PRICING OPTIONS AND COMPUTING IMPLIED VOLATILITIES USING NEURAL NETWORKS

Redwan Mekrami

Contents

1 Introduction

An option contract is a financial derivative that represents the right, but not the obligation, to buy (call) or sell (put) a particular security on (European type) or before (American type) an expiration date. Different well-known methods are available to price an option: binomial model, Monte Carlo method or using Black-Scholes formula. The choice of the accurate mathematical model and numerical method generally depends on the option features: path-dependent, early exercice or not, low or high dimension.

Nevertheless, these valuation methods make specific assumptions of the underlying dynamic and could mismatches empirical findings. For instance, the Black-Scholes model assumes constant underlying volatility and consequently fails to explain the relationship between the implied volatility and option moneyness. More generally, the multi-dimensionality of real-world derivative valuation problems does not always guarantee the existence of a closed-form formula to price a derivative.

In order to overcome that, deep learning algorithms have become popular by making a trade-off between accuracy and efficient numerical performance. Among them, Artificial Neural Networks (ANN) have the advantage of being able to accurately and efficiently approximate non-linear price relationships without making specific hypothesis regarding the underlying dynamic.

This project aims to reproduce the work introduced in [1] in which the authors propose a neural-network based approach to price European call options and compute implied volatilities. After having established the mathematical framework of this work (Section 2), we challenge the ANN approach (Section 3) with analytical solutions: Black-Scholes equation, COS method for the Heston stochastic volatility model and Brent's iterative root-finding method for the calculation of implied volatilities (Section 4).

2 Asset models and vanilla call pricing

In this section we introduce the mathematical tools that will be used as a benchmark and in order to generate the artificial market data (Black-Scholes and Heston models), as well as the numerical methods that will be used to price call options under the Heston model (COS method) and compute their associated implied volatility (Brent's method).

2.1 Black-Scholes model

A well-known model for the underlying price S_t is the Black-Scholes model [2] where S_t follows a geometric Brownian Motion which satisfies the following SDE

$$dS_t = S_t \left(r dt + \sigma dW_t \right) \tag{2.1}$$

where r is the risk-free rate, σ the volatility of the underlying asset and the process $(W_t)_{t \geqslant 0}$ is a Brownian Motion under the $\mathbb{Q}$ risk-neutral probability measure. By using a martingale approach or self-financing replicating portfolio, a European call option price must satisfies the PDE

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0 \tag{2.2}$$

with the final condition at time T representing the call option payoff

$$V(t=T,S)=(S_T-K)_+ \tag{2.3}$$

The solution to the equation (2.2) is given by Feymann-Kac formula that states the value at each time t of an European option can be expressed as the conditional expectation of the discounted expected values of the option's payoff function at maturity T, under the risk-neutral measure $\mathbb{Q}$

$$V(t,S)=e^{-r(T-t)}E_{\mathbb{Q}}\left[(S_T-K)_+|\mathcal{F}_t\right] \tag{2.4}$$

Solving the SDE (2.1) and developing the quantity (2.4) above leads to a closed-form formula for the call price V

$$V(t,S)=S\mathcal{N}\left(d_1\right)-Ke^{-r(T-t)}\mathcal{N}\left(d_2\right) \tag{2.5}$$

where

$$d_1=\frac{\log(S/K)+\left(r-0.5\sigma^2\right)(T-t)}{\sigma\sqrt{T-t}} \quad ,d_2=d_1-\sigma\sqrt{T-t} \tag{2.6}$$

Note that this model stays valid under the following hypotheses:

1. There is not arbitrage oppportunity
2. There are no bid-ask spread or transaction costs
3. Trading is continuous in all quantities and short-selling of securities is allowed
4. Risk-free rate r and volatility σ are supposed to be constant
5. The discounted underlying price follows a log-normal process $\ln\mathcal{N}\left(\left(r-\frac{\sigma^2}{2}\right)t,\sigma^2t\right)$
6. There are no dividend during the life of the option

2.2 Implied volatility

Implied volatility is a forecast of how variable the underlying price is expected to be over the remaining life of the option [2]. The forecast is calculated from an option price as the level of volatility gives the theoretical Black-Scholes option price equal to its market price

$$\sigma^{implied}(K,T)=BS^{-1}\left(V^{mkt};S,K,T-t,r\right) \tag{2.7}$$

where BS^{-1} represents the inverse of the Black-Scholes formula for an European call option (2.3) and V^{mkt} denotes a given observed market call option price. Unfortunately, inverse problems such as (2.5) do not have a closed-form solution. Instead, a root finding technique is often used to solve:

$$g\left(\sigma^{implied}\right)=BS\left(S,T-t,K,r,\sigma^{implied}\right)-V^{mkt}(S,T-t;K)=0 \tag{2.8}$$

Two of the most commonly used are Newton's method and Brent's method. Newton's method provides rapid convergence, as long as the first partial derivative of the option's theoretical value with respect to volatility (called vega) does not convergence to zero. However, this assumption is no fulfilled by deep in or out-the-money European call option and lead us to consider other approach, such as the Brent's method.

2.3 Brent's method

R. Brent [3] had established an algorithm that alternates between bisection, inverse quadratic interpolation and the secant method. The idea is at each iteration, the algorithm decides which of these three methods is likely to best approximate zero, and performs an iteration using that method. In the article and likewise in this project, the Brent method is used to compute the Black-Scoles implied volatility related to the Heston option prices (Section 4.3). A pseudo-code of the method is available in the Appendix 6.1.

2.4 Heston model

The Heston model is described by the bivariate stochastic process for the stock price process $(S_t)_{t\geqslant 0}$ and its variance v_t which follows the Cox-Ingersoll-Ross (CIR) model

$$\begin{aligned} dS_t &= rS_t dt + \sqrt{v_t} S_t dW_{1,t} \\ dv_t &= \kappa\left(\theta - v_t\right) dt + \gamma\sqrt{v_t} dW_{2,t} \end{aligned} \tag{2.9}$$

where r is the risk-neutral interest rate, θ is the long run mean, κ is the rate at which the volatility converge to θ, γ is the volatility of the volatility and ρ is the corellation between the two brownian motion such that: $\mathbb{E}\left[dW_{1,t} dW_{2,t}\right] = \rho dt$.

The volatility is always positive and cannot be zero or negative if the Feller condition $2\kappa\theta > \gamma^2$ is satisfied. The price of the option increases if θ increases. The parameters ρ and γ affect the Skewness and the Kurtosis respectively of the return distribution.

The Heston model does not have analytic solution, and thus it is solved classically by Monte Carlo method or Finite Differences on the PDE associated to the Heston's option price:

$$\begin{aligned} &\frac{\partial V}{\partial t} + rS\frac{\partial V}{\partial S} + \kappa(\bar{v} - v)\frac{\partial V}{\partial v} + \frac{1}{2} v S^2 \frac{\partial^2 V}{\partial S^2} \\ &\quad + \rho\gamma S v \frac{\partial^2 V}{\partial S \partial \nu} + \frac{1}{2}\gamma^2 v \frac{\partial^2 V}{\partial v^2} - rV = 0 \end{aligned} \tag{2.10}$$

2.5 COS method

Integration methods present the advantage to be used used for calibration and pricing purposes whenever the characteristic function of the underlying process is known analytically. In practise, the probability density function of the underlying price process is not always known, like in the Heston model.

The COS method was originally developed by Fang and Oosterlee [4] in the application of European option pricing. The proposed method approximates the stock price probability density function by a cosine expansion using the characteristic function of the log spot price. We present bellow the main computation steps to perform this task in the Heston framework.

Let us denote the log-asset prices by

$$x := \ln\left(S_0/K\right) \text{ and } y := \ln\left(S_T/K\right),$$

Thus, the pay-off of an European call option could be re-expressed as follows

$$(S_T - K)_+ = (K(e^y - 1))_+ =: v(y, T) \tag{2.11}$$

By subsidizing this new pay-off in equation (2.4), it leads to

$$V(t_0, x) = e^{-r(T-t)} \int_a^b v(y, T) f(y \mid x) dy \tag{2.12}$$

where $f(y \mid x)$ is the probability density function of the log-spot price and a and b are the truncation range (detailed further).

We now replace the density f by its cosine expansion in y

$$f(y \mid x) = \sum_{k=0}^{\infty} A_k(x) \cdot \cos\left(k\pi \frac{x-a}{b-a}\right) \tag{2.13}$$

with for all $k \in \mathbb{N}$,

$$A_k(x) := \frac{2}{b-a} \int_a^b f(y \mid x) \cos\left(k\pi \frac{y-a}{b-a}\right) dy$$

It yields to

$$V(t_0, x) = \frac{1}{2}(b-a)e^{-r(T-t)} \cdot \sum_{k=0}^{N-1} A_k(x) V_k^{call} \tag{2.14}$$

with

$$V_k^{call} = \frac{2}{b-a} K\left(\chi_k(0, b) - \psi_k(0, b)\right)$$

The functions χ and ϕ are given by

$$\begin{aligned}\chi_k(c, d) :=& \frac{1}{1 + \left(\frac{k\pi}{b-a}\right)^2} \left[\cos\left(k\pi \frac{d-a}{b-a}\right) e^d - \cos\left(k\pi \frac{c-a}{b-a}\right) e^c \right. \\ &\left. + \frac{k\pi}{b-a} \sin\left(k\pi \frac{d-a}{b-a}\right) e^d - \frac{k\pi}{b-a} \sin\left(k\pi \frac{c-a}{b-a}\right) e^c \right]\end{aligned}$$

and

$$\psi_k(c, d) := \begin{cases} \left[\sin\left(k\pi \frac{d-a}{b-a}\right) - \sin\left(k\pi \frac{c-a}{b-a}\right)\right] \frac{b-a}{k\pi}, & k \neq 0, \\ (d-c), & k = 0. \end{cases}$$

Besides, the characteristic function of the log-asset price φ_{Hes} in the Heston model (2.9) reads

$$\begin{aligned}\varphi_{Hes}(\omega; u_0) =& \exp\left(i\omega\mu(T-t_0) + \frac{u_0}{\gamma^2}\left(\frac{1 - e^{-D(T-t_0)}}{1 - Ge^{-D(T-t_0)}}\right)(\kappa - i\rho\gamma\omega - D)\right) \\ &\cdot \exp\left(\frac{\kappa}{\gamma^2}\left((T-t_0)(\kappa - i\rho\gamma\omega - D) - 2\log\left(\frac{1 - Ge^{-D(T-t_0)}}{1-G}\right)\right)\right)\end{aligned}$$

with

$$D = \sqrt{(\kappa - i\rho\gamma\omega)^2 + (\omega^2 + i\omega)\,\gamma^2} \quad \text{and} \quad G = \frac{\kappa - i\rho\gamma\omega - D}{\kappa - i\rho\gamma\omega + D}$$

Finally, the price of European call option at initial time t_0 is approximated by

$$V(t_0, x, v_0) \approx Ke^{-r(T-t_0)} \cdot \mathrm{Re}\left\{ \sum_{k=0}^{N-1} \varphi_{Hes}\left(\frac{k\pi}{b-a}; v_0\right) V_k^{call} \cdot e^{ik\pi\frac{x-a}{b-a}} \right\} \tag{2.15}$$

where v_0 represents the volatility of the underlying asset at initial time t_0.

The truncation range is set as follow:

$$[a, b] := \left[c_1 - L\sqrt{c_2 + \sqrt{c_4}}, \quad c_1 + L\sqrt{c_2 + \sqrt{c_4}} \right] \tag{2.16}$$

where c_n is the n-th cumulants of $log(S_T/K)$.

The cumulents c_1 and c_2 are specified in the Appendix 6.2. The cumulant c_4 is null as it only concerns Levy's processes. The parameter L is set to be $L = 12$. The parameter N is found to be suitable for accurately pricing option if $N = 160$.

In contrast, the authors of the paper used $L = 50$ and $N = 1500$. These parameters were tested and found to be unreliable. We thus kept the configuration of the original COS method paper.

3 Artificial neural network approach

In this section, we provide details regarding the neural network structures that will be used to price call options (BS-ANN, Heston-ANN) and calculate implied volatilities (IV-ANN). After generating option prices market data, the networks are calibrated based on a minimizing error metric criteria. In order to accurately train the model, we explain the determination of the accurate optimal-hyper-parameters: activation function, number of nodes and layers. We also motivate the choice of the optimal learning rate.

3.1 Data generation

The performance of the neural network heavily depends on the quality of the dataset. In order to generate a viable dataset, the authors use the Latin Hypercube Sampling (LHS) technique that was initially introduced in [5]. The method is a generalisation of a Latin square which has a single sample in each row and column. The method thus, generates random samples from a multidimensional distribution that results in a viable representation of the data. Sampling n samples from a random variable would require to divide its support into n intervals and draw on sample from each one.

The LHS is used to sample from the intervals of the parameter of the model whether it is the Heston model or the Black-Scholes model. Next, for each sample, the corresponding price is computed either by the closed form formula or by the COS method. The charateristic of each dataset will be further detailed in the chapter 4.

3.2 Structure of the neural network

S. Liu, Cornelis W. Oosterlee and S. M.Bohte [1] have used a random search combined with a 3-fold cross validation in order to find the optimal hyper-parameter configuration for the neural networks. The setting for the random search is described in table 1.

Parameters	Options or Range
Activation	ReLu, tanh, sigmoid, elu
Dropout rate	$[0.0, 0.2]$
Neurons	$[200, 600]$
Initialization	uniform, glorot_uniform, he_uniform
Batch normalization	yes, no
Optimizer	SGD, RMSprop, Adam
Batch size	$[256, 3000]$

Table 1: The setting of random search for hyper-parameters optimization

The result of this experience has lead the authors to choice an architecture with four hidden layers containing $n = 400$ nodes and ReLu activation function. The full optimal configuration is described in table 2

Parameters	Options
Hidden layers	4
Neurons(each layer)	400
Activation	ReLu
Dropout rate	0.0
Batch-normalization	No
Initialization	Glorot_uniform
Optimizer	Adam
Batch size	1024

Table 2: The selected model after the random search

The general structure of the network with four hidden layers containing $n = 400$ neurons is defined by

$$(\boldsymbol{W}, \boldsymbol{b}) := (\mathbf{W}_1, \mathbf{b}_1, \mathbf{W}_2, \mathbf{b}_2, \ldots, \mathbf{W}_L, \mathbf{b}_L)$$

where $L = 6$ is the depth of the network (one input layer, 4 hidden layers, one output layer), $\mathbf{W}_j$ is a weight $(n \times n)$ matrix and $\mathbf{b}_j$ is the bias $(n \times 1)$ vector of the L-th neural layer.

with $z_j^{(l)}$ denote the value of the j-th node in the l-th layer,

$$z_j^{(l)} = \max\left(\sum_i w_{ij}^{(l)} z_i^{(l-1)} + b_j^{(l)}, 0\right)^{(l)}$$

$z_i^{(l-1)}$ is the output value of the i-th node in the $(l-1)$-th layer, $w_{ij}^{(l)} \in \mathbf{W}_l$, and $b_j^{(l)} \in \mathbf{b}_l$.

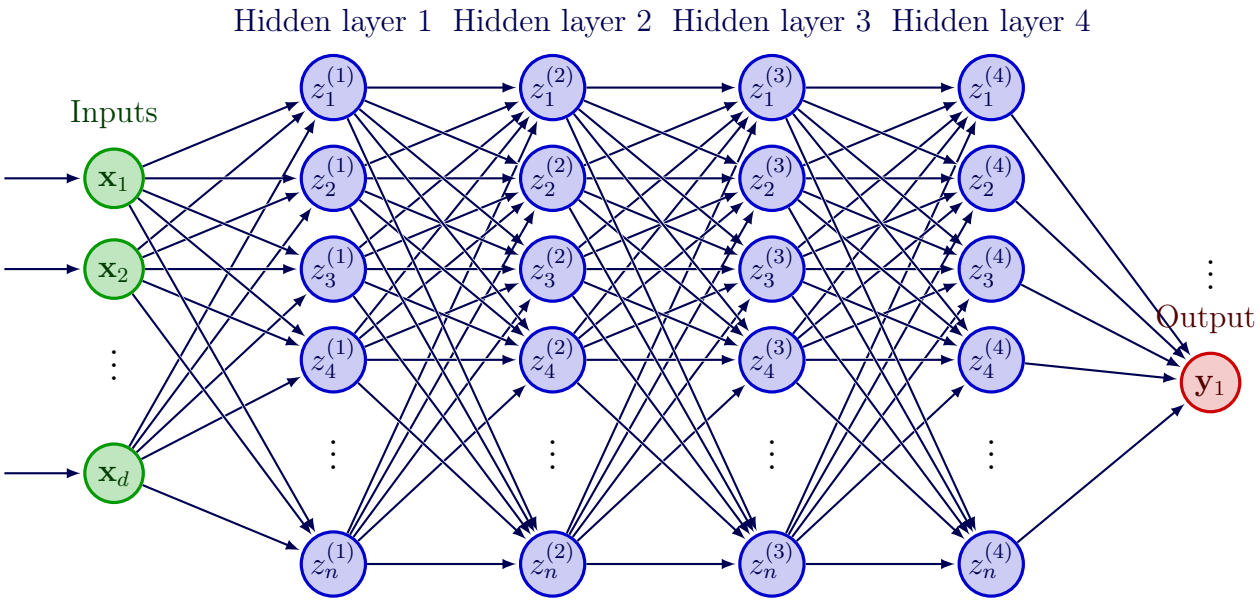

Figure 1: ANN structure with 4 hidden layers

For a $(1 \times d)$ input vector $\mathbf{x}$, the "prediction function" of the network reads

$$\mathbf{y}(\mathbf{x}) = \mathbf{F}(\mathbf{x}, (\boldsymbol{W}, \boldsymbol{b}))$$

We can now define the loss function of the network

$$L((\boldsymbol{W}, \boldsymbol{b})) := D(f(\mathbf{x}), F(\mathbf{x} \mid \boldsymbol{W}, \boldsymbol{b}))$$

where f is an objective function. The training process can be formulated as the following optimization problem

$$\arg\min_{(\boldsymbol{W},\boldsymbol{b})} L((\boldsymbol{W}, \boldsymbol{b}) \mid (\mathbf{x}, \mathbf{y}))$$

3.3 Determination of the optimal learning rate

Estimating an optimal learning rate, i.e at which rate the weight are updated during training phase, is an important step when configuring a neural network. Indeed, a large learning rate leads to a divergence, and, on the contrary, a small learning rate could lead to a slow convergence.

In the article, the authors used a simple method introduced in [6] to find the optimal interval for the learning rate. The method consists of gradually increasing the learning rate while training and observe its effect on the averaged Loss. By observing the figure 2, we can see that the Loss plateaus in the beginning, starts to drop rapidly around 10^{-6} and then starts to oscillates past 10^{-4}. Thus the best learning rate interval is from 10^{-4} to 10^{-6}.

3.4 Performance metrics

In order to assess the performance of each ANN for pricing and computing the implied volatility, we consider the metrics summurized in table 3

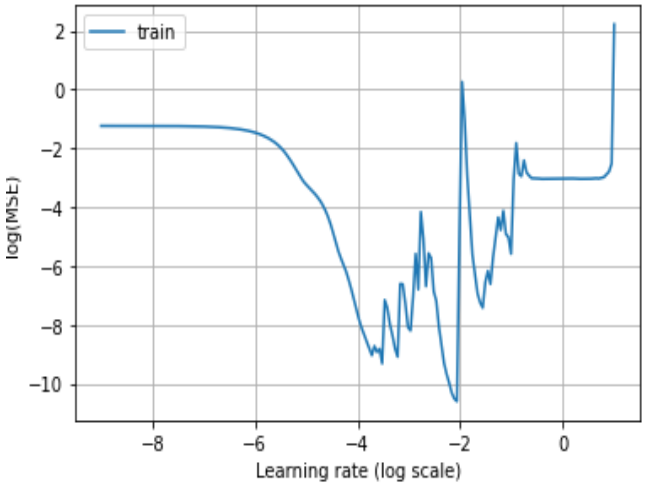

Figure 2: $\log(MSE)$ on training set as a function of log learning rate

Error metric	Formula
MSE	$\frac{1}{N}\sum_{k=1}^{N}(y_k - \widehat{y_k})^2$
RMSE	$\sqrt{MSE} = \sqrt{\frac{1}{N}\sum_{k=1}^{N}(y_k - \widehat{y_k})^2}$
MAE	$\frac{1}{N}\sum_{k=1}^{N}\lvert y_k - \widehat{y_k}\rvert$
MAPE	$\frac{1}{N}\sum_{k=1}^{N}\frac{\lvert y_k - \widehat{y_k}\rvert}{\lvert y_k\rvert}$
R2	$1 - \frac{\sum_{k=1}^{n}(y_k - \hat{y}_k)^2}{\sum_{k=1}^{n}(y_k - \bar{y})^2}$

Table 3: Error metrics

The MSE is used as a loss function for the training of the neural networks. The RMSE is used to return the MSE error in its original unit by applying the square root. The MAE is utilised to give additional information as the MSE gives greater importance to severe prediction errors. The MAPE is used to give a scaled perspective of the MAE as the summation is weighed by each sample. Finally the R2 metric compares the goodness of the model compared to a naive mean prediction.

3.5 Experimental methodology

The experiments conducted by the authors seek at demonstrating the ability of the neural network to price the vanilla call options and compute their implied volatility. The authors have done the following experiments:

- Train a neural network used to price the vanilla call option with the underlying following the Black-Scholes model. The parameters are sampled using the LHS Method and the call prices using the BS closed form formula.
- Train a neural network used to compute the implied volatility of the vanilla call option. The parameters are sampled using the LHS method and the data is generated by switching the option price and its volatility from the previous dataset.

- Train a neural network used to price the vanilla call option with the underlying following the Heston model. The parameters are sampled using the LHS Method and the call prices using the COS method.
- Compare a neural network approach for pricing and computing the implied volatility in comparison with the numerical result consisting of the combination of COS and BRENT method.

The next chapter is dedicated to the numerical results and an in-depth overview of these experiments.

4 Numerical results

In this chapter, we cover the numerical results over the different neural networks based on the error metrics previously introduced. In each subsection we will proceed to compare our results and the author's results.

4.1 Black-Scholes network

The training data for the Black-Scholes neural network (BS-ANN) is detailed in table 4. The BS-ANN takes the following inputs: $\{S_0/K, \tau, r, \sigma\}$. The output is the scaled option price $\{V/K\}$ computed using the Black-Scholes formula. The model is minimizing the MSE loss. The Pytorch's ADAM optimizer is used for the weight update. The scheduler start at 10^{-4} and decreases every 1000 epoch to reach 10^{-6}. The training is done over 3000 epoch in total.

The authors distinguish two dataset types: Wide range and Narrow range. The training of the BS-ANN is done on both datasets separately. The goal here is to observe if the network is more accurate when the input data is not close to the input domain boundaries.

	Parameters	Wide Range	Narrow Range	Unit
Input	Stock price (S_0/K)	$[0.4, 1.6]$	$[0.5, 1.5]$	–
	Time to Maturity (τ)	$[0.2, 1.1]$	$[0.3, 0.95]$	year
	Risk free rate (r)	$[0.02, 0.1]$	$[0.03, 0.08]$	–
	Volatility (σ)	$[0.01, 1.0]$	$[0.02, 0.9]$	–
Output	Call price (V/K)	$(0.0, 0.9)$	$(0.0, 0.73)$	–

Table 4: Wide and narrow Black-Scholes parameter ranges

The BS-ANN performance is detailed in table 5. The second line covers the authors results while the third line covers our results.

The training of the network was only done with 10^5 samples because of Google Colab resources restrictions. The authors in contrast trained their network with 10^6 samples. Bearing this in mind, table 5 shows that the error metrics that we obtained are only different by a factor of 10^{-1} in maximum.

We have thus successfully replicated the authors findings. Likewise we first observe that the BS-ANN does not suffer from any over-fitting as the test and train metrics are roughly similar. Next, we observe that the network performs slightly better on the narrow data set,

thus training the network on a wider dataset is not a bad practice. The testing RMSE is approximately $1 \cdot 10^{-4}$ meaning that the average pricing error is around 0.01%.

BS-ANN		MSE	RMSE	MAE	MAPE
Article	Training-wide	$8.04 \cdot 10^{-9}$	$8.97 \cdot 10^{-5}$	$6.73 \cdot 10^{-5}$	$3.75 \cdot 10^{-4}$
	Testing-wide	$8.21 \cdot 10^{-9}$	$9.06 \cdot 10^{-5}$	$6.79 \cdot 10^{-5}$	$3.79 \cdot 10^{-4}$
	Testing-narrow	$7.00 \cdot 10^{-9}$	$8.37 \cdot 10^{-5}$	$6.49 \cdot 10^{-5}$	$3.75 \cdot 10^{-4}$
Project	Training-wide	$2.62 \cdot 10^{-8}$	$1.62 \cdot 10^{-4}$	$1.20 \cdot 10^{-4}$	$1.24 \cdot 10^{-1}$
	Testing-wide	$2.59 \cdot 10^{-8}$	$1.61 \cdot 10^{-4}$	$1.20 \cdot 10^{-4}$	$1.16 \cdot 10^{-1}$
	Testing-narrow	$1.55 \cdot 10^{-8}$	$1.24 \cdot 10^{-4}$	$9.35 \cdot 10^{-5}$	$2.90 \cdot 10^{-1}$

Table 5: BS-ANN performance on the test data set

Figure 3 shows that the histograms of the prediction errors for both the narrow and wide set are normally distributed. The maximum absolute error is 0.07% and 0.06% respectively for the wide and narrow datasets.

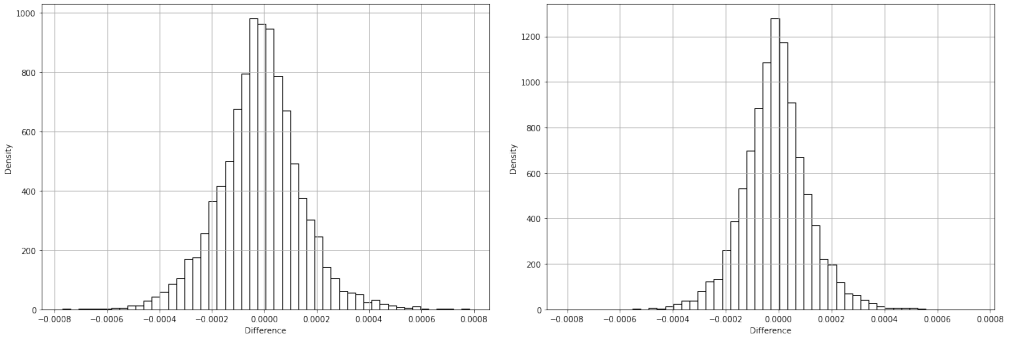

Figure 3: Left: BS-ANN error histogram on the wide test data set. Right: BS-ANN error histogram on the narrow test data set.

4.2 Implied volatility network

The implied volatility network (IV-ANN) goal is to learn the implicit relationship between the option price and its implied volatility. The neural network inputs are detailed in table 6. Rather than using directly the scaled option value $\{V/K\}$, the authors feed to the neural network the log of the time value of the option $\{log(\tilde{V}/K)\}$. The time value is defined as follow:

$$\tilde{V} = V_t - \max\left(S_t - Ke^{-r\tau}, 0\right) \tag{4.1}$$

This log transformation is done to avoid the steep gradient problem in the training of the IV-ANN which may lead to a poor performance. A comparison over a standard training using $\{V/K\}$ and $\{log(\tilde{V}/K)\}$ is done to cover the effect of this log-transformation. In addition, small values of $\tilde{V}$ like those for which $\tilde{V} < 10^{-7}$ are deleted before applying the logarithm. Table 6 summarizes the network input with their corresponding ranges.

	Parameters	Range	Unit
NN Input	Stock price (S_0/K)	$[0.5, 1.4]$	–
	Time to maturity (τ)	$[0.05, 1.0]$	year
	Risk-free rate (r)	$[0.0, 0.1]$	–
	Scaled time value ($\log(\tilde{V}/K)$)	$[-16.12, -0.94]$	–
NN output	Volatility (σ)	$(0.05, 1.0)$	–

Table 6: Parameter range of data set

The IV-ANN performance on the test dataset is detailed in table 7. The second line covers the authors results while the third line covers our results. It shows that the error metrics that we obtained are only different by a factor of 10^{-1} in maximum. We observe that the log-transformation drastically improves the neural network performance.

Figure 4 shows that the histograms of the prediction errors for test dataset is normally distributed and that the maximum absolute error is $4 \cdot 10^{-4}$. The IV-ANN successfully captured the relationship between the option prices and the implied volatility.

IV-ANN		MSE	MAE	MAPE	R2
Article	Standard	$6.36 \cdot 10^{-4}$	$1.24 \cdot 10^{-2}$	$7.67 \cdot 10^{-2}$	0.97510
	Log transform	$1.55 \cdot 10^{-8}$	$9.73 \cdot 10^{-5}$	$2.11 \cdot 10^{-3}$	0.9999998
Project	Standard	$7.21 \cdot 10^{-4}$	$8.54 \cdot 10^{-3}$	$3.84 \cdot 10^{-2}$	0.99030
	Log transform	$6.25 \cdot 10^{-7}$	$5.69 \cdot 10^{-4}$	$1.56 \cdot 10^{-3}$	0.9999909

Table 7: Out-of-Sample ANN performance comparison

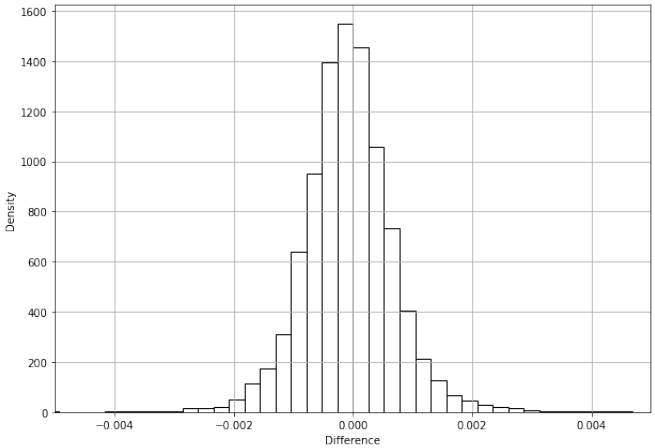

Figure 4: Out-of-Sample IV-ANN error histogram on the scaled input.

4.3 Heston network

The training data for the Heston neural network (Heston-ANN) is detailed in table 8. The neural network inputs are the parameters of the Heston SDE and the call option features:

$(r, \rho, \kappa, \bar{\nu}, \gamma, \nu_0, \tau, S_0/K)$ with the output being the scaled option price $\{V/K\}$. The option scaled prices are computed using the COS method that was introduced in section 2.5.

The training of the network is done over 3000 epoch starting at a learning rate of 10^{-4} that decays to 10^{-6} each 1000 epoch.

ANN	Parameters	Range	Method
NN Input	Moneyness, $m = S_0/K$	$(0.6, 1.4)$	LHS
	Time to maturity, τ	$(0.1, 1.4)$(year)	LHS
	Risk free rate, r	$(0.0\%, 10\%)$	LHS
	Correlation, ρ	$(-0.95, 0.0)$	LHS
	Reversion speed, κ	$(0.0, 2.0)$	LHS
	Long average variance, $\bar{v}$	$(0.0, 0.5)$	LHS
	Volatility of volatility, γ	$(0.0, 0.5)$	LHS
	Initial variance, v_0	$(0.05, 0.5)$	LHS
NN output	European call price, V	$(0, 0.67)$	COS

Table 8: The Heston parameter ranges for traing the ANN

The Heston-ANN performance is detailed in table 9. It shows that the error metrics that we obtained are only different by a factor of 10^{-1} in maximum in comparison with the authors results excluding the MAPE. We observe that neural network is not overfitting and that it accurately computes the option prices with a RMSE of approximately $4 \cdot 10^{-4}$ suggesting that the pricing error is around 0.04%.

Figure 5 shows that the histograms of the prediction errors for test dataset is normally distributed and that the maximum absolute error is $2 \cdot 10^{-3}$.

Heston-ANN		MSE	MAE	MAPE	R2
Article	Training	$1.34 \cdot 10^{-8}$	$8.92 \cdot 10^{-5}$	$5.66 \cdot 10^{-4}$	0.9999994
	Testing	$1.65 \cdot 10^{-8}$	$9.51 \cdot 10^{-5}$	$6.27 \cdot 10^{-4}$	0.9999993
Project	Training	$7.91 \cdot 10^{-8}$	$2.22 \cdot 10^{-4}$	$4.46 \cdot 10^{-2}$	0.9999996
	Testing	$1.67 \cdot 10^{-7}$	$3.11 \cdot 10^{-4}$	$1.29 \cdot 10^{-2}$	0.9999992

Table 9: The trained Heston-ANN performance

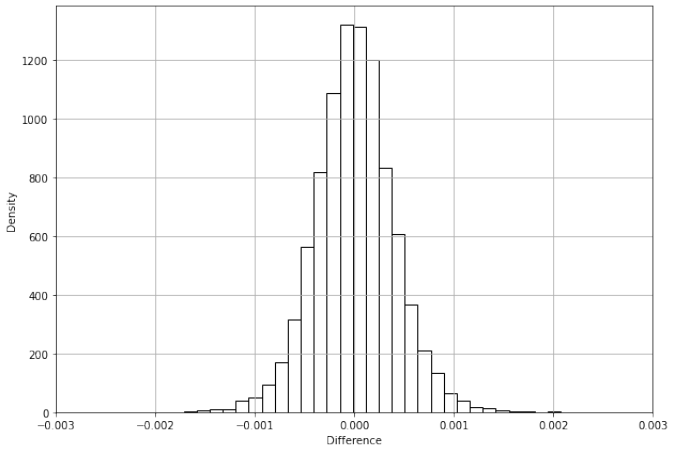

Figure 5: Out-of-Sample HESTON-ANN error histogram on the scaled input.

4.4 Machine Learning approach versus numerical approach

The final experiment conducted by the authors seeks to demonstrate the ability of the trained neural networks to price the vanilla call option and recover its implied volatility. The neural network approach is thus compared to the classical approach that utilises the COS and BRENT methods for pricing and computing the implied volatility. Figure 6 summarizes the two pipelines. Since the neural network performs worse at the domain parameters, the experiment is conducted on a narrow and wide datasets: Case 1: $\tau \in [0.3, 1.1], m \in [0.7, 1.3]$ and Case 2: $\tau \in [0.4, 1.0], m \in [0.75, 1.25]$.

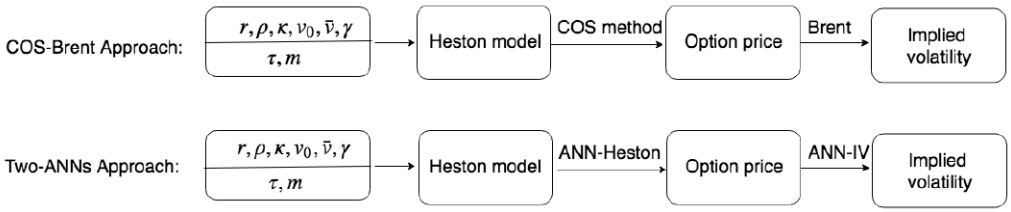

Figure 6: Two approaches of computing implied volatility for Heston model.

Table 10 summarizes the neural network performance on both cases. Likewise we observe that the neural network approach performs better on the narrow range. In a addition, we observe that we have a slightly better MSE error in comparison to the authors results.

Heston-ANN & IV-ANN		MSE	MAE	MAPE	R2
Article	Case 1: $\tau \in [0.3, 1.1], m \in [0.7, 1.3]$	$7.12 \cdot 10^{-4}$	$4.19 \cdot 10^{-4}$	$1.46 \cdot 10^{-3}$	0.999966
	Case 2: $\tau \in [0.4, 1.0], m \in [0.75, 1.25]$	$5.53 \cdot 10^{-4}$	$3.89 \cdot 10^{-4}$	$1.14 \cdot 10^{-3}$	0.999980
Project	Case 1: $\tau \in [0.3, 1.1], m \in [0.7, 1.3]$	$5.43 \cdot 10^{-5}$	$3.44 \cdot 10^{-3}$	$7.44 \cdot 10^{-3}$	0.994550
	Case 2: $\tau \in [0.4, 1.0], m \in [0.75, 1.25]$	$2.07 \cdot 10^{-6}$	$1.11 \cdot 10^{-3}$	$2.36 \cdot 10^{-3}$	0.999780

Table 10: Out-of-sample performance of the Heston-ANN plus the IV-ANN

In order to visualize the IV-ANN capabilities at capturing the implied relationship between the option prices and their implied volatility's, the authors display the volatility surface by varying $m \in [0.7, 1.3]$ and $\tau \in [0.5, 1.0]$ and fixing the Heston parameters $\rho = -0.05, \kappa = 1.5, \gamma = 0.3, \bar{v} = 0.1, v_0 = 0.1$ and $r = 0.02$. Figure 7 shows the initial volatility surface obtained by the IV-ANN and the final result after performing a B-spline surface interpolation. The corresponding errors are displayed as a heat-map in figure **??**. The maximum absolute error is $1 \cdot 10^{-3}$

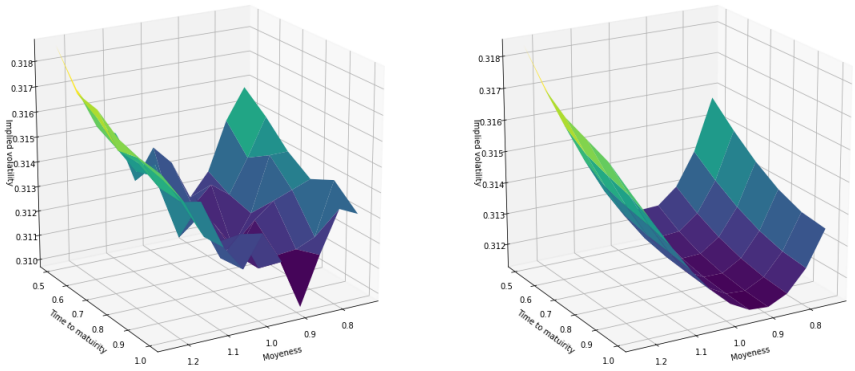

Figure 7: Left: Heston implied volatility surface generated by IV-ANN. Right: Heston smoothed implied volatility surface generated by IV-ANN.

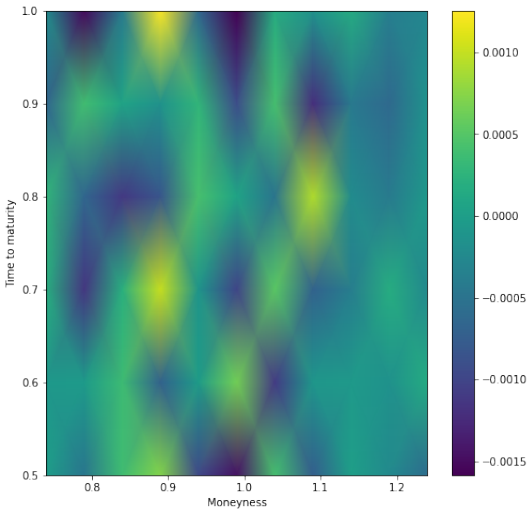

Figure 8: Heston implied volatility difference between Brent's method and IV-ANN

5 Conclusion

The aim of this project was to replicate the authors experiments which seek to demonstrate the capabilities of the neural network at pricing the vanilla call option and at computing their implied volatility. Training the neural network required a good space-filling technique. The LHS thus was used to generate a viable training dataset. The option prices were computed using the BS closed formula and the COS method. The implied volatility was computed using the BRENT method. We successfully reproduced the conducted experiments on the BS-ANN, IV-ANN and the Heston-ANN. Finally, we demonstrated that a full neural network approach is a viable alternative to a traditional numerical approach at both pricing and generating the volatility surface. The training could be done on wider parameter range and the ANN provides a fast way for executing numerical computation although the training would take much longer.

6 Appendix

6.1 BRENT algorithm

Algorithm 1 Brent's method applied to a function f on the subset $[a, b]$

Fix a, b
Fix ϵ
Compute $f(a)$, $f(b)$
If $f(a)f(b) \geq 0$
Exit
If $|f(a)| < |f(b)|$
Switch (a, b)

Set $c = a$
Repeat

If $f(a) \neq f(c)$ and $f(b) \neq f(c)$
$m = \frac{af(b)f(c)}{(f(a)-f(b))(f(a)-f(c))} + \frac{bf(a)f(c)}{(f(b)-f(a))(f(b)-f(c))} + \frac{cf(a)f(b)}{(f(c)-f(a))(f(c)-f(b))}$
Else
$m = b - f(b)\frac{b-a}{f(b)-f(a)}$
Else
$m = \frac{a+b}{2}$

Until $|b - a| < \epsilon$

Compute $f(m)$
Set $d = c$, $c = b$

If $f(a)f(m) < 0$
$b = m$
Else
$a = m$
If $|f(a)| < |f(b)|$
Switch (a, b)

Return b or m

6.2 Heston cumulants

$$
\begin{aligned}
c_1 =& \mu T + \left(1 - e^{-\lambda T}\right) \frac{\bar{u} - u_0}{2\lambda} - \frac{1}{2}\bar{u}T \\
c_2 =& \frac{1}{8\lambda^3} \left(\eta T \lambda e^{-\lambda T} \left(u_0 - \bar{u}\right) \left(8\lambda\rho - 4\eta\right)\right. \\
& + \lambda\rho\eta \left(1 - e^{-\lambda T}\right) \left(16\bar{u} - 8u_0\right) \\
& + 2\bar{u}\lambda T \left(-4\lambda\rho\eta + \eta^2 + 4\lambda^2\right) \\
& + \eta^2 \left(\left(\bar{u} - 2u_0\right) e^{-2\lambda T} + \bar{u} \left(6e^{-\lambda T} - 7\right) + 2u_0\right) \\
& \left. + 8\lambda^2 \left(u_0 - \bar{u}\right) \left(1 - e^{-\lambda T}\right)\right)
\end{aligned}
\tag{6.1}
$$

References

[1] C. W. O. Shuaiqiang Liu and S. M.Bohte, "Pricing options and computing implied volatilities using neural networks," 2018.

[2] M. Bouzouba and A. Osseiran, *Exotic Options and Hybrids: A guide to Structuring, Pricing and Trading.* 2010.

[3] R. Brent, "An algorithm with guaranteed convergence for finding a zero of a function," *Algorithms for Minimization without Derivatives*, 1973.

[4] C. Fang, F.; Oosterlee, "Novel pricing method for european options based on fourier-cosine series," *Society for Industrial and Applied Mathematics*, vol. 2, pp. 826–848, 2008.

[5] M. B. McKay, "A comparison of three methods for setting values of input variables in the analysis of output from a computer code," *Technometrics*, vol. 21, no. 2.

[6] L. Smith, "Cyclical learning rates for training neural networks," 2015.