Deep Calibration of the Variance Gamma Model

Mtsetfosisekelo Mlondi Mhlanga (mtsetfosisekelo@aims.ac.za)
African Institute for Mathematical Sciences (AIMS)

Supervised by: Professor Peter Ouwehand
University of Cape Town, South Africa

26 May 2022

Submitted in partial fulfillment of a structured masters degree at AIMS South Africa

Abstract

Option pricing models have long been used for financial purposes in the market and the choice of which model to use depends on many factors one being how well that model fits into the real world. In this dissertation we discuss the Variance Gamma (VG) model which comes as an alternative to the famous Black-Scholes model as it able to better match market observed option prices. The Variance Gamma model is a pure jump process that is able to cater for the rapid changes often observed over a small period of time as the model implies infinitely many time jumps in a finite time interval. VG implied option prices are recovered through the COS method which utilizes the characteristic function of the model to compute option prices.

This dissertation is specifically focused on achieving the calibration of the parameters of the VG model through the use of Artificial Neural Networks (ANN). Calibration of a financial model is finding the model parameters that match market observed prices. We make use of a two-step approach where in the first step a pricing ANN learns the pricing map imposed by the VG model when priced by the COS method and in the second step another ANN which uses weights obtained from the first step performs the actual calibration process (Liu et al., 2019a). The pricing and calibrating ANN are tested for their performance to execute the task at hand and the results show that the pricing ANN is able to learn the pricing map relatively good and while the calibrating ANN was able to mostly return calibrated parameters it did present few instances where parameters fail to match observed prices. The ANN approach to achieve calibration proved to be computationally fast which is due to the offline learning of the pricing map in the first step which needs only be done once.

Declaration

I, the undersigned, hereby declare that the work contained in this research project is my original work, and that any work done by others or by myself previously has been acknowledged and referenced accordingly.

Mtsetfosisekelo Mlondi Mhlanga, 26 May 2022

Contents

1. Introduction

Financial models need to be calibrated to market observed prices before being applied for financial purposes such as hedging, risk management, option pricing, etc. However the calibration procedure is deemed a computationally expensive task when utilizing traditional approaches and this is an issue since the speed taken to calibrate a model is an important factor. With the development of artificial neural networks (ANN) many authors: (Liu et al., 2019a), (Hernandez, 2016), (Itkin, 2019), etc, have ventured into the application of ANN to achieve the calibration procedure.

Two approaches have been used thus far to achieve calibration: (i) the first being to directly train an ANN to directly return calibrated parameters and (ii) the second is to first train the ANN to learn the pricing map for the proposed pricing model through a feed forward pass, and then perform the calibration through a backward pass by fixing the weights obtained in the forward pass. The offline training of ANN has greatly reduced the time constraint imposed on traditional methods and in addition the pricing ANN need to be trained only once. The use of the model to generate a large training data set ensures that the ANN can efficiently learn the pricing map and the issue of a model over-fitting can be overcome by training with a new data set in every training instance. When utilizing artificial neural networks for calibration the problem lies in the architecture of the network as finding the right hyperparameters can be time consuming. However one can make reference to already existing work in the field as a starting point to create the model.

The number of parameters to be calibrated depends on the choice of model being used. For example, in the Black-Scholes model the only parameter to consider is the volatility while models like the Variance Gamma or the Heston Model have more parameters. Models with more parameters are often more difficult to calibrate as all parameters need to be calibrated at the same time and sometimes some of these are constrained. This dissertation explores the use of ANN to calibrate the three-parameter Variance Gamma (VG) model.

1.1 Objective

- The objective of this dissertation is to use the two-step approach in creating an artificial neural network to achieve the calibration process of the Variance Gamma model parameters.

2. Background

2.1 Option Pricing

Options are one of the most widely traded securities in the financial markets and they can be defined as financial derivatives or contracts that offer the owner the right to sell or to buy an underlying stock at some fixed price and particular time (Kumar, 2018). Options can be traded either as "calls" or "puts". A call option offers the owner the right to buy a share of an underlying stock while put options offer the right to sell. Financial markets have been for a long dominated by two major styles of option: (i) American options which can be exercised at any time t before, and on the Maturity time T and (ii) European options which can only be exercised at the maturity time T, however there has been an increase in the popularity of some exotic option such as the Barrier Options. Other traded exotic options include: Bermuda options, Asian options, etc.

There are many financial models or methods used to determine the theoretical value of an option and many if not all of these models will have the following parameters in common: $\mathbf{S}_0$ the current price of the stock, **K** the strike price at which the contract owner buys or sell, time **T** at which the option matures, risk free interest rate r, the dividend rate q, and the volatility σ. For example when using the famous Black-Scholes model the price (C) for a vanilla call option is given by

$$C = S_0 e^{-qT} N(d_1) - K e^{-rT} N(d_2), \tag{2.1.1}$$

$$\begin{aligned} d_1 &= \frac{\ln(\frac{S_0}{K}) + (r - q + \frac{\sigma^2}{2})T}{\sigma\sqrt{T}}, \\ d_2 &= d_1 - \sigma\sqrt{T}, \end{aligned} \tag{2.1.2}$$

where $N(\cdot)$ is the cumulative distribution function for the normal distribution. Once priced the payoff for the call option is given as

$$C_{\text{payoff}} = max\{S_T - K, 0\} = (S_T - K)^+,$$

while for put options it is defined as $(K - S_T)^+$. Knowing the price of a call is enough for one to be able to recover the price of the put option counter part and vice versa through the put-call parity;

$$\mathsf{P} = \mathsf{C} + K e^{-rT} - S_0. \tag{2.1.3}$$

Other pricing models may include: the Binomial model, Variance Gamma model, Heston model, etc. Financial models only provide a theoretical quote rather than the actual market observed price hence the need for calibration which aims at reducing the error between the theoretical quote and market observed prices.

Volatility is one of the most important concepts in option pricing such that traders often prefer to quote options with implied volatility rather than their prices as this provides a better view of the market (Orlando and Taglialatela, 2017). With respect to the Black Scholes model defined by (2.1.1) implied volatility, σ^{BS}, is the value of the volatility parameter in the model that yields market observed prices. Traders make use of implied volatility surfaces defined at some time, t, to find the price of all vanilla option at that particular date (Cont and Da Fonseca, 2002). The use of implied volatility is often extended or used for hedging purposes.

Hedging is an investment made towards a derivative, which includes options, that aims at reducing risk associated to price movements. Being able to determine prices for all calls and puts at a particular date therefore puts traders on better position to be able to hedge their portfolios such as using the put options to hedge call options.

Options are sometimes classified with respect to the ratio between the strike price and the exercise price and this is termed moneyness. With respect to moneyness an option can be: In-the-money (ITM), Out-of-the-money (OTM) or At-the-money(ATM). Some pricing models such as the the cos model make use of moneyness,or log-moneyness in the pricing formula, hence making moneyness an important concept in finance.

Moneyness	Call Option	Put Option
ATM	Exercise Price $=$ Market Price	Exercise Price $=$ Market Price
ITM	Exercise Price $<$ Market Price	Exercise Price $>$ Market Price
OTM	Exercise Price $>$ Market Price	Exercise Price $<$ Market Price

Table 2.1: Moneyness

Options can be held in terms of long or short positions, and given the two types of trading options as a results we have the following positions:

- **Long Call:** The holder pays a premium which earns the right to buy the stock at the strike price at the maturity time.
- **Long put:** Similar to the long call, the holder pays a premium which grants him the right to sell at the strike price at maturity.
- **Short call:** The writer receives a premium which commits them to deliver the asset at the strike price at time T if the holder chooses to exercise their right.
- **Short put:** The writer receives a premium which commits them to buy the asset at the exercise price at time T if the owner chooses to sell.

2.2 Variance Gamma Model

The Variance Gamma (VG) model was originally introduced in the paper by Madan and Seneta (1990), and in this section we provide a brief summary of this model with reference to the original paper together with work that came afterwards, such as from Madan and Milne (1991), Madan et al. (1998), etc. The starting point for the VG model is the Brownian Motion(BM) which is defined as

$$b(t;\theta,\sigma) = \theta t + \sigma W(t), \tag{2.2.1}$$

where θ is the drift coefficient, σ is the volatility coefficient, and $W(t)$ is the normal Wiener process or Brownian motion (Madan et al., 1998). A gamma process, $\gamma(t;\mu,\nu)$, where μ is the mean rate and ν is the variance rate, is a stochastic process with independent positive increments following the gamma distribution [(Madan et al., 1998), (Van Noortwijk et al., 2005)] thus making the variance gamma to be pure jump process with infinitely many jumps in a every finite time interval. The variance gamma process, $X(t;\sigma,\nu,\theta)$, which is a three parameter process is then defined as

$$\begin{aligned} X(t;\sigma,\nu,\theta) &= b(\gamma(t;1,\nu);\theta,\sigma) \\ &= \gamma(t;1,\nu)\theta + \sigma W(\gamma(t;1,\nu)), \end{aligned} \tag{2.2.2}$$

where the gamma process is evaluated with mean rate $\mu = 1$. The parameters σ and θ are defined the same way as in the BM where they are the volatility and drift, respectively. The parameter ν is the variance rate of the gamma process time change. θ is responsible for modeling the skewness of the VG process, while ν models the kurtosis (Madan et al., 1998). For a distribution, skewness is the measure of symmetry that compares the left and right side of of the distribution with respect to the mid-point, while Kurtosis measures how a distribution is tailed with respect to the normal distribution. The VG process can also be represented as a sum of two independent gamma processes (Fiorani, 2004) as

$$X(t;\sigma,\nu,\theta) = \gamma_p\left(t;\mu_p,\nu_p\right) - \gamma_n\left(t;\mu_n,\nu_n\right), \tag{2.2.3}$$

where γ_p represents an upward movement while γ_n is for a down movement and for option pricing these would be a price increase and decrease, respectively. To ensure that (2.2.2) and (2.2.3) are both modeling the same scenario the variables in (2.2.3) are defined as

$$\begin{aligned}
\mu_p &= \frac{1}{2}\sqrt{\theta^2 + \frac{2\sigma^2}{\nu}} + \frac{\theta}{2} \\
\mu_n &= \frac{1}{2}\sqrt{\theta^2 + \frac{2\sigma^2}{\nu}} - \frac{\theta}{2} \\
\nu_p &= \left(\frac{1}{2}\sqrt{\theta^2 + \frac{2\sigma^2}{\nu}} + \frac{\theta}{2}\right)^2 \nu \\
\nu_n &= \left(\frac{1}{2}\sqrt{\theta^2 + \frac{2\sigma^2}{\nu}} - \frac{\theta}{2}\right)^2 \nu
\end{aligned} \tag{2.2.4}$$

Pricing models in finance can be implemented in different ways and some implementations such as the COS method and Fast Fourier Transforms (FFT) (Fang and Oosterlee, 2009) depend on the existence of a characteristic function or the density function for the given process. With respect to the gamma time change, $g = \gamma(t+h;\mu,\nu) - \gamma(t;\mu,\nu)$, the characteristic and density functions (Fiorani, 2004) are

$$\phi_{X(t)}(u) = \left(\frac{1}{1 - i\theta\nu u + \left(\frac{\sigma^2\nu u^2}{2}\right)}\right)^{t/v}, \tag{2.2.5}$$

$$f_{X(t)}(X) = \int_0^{+\infty} \frac{1}{\sigma\sqrt{2\pi p}} \exp\left(-\frac{(X-\vartheta p)^2}{2\sigma^2 p}\right) \frac{p^{\frac{t}{\nu}-1}\exp\left(-\frac{p}{\nu}\right)}{\nu^{\frac{t}{\nu}}\Gamma\left(\frac{t}{\nu}\right)} dp \tag{2.2.6}$$

Option pricing is done under the risk neutral measure which for the VG model is obtained by replacing the brownian motion with the VG process in the Black-Scholes stock dynamic equation to give

$$S(t) = S(0)\exp\left(rt + X\left(t;\sigma_{RN},\nu_{RN},\theta_{RN}\right) + \omega_{RN}t\right) \tag{2.2.7}$$

where r is a continuously compounded interest rate and

$$\omega_{RN} = \frac{1}{\nu_{RN}}\ln\left(1 - \theta_{RN}\nu_{RN} - \frac{\sigma_{RN}^2\nu_{RN}}{2}\right). \tag{2.2.8}$$

Since here we provide a brief summary of the VG model the analytical formula will not be derived rather we present it with respect to the paper by Madan et al. (1998). When considering vanilla call options they are priced by

$$\begin{aligned} c(S(0);K,t) = S(0)\Psi\left(d\sqrt{\frac{1-c_1}{\nu}}, (\zeta s+s)\sqrt{\frac{v}{1-c_1}}, \frac{t}{\nu}\right) \\ - Ke^{-rt}\Psi\left(d\sqrt{\frac{1-c_2}{\nu}}, \zeta s^2\sqrt{\frac{v}{1-c_2}}, \frac{t}{\nu}\right) \end{aligned} \tag{2.2.9}$$

where

$$\zeta = -\frac{\theta}{\sigma^2} \tag{2.2.10}$$

$$s = \frac{\sigma}{\sqrt{1+\left(\frac{\theta}{\sigma}\right)^2\frac{\nu}{2}}} \tag{2.2.11}$$

$$c_1 = \frac{v(\alpha+s)^2}{2} \tag{2.2.12}$$

$$c_2 = \frac{v\alpha^2}{2} \tag{2.2.13}$$

$$d = \frac{1}{s}\left[\ln\left(\frac{S(0)}{K}\right) + rt + \frac{t}{v}\ln\left(\frac{1-c_1}{1-c_2}\right)\right] \tag{2.2.14}$$

Ψ is defined as a Bessel function of the 2^{nd} kind, and is given by

$$\Psi(\alpha,\beta,\gamma) = \int_0^{+\infty} N\left(\frac{\alpha}{\sqrt{x}} + \beta\sqrt{x}\right)\frac{x^{\gamma-1}e^{-x}}{\Gamma(\gamma)}dx \tag{2.2.15}$$

An alternative closed form solution was later presented by (Madan and Milne, 1991) is defined as

$$\begin{aligned} c(S(0);K,t) = S(0)\left(1-\frac{\nu(\zeta s+s)^2}{2}\right)^{\frac{t}{\nu}}\exp\left(\frac{g(\zeta s+s)^2}{2}\right)\cdot N\left[\frac{d}{\sqrt{g}} + (\zeta s+s)\sqrt{g}\right] \\ -Ke^{-rt}\left[\left(1-\frac{\nu(\zeta s)^2}{2}\right)^{\frac{t}{\nu}}\right]\exp\left(\frac{g(\zeta s)^2}{2}\right)\cdot N\left(\frac{d}{\sqrt{g}} + (\zeta s)\sqrt{g}\right) \end{aligned} \tag{2.2.16}$$

where $N(\cdot)$ is the cumulative distribution function for the normal distribution.

Provided with the pricing formula one would be interested in knowing how each parameter in the VG model affects option prices. Using literature bounds from Madan et al. (1998) for the parameters together with the input set $\mathbf{p} = [S_0 = 100, K = 120, T = 1, r = 0.1]$, Figure2.1 shows the effect of each parameter on a call option price. When varying one parameter, the others were kept constant at $[\sigma = 0.1, \theta = -0.2, \nu = 0.5]$.

The VG model comes as an improvement of the BS with a random time change that is able to model better the rapid changes in the market over short periods of time. The model further corrects the bias of the BS model when it comes to strike and maturity (Madan et al., 1998). For the risk neutral measure the VG model correctly rejects the zero skewness and zero kurtosis over the normal distribution (Madan et al., 1998).

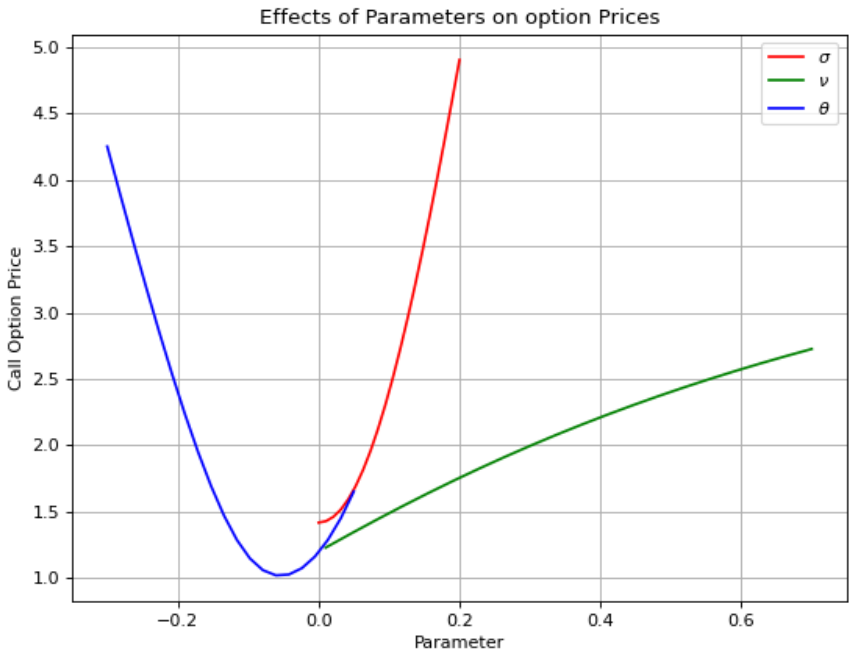


Figure 2.1: Effects of Changing VG Model Parameters

2.3 COS Method

2.3.1 Derivation.

Option pricing by VG model in this paper is implemented through numerical means via the Fourier cosine expansion rather than using the analytical formulas presented by (2.2.9) and (2.2.16) . This next section provides an overview of this Fourier cosine expansion also called the COS method [(Fang and Oosterlee, 2009),(Hirsa, 2013)] and its implementation on pricing Vanilla options. To present the COS Method pricing formula we first look at the Fourier cosine expansion of some function $f(\theta)$ on the interval $[0, \pi]$ which is given as

$$f(\theta) = \frac{1}{2}A_0 + \sum_{k=1}^{\infty} A_k \cos(k\theta) = \sum_{k=0}^{\infty}{}' A_k \cdot \cos(k\theta), \tag{2.3.1}$$

where

$$A_k = \frac{2}{\pi}\int_0^{\pi} f(\theta)\cos(k\theta)d\theta. \tag{2.3.2}$$

The prime ($'$) on the summation of (2.3.1) indicates that the first term, $k = 0$, is weighted by a factor of half. The main idea in the COS method is to approximate the cosine coefficients (A_k) in (2.3.2) with respect to the characteristic function of the pricing model. The cosine expansion can be extended from the interval $[0, \pi]$ to some finite interval $[a, b]$ through a change of variables defined as

$$\theta = \frac{x-a}{b-a}\pi \implies x = \frac{b-a}{\pi}\theta + a \tag{2.3.3}$$

which results into the following transformation

$$f(x) = \sum_{k=0}^{\infty}{}' A_k \cdot \cos\left(k\pi \frac{x-a}{b-a}\right) \tag{2.3.4}$$

where

$$A_k = \frac{2}{b-a}\int_a^b f(x)\cos\left(k\pi\frac{x-a}{b-a}\right)dx \tag{2.3.5}$$

A characteristic function is defined over the entire real line thus a finite interval $[a,b]$ only provides an approximation and this interval is chosen such that it provides a good approximation. This approximation is called a truncated version of the characteristic function and their relationship is given as

$$\phi(u) = \int_{-\infty}^{\infty} e^{iux} f(x)dx \approx \int_a^b e^{iux} f(x)dx = \widehat{\phi}(u) \tag{2.3.6}$$

The next step is evaluating $\widehat{\phi}(u)$ at $u = \frac{k\pi}{b-a}$, and then multiplying by $(e^{-i\frac{k\pi a}{b-a}})$ to get

$$\begin{aligned}\widehat{\phi}\left(\frac{k\pi}{b-a}\right)e^{-i\frac{k\pi a}{b-a}} &= e^{-i\frac{k\pi a}{b-a}}\int_a^b e^{i(\frac{k\pi}{b-a})x} f(x)dx \\ &= \int_a^b \left[\cos\left(k\pi\frac{x-a}{b-a}\right) + i\sin\left(k\pi\frac{x-a}{b-a}\right)\right] f(x)dx \end{aligned} \tag{2.3.7}$$

$$\implies \mathrm{Re}\left\{\widehat{\phi}\left(\frac{k\pi}{b-a}\right)\exp\left(-i\frac{ka\pi}{b-a}\right)\right\} = \int_a^b \cos\left(k\pi\frac{x-a}{b-a}\right) f(x)dx \tag{2.3.8}$$

When compared with (2.3.5), we get that

$$A_k = \frac{2}{b-a}\,\mathrm{Re}\left\{\widehat{\phi}\left(\frac{k\pi}{b-a}\right)\exp\left(-i\frac{ka\pi}{b-a}\right)\right\} \tag{2.3.9}$$

The objective is to approximate the cosine coefficients with respect to the characteristic function, which has been achieved on (2.3.9), thus we make the assumption that $A_k \approx F_k$, where F_k is defined in terms of the original characteristic function rather than the truncated version. As a result we have

$$F_k = \frac{2}{b-a}\,\mathrm{Re}\left\{\phi\left(\frac{k\pi}{b-a}\right)\exp\left(-i\frac{ka\pi}{b-a}\right)\right\} \tag{2.3.10}$$

Substituting F_k for A_k in (2.3.4) and taking the summation over a finite interval we get the cosine expansion for $f(x)$ in terms of the characteristic function is approximated as

$$\widetilde{f}(x) = \frac{2}{b-a}\sum_{k=0}^{N-1}{}'\,\mathrm{Re}\left\{\phi\left(\frac{k\pi}{b-a}\right)\exp\left(-i\frac{ka\pi}{b-a}\right)\right\}\cos\left(\frac{x-a}{b-a}k\pi\right) \tag{2.3.11}$$

2.3.2 Truncation.

The COS method carries two truncation errors from its derivation; the first being in the truncation of the characteristic function into a finite interval $[a,b]$ and also the truncation error from the summation. Regardless of these errors the COS method still provides a good approximation for option prices, and is know to be the fasted amongst Fourier transforms. In the paper (Fang and Oosterlee, 2009) we are provided with an approximation for the interval $[a,b]$ as

$$[a,b] = \left[c_1 - L\sqrt{c_2+\sqrt{c_4}},\quad c_1 + L\sqrt{c_2+\sqrt{c_4}}\right] \tag{2.3.12}$$

where c_n is the n^{th} cumulant with respect to log-moneyness, and $L \in [6, 12]$. For the VG model we have that these cumulants defined as:

$$\begin{aligned} c_1 &= (\mu + \theta)T, \\ c_2 &= \left(\sigma^2 + v\theta^2\right) T, \\ c_4 &= 3\left(\sigma^4 v + 2\theta^4 v^3 + 4\sigma^2\theta^2 v^2\right) T, \\ \mu &= r - q + w, \\ w &= \frac{1}{\nu} \ln\left(1 - \theta\nu - \sigma^2\nu/2\right). \end{aligned} \tag{2.3.13}$$

2.3.3 Pricing Model - Vanilla Options.

Suppose the modeled quantity x, often the log price, is given at time t, and its its value y is known at the maturity time T, the pricing COS formula is

$$v(x,t) = e^{-r\Delta t} \int_a^b v(y,T) f(y \mid x) dy, \tag{2.3.14}$$

where $v(y,T)$ is the payoff at the maturity time, r is the risk neutral interest rate, and $f(y|x)$ is the density function for a given process. The density function is replaced by its approximation given by the Fourier cosine expansion of the characteristic function as presented in (2.3.11), thus

$$v(x,t) = e^{-r\Delta t} \cdot \int_a^b v(y,T) \sum_{k=0}^{\infty}{}' A_k \cos\left(\frac{y-a}{b-a} k\pi\right) dy \tag{2.3.15}$$

where

$$A_k = \frac{2}{b-a} \int_a^b f(y|x) \cos\left(\frac{y-a}{b-a} k\pi\right) dy \tag{2.3.16}$$

Re-arranging (2.3.15) to get

$$v(x,t) = e^{-r\Delta t} \cdot \sum_{k=0}^{\infty\prime} A_k \left[\int_a^b v(y,T) \cos\left(\frac{y-a}{b-a} k\pi\right) dy\right] \tag{2.3.17}$$

Defining V_k as

$$V_k = \frac{2}{b-a} \int_a^b v(y,T) \cos\left(\frac{y-a}{b-a} k\pi\right) dy \tag{2.3.18}$$

$$\implies v(x,t) = \frac{b-a}{2} e^{-r\Delta t} \sum_{k=0}^{N-1}{}' A_k V_k \tag{2.3.19}$$

Recalling that $A_k \approx F_k$ where F_k is defined in (2.3.10) we get an approximation for $v(x,t)$ which is the COS Method for underlying process

$$v(x,t) \approx e^{-r\Delta t} \sum_{k=0}^{N-1}{}' \operatorname{Re}\left\{\phi\left(\frac{k\pi}{b-a}\right) \exp\left(-ik\pi \frac{a}{b-a}\right)\right\} V_k \tag{2.3.20}$$

For pricing vanilla options V_k is analytically defined with respect to x and y where

$$x := \ln\left(\frac{S_0}{K}\right) \quad \text{and} \quad y := \ln\left(\frac{S_T}{K}\right), \tag{2.3.21}$$

The payoff function, $v(y,T)$, for vanilla options is given as

$$v(y,T) = [\alpha K(e^y - 1)]^+ \tag{2.3.22}$$

where $\alpha = 1$ for the call option and $\alpha = -1$ for the put option. To simplify the evaluation of V_k, the following function are defined as

$$\begin{aligned}\chi_k(c,d) &= \int_c^d e^y \cos\left(k\pi\frac{y-a}{b-a}\right) dy \\ &= \frac{1}{1+\left(\frac{k\pi}{b-a}\right)^2}\left[\cos\left(k\pi\frac{d-a}{b-a}\right)e^d - \cos\left(k\pi\frac{c-a}{b-a}\right)e^c\right. \\ &\quad \left. + \frac{k\pi}{b-a}\sin\left(k\pi\frac{d-a}{b-a}\right)e^d - \frac{k\pi}{b-a}\sin\left(k\pi\frac{c-a}{b-a}\right)e^c\right]\end{aligned} \tag{2.3.23}$$

$$\begin{aligned}\varphi_k(c,d) &= \int_c^d \cos\left(k\pi\frac{y-a}{b-a}\right) dy \\ &= \begin{cases}\left[\sin\left(k\pi\frac{d-a}{b-a}\right) - \sin\left(k\pi\frac{c-a}{b-a}\right)\right]\frac{b-a}{k\pi} & k \neq 0 \\ (d-c) & k = 0.\end{cases}\end{aligned} \tag{2.3.24}$$

It then follows that

$$\begin{aligned}V_k^{\text{call}} &= \frac{2}{b-a}\int_a^b K\,(e^y - 1)^+ \cos\left(k\pi\frac{y-a}{b-a}\right) dy \\ &= \frac{2}{b-a}K\,(\chi_k(0,b) - \varphi_k(0,b))\end{aligned} \tag{2.3.25}$$

$$\begin{aligned}V_k^{\text{put}} &= \frac{2}{b-a}\int_a^b K\,(1 - e^y)^+ \cos\left(k\pi\frac{y-a}{b-a}\right) dy \\ &= \frac{2}{b-a}K\,(-\chi_k(a,0) + \varphi_k(a,0))\end{aligned} \tag{2.3.26}$$

From combining (2.3.20) with either (2.3.25) for call options or with (2.3.26) for put options we get the numerical VG COS method.

2.4 Calibration and Calibration Neural Networks (CaNN)

This section properly introduces the calibration problem. Consider a pricing model $\mathcal{M}(\Theta, \mathbf{p})$ where $\Theta \in \mathbf{R}^n$ is the set over which the parameters range and $\mathbf{p}$ is a set of inputs used in the pricing process, such as time to maturity, strike, stock price, interest rate, etc. For the VG model, $\sigma, \theta, \nu \in \Theta$, and when implementing it through COS method, $\mathbf{p} = \{x, r, T\}$, where $x = \ln(S_0/K)$ is the log moneyness. The model $\mathcal{M}(\Theta, \mathbf{p})$ creates a mapping

$$\mathcal{M} : (\Theta, \mathbf{p}) \to C \tag{2.4.1}$$

where C is the theoretical price of the underlying asset. Given observed market prices, C^*, calibration is the process of finding the optimal model parameters, $\theta_i \in \Theta$, that minimize the error between model prices and the market prices [(Itkin, 2019), (Hernandez, 2016), (Liu et al., 2019a)]. The difference between model and observed prices is measured by some cost function of choice and letting δ be the cost function then the calibration problem can be expressed as

$$\theta^* = \underset{\theta \in \Theta}{\operatorname{argmin}} \quad \delta(C, C^*) \tag{2.4.2}$$

The commonly used cost function is the weighted least squares function thus (2.4.2) under this function would be

$$\theta^* = \underset{\theta \in \Theta}{\operatorname{argmin}} \sum_{i=1}^{N} \Omega_i \left(C_i - C_i^*\right)^2 \tag{2.4.3}$$

for an appropriate weight Ω_i. The calibration process can be sometimes be viewed as solving the inverse of a pricing model. First, use the mapping in (2.4.1) to price options then re-arrange the inputs and parameters to solve the inverse problem, i.e.

$$\mathcal{M}^{-1}(C, \mathbf{p}) \to \Theta \tag{2.4.4}$$

The factors to consider when calibrating financial models are greatly dependent on what purpose the model is being used for, for example option pricing, arbitrage detection, and risk management models are calibrated to different factors (Hirsa, 2013). Calibration of an option pricing model may be more focused on the accuracy and time taken achieve market observed prices while in the case of risk management one may be interested in the long term behaviour of the model.

Calibrating to implied volatility is often the common practice in finance as it yields more accurate results compared to calibrating to option prices. Furthermore in the paper by Liu et al. (2019a) calibration to implied volatility for vanilla options is favoured as it helps to specify prices for both call and put options.

Many pricing models in finance make use of numerical methods for the pricing process hence making their calibration process to be time consuming such that by the time the model has to implemented the market has greatly changed. This has lead to the venture of using Artificial Neural Networks (ANN) to accomplish the calibration process and in financial terminology this is called Calibration Neural Networks (CaNN). CaNN have been favoured due to that after training of a suitable model the calibration process is very fast.

2.4.1 Calibration Neural Networks (CaNN).

Calibration models differ from one pricing model to another especially when it comes to the number of hidden layers and neurons used. However once a model has been designed CaNN offers a faster method to achieve calibration by offloading the time consuming training step into an offline process in return for a fast online calibration step. The advantage is in that even though training can take several hours or even days to complete, a model need only to be trained once. Two approaches have been investigated into using ANN to achieve calibration and these are;

One Step Approach: With the single step approach, also referred to as the direct method, an ANN is trained to directly return calibrated parameters. When training under the direct method the inputs are the option prices and the pricing inputs "**p**" and the output are the model parameters "Θ" and this will be the case even when calibrating the model.

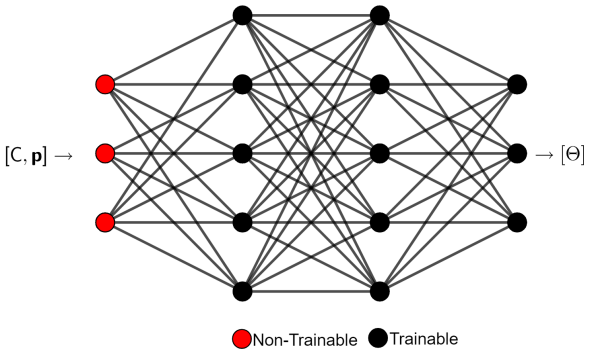
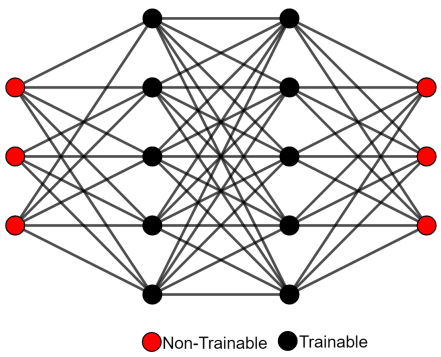
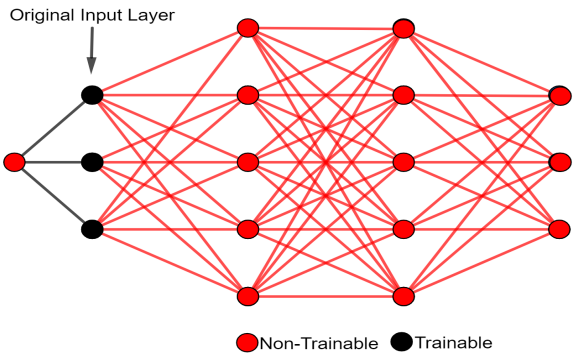


Figure 2.2: One Step Approach

Two Step Approach: The two step approach, also referred to as the indirect method, is in itself composed of two steps where in the first step an ANN is trained to learn the pricing map imposed by the model of choice, shown by $Figure$ 2.3(a) and the second step is the calibration step achieved by solving the inverse problem of the pricing map, shown by $Figure$ 2.3(b). The pricing map step optimizes the weights of the hidden layers while the calibration step makes use of the optimized weights to find the parameters that result into observed market prices.



(a) Pricing Map



(b) Calibration Step

Figure 2.3: Two Step Approach

The calibration step is achieved by converting the original input layer into a hidden layer and then using a new input layer that takes in the value "1", thus of interest are the weights of the first hidden layer which would represent the model parameters. This is the approach later used in this thesis to achieve the calibration of the Variance Gamma model.

2.5 Artificial Neural Networks

Artificial Neural Networks (ANN) or commonly referred as Neural Networks (NN) in the field of machine learning (ML) can be thought off as a collection of nodes/artificial neurons that provide a mapping from an input vector to its corresponding output vector via the use of hidden layers. The basic architecture of an ANN is composed of an input layer, one or more hidden layers, and an output layer (Mostafa et al., 2021) and once more hidden layers are incorporated into a network the resulting structure is called a deep neural network (DeepNN) $Figure$ 2.4 and this is classified into a subfield of ML called called Deep Learning (DL). The complexity and accuracy of the mapping created through the use of a NN is dependent on the training the NN undergoes and the final hyperparameters chosen.

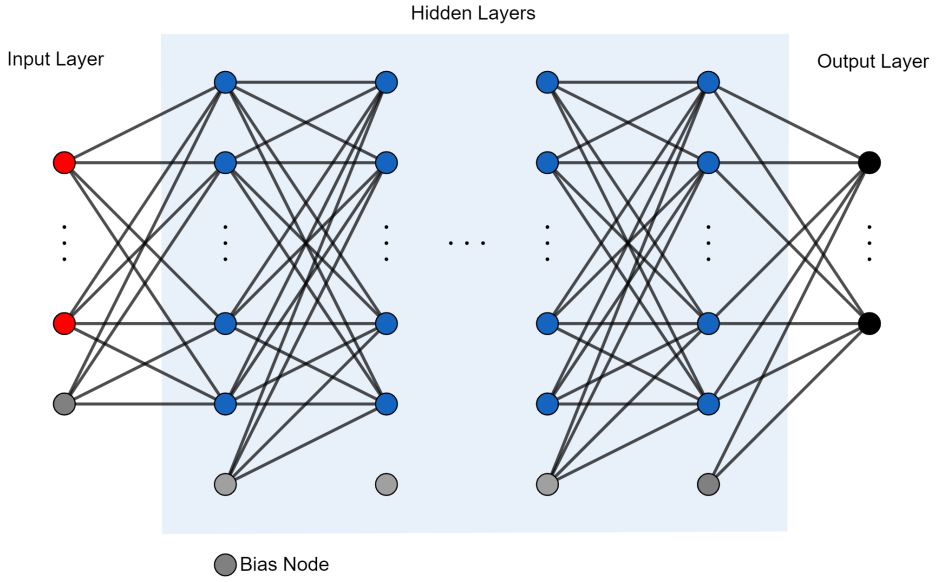


Figure 2.4: Artificial Neural Network

Most ANN follow the map shown in $Figure$ 2.5 when constructing their models and the process of finding the best hyperparameters falls into the design stage where the user decides on some of the inputs that the ANN will make us in order to learn the problem at hand. Hyperparameters may include: activation functions, number of training epochs, hidden layers, number of neuron/nodes per hidden layer, loss function, etc. Some hyperparameters are dependent on the type of problem being modeled by the NN, for example a ReLU activation which outputs $max\{0, x\}$ can not be chosen to activate the output layer in a problem where negative values are a possibility.

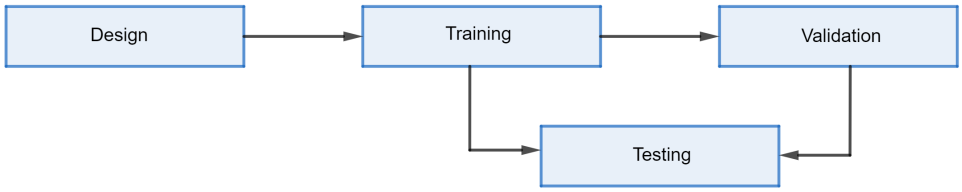


Figure 2.5: ANN Basic Construct

ANN can either be fully connected as shown in $Figure$ 2.4 where all nodes in the current layer are connected to all nodes in the next layer or partially connected where a node in one layer is not connected to all nodes in the next layer (Hamid and Habib, 2005). In a fully connected ANN the input for a node is a weighted sum of all the nodes in the previous layer including the bias node which is then operated or passed through some activation function. Suppose the input of a node j in the a layer is given by I_j, with the exception of the input layer of course, and let Φ_j be the activation for that layer then

$$I_j = \Phi_j \left(\sum_i x_i \omega_{i,j} + b_j \right) \tag{2.5.1}$$

where x_i is a node in previous layer and $\omega_{i,j}$ is the weight value associated with node x_i and node j. For a node to be activated (2.5.1) has to be greater than some threshold value. The importance of the bias node is that it activates the node in a case where the summation part in (2.5.1) is zero thus the bias will be such that it is greater than the threshold value.

The training of a NN can be achieved via Supervised Learning, Unsupervised Learning, Reinforcement Learning and Stochastic Learning (Sathya et al., 2013). The method of choice in this dissertation is Supervised Learning where the NN is trained by using an existing or historical dataset which consist of an input vector and its output vector. The training process is aimed at optimizing the weights of the network so they predict the output with the highest accuracy possible. This is mostly achieved by utilizing gradient descent to minimize some loss function. Suppose the loss function being used is the mean squared error (mse) then the difference between actual values $\hat{y}_i$ and predicted values y_i is measured by

$$MSE = \frac{1}{N} \sum_{i=0}^{N} (y_i - \hat{y}_i)^2. \tag{2.5.2}$$

ANN minimizes this error by optimizing the weights of the model through an update process where the minimum of the loss function is computed with respect to the weight of the NN via the gradient process thus

$$\begin{aligned} W_i^{new} &= W_i^{old} - \alpha \cdot \text{grad (MSE)} \\ &= W_i^{old} - \alpha \cdot \frac{\partial}{\partial W_i^{old}} (MSE) \end{aligned} \tag{2.5.3}$$

where α is the learning rate of the NN. Gradient decent minimizes the weights by moving in the negative direction of the gradient which is called the steepest descent. The platforms used for ANN modelling such as TensorFlow and PyTorch have built in commands that speed up the calculations of gradients

for the loss function. The choice of the cost function also falls under the category of hyperparameters for NN.

In most cases the training of a network is often done with the help of a validation data set which acts as a benchmark for training. The validation set is used for finding a combination of hyper parameters that yield the best results and as a consequence NN are often trained multiple times while varying the hyper parameters in each case. The use of the validation set is not a mandatory step when dealing with NN however it is a recommended practice that helps users to monitor the training process.

After training comes the testing process where a set of an input vector with its corresponding output vector is used to test the performance of the model on new unknown data. If the network output is satisfactory then one may proceed to deploy the model.

3. Methodology

To achieve calibration for the VG model the two-step approach as illustrated in $Figure$ 2.3 was implemented where in step 1 the ANN learns the pricing map and in step 2 the ANN uses weights from step 1 to achieve calibration. The calibration was done with respect to Vanilla call options which can then be extended to put options by using the put-call parity (2.1.3).

All python codes are available here

3.1 Data Collection

All the data sets used for training, validation and testing were generated through implementing the VG model via the COS method whose numerical formula can be given explicitly by combining (2.3.20) and (2.3.25) to get

$$C = e^{-rt} \sum_{k=0}^{N-1}{}' \operatorname{Re}\left\{ e^{iu\mu t} \cdot \left(1 - iu\theta\nu + \frac{\sigma^2 \nu u^2}{2}\right)^{-\frac{t}{\nu}} \cdot e^{iux} \cdot e^{-iua} \right\} \cdot V_k^{call} \tag{3.1.1}$$

where

$$u = \frac{k\pi}{b-a} \tag{3.1.2}$$

$$\mu = r - q + \frac{1}{\nu} \ln\left(1 - \theta\nu - \frac{\sigma^2 \nu}{2}\right), \tag{3.1.3}$$

$$x = \log\left(\frac{S_0}{K}\right). \tag{3.1.4}$$

The parameters, $\Theta = [\sigma, \theta, \nu]$, of the VG model were chosen randomly from literature defined bounds (Madan et al., 1998) where

$$\begin{aligned} \sigma &\in [0, 0.2], \\ \theta &\in [-0.3, 0.05], \\ \nu &\in [0, 0.7]. \end{aligned} \tag{3.1.5}$$

The pricing inputs, $\mathbf{p} = [S_0, K, T, r]$, were defined as follows:

$$\begin{aligned} T &\in \{0.5, 0.75, 1.0, 1.25\}, \\ K &\in \{0.8, 0.9, 1.0, 1.1, 1.2\}, \\ r &\in [0, 0.2], \\ S_0 &= 1. \end{aligned} \tag{3.1.6}$$

For each random combination of $[r, \sigma, \theta, \nu]$ a set of 20 option were produced corresponding to to all strike and maturities. For options with $S_0 \neq 1$ and strike K we multiply with the option price obtained for $S - 0 = 1$ and with strike K/S_0. The model was trained for 100 epochs with each containing 1024 random combinations of $[r, \sigma, \theta, \nu]$, thus resulting into a total of $1\ 048\ 000$ option prices. The model's validation was run separately from training due to how the code was set up with a validation data set of size 2048 resulting into $40\ 960$ option prices.

3.2 Pricing Neural Network

Multiple pricing ANN were implemented and tested for their pricing performance by choosing different hyper parameters and the model that was settled for is presented by $Figure$ 3.1.

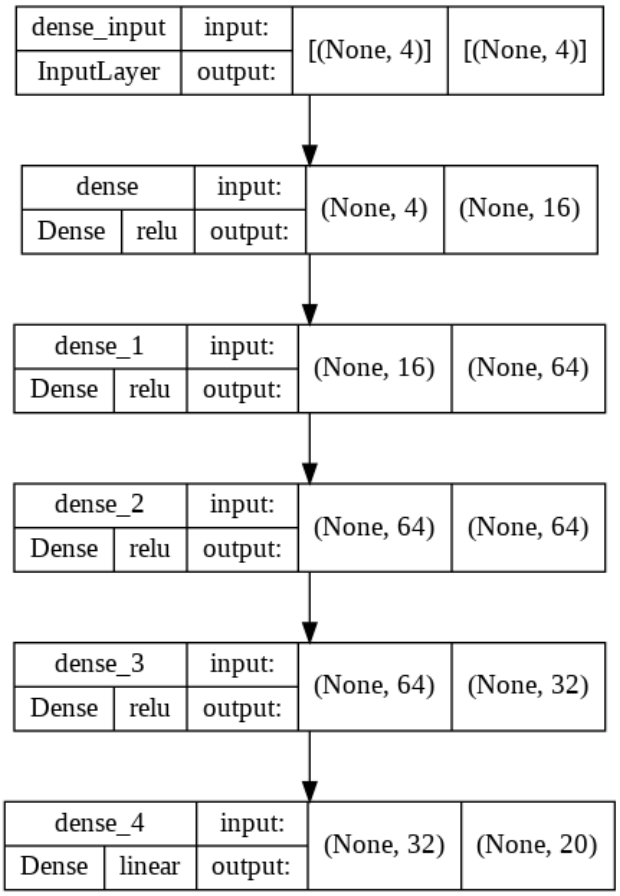


Figure 3.1: Pricing Neural Network Design

Training Epochs: The model was trained with 100 epochs. The training data size with 100 was large enough to achieve good results while keeping the training time relatively low as to allow hyperparameter fine tuning

Activation function: ReLU was chosen as it seems to be the standard choice in many studies. The input and output layers made use of the default linear activation function. The ReLU function is defined by

$$f(x) = \max\{x, 0\}.$$

Loss Function = Mean Squared Error(MSE): The MSE loss function 2.5.2 retained the highest pricing accuracy when compared to the Mean Absolute Error (MAE) thus it was the preferred choice.

Optimizer = Adam: Derived from "adaptive moment estimation" the Adam optimizer separately calculates an adaptive learning rate for the first and second moments of the gradient (Kingma and Ba, 2014). The Adam optimizer makes use of first order gradients and requires little memory.

Learning Rate $= 0.001$**:** This initial learning rate was decrease by a factor of 0.95 every fifth epoch.

3.3 Calibrating Neural Network

The calibrating NN differs from the pricing NN by that the input layer of the pricing NN is made into a hidden layer in the calibrating NN and then a new single node input is used instead. This new input layer inputs the value one into the NN and with the bias deactivated this allows the NN to learn the weights of the first hidden layer which represent the parameters for calibration. All other hidden layers make use of the weights obtained from training the pricing NN.

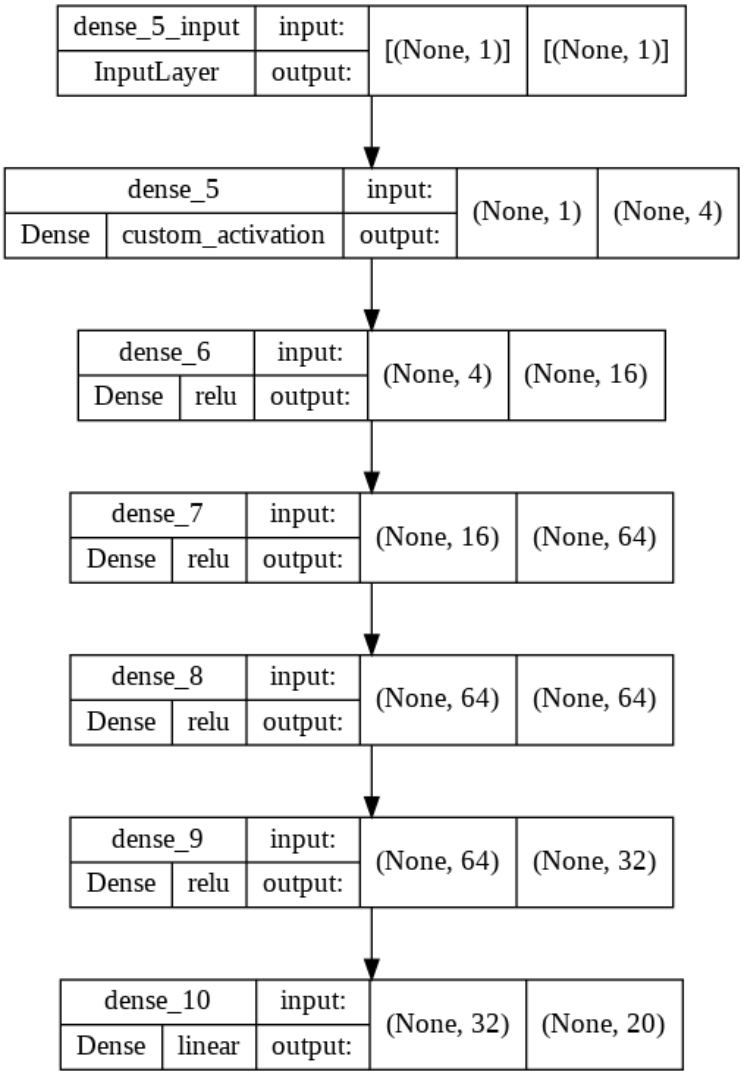


Figure 3.2: Calibration Neural Network Design

Activation Function: Many combination of $[r, \sigma, \theta, \nu]$ could result into the same option price thus to make the model more stable a custom activation function was used to bound the search space of the weights to be within the range of the parameters. This custom activation was obtained through modification of the sigmoid function

$$f(x) = \frac{1}{1 + e^{-x}}, \tag{3.3.1}$$

which outputs in the range $[0, 1]$. The custom function outputs in the range $[a, b]$ and this range depends on the parameter of interest, for example the volatility σ would range in $[0, 0.2]$. The custom function is the form

$$F(x) = (b - a) \cdot \frac{1}{1 + e^{-\left(\frac{x-c}{d}\right)}} + a, \tag{3.3.2}$$

where d controls the slope of the function and c is chosen such that it centers the range $[a, b]$ and again using the volatility as an example one would choose $c = 0.1$. Figure 3.3 shows how the custom sigmoid function looks like with respect to the bounds of each parameter.

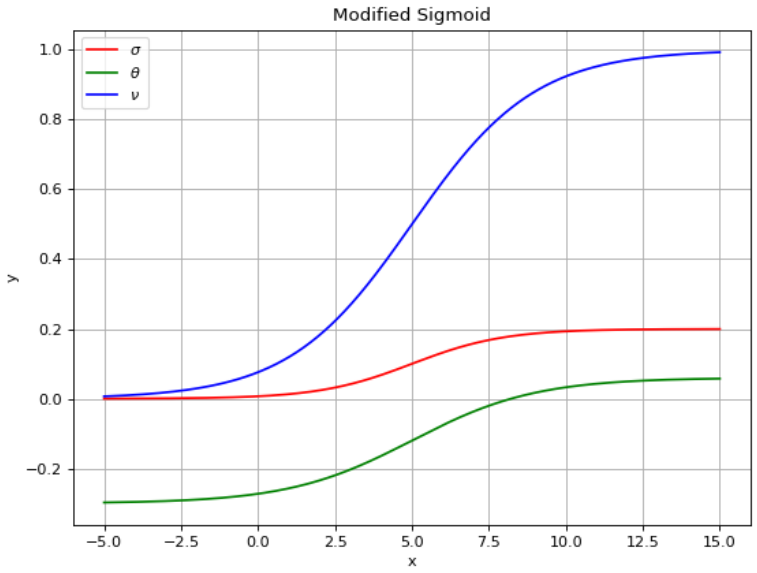


Figure 3.3: activation Function

Loss Function: The loss function used for the calibrator was also the MSE (2.5.2) as it again performed much better than MAE.

Initial First Hidden Layer Weights: To further reduce the search space the weights of the first hidden layer were initialized randomly from a normal distribution with mean zero and standard deviation of 0.5.

Training Epochs: The calibrator NN was trained for 100 epochs in a loop that looped eight times. This proved to yield good results.

Learning Rate: The calibrator NN started with an initial learning rate of 0.001 which was decreased through a learning rate scheduler by a factor of 0.95 every 10 epochs.

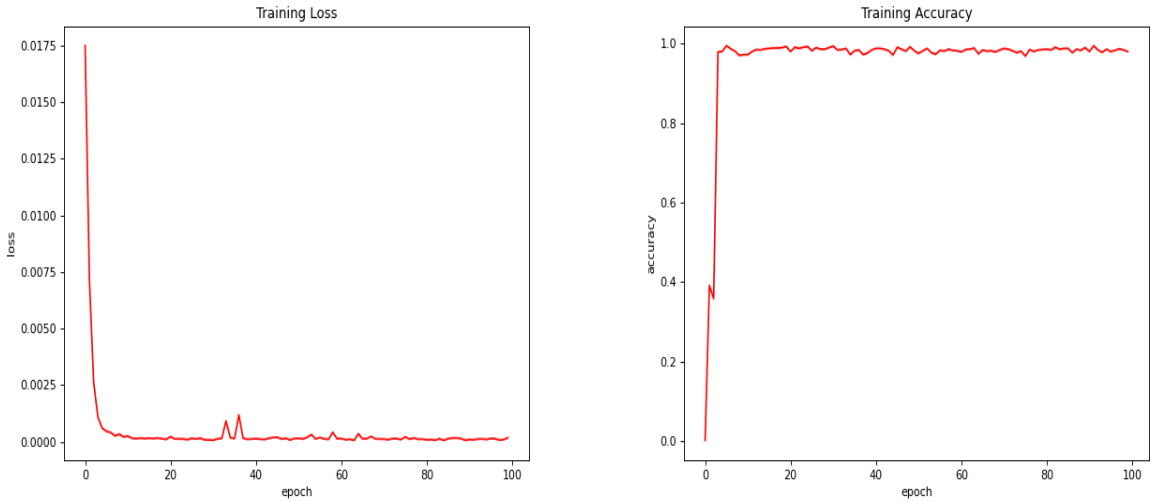
3.4 Hardware and Software

All computations were run using Google Colaboratory which on the *25 May 2022* was running the following: TensorFlow 2.8.0, Python 3.7.13, and Intel(R) Xeon(R) CPU 2.20GHz. Codes were only run using the CPU only.

4. Results

4.1 Pricing NN

During the training process the pricing NN attained good training levels in terms of the loss and accuracy. Even though the graphs presented on figure 4.1 are not smooth curves they do show successful training of the model with the highest training accuracy of about 98%. The training loss is with respect to the MSE loss function (2.5.2). The unevenness of the graphs may have been due to many factors such as the initial learning rate or maybe the fact that model was trained with a new data set in every epoch thus it had to learn to fit a wide range of option prices for different parameters. These results did prove to be sufficient in learning the pricing model.



(a) Loss (MSE)

(b) Accuracy

Figure 4.1: Training Performance

Training the pricing NN proved to be time consuming as it would take hours to train and even though this was an expected outcome it did make it difficult to fine tune the hyperparameters since changing a single hyperparameter meant that the model had to be trained all over again. Once trained the pricing NN did prove to be slightly faster than the COS method. The speed between these two models was tested by pricing 1000 different parameters each resulting into 20 option prices and the pricing NN was about 0.667 faster than the COS method.

To test the performance of the pricing NN option prices obtained from it were plotted against prices obtained from the COS method for the same parameters: r, σ, θ and ν. The two models were tested with 30 different values of the parameters per simulation hence resulting into 600 options being compared in order to gauge the overall performance of the pricing NN. Figure 4.2 shows the pricing NN performed relatively well when compared to the COS method. However there were instances where the pricing NN failed to properly match the prices of the COS method as shown by figure 4.3. Even though sometimes the pricing NN would fail to properly price some options, it would only be a few options and most likely the ones with high strike prices.

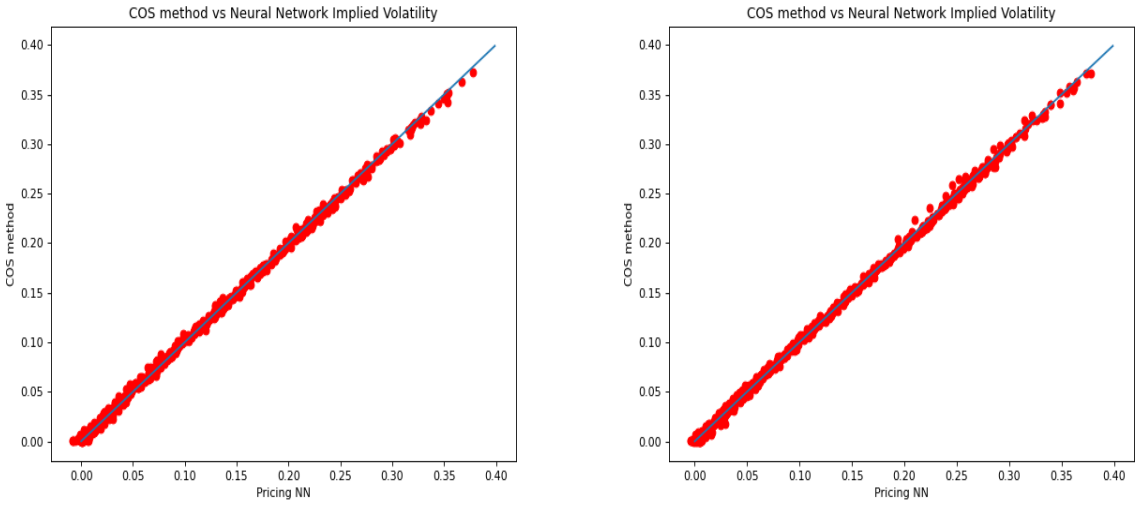


Figure 4.2: Pricing NN vs COS Method

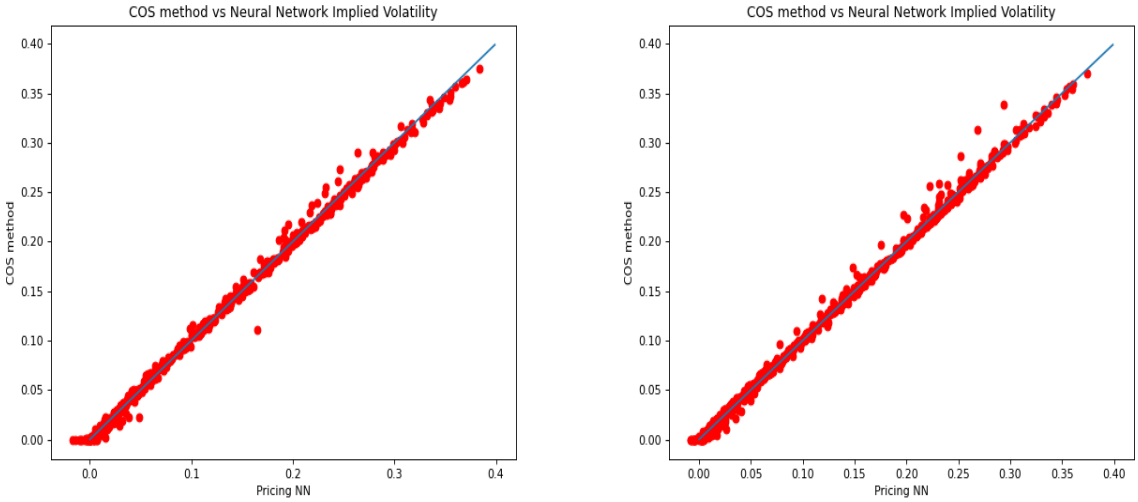


Figure 4.3: Pricing NN vs COS Method

4.2 Calibrator NN

A similar procedure was applied when testing the performance of the calibrating NN where prices were compared for actual and calibrated parameters. Figure 4.4 show two cases where the Calibrating NN performed relatively good as the predicted parameters produced prices that match the actual parameters. Figure 4.5 present a case where the overall behaviour of the NN can be deemed good while showing instances where some parameters failed to match the actual parameters.

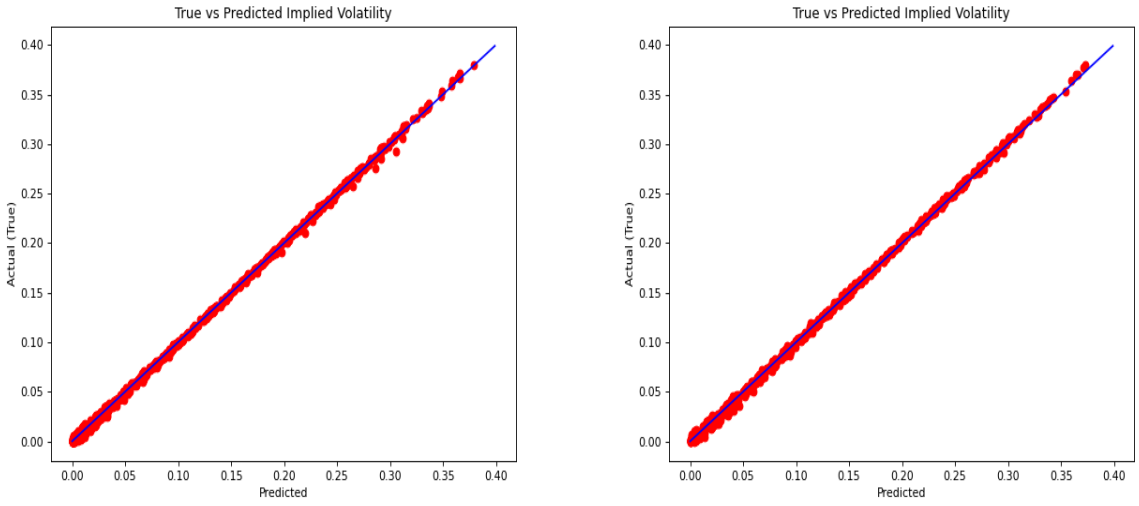


Figure 4.4: Actual vs Calibrated Parameters

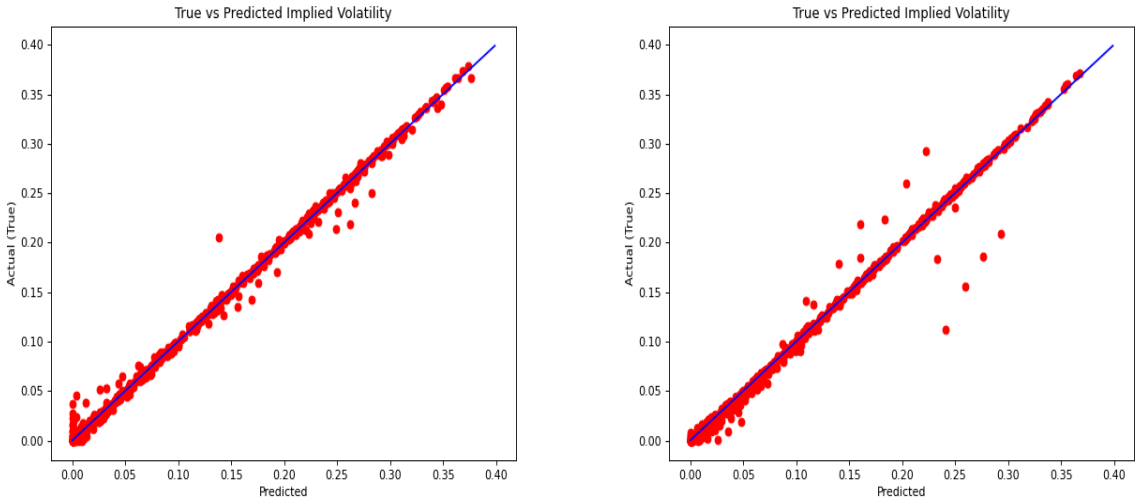


Figure 4.5: Actual vs Calibrated Parameters

However figures 4.4 and 4.5 do not show whether the parameters were calibrated or the calibrating NN just return parameters that result into the same prices as the actual parameters. For example the predicted parameters in table 4.1 recover the prices obtained from the actual parameters as shown by figure 4.6.

	r	σ	θ	ν
Actual	0.11591396	0.02578096	-0.21010623	0.61750327
Predicted	0.12739283	0.1292924	-0.01364869	0.16664836

Table 4.1: Actual vs Price Recovery Parameters

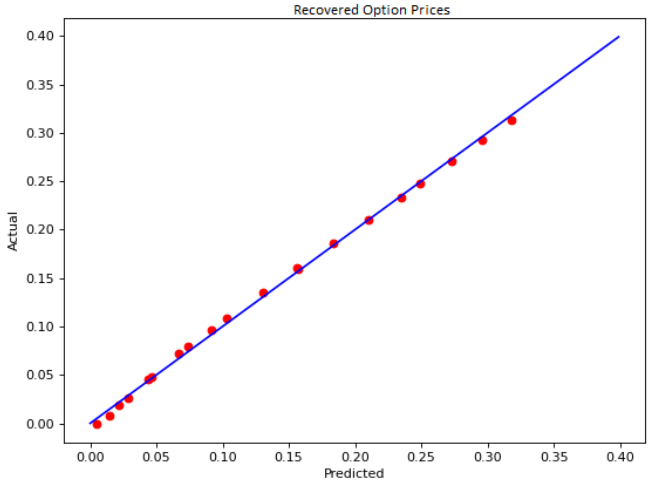


Figure 4.6: Actual vs Price Recovery Parameters

In most cases, about more than 95% of the time, the calibrator NN was able to return good calibrated parameters. Table 4.2 and figure 4.7 show an example of a random case where the calibrator yielded proper calibrated parameters. Parameters that recover option prices were mostly observed with the first few executions of the calibrator NN after which it became stable in returning proper calibrated parameters.

	r	σ	θ	ν
Actual	0.19196982	0.1188308	0.04525136	0.29421652
Predicted	0.19196849	0.11882527	0.04524997	0.29427677

Table 4.2: Actual vs Calibrated Parameters

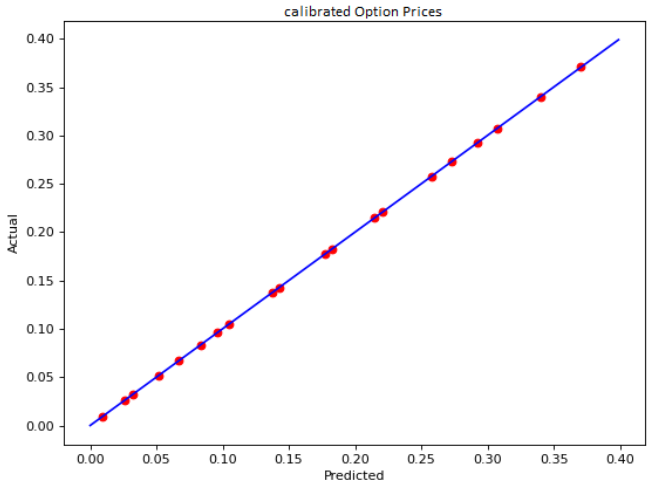


Figure 4.7: Actual vs calibrated Parameters

In terms of the amount of time spent for the calibration process, provided with 20 observed prices the calibrator NN was able to return calibrated parameters in an average of about 6 seconds. This speed was found to be acceptable when compared to traditional methods.

In reality model parameters are not observed from the market and hence all that matters is whether the model can return parameters that match the observed prices or not. The issue presented by table 4.1 and 4.2 does not really exist as there would be no benchmark parameters to compare to. Since the calibrating NN proposed in this dissertation is able to recover parameters that match the observed prices, then we could say the model was able to achieve the calibration task.

5. Conclusion

The aim in this dissertation was to achieve the calibration of the Variance Gamma model via the two-step approach using neural networks. In the first step a pricing neural network model was trained to learn the mapping imposed by the Variance Gamma model which was priced via the COS method. The neural network was trained by utilizing supervised learning in a feed forward pass which proved to be a time consuming process, however the pricing neural network was able to adequately learn the pricing map with a high accuracy. The model did exhibit cases where it returned prices which did not match those obtained from the COS method but this was not a generally observed phenomenon and was only for large option strike prices.

In the second step the pricing network is modified by adding a new input layer of a single input and the original input layer from the pricing neural network becomes the first hidden layer thus creating the calibration neural network. The weights from the pricing neural network are fixed into the calibrator neural network and the bias in the first hidden layer of the calibrator is deactivated so that the weights are the parameters being actually optimized by the neural network. Since the input into the calibrator is "one", the optimal weights obtained from the first hidden layer represent the parameters of the Variance Gamma model. Since model parameters are unknown the task is for the model to return parameters that actually match market observed prices and judging by the presented results, this was achieved successfully. The calibration process was fast to return calibrated parameters as it took about $6s$, which is much faster than most traditional methods.

All computations done in the dissertation were executed online through Google Colaboratory only using the CPU therefore more hyperparameters could have been tested using the GPU possibly improving the performance of the model.

Acknowledgements

I would like to acknowledge AIMS and it's funders for the opportunity to have been part of the AIMS community, the experience I gained has been transforming to me. I would to thank my supervisor and project tutor for the patience, time, dedication and guidance they provided throughout this dissertation. I want to also acknowledge my family and friends for the support they provided during my stay at AIMS South Africa. Lastly, I would like to thank myself for not giving up and trying my best.

References

Bayer, C., Horvath, B., Muguruza, A., Stemper, B., and Tomas, M. On deep calibration of (rough) stochastic volatility models. *arXiv preprint arXiv:1908.08806*, 2019.

Cont, R. and Da Fonseca, J. Dynamics of implied volatility surfaces. *Quantitative finance*, 2(1):45, 2002.

Fang, F. and Oosterlee, C. W. A novel pricing method for european options based on fourier-cosine series expansions. *SIAM Journal on Scientific Computing*, 31(2):826–848, 2009.

Fiorani, F. Option pricing under the variance gamma process. *Available at SSRN 1411741*, 2004.

Hamid, S. A. and Habib, A. Can neural networks learn the black-scholes model? a simplified approach. 2005.

Hernandez, A. Model calibration with neural networks. *Available at SSRN 2812140*, 2016.

Hirsa, A. *Computational methods in finance*. CRC Press Boca Raton, FL, 2013.

Itkin, A. Deep learning calibration of option pricing models: some pitfalls and solutions. *arXiv preprint arXiv:1906.03507*, 2019.

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

Kumar, A. A study on risk hedging strategy: Efficacy of option greeks. *Abhinav National Monthly Refereed Journal of Research in Commerce & Management*, 7(4):77–85, 2018.

Liu, S., Borovykh, A., Grzelak, L. A., and Oosterlee, C. W. A neural network-based framework for financial model calibration. *Journal of Mathematics in Industry*, 9(1):1–28, 2019a.

Liu, S., Oosterlee, C. W., and Bohte, S. M. Pricing options and computing implied volatilities using neural networks. *Risks*, 7(1):16, 2019b.

Madan, D. B. and Milne, F. Option pricing with vg martingale components 1. *Mathematical finance*, 1(4):39–55, 1991.

Madan, D. B. and Seneta, E. The variance gamma (vg) model for share market returns. *Journal of business*, pages 511–524, 1990.

Madan, D. B., Carr, P. P., and Chang, E. C. The variance gamma process and option pricing. *Review of Finance*, 2(1):79–105, 1998.

Mostafa, B. M., El-Attar, N., Abd-Elhafeez, S., and Awad, W. Machine and deep learning approaches in genome. *Alfarama Journal of Basic & Applied Sciences*, 2(1):105–113, 2021.

Orlando, G. and Taglialatela, G. A review on implied volatility calculation. *Journal of Computational and Applied Mathematics*, 320:202–220, 2017.

Sathya, R., Abraham, A., et al. Comparison of supervised and unsupervised learning algorithms for pattern classification. *International Journal of Advanced Research in Artificial Intelligence*, 2(2): 34–38, 2013.

Van Noortwijk, J., Kallen, M., and Pandey, M. Gamma processes for time-dependent reliability of structures. *Advances in safety and reliability, proceedings of ESREL*, pages 1457–1464, 2005.