\}_{i=1}^{N}$ follows the relation

$$y_i = f(x_i, w) + \varepsilon_i, \tag{3}$$

where y_i is again the true output data, $f(x_i, w)$ the output of the ANN and $\varepsilon_i \sim \mathcal{N}(0, \beta^{-1})$ is the noise in the data.

Furthermore, we also need Bayes inference to derive the Bayesian aspect of the loss function. Let w be the weight vector of a neural network with the additional assumption that these are random variables with probability distribution $p(w)$. Interpreting this as a prior distribution we can use Bayes inference to get to the posterior distribution

$$p(w \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid w)p(w)}{p(\mathcal{D})}, \tag{4}$$

where $\mathcal{D}$ represents the option data. Intuitively spoken, through the data we get a better idea of how the distribution over the weights looks like.

In practice, the expression $p(\mathcal{D})$ is usually not known and computationally expensive to calculate. But what one gets from Bayes' formula is

$$p(w \mid \mathcal{D}) \propto p(\mathcal{D} \mid w)p(w).$$

This relation is very important in order to find the maximum a posteriori (MAP) estimate. For this concept, we need to find the w, which maximizes $p(w \mid \mathcal{D})$, and therefore $p(\mathcal{D})$ can be seen as a constant in optimization, because it is independent of w.

By using Equation (3), we have the following likelihood function

$$\prod_{i=1}^{N} \mathcal{N}(y_i; f(x_i, w), \beta^{-1}).$$

i.e. $y_i \overset{iid}{\sim} \mathcal{N}(f(x_i, w), \beta^{-1})$. Now we want to regularize the weights w by introducing the prior $\prod_{j=1}^{N_w} N(w_j; 0, \alpha^{-1})$ with α being a strict positive constant value. Hence, following Bayes inference by combining the above defined likelihood function with the prior, we get

$$\prod_{i=1}^{N} \mathcal{N}(y_i; f(x_i, w), \beta^{-1}) \prod_{j=1}^{N_w} \mathcal{N}(w_j; 0, \alpha^{-1}). \tag{5}$$

In order to find the MAP estimate, we need to maximize the posterior distribution given in Equation (5). For this, we take the logarithm and switch to density notation. After dropping the constant values, because they do not affect the optimization, we end up with the following optimization equation

$$\beta \sum_{i=1}^{N} (y_i - f(x_i, w))^2 + \alpha \sum_{j=1}^{N_w} w_j^2,$$

which is identical to the loss function defined in Equation (2). In this case, this boils down to maximum likelihood estimation with a Bayesian regularization term in the loss function to prevent over-fitting problems (MacKay (1994), Burder and Winkler (2008)).

The BNN used in this paper consists of 3 hidden layers of 64, 256 and 128 nodes, respectively. The input consists of moneyness and maturity of the put options. The hyperparameters of the BNN are optimized with the help of a grid search. This resulted in $\beta = 1$ and $\alpha = 10^{-7}$ for the loss function. The other specifications of the BNN are learning rate = 0.01, batch size = 256, epochs = 500, for the activiation function ($\sigma(.)$) we use ReLU and for the output function ($h(.)$) a linear function. For initialization of the weights, we sample from a normal distribution with mean 0 and variance $2/n_{in}$, where n_{in} is the number of incoming connections from the previous layer. He et al. (2015) showed that initializing the weight in this manner results in earlier convergence when ReLU hidden layers are used (as in our case).

2.3.3 Fully Bayesian Neural Network

The problem with classic BNN is that this model does not account for uncertainty, meaning once trained it always returns the same prediction given the same input. This overconfidence is especially problematic in areas where uncertainty plays a major role, such as financial markets. Facing this issue led researchers to come up with fully Bayesian neural networks[3] (henceforth, FBNN).

The FBNN is trained using variational inference for Bayesian models (also called variational Bayes) to fit a ”surrogate” posterior to the posterior distribution of the weights. This method tends to be faster than classical methods, such as MCMC sampling (Blei et al. (2017)). In this paper, a multivariate normal distribution is used as the ”surrogate” posterior where the mean, variance and covariances are trainable. For the prior, we assume a non-trainable multivariate normal distribution with the identity matrix as covariance matrix.

After training the ”surrogate” posterior, we can draw a random sample from this posterior. The output of the node, by applying the activation function $\sigma(.)$ for each hidden layer, is calculated using those sampled weights. An immediate implication of the sampled weights is that the output differs each time, which eventually gives us a picture of how certain the network is about its own predictions.

The FBNN used in this paper consists of 2 hidden layers of 32 nodes. The inputs are the moneyness and maturity of the put option data. As for the BNN, a grid search is conducted

[3]Note that these are also often called Bayesian neural networks in the literature, but to prevent confusion we stick to the term fully Bayesian

to optimize the hyperparameters of the model. This gives $\beta = 10^{-7}$ and $\alpha = 1$, which is remarkable since it is the opposite of the BNN. Moreover, a learning rate of 0.01, batch size of 256, 500 epochs, ReLU activation function and a linear output function is used. Furthermore, we experimented with different loss functions: MSE, negative log likelihood and Equation (2). Eventually, using Equation (2) yielded the best results, which is a little counterintuitive since it then has a double Bayesian flavor.

2.4 Generative Bayesian Neural Network

Since classical machine learning methods have problems pricing ITM and OTM options properly due to bad data, a generative approach is presented here, named Generative Bayesian Neural Network (GBNN). The idea hereby is to use a well-known market model, which fulfills classical necessities such as returning arbitrage-free prices, and use this model to generate data for the GBNN, which then self-evolves from day to day while keeping the learned properties. More precisely, the market model is only used on the first trading day for generating arbitrage-free prices for ITM and OTM options. ATM option prices are provided from actual market data. Once the GBNN has learned from the artificial data and market data in the first trading day, it will predict for itself ITM and OTM option prices for the next trading day, which are then combined with actual market ATM option data again. Following this, the GBNN is again trained on the newly obtained dataset. This continues from day to day until the network is fully trained on the provided data. The inputs for the market model as well as for the GBNN are moneyness and maturity in our case.

In the following we derive the above within a mathematical framework. For this, we assume that the option price y_i^t follows the relation

$$y_i^t = f(x_i^t, w) + \varepsilon_i$$

for all $i = 1, \dots, n_t$, where $f(x_i^t, w)$ represents a neural network model with weight vector w and input data x_i^t, and n_t stands for the amount of option data within one trading day. Furthermore, ε_i represents the noise in the data and fulfills $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$. Rewriting this into a probability distribution, we immediately get

$$p(\mathcal{D} \mid w) = \prod_{i=1}^{n_t} \mathcal{N}(y_i^t; f(x_i^t, w), \sigma^2).$$

As mentioned before, the model will be first trained with a set of data points and a set of option prices generated by the market model. In order to keep the learned properties from the market model in the ongoing training procedure, we want the predicted prices of the model close to the previous learned prices, at least for ITM and OTM options. This can be translated into a prior distribution

$$p(w \mid w^{t-1}) = \prod_{k=1}^{\nu} \mathcal{N}(f(x_k, w); f(x_k, w^{t-1}), \sigma_0^2),$$

where ν is the number of prior samples generated in the first step by the market model and later by the GBNN itself. With this we achieved that the OTM and ITM option values are going to be close to previous predicted ones up to gaussian noise.

Remembering Bayes' rule, equation (4) from Section 2.3.2, we can derive the log posterior distribution

$$\begin{aligned}\ln\big(p(w \mid \mathcal{D})\big) &= \ln(p(\mathcal{D} \mid w)) + \ln(p(w \mid w^{t-1})) - \ln(p(\mathcal{D}))\\ &= \sum_{i=1}^{n_t} \ln\Big(\mathcal{N}\Big(y_i; f(x_i, w), \sigma^2\Big)\Big) + \sum_{k=1}^{\nu} \ln\Big(\mathcal{N}(f(x_k, w); f(x_k, w^{t-1}), \sigma_0^2)\Big) - \ln(\mathcal{D}).\end{aligned}$$

Again, we switch to density notation and get

$$\begin{aligned}\ln\big(p(w \mid \mathcal{D})\big) &= -\frac{1}{2\sigma^2}\sum_{i=1}^{n_t}(f(x_i^t, w) - y_i)^2 - \frac{1}{2\sigma_0^2}\sum_{k=1}^{\nu}(f(x_k, w) - f(x_k, w^{t-1}))^2 + \text{const}\\ &= -\Big(\frac{1}{2\sigma^2}\sum_{i=1}^{n_t}(f(x_i^t, w) - y_i)^2 + \frac{1}{2\sigma_0^2}\sum_{k=1}^{\nu}(f(x_k, w) - f(x_k, w^{t-1}))^2\Big) + \text{const}\end{aligned}$$

In order to find the MAP estimate the expression on the right-hand side of the equation has to be maximized. But instead of maximizing the expression, we can equivalently minimize

$$E(w) = \sum_{i=1}^{n_t}(f(x_i^t, w) - y_i)^2 + \lambda\sum_{k=1}^{\nu}(f(x_k, w) - f(x_k, w^{t-1}))^2, \tag{6}$$

where λ is defined via $\lambda = \frac{\sigma^2}{\sigma_0^2}$, which follows from rearranging terms.

Algorithm 1: Generative Bayesian Neural Network

```
Require: Preprocessed option data $\mathcal{D} = \{(x_i^t, y_i^t) \mid i = 1, \dots, n_t, t = 1, \dots, l\}$, prior
  samples $\{(x_k, f(x_k, w^0)) \mid k = 1, \dots, \nu\}$ generated by the market model where
  $f(x_k, w^0)$ is the option value calculated by the market model
Output: Trained GBNN
for $t = 1 : l$ do
    Step 1: Prior sampling
    Generate $\nu$ prior samples $\mathcal{S} = \{(x_k, f(x_k, w^{t-1})) \mid k = 1, \dots, \nu\}$ with the help of
      the weights from the previous period
    Step 2: Training of GBNN
    First: Merge intraday data $\mathcal{D}_t = \{(x_i^t, y_i^t) \mid i = 1, \dots, n_t\}$ with generated prior
      samples $\mathcal{S}$: $\tilde{\mathcal{D}} = \mathcal{D}_t \cup \mathcal{S}$
    Second: Optimize the network weights $w$ according to the error function (6)
      using the merged dataset $\tilde{\mathcal{D}}$
end
```

Having now seen the idea of the model and the theoretical justification, we now can summarize all of the above with the help of Algorithm 1.

The question now still remains what kind of neural network model for f to choose. As already discussed in previous sections, we addressed this by doing a grid search and comparing different model combinations. Eventually we ended up with a two hidden layer network with 128 and 64 hidden nodes respectively. Furthermore, the learning rate is set to 0.001 and again the ADAM optimizer is used. For λ a value of 0.9 worked best (we also experimented with a dynamic λ, see Section A.1). Additionally, since the first training step sets the foundation for everything else, we decided to set the epochs to 1000 for this trading day and for every other that follows decrease this number to 100. Regarding the weight initialization, we used the same as for the BNN, since also here the ReLU activation function was used within the network. For prior sampling, we generated $\nu = 50$ ($\nu = 200$ for the first day) option prices in total per trading day by using a simple linspace for different levels of moneyness for each of the maturities available for trade that day.

2.4.1 Generative Fully Bayesian Neural Network

Given that within one trading day, only a limited amount of ATM option data is available, (ITM and OTM data is sampled by the user), we decided to also use the fully Bayesian neural network in place of the Bayesian one, since the fully Bayesian is very data efficient as they can learn from limited data without overfitting (Jospin et al. (2020)). The algorithm itself remains the same, but instead of f being a BNN, it is now a FBNN, as already defined in section 2.3.3, with the same prior distribution and posterior distribution. Therefore, we also named it Generative fully Bayesian neural network, short GFBNN.

For this network, we used the same hyperparameter tuning as usual. We ended up having a network with two hidden layers with each having 8 hidden nodes. The learning rate was set to 0.001, RMSprop was used as the optimizer, and the epochs for the first trading day were 3500 and for every other trading day 800, because overfitting is very hard to achieve with these kind of networks. Furthermore, we chose $\nu = 100$ ($\nu = 200$ for the first day). Similarly to Section 2.3.3, we experimented with different loss functions and using Equation (6) yielded the best results.

3 Empirical Results

With all the models introduced and their corresponding ideas discussed from a theoretical point of view in the previous chapter, we now want to compare their practical capabilities. For this we start off with describing the data we use, move on to how we measure a models performance and conclude by presenting the results we achieved.

3.1 Data

In this paper, S&P 100 index options are used. The S&P 100 index is a weighted index of United States stocks maintained by Standard & Poor‘s. The index is a subset of the well known S&P 500 index and consists (at the moment of writing) of companies such as Alphabet, JPMorgan and Oracle.

The S&P 100 index options offer two different types: American options (which allow for exercising before maturity) and European options (which only can be exercised at the time of maturity). The American type is also known as OEX and the European type as XEO, both of which are traded on the Chicago Board Options Exchange (CBOE).

In this research, OEX put option data from 01-01-2003 until 31-12-2012 is used. The option data consists of the expiration date, strike price, best bid, best offer among others and is obtained from OptionMetrics. We work with time to maturity in trading days (250 in a year). The moneyness and time value of the options are calculated as follows

$$\begin{aligned}\text{Moneyness} &= \frac{\text{underlying}}{\text{strike price}},\\ \text{Time value} &= \text{market price} - (\text{strike price} - \text{underlying}),\end{aligned}$$

where the market price is chosen as the average between the bid and ask.

The time value, as calculated above, is used to clean the option data by the conventional data preprocessing steps in the literature. The applied data cleaning steps are mainly taken from Ruf and Wang (2020a), combined with those in Jang and Lee (2019), meaning the following samples are removed:

- Samples where the maturity is < 7 or > 90.
- Samples where the implied volatility is unknown.
- Samples where the implied volatility is ≤ 0.01 or ≥ 1.
- Samples where the volume is ≤ 0.
- Samples where bid ≤ 0.05.
- Samples where time value ≤ 0.
- Samples where offer $\geq 2 \times$ bid.
- Samples where the next trade price is not available.
- Samples which are not traded on the current date.

Table 3: Summary statistics of the S&P American put option data from 2003 to 2012

		Maturity							
		<30		30-60		>60		All	
Moneyness		Mean	Std. dev.	Mean	Std. dev.	Mean	Std. dev.	Mean	Std. dev.
<0.94	Price	58.52	27.26	67.75	40.51	75.86	42.0	65.4	36.43
	Observations	1900		2064		920		4677	
0.94-0.97	Price	27.36	5.57	30.24	6.03	34.31	7.01	29.55	6.43
	Observations	2892		2804		916		6351	
0.97-1.00	Price	13.62	4.88	17.67	5.71	21.44	6.71	16.65	6.25
	Observations	6061		6772		2477		14737	
1.00-1.03	Price	5.72	3.83	10.47	5.17	14.43	6.24	9.51	5.93
	Observations	6395		8042		3428		17186	
1.03-1.06	Price	2.79	2.9	6.31	4.41	9.75	5.49	5.65	4.87
	Observations	5747		7228		2855		15234	
>1.06	Price	1.42	1.92	2.91	3.0	4.49	3.99	2.82	3.19
	Observations	13507		23782		10012		45350	
All	Price	9.44	15.41	10.72	16.61	13.42	18.62	10.78	16.59
	Observations	36487		50674		20600		103496	

The reason for removing these options is that they can lead to distortions in the experiments. For example short time-to-maturity options tend to be problematic because of low time premium and the bid-ask spread (Jang and Lee (2019)). After applying the above mentioned preprocessing steps, we are left with 103.496 observations and summary statistics of the remaining option data is given in Table 3. Also, the trading volume (per year) is shown in Table 4. We should note that this data is not the same as Jang and Lee (2019) (where the data preprocessing is not fully explained), but the dataset has similar characteristics. However, one important difference is that our dataset contains significantly more OTM options.

Table 4: Trading volume of the S&P American put option data from for each year with respect to moneyness and time

	Trading volume by moneyness			Trading volume by maturity			
	< 0.97	0.97-1.03	>1.03	< 30	30-60	> 60	All
Year	ITM	ATM	OTM				
2003	1232	2873	6258	3160	4949	2254	10363
2004	615	3193	4845	2577	4185	1891	8653
2005	505	3301	4038	2403	3725	1716	7844
2006	364	3541	4274	2292	3948	1939	8179
2007	802	3873	6219	3368	5351	2175	10894
2008	2742	3191	7469	4541	6480	2381	13402
2009	1730	2507	8165	4034	6253	2115	12402
2010	1461	3009	7499	3955	6113	1901	11969
2011	1012	3138	6872	4200	5407	1415	11022
2012	562	3270	4936	3358	4263	1147	8768

3.2 Performance measurements

To measure the performance of the different models we distinguish two types of performance, (in-sample) estimation and (out-of-sample) prediction. Estimation refers to the model predicting values it has already seen during training, which shows how well the model can describe existing data/behavior. Prediction on the other hand refers to the model predicting values it has not seen, typically those that come after the training set chronically. Given a value y_i and the model's estimation/prediction $\hat{y}_i$, we define the error $\varepsilon_i = y_i - \hat{y}_i$, for observations $i = 1, ..., N$. With this, we define two performance measurements:

$$\text{MAPE} = \frac{1}{N}\sum_{i=1}^{N} \frac{|\varepsilon_i|}{y_i}$$

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} \varepsilon_i^2}$$

where MAPE stands for mean absolute percentage error and RMSE for root mean squared error. Note that the MAPE is typically more sensitive to options with a smaller option value, i.e. OTM options, with the opposite being true for the RMSE. Each metric will be discussed for OTM, ATM and ITM options seperately as well as for all options together.

We split the performance measures into three time windows, pre-crisis 2003-2006, crisis 2007-2009 and post-crisis 2010-2012. Estimation and prediction is generally done in an expanding window type way. The model is trained on all data up until the last 6 months of the current time window (so for example pre-crisis data is included in the training for crisis), after which the last 6 months are predicted and used for calculating the prediction performance. Next, we train the model on all data until the end of the current time window, after which all data from the current time window is estimated to track the estimation performance. This method prevents look-ahead biases. Two notable exceptions are the Heston model and the SVR. For Heston, the model is calibrated on the S&P 100 index and all European options traded between 2003 and 2012 (hence this approach does have a look-ahead bias[4]), therefore there is no distuingishment between prediction and estimation, and we report the values from the Heston model in both tables. For the SVR, training on over 100.000 datapoints is computationally infeasible, as discussed in Section 2.3.1, thus we choose to discard data from previous time windows for this method.

With this approach come three remarks. First, the choice for expanding window over a rolling window is fairly arbitrary. Generally for deep learning techniques more data is better, but only under the assumption that there is no structural breaks in the underlying data generating process. Given that the dataset contains a crisis, this assumption is unrealistic, but it is hard to say whether this is enough reason to justify the loss of training data from switching to a rolling window approach (could potentially be investigated in further studies). Second, from an investors point of view, estimation and prediction could be done on a daily basis, i.e. the model is trained up to a day, after which estimation for the same day and prediction for the next day is done. However, this procedure is too computationally expensive for our current implementations and limited time. Especially for models that cannot be continually trained, but instead have to start over on each day (in our case Heston and SVR), so to keep comparison fair, we use the same procedure described above for each model. However, because this daily approach could significantly improve performance, we include it for the best performing model in the appendix (Section A.2). Lastly, Jang and Lee (2019) work with the same 3 time windows, but do not explicitly state the approach chosen to do estimation and prediction, making a direct comparison difficult.

3.3 Model performance

Before diving deep into the performance measures of each model, one important thing to mention is that the calibration of the Heston model yields option prices which are a little off, especially for ITM and OTM options. Since this lays out the foundation for the GBNN and GFBNN due to the first step in the algorithm, the results are most likely worse than usual as if one had a better calibrated model. Due to time constraints, we were not able to further improve the calibration of the model. So to partly solve this issue, we also include actual ITM and OTM data (amounts to very few data points per day) into the generative networks, which improved training stability, but on the other hand does not guarantee us arbitrage free prices anymore. These issues become clearer when we take a deeper look at the actual errors, because there we will see that the MAPE is usually high for the GBNN and GFBNN for OTM options while the RMSE is high for both for ITM options. This needs to be kept in mind while looking at the tables.

We start off with the estimation performance, where an overview is given in Table 5. As previously denoted, for the estimation performance, an expending window type way is used except for the SVR model because of computational reasons.

[4]Calibration of the Heston model is computationally expensive, therefore we only do it once, and thus use the full sample

Table 5: Estimation performance

Estimation error during the pre-crisis, from 2003 to 2006.

	MAPE				RMSE			
Model	ITM	ATM	OTM	All	ITM	ATM	OTM	All
AH-BS	0.1447	0.6555	0.9240	0.7647	7.6435	5.0746	2.6993	4.2488
Heston*	0.2756	0.5703	0.9052	0.7330	8.5406	6.2061	2.7970	4.9171
SVR*	0.0248	0.1880	0.3911	0.2879	1.3186	2.4598	1.6240	1.9558
BNN	0.0511	0.2411	0.6254	0.4393	2.4808	2.5411	1.7402	2.1293
FBNN	0.1079	0.2940	0.8671	0.5971	4.8716	2.7166	1.9232	2.5705
GBNN	0.0971	0.2016	1.0822	0.6814	7.4975	2.8739	2.3369	3.2289
GFBNN	0.1557	0.2161	1.1083	0.7058	7.3294	2.8151	2.2431	3.1419

Estimation error during the financial-crisis, from 2007 to 2009.

	MAPE				RMSE			
Model	ITM	ATM	OTM	All	ITM	ATM	OTM	All
AH-BS	0.2317	0.4717	0.9425	0.7176	18.8310	7.1930	4.6597	8.7969
Heston*	0.1472	0.3726	0.9665	0.6939	7.2224	6.9228	6.1995	6.5485
SVR*	0.0760	0.2858	0.5666	0.4229	5.3421	4.9628	3.4916	4.2172
BNN	0.1191	0.3016	0.7782	0.5592	9.4651	6.0387	4.1301	5.7046
FBNN	0.1278	0.3296	0.9692	0.6815	11.0497	5.5933	4.6731	6.2217
GBNN	0.1644	0.3332	0.6190	0.4791	11.2890	6.6794	4.1815	6.3532
GFBNN	0.1411	0.3201	0.7699	0.5622	18.0693	6.2487	4.5214	8.3234

Estimation error during the post-crisis, from 2010 to 2012.

	MAPE				RMSE			
Model	ITM	ATM	OTM	All	ITM	ATM	OTM	All
AH-BS	0.1492	0.6351	1.0059	0.8141	8.9260	7.0103	3.7282	5.5349
Heston*	0.3054	0.4287	0.9586	0.7390	11.3282	6.5980	4.3161	6.0413
SVR*	0.0367	0.1413	0.3279	0.2448	2.0875	2.7903	1.9907	2.2659
BNN	0.0697	0.1783	0.5956	0.4216	3.8853	3.1521	2.2547	2.7348
FBNN	0.1034	0.2453	0.7326	0.5280	5.9372	3.9959	2.8588	3.6155
GBNN	0.1348	0.3250	0.4835	0.4032	7.9738	3.4380	2.2457	3.5562
GFBNN	0.0777	0.2597	0.8690	0.6127	4.6587	3.0891	3.0632	3.2570

Estimation performance of the different models evaluated as described in Section 3.2. The moneyness is divided into ITM (moneyness < 0.97), ATM (moneyness 0.97-1.03) and OTM (moneyness > 1.03).
*The Heston and SVR model use a different estimation/prediction procedure, see Section 3.2

Looking at the estimation errors during the pre-crisis in Table 5, it becomes clear that the SVR model performs the best based on all options for the MAPE (0.2879) and RMSE (1.9558), but this might be due to overfitting since the prediction performance for the corresponding period is already considerably worse. For the other periods we will not consider the SVR's performance due to already mentioned reasons and would lead otherwise to an unjust comparison. The GBNN and GFBNN seem to be outperformed by the BNN and FBNN in both measurements. But looking closer at the errors we see that the reason for this stems from the errors for ITM and OTM options. As mentioned before, this is most likely due to the option prices provided by the Heston model in the first step of Algorithm 1.

Regarding the financial crisis period, the MAPE for all options shows that the BNN and GBNN model perform relatively well with a MAPE of 0.5592 and 0.4791, respectively. This

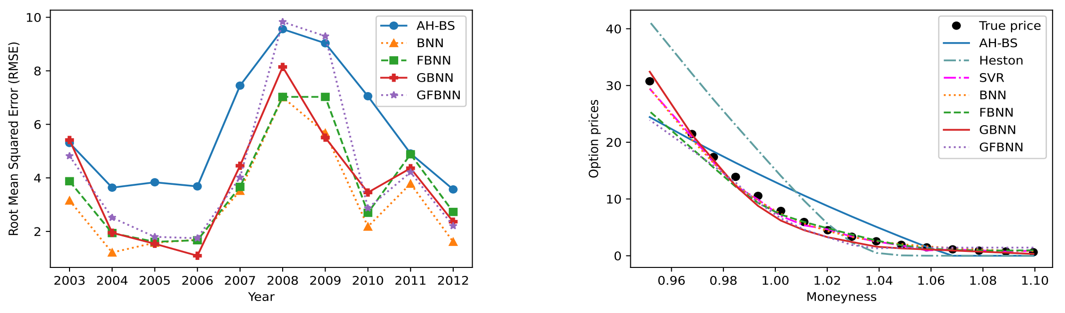

Figure 2: Yearly estimation RMSE (left) and estimated prices by different models (right)

is mainly due to the relatively low estimation error for OTM options, for which MAPE is very sensible. The RMSE performance measure also confirms the good performance for the BNN model although it does not have the best results for each option category. Furthermore, the GBNN is not too far behind since a difference of approximately 0.5 for the RMSE is not immense.

Considering now the post-crisis period, we can see that based on the MAPE performance measure, the BNN model performs well for ITM and ATM options. The GBNN model exhibits a solid performance for OTM options and therefore its overall performance is good. Looking at the RMSE, the BNN model performs the best for ITM, ATM and all moneyness level options. For OTM options, the GBNN performs slightly better than the BNN model.

From Figure 2 we get two more insights into the estimation performance of the different models. From the plot on the left-hand side in that figure we see that the BNN has the lowest RMSE based on yearly estimations. The other plot tells us how well the different models estimate option prices on a given date and a given maturity. Options plotted there are traded in the first month of 2004, with a maturity of 30 days. We see that the BNN and GBNN seem to have learned the option prices the best. Interesting to note is that the BNN is more off the further the option is ITM. This is probably due to the fact that there is little data to learn from in that area. Nevertheless we can conclude, that the BNN model seems to be a valid candidate for its estimation performance. But by considering problems mentioned before, the GBNN still manages to perform very well and we believe that there is definitely room for improvement.

Turning our attention now towards the prediction performance, for which the results are summarized in Table 6, we see right away that the GBNN outperforms every other model regarding both measurement criteria. Another thing to notice is that the GFBNN also seems to have improved its standing considering its relative bad performance in the estimation section. The BNN and FBNN exhibit a worse performance than for the respective estimation section. One might think that this is due to overfitting, which we cannot confirm since we also tested models on validation sets to avoid overfitting.

For the crisis-period the GBNN has an incredible MAPE but not such a great RMSE. This is due to the immense RMSE for the ITM options. The same holds for the GFBNN, which even more mispriced ITM options. Furthermore, the FBNN does not seem to be up for

Table 6: Prediction performance

Prediction error during the pre-crisis, from 2003 to 2006.

	MAPE				RMSE			
Model	ITM	ATM	OTM	All	ITM	ATM	OTM	All
AH-BS	0.4568	0.5986	0.9566	0.7794	14.9483	5.8512	1.9043	4.9307
Heston*	0.3661	0.7687	0.8478	0.7970	10.1597	7.1658	1.4042	5.2367
SVR*	0.1063	0.3372	1.2899	0.8229	8.2536	2.6560	2.0790	2.7476
BNN	0.0731	0.4951	1.3938	0.9469	3.3234	3.0895	1.8584	2.5378
FBNN	0.1119	0.4824	0.7334	0.6006	4.7709	2.3778	1.3955	2.0666
GBNN	0.0685	0.1163	0.5580	0.3434	2.7866	0.9163	0.6759	0.9263
GFBNN	0.1972	0.2025	1.4088	0.8265	7.6823	1.1723	0.9404	1.7094

Prediction error during the financial-crisis, from 2007 to 2009.

	MAPE				RMSE			
Model	ITM	ATM	OTM	All	ITM	ATM	OTM	All
AH-BS	0.3731	1.5464	2.8732	2.3269	23.0709	17.4340	9.7628	13.6181
Heston*	0.2723	0.2872	0.9716	0.7468	10.2869	4.3127	3.7058	4.8731
SVR*	0.2667	0.2340	3.7183	2.5844	57.0133	3.6286	5.4609	18.3617
BNN	0.1947	0.1407	0.9540	0.6938	16.0931	2.3304	1.5409	5.2790
FBNN	0.1605	0.1436	2.6347	1.8234	18.0752	2.0156	2.4466	6.0440
GBNN	0.1357	0.0763	0.3440	0.2624	22.8218	1.1836	0.6757	7.1378
GFBNN	0.1835	0.1229	0.7535	0.5536	32.9438	1.6707	1.1076	10.3111

Prediction error during the post-crisis, from 2010 to 2012.

	MAPE				RMSE			
Model	ITM	ATM	OTM	All	ITM	ATM	OTM	All
AH-BS	0.5130	0.6090	0.9733	0.7961	22.0771	7.6157	3.3892	7.5637
Heston*	0.4369	0.5473	0.9240	0.7402	15.3959	7.7987	2.7082	6.5056
SVR*	0.1627	0.1887	0.9917	0.6115	16.7541	2.2380	2.6353	4.6154
BNN	0.1607	0.1352	0.7288	0.4503	7.2774	1.6411	1.1310	2.1750
FBNN	0.2536	0.2858	0.8392	0.5764	10.5218	2.7273	1.1853	3.1576
GBNN	0.0938	0.1031	0.3731	0.2452	6.0479	1.0399	0.7662	1.6692
GFBNN	0.0408	0.1778	0.7253	0.4595	4.0219	1.3117	1.3433	1.6018

Prediction performance of the different models evaluated as described in Section 3.2. The moneyness is divided into ITM (moneyness < 0.97), ATM (moneyness 0.97-1.03) and OTM (moneyness > 1.03).
*The Heston and SVR model use a different estimation/prediction procedure, see Section 3.2

discussion since it managed to misprice OTM options by more than twice the value, which might even give room for arbitrage. For the BNN model we kind of see a similar pattern as for the others, high MAPE for OTM options whereas a high RMSE for ITM options and therefore it seems to be rather in a middle position.

In the last period up for discussion, the GBNN again shows a remarkable performance. It has by far the lowest overall MAPE, but also the lowest MAPE for each individual option category. The same almost also holds for the RMSE, where it only looses to the GFBNN model for ITM options. Apart from that, we note that the GFBNN also managed to kind of come back and seems to have lost only to the GBNN regarding performance. The BNN and FBNN show in this period their best prediction performance, but still did not stand a chance eventually.

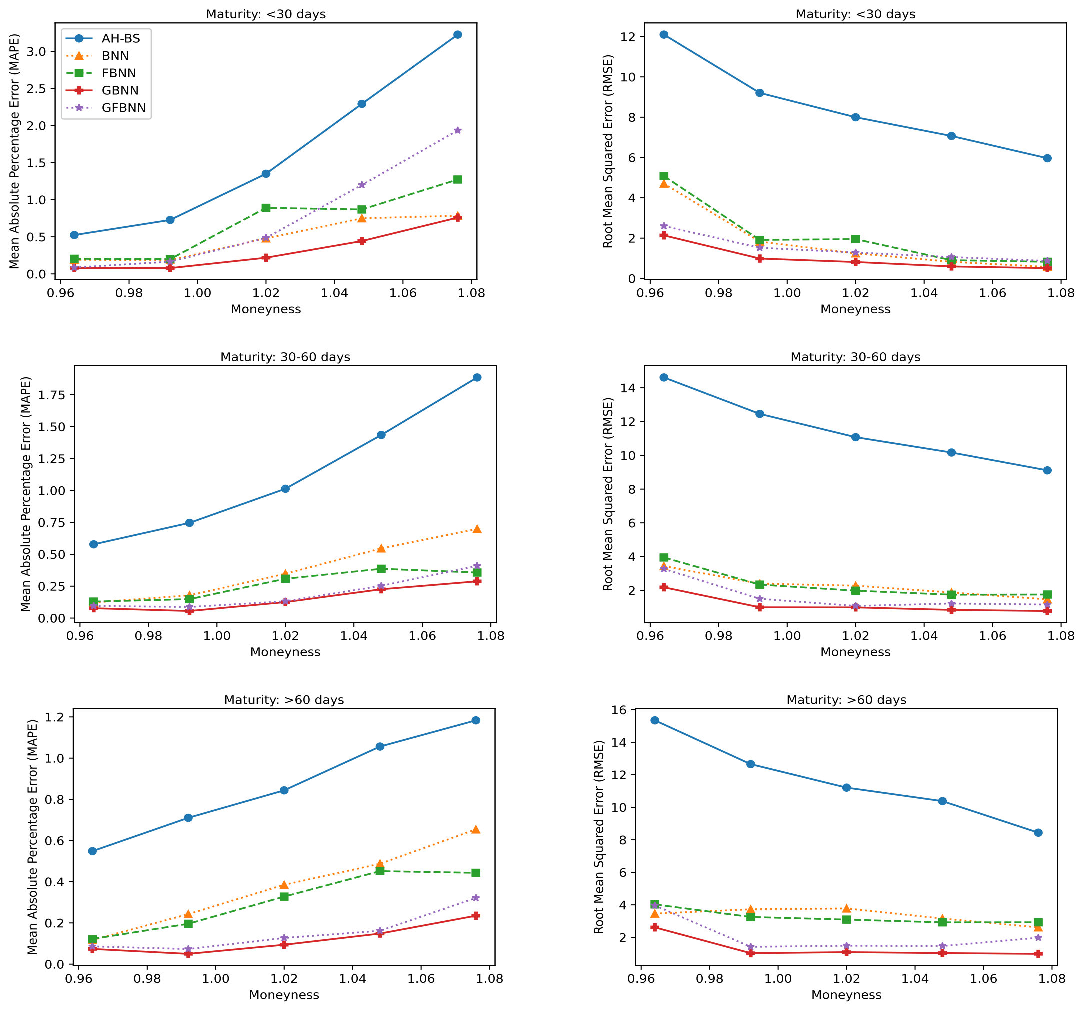

Figure 3: prediction MAPE (left) and prediction RMSE (right) performance for different levels of moneyness and maturity for each model, excluding Heston and SVR

To get even a better idea of the prediction performance, we take a look at the two metrics for options of different maturities. For this we split the maturity into three categories: maturity below 30 days, between 30 and 60 days and above 60 days. The results are summarized in Figure 3. Here again we see the impressive prediction performance by the GBNN, which manages to have the lowest error in each category for each of the two measurements. As a close runner up we have the GFBNN, which is not too surprising because it is based on the same algorithm. A common pattern we see in general is that MAPE is higher for OTM options and lower for ITM options whereas it is for the RMSE the other way around. This structure was already explained in Section 3.2.

Eventually, we can say that the GBNN is the best model for prediction given that it is superior to the other models in almost every category. The GFBNN also managed to do very well, but since its benchmark is the GBNN, it can not be considered as a solid candidate given the state it is in.

Table 7: Approximate training time and pricing time for each model

Model	Training time	Option pricing time
AH-BS	1 minute	0.00001 seconds
Heston	1 hour	0.04172 seconds
SVR	15 minutes	0.00123 seconds
BNN	10 minutes	0.00001 seconds
FBNN	45 minutes	0.00009 seconds
GBNN	30 minutes	0.00005 seconds
GFBNN	5 hours	0.00265 seconds

Training time refers to training on all time windows combined. Pricing time refers to average pricing time of one option, using the trained models.

Before we reach a final conclusion, we take a quick look at the training time and option pricing time. The respective numbers can be found in Table 7. From this we see that the GBNN does very well in both categories whereas the GFBNN takes by far the longest to train.

4 Conclusion

Having seen the results for the estimation and prediction performance as well as computational aspects, we can now sum up the findings and reach a conclusion. The GBNN model was presented as a new option pricing technique which is supposed to be superior to others. Given the results we achieved, we support this point of view for several reasons. First, it beat every other model when it came to prediction performance. Even though it did not achieve the greatest results in the estimation section, it was also not too far off from the best models. Second, the GBNN yielded better results for the prediction section than for the estimation section, which is clearly a sign that it is not overfitting to data, whereas for other models it was the other way around. Lastly, if we remind ourselves of the issue, that the GBNN was originally fed market prices for ITM and OTM options from the Heston model, which were rather off, next to a few market data points, it is remarkable that it still performed so well. With a better calibrated market model, one can expect even better results.

Apart from that, our idea of using fully Bayesian neural networks did not manage to beat the other models. They were inferior to their respective underlying model ideas and were computationally much more expensive. Even though, we believe that for the BFNN and the GFBNN there is lots of room for improvement, it is questionable they will reach the level of the BNN and GBNN. We can not answer this question with full confidence, since we also lacked computational power and had limited time.

4.1 Discussion

The models discussed throughout the paper are quite delicate in their calibration/tuning. Therefore, employing the same techniques to different datasets might lead to wildly varying results. This also means that the results obtained should be evaluated carefully, as there might exist combinations of parameters that could still vastly improve performance of certain models. Especially the fully Bayesian neural networks could have room for improvement, as their large computation time did not allow us to experiment the full parameter space.

Moreover, the Heston model calibration is also very unstable. For this reason, different tuning methods, or a completely different (and more sophisticated) market model, would be worth considering. Especially considering it lies at the basis of data generation for the generative networks. In Jang and Lee (2019), the CGMY model (Carr et al. (2003)) is considered for data generation. We opted for the Heston model because the CGMY model is even more difficult to calibrate, and our main focus is on deep learning techniques.

The central paradigm in classical option pricing literature is No Arbitrage (NA). With our implementation of deep learning techniques, NA conditions are ignored. Jang and Lee (2019) suggest the generative neural network models are expected to learn prices that satisfy NA conditions, by incorporating prior information from an arbitrage-free model. However, we are a bit more reserved with this type of claim, especially given that our Heston model calibration is not necessarily ideal. Moreover, one should be very careful in using the model for options which have different characteristics (for example a different index), as prices might be wildly off and there is no guarantee of NA.

Comparing our findings to the reference paper by Jang and Lee (2019) is a difficult task, even though we employ the same methods on the same dataset. This is because Jang and Lee (2019) are not very explicit in discussing the exact implementations used. Therefore, there could be differences in data preprocessing, parameter tuning, chosen parameters and methods of estimation and prediction among others. Consequently, attempts to compare results 1 on 1 are very far-fetched. Nevertheless, we did show in Section A.2 that our GBNN model with daily estimation and prediction could closely match the performance of the corresponding GBNN model in Jang and Lee (2019).

For further research, investigating different, more sophisticated tuning methods would be a very interesting direction. Furthermore, trying to incorporate NA conditions into the deep learning models is also an interesting area of research, that could be experimented within this setup. Moreover, as deep learning models are flexible in what can be used as input data, it could be interesting to see if additional inputs could improve performance (also suggested in Jang and Lee (2019)). Examples of such inputs include VIX data, other measures of volatility like implied- or realized volatility and sentiment data obtained from natural language processing techniques.

References

AitSahlia, F., Goswami, M., and Guha, S. (2010). American option pricing under stochastic volatility: an empirical evaluation. *Computational Management Science*, 7(2):189–206.

Amornwattana, S., Enke, D., and Dagli, C. H. (2007). A hybrid option pricing model using a neural network for estimating volatility. *International Journal of General Systems*, 36(5):558–573.

Anders, U., Korn, O., and Schmitt, C. (1998). Improving the pricing of options: A neural network approach. *Journal of forecasting*, 17(5-6):369–388.

Awad, M. and Khanna, R. (2015). Support vector regression. In *Efficient learning machines*, pages 67–80. Springer.

Barone-Adesi, G. and Whaley, R. E. (1987). Efficient analytic approximation of american option values. *The Journal of Finance*, 42(2):301–320.

Bates, D. S. (1996). Jumps and stochastic volatility: Exchange rate processes implicit in deutsche mark options. *The Review of Financial Studies*, 9(1):69–107.

Berkowitz, J. (2009). On justifications for the ad hoc black-scholes method of option pricing. *Studies in Nonlinear Dynamics & Econometrics*, 14(1).

Black, F. and Scholes, M. (1973). The pricing of options and corporate liabilities. *Journal of Political Economy*, 81(3):637–654.

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (2017). Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877.

Bollerslev, T., Engle, R. F., and Nelson, D. B. (1994). Arch models. *Handbook of econometrics*, 4:2959–3038.

Bottou, L. and Lin, C.-J. (2007). Support vector machine solvers. *Large scale kernel machines*, 3(1):301–320.

Buehler, H., Gonon, L., Teichmann, J., and Wood, B. (2019). Deep hedging. *Quantitative Finance*, 19(8):1271–1291.

Burder, F. and Winkler, D. (2008). *Bayesian regularization of neural networks.* Artificial Neural Networks: Methods and Applications.

Carr, P., Geman, H., Madan, D. B., and Yor, M. (2003). Stochastic volatility for lévy processes. *Mathematical finance*, 13(3):345–382.

Carr, P. and Madan, D. (1999). Option valuation using the fast fourier transform. *Journal of computational finance*, 2(4):61–73.

Chen, S.-H. and Lee, W.-C. (1999). Pricing call warrants with artificial neural networks: the case of the taiwan derivative market. In *IJCNN'99. International Joint Conference on Neural Networks. Proceedings (Cat. No. 99CH36339)*, volume 6, pages 3877–3882. IEEE.

Chen, Y. and Wan, J. W. (2021). Deep neural network framework based on backward stochastic differential equations for pricing and hedging american options in high dimensions. *Quantitative Finance*, 21(1):45–67.

Cox, J. C., Ross, S. A., and Rubinstein, M. (1979). Option pricing: A simplified approach. *Journal of financial Economics*, 7(3):229–263.

Engle, R. and Lee, G. (1996). *Modeling stock market volatility.* Academic Press, Inc, New York.

Gourieroux, C., Monfort, A., and Renault, E. (1993). Indirect inference. *Journal of applied econometrics*, 8(S1):S85–S118.

Hagan, P. S., Kumar, D., Lesniewski, A. S., and Woodward, D. E. (2002). Managing smile risk. *The Best of Wilmott*, 1:249–296.

Han, G.-S. and Lee, J. (2008). Prediction of pricing and hedging errors for equity linked warrants with gaussian process models. *Expert Systems with Applications*, 35(1-2):515–523.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034.

Heston, S. (1993). A closed-form solution for options with stochastic volatility with applications to bond and currency options. *Review of Financial Studies*, 6:327–343.

Hutchinson, J. M., Lo, A. W., and Poggio, T. (1994). A nonparametric approach to pricing and hedging derivative securities via learning networks. *The Journal of Finance*, 49(3):851–889.

Jang, H. and Lee, J. (2019). Generative bayesian neural network model for risk-neutral pricing of american index options. *Quantitative Finance*, 19(4):587–603.

Jospin, L. V., Buntine, W., Boussaid, F., Laga, H., and Bennamoun, M. (2020). Hands-on bayesian neural networks – a tutorial for deep learning users.

Kazem, A., Sharifi, E., Hussain, F. K., Saberi, M., and Hussain, O. K. (2013). Support vector regression with chaos-based firefly algorithm for stock market price forecasting. *Applied soft computing*, 13(2):947–958.

Kelly, D. L., Shorish, J., et al. (1994). Valuing and hedging american put options using neural networks. *Unpublished manuscript, Carnegie Mellon University.*

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980.*

Kou, S. G. (2002). A jump-diffusion model for option pricing. *Management science*, 48(8):1086–1101.

Lajbcygier, P. R. and Connor, J. T. (1997). Improved option pricing using artificial neural networks and bootstrap methods. *International journal of neural systems*, 8(04):457–471.

Longstaff, F. A. and Schwartz, E. S. (2001). Valuing american options by simulation: a simple least-squares approach. *The review of financial studies*, 14(1):113–147.

MacKay, D. (1994). A practical bayesian framework for backpropagation networks. *Neural Computation*, 14:448–472.

Meissner, G. and Kawano, N. (2001). Capturing the volatility smile of options on high-tech stocks—a combined garch-neural network approach. *Journal of economics and finance*, 25(3):276–292.

Pires, M. M. and Marwala, T. (2004). American option pricing using multi-layer perceptron and support vector machine. In *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No. 04CH37583)*, volume 2, pages 1279–1285. IEEE.

Pires, M. M. and Marwala, T. (2005). American option pricing using bayesian multi-layer perceptrons and bayesian support vector machines. In *IEEE 3rd International Conference on Computational Cybernetics, 2005. ICCC 2005.*, pages 219–224. IEEE.

Ruf, J. and Wang, W. (2020a). Hedging with linear regressions and neural networks. *arXiv:2004.08891.*

Ruf, J. and Wang, W. (2020b). Neural networks for option pricing and hedging: a literature review. *Journal of Computational Finance, Forthcoming.*

Vapnik, V. (2013). *The nature of statistical learning theory.* Springer science & business media.

Wang, P. (2011). Pricing currency options with support vector regression and stochastic volatility model with jumps. *Expert Systems with Applications*, 38(1):1–7.

Yao, J., Li, Y., and Tan, C. L. (2000). Option price forecasting using neural networks.

Omega, 28(4):455–466.

Zhang, J. E. and Shu, J. (2003). Pricing s&p 500 index options with heston's model. In *2003 IEEE International Conference on Computational Intelligence for Financial Engineering, 2003. Proceedings.*, pages 85–92. IEEE.

Zheng, Y., Yang, Y., and Chen, B. (2019). Gated neural networks for implied volatility surfaces. *arXiv preprint arXiv:1904.12834*.

A Appendix

A.1 Dynamic λ

While the parameter λ is statically chosen in the generative networks, it might be more realistic that it would change over time. To investigate this, we do a small simple experiment, where we let λ linearly increase (decrease) between 0.1 (1.7) and 1.7 (0.1) (so that the reference value 0.9 is in the middle) over time. Results of this experiment are shown in Table 8. Here we observe that these dynamic choices for λ do not influence the results by much, and that deviations are just as likely to be from randomness during the optimization. Of course the chosen schemes are very simplistic, and more sophisticated, data-driven approaches would be preferred.

Table 8: Different λ schemes

Error during the pre-crisis, from 2003 to 2006.

	MAPE				RMSE			
Model	ITM	ATM	OTM	All	ITM	ATM	OTM	All
GBNN incr. λ est.	0.0810	0.2122	0.5434	0.3855	10.3129	2.7947	1.9583	3.6395
GBNN decr. λ est.	0.0795	0.2254	1.7564	1.0624	8.1133	2.7424	3.3489	3.7533
GBNN cons. λ est.	0.0971	0.2016	1.0822	0.6814	7.4975	2.8739	2.3369	3.2289
GBNN incr. λ pred.	0.0599	0.1163	0.6538	0.3926	2.7547	0.9587	0.7280	0.9624
GBNN decr. λ pred.	0.0765	0.1397	0.7129	0.4343	3.2722	0.9971	0.6580	1.0037
GBNN cons. λ pred.	0.0685	0.1163	0.5580	0.3434	2.7866	0.9163	0.6759	0.9263

Error during the financial-crisis, from 2007 to 2009.

	MAPE				RMSE			
Model	ITM	ATM	OTM	All	ITM	ATM	OTM	All
GBNN incr. λ est.	0.1606	0.3190	0.6847	0.5140	10.7473	6.3610	4.2155	6.1428
GBNN decr. λ est.	0.1615	0.3254	0.7278	0.5415	11.0802	6.5134	4.3201	6.3105
GBNN cons. λ est.	0.1644	0.3332	0.6190	0.4791	11.2890	6.6794	4.1815	6.3532
GBNN incr. λ pred.	0.1318	0.0821	0.3655	0.2778	20.7330	1.1847	0.6884	6.4939
GBNN decr. λ pred.	0.1344	0.0789	0.6287	0.4547	20.3246	1.2398	0.7899	6.3783
GBNN cons. λ pred.	0.1357	0.0763	0.3440	0.2624	22.8218	1.1836	0.6757	7.1378

Error during the post-crisis, from 2010 to 2012.

	MAPE				RMSE			
Model	ITM	ATM	OTM	All	ITM	ATM	OTM	All
GBNN incr. λ est.	0.1344	0.4139	0.8453	0.6495	6.6732	3.9752	2.2970	3.4855
GBNN decr. λ est.	0.1261	0.3275	0.6881	0.5275	6.1334	3.4516	2.2071	3.1763
GBNN cons. λ est.	0.1348	0.3250	0.4835	0.4032	7.9738	3.4380	2.2457	3.5562
GBNN incr. λ pred.	0.0611	0.1751	0.7719	0.4841	4.6720	1.4950	1.4270	1.7931
GBNN decr. λ pred.	0.1045	0.1029	0.3357	0.2260	6.2883	1.0041	0.7565	1.7066
GBNN cons. λ pred.	0.0938	0.1031	0.3731	0.2452	6.0479	1.0399	0.7662	1.6692

A.2 Daily estimation and prediction

Instead of doing all estimation and prediction with the model trained on the full sample, it is possible to estimate and predict after each day of training. This is more computationally expensive, therefore we only implement this for the GBNN model, which is the best performing model. Results of this are compared to the corresponding results of the GBNN in Jang and Lee (2019) in Table 9. From this table it is observed that this method of prediction is able to closely replicate the best results obtained in Jang and Lee (2019).

Table 9: GANN compared to Jang and Lee (2019)

Error during the pre-crisis, from 2003 to 2006.

	MAPE				RMSE			
Model	ITM	ATM	OTM	All	ITM	ATM	OTM	All
GBNN daily est.	0.0447	0.0516	0.3588	0.2213	2.9323	0.6552	0.5045	0.9827
Jang and Lee (2019) est.	0.0409	0.1098	0.1990	0.1426	2.1862	1.4548	1.0964	1.4378
GBNN daily pred.	0.0685	0.1163	0.558	0.3434	2.7866	0.9163	0.6759	0.9263
Jang and Lee (2019) pred.	0.0477	0.1139	0.2060	0.1490	2.2307	1.1336	0.7280	1.663

Error during the financial-crisis, from 2007 to 2009.

	MAPE				RMSE			
Model	ITM	ATM	OTM	All	ITM	ATM	OTM	All
GBNN daily est.	0.0373	0.0402	0.2917	0.1896	3.7791	0.8347	0.6402	1.5742
Jang and Lee (2019) est.	0.0403	0.1269	0.1770	0.1385	4.9511	3.5672	2.5193	3.4489
GBNN daily pred.	0.1357	0.0763	0.344	0.2624	22.8218	1.1836	0.6757	7.1378
Jang and Lee (2019) pred.	0.0608	0.1375	0.2373	0.1767	10.0039	3.9517	2.9400	5.4924

Error during the post-crisis, from 2010 to 2012.

	MAPE				RMSE			
Model	ITM	ATM	OTM	All	ITM	ATM	OTM	All
GBNN daily est.	0.0338	0.054	0.275	0.1864	3.8402	0.75	0.5543	1.3277
Jang and Lee (2019) est.	0.0345	0.1045	0.1813	0.1338	2.6295	1.7108	1.4304	1.7562
GBNN daily pred.	0.0938	0.1031	0.3731	0.2452	6.0479	1.0399	0.7662	1.6692
Jang and Lee (2019) pred.	0.0450	0.1142	0.2391	0.1667	3.6036	1.8372	1.6229	2.1386