YUAN_BO_01781777.pdf

by Bo Yuan

Submission date: 05-Sep-2022 09:07PM (UTC+0100)
Submission ID: 185723983
File name: YUAN_BO_01781777.pdf (4.03M)
Word count: 14627
Character count: 74541

Imperial College London

Imperial College London

Department of Mathematics

Deep learning interpretability of the parameters to smile map in a rough volatility model

Author: Bo Yuan (CID: 01781777)

A thesis submitted for the degree of

MSc in Mathematics and Finance, 2021-2022

Declaration

The work contained in this thesis is my own work unless otherwise stated.

Acknowledgements

First and foremost, I would like to express my deep gratitude to Prof. Damiano Brigo for his continuous encouragement and timely feedback on my MSc project. His advice was invaluable during the research and writing of the thesis. I also thank Dr. Jack Jacquier for his comments and support with data generation using rHeston. I would also like to thank Dr. Mikko Pakkanen for his insightful advice on NN training.

Particularly, I would like to thank Mediobanca for having me work on the project. Special thanks to Dr. Nicola Pede for his time and advice throughout the internship.

Last but not least, I want to thank my family and friends for their ongoing encouragement and support.

Abstract

The rough volatility model, which can reproduce the implied volatility's at-the-money skew, is widely applied. However, one major drawback of the rough volatility model is the slow calibration, which deep learning approaches can expedite. Because neural networks (NN) are black boxes, interpretability is a critical consideration.

In this research, we construct volatilities using the rough Heston (rHeston) model and the Fourier inversion technique, then fit feedforward neural networks (FNN) to the synthetic data, and last investigate the interpretability of the deep learning model we fitted.

We discover that FNN with basic architectures are capable of obtaining predicting accuracy of more than 90%. The self-defined boundaries of parameters we specify for rHeston matter in the predictive performance of the fitted FNN. Furthermore, when it comes to attributions for particular parameters, the model interpretation from rHeston matches the results from interpretability analysis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

We give an introduction by discussing why this topic is meaningful, what we do, and how we contribute to the existing literature.

1.1 Motivation

The financial industry has developed drastically during the last decade. One major flaw in classical stochastic volatility models is their inability to replicate the at-the-money (ATM) skew of implied volatility that is seen in the market. It has been demonstrated [3] that approximate fractional processes allow us to reconstruct the behaviour of historical volatility time series quite well. Rough volatility models have been created to replicate the pattern of past volatility evidence and match the volatility surface amazingly well with few parameters. The improved performance of the rough volatility model pushes us to continue working on it.

In fact, one of the few possible limitations of such models is their complexity in execution, as the calibration procedure is fairly sluggish. As a result, it is sensible to explore integrating deep learning techniques to accelerate calibration.

On the occasion that it comes to machine learning models, interpretability, which is characterized as "the extent to which a human can comprehend the cause of a decision" [4] or "a human can reliably anticipate the model's conclusion" [5], is an essential consideration since it is beneficial for troubleshooting the model and making informed judgments about how to improve it. Thus, we work on the interpretability of the rough volatility model to study the contribution of each feature (price/volatility) to the label (parameter of the rough volatility model).

The project can be divided into three parts broadly as illustrated in the Figure 1.1. First, we generate a data set of volatilities/prices using rHeston, which maps the model parameters to the volatilities/prices. Then, we fit the data set with the NNs and compare performances across them, which maps the generated volatilities/prices back to the model parameters of rHeston. Finally, we study interpretability of the deep learning models we fitted.

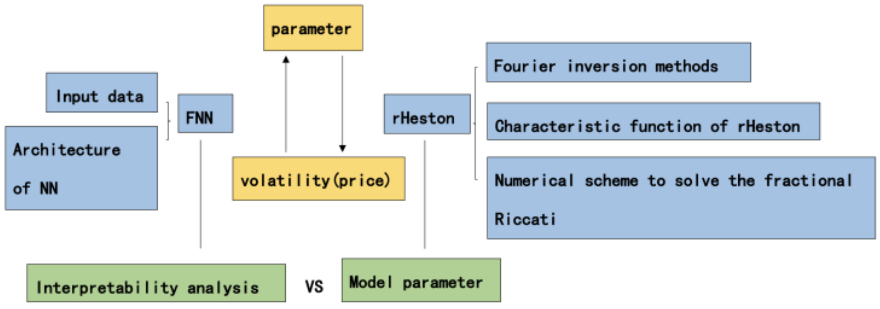


Figure 1.1: Framework of the project

1.2 Literature review

Previous work related to our project can be categorised in three subgroups: pricing with the rough stochastic volatility, deep calibration of the volatility model, and interpretability of NNs.

1.2.1 Pricing with the rough stochastic volatility

According to Gatheral et al. [3], a fractional Brownian motion (fBm) has a similar behaviour pattern to the logarithm of realised variance. Specifically, it is like fBm with Hurst exponent H of order=0.1. This was achieved by evaluating the history of realised variance using recent high-frequency data. The final Rough Fractional Stochastic Volatility (RFSV) model exhibits striking agreement with the financial data over time. This introduced the rough volatility model.

On the other hand, the variance of the rough volatility model is not semi-martingale due to the reason that the Markovian property no longer exists. Then, it's highly likely that the option payoff cannot be represented in a direct analytical stochastic process. The characteristic function of the density of these processes can be projected to the payout in Fourier space to obtain the option values. The inversion methods are a product of this approach. Stein and Stein [6] initially suggested the inversion approach as a way to determine the distribution that underlies a stochastic volatility model. Then, Heston [7] discovered an analytical method for European option pricing.

There are two types of inversion procedures described in the literature. Bankshi and Madan [8] propose a Black-Scholes-like formula for inverting the cumulative distribution function for option pricing. Having gained insights from the fast Fourier Transform, Carr and Madan [9] devise a numerical technique directly projecting the Fourier transform to the whole option price by the characteristic function.

1.2.2 Interpretability of NNs

Machine learning has grown in popularity in recent years, and its use has spread to almost every industry. The review of the majority of current deep learning applications in finance is provided by [10]. [11] offers extensive insights on the idea of interpretability in machine learning and widely adopted interpretability models as a response to the black-box issue.

In reality, there are several ways to classify machine learning interpretability methodologies. There are several points of view that set apart and could further segment these methodologies ([12]). Figure 1.2 shows a mind-map that highlights the various criteria that might be used to categorise an interpretability model.

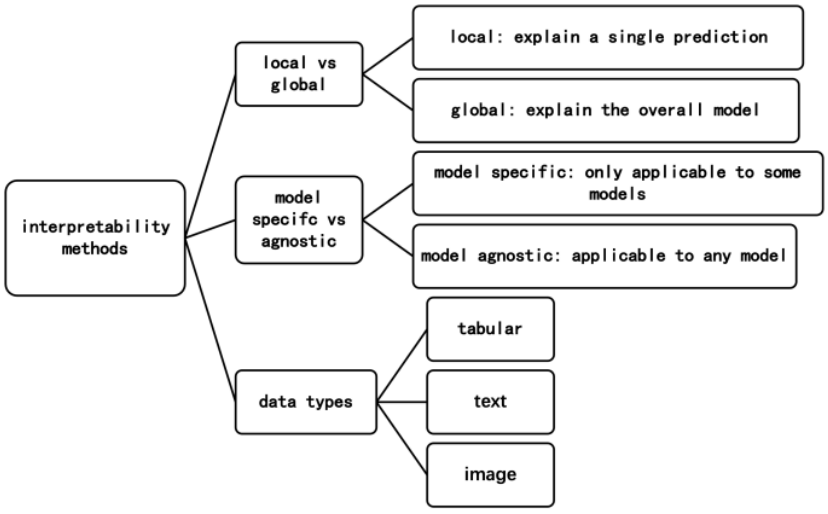


Figure 1.2: Classfication of interpretability models

The increased application of deep learning approaches, as well as their natural association with the black-box mechanism, has resulted in a substantial quantity of experimentation and research effort centred on them, as well as techniques for interpreting their results. LIME, which stands for Local Interpretable Model-agnostic Explanations, is a solution proposed by [13]. It focuses on interpreting individual predictions locally. The Gradient*Input method, which was used to create saliency maps following the gradient method, was improved significantly by multiplying the gradient with the input feature, which is indeed a Taylor approximation (with 1st order) of the shift of the come with input as zero. DeepLIFT, as described in [14], is an improvement over this method. By propagating backwards from predictions made by very complicated deep NNs, Layer-wise Relevance Propagation (LRP) [15] is a breakdown of nonlinear classifier approach that makes these networks interpretable. A game-theory based technique called Shapley Additive explanations (SHAP) [16] aims to improve interpretability by estimating the significance values of characteristics for certain predictions. It can interpret global model predictions. [17] experiment with a variety of interpretability approaches, including gradient-based attribution methods like DeepLIFT, LRP, Saliency maps, Integrated Gradients, and Gradient*Input, as well as perturbation-based attribution methods [18].

1.2.3 Deep calibration of the volatility model

There are two main groups of techniques in the literature using an artificial NN to calibrate rHeston. Horvath, Muguruza, and Tomas [19] describe a two-step deep calibration technique in which a NN first learns the map from model parameters to inferred volatilities, and then a standard solver is used to calibrate the majority of optimum model parameters. Recently, Roeder and Dimitrof [20] propose an alternative direct technique in which a FNN approximates the association between implied volatilities and model parameters. Plus, they evaluate the findings of the direct and indirect procedure, and calrify that the direct method of the rough volatility model calibration is more efficient and accurate.

Additionally, the Heston model's results [2] show that the FNN performs better than the convolutional neural network (CNN). In this project, described in Section 3, we reproduce the direct inverse map technique and build a FNN.

1.2.4 Our contribution

Our work further develops the work in the following aspects.

First and foremost, we extend the study on deep learning interpretability of a general volatility model (Heston) [2] to a rough volatility model (rHeston). Second, by comparing the performances of different NNs with the volatilities/prices as the features and the parameters as the labels, the project sheds light on how to calibrate the rough volatility model relatively fast as well as how to choose the architecture of the NNs. What's more, the work highlights that the range of the parameters matters in the performance of FNN.

Plus, employing NNs, the study carries out both local and global interpretability analysis and assesses whether they are good enough to provide insightful information into the calibration procedure.

Last but not least, the project provides a relatively fast approach to pricing with rHeston by using inversion of the Fourier Transform as well as numerically solving the fractional Riccati equation.

For further work, we suggest obtaining turn to the noisy actual market data.

1.3 Outline of the thesis

We organise the structure of the paper as below. Section 2 begins with an introduction of the rough volatility model as well as an overview of various rHeston representations, then moves on to the three steps of pricing using rHeston, and finally touches on the specific settings of our implementation. Section 3 discusses how we preprocess the data before fitting the FNN, how we construct the FNN architecture, how self-defined parameter boundaries affect FNN performance, and how the fitted FNN behaves within and outside of the sample. Section 4 introduces several local interpretability models, certain gradient-based explanatory approaches and one global interpretability model (SHAP). The results of the interpretability study are then discussed and compared to the

interpretation of rHeston for each parameter. Section 5 summarises our contributions and suggests the future study.

Chapter 2

Data generation with rHeston

2.1 rHeston: introduction

2.1.1 Rough volatility model

It's seen that the traditional stochastic models are not able to generate the real volatility surface observed in the market. Statistical analyses on high-frequency market data support the notion that spot volatility operates in a "rougher" manner than previously assumed [3].

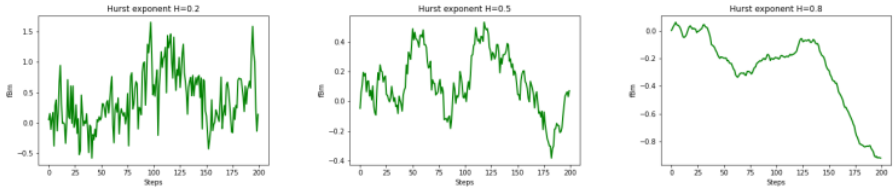
The standard approach is to add extra volatility components. This is how the rough fractional stochastic volatility model was introduced [3]. As a result, the financial industry has begun to characterise volatility dynamics as fBm W^H with Hurst parameter $0 < H < \frac{1}{2}$, which is a generalisation of the classical Brownian Motion (BM) previously used but the increments are not independent.

The fBm is a continuous-time centred self-similar Gaussian process indicated as $\left\{W_t^H; t \in \mathbb{R}\right\}$ on $[0, T]$, of which the mean for all $t \in [0, T]$ and the initial value are both zero. It has stationary increments and covariance function is:

$$\mathbb{E}\left[W_t^H W_s^H\right] = \frac{1}{2}\left\{|t|^{2H} + |s|^{2H} - |t-s|^{2H}\right\}$$

where H is a real number in $(0, \frac{1}{2})$, called Hurst parameter [21]. This parameter describes the roughness/raggedness of the motion. For $H = \frac{1}{2}$, the process is indeed a classical Brownian Motion.

For illustration, I follow the Cholesky decomposition method [22] to simulate three fBm with Hurst exponent H=0.2, 0.5, and 0.8, respectively. Figure 2.1 displays the three paths by simulation, through which we notice H describes the smoothness of the process.



(a) H=0.2 (b) H=0.5 (c) H=0.8

Figure 2.1: Simulated fBm paths

To make it more clear, some important properties of fBm are listed as below:

- fBm W_1^H is a Gaussian process. To be more specific, it is with mean of zero and variance of one.
- fBm W is self-similar with Hurst exponent H. Mathematically,

$$W_{kt}^H \overset{dist}{=} W_t^H$$

where k should be positive, and $\overset{dist}{=}$ indicates that the processes on both sides have the same distributions.

- fBm W^H has stationary increments across time. Mathematically,

$$W^H_{t+k} - W^H_t \overset{dist}{=} W^H_k$$

To put it in other words, increments of fBm with the same time intervals are identically distributed. It should be highlighted that the average increments across time intervals are zero all the time.

- It should be noted that the increments of fBm are not independent of each other with the correlation:

$$\rho_k = \frac{(t+1)^{2H} - t^{2H} - (t^{2H} - (t-1)^{2H})}{2}$$

where k is the time interval and t is the initial starting time. Furthermore, if $H \in (\frac{1}{2}, 1)$, $\rho>0$, then the process is called persistent. If $H \in (0, \frac{1}{2})$, $\rho<0$, then the process is called anti-persistent. We can view the property in a way that if W^H was increasing in the past across time, it is expected to grow as well in the future and vice versa. In a similar way, if W^H diminished in the past as time went by, it is expected to go up in the future and vice versa.

- It should be emphasized that for $H \neq \frac{1}{2}$, fBm is not a martingale or a Markov process or a semimartingale.

There are multiple representations of fBm in terms of BM. The most common one is the Mandelbrot-Van Ness representation [23], where we are able to build a fBm W^H from a two-sided BM:

$$\begin{aligned} W^H_t = &\frac{1}{\Gamma(H+1/2)} \int_{-\infty}^{0} \left((t-s)^{H-1/2} - (-s)^{H-1/2} dW_s + \right. \\ &+ \frac{1}{\Gamma(H+1/2)} \int_0^t (t-s)^{H-1/2} dW_s \end{aligned}$$

The models which exploit the property of fBm are called rough volatility model with the general formula [24]:

$$\begin{aligned} dS_t &= \mu S_t dt + \sqrt{\nu_t(t)} S_t dW_t \\ d\nu_t(u) &= \lambda(t, u, \nu_t)\, dW^H_t \\ \rho dt &= \mathbb{E}\left[dW_t dW^H_t \right] \end{aligned}$$

The parameter set is $\{\mu, \nu_0, \lambda, \rho\}$. The interpretation of the elements in the formula is listed below:

- W is BM, W^H is fBm, ρ measures correlation between BM W and fBm W^H
- S is asset price, μ is drift of asset price
- ν_t is instantaneous volatility, which starts from ν_0

The most notable improvement of the rough world is that the models can recreate better fitting with not only the implied volatility surface but also the historical volatility dynamics. It should be noticed that the rough fractional driver has limited memory, which is contradictory to the traditional long-term memory of volatility.

2.1.2 Representations of rHeston

Specifically, to generate enough volatility smiles to train the network, this paper works on rHeston to find a fast way to price options in the rough model. Plus, it would be a natural choice given that the original study on this topics used Heston [2]. It is assumed in the basic Heston model that asset price S_t can be written in the form as below:

$$
\begin{aligned}
dS_t &= \mu S_t dt + \sqrt{\nu_t(t)} S_t dW_t^S \\
d\nu_t &= \kappa(\theta - \nu_t)dt + \xi\sqrt{\nu_t} dW_t^\nu \\
\rho dt &= \mathbb{E}\left[dW_t^S dW_t^\nu\right]
\end{aligned}
$$

The parameter is set $\{\mu, \nu_0, \kappa, \theta, \xi, \rho\}$. The interpretation of the elements in the formula is listed below.

- W_t^S and W_t^ν are BM with correlation ρ
- S_t is asset price, μ is drift of asset price
- ν_t is positive instantaneous volatility with the Feller condition $2\kappa\theta > \xi^2$, ν_0 is the initial value of the variance
- κ measures how fast the process converges to the mean
- θ is the average variance of asset price over a long period
- ξ is volatility of volatility

There is no formal definition of rHeston, thus different versions appeared in the existing literature. As proposed in [25], using the Mandelbrot-Van Ness representation, rHeston can be written as:

$$
\frac{dS_t}{S_t} = \sqrt{V_t}\left\{\rho dB_t + \sqrt{1-\rho^2} dB_t^\perp\right\}
$$

with

$$
V_u = V_t + \frac{\kappa}{\Gamma\left(H+\frac{1}{2}\right)} \int_t^u \frac{\theta^t(s) - V_s}{(u-s)^{\frac{1}{2}-H}} ds + \frac{\nu}{\Gamma\left(H+\frac{1}{2}\right)} \int_t^u \frac{\sqrt{V_s}}{(u-s)^{\frac{1}{2}-H}} dB_s, \quad u \geq t
$$

All the free parameters are collected in the set $\{\rho, V_0, \lambda, H, \theta^t, \nu\}$. The interpretation of the elements in the formula is listed below.

- B_t and $B_t^\perp$ are perpendicular BMs, which is the decomposition of the BM of the asset price S_t. $\rho \in [-1, 1]$ measures the correlation between the asset price S_t and the volatility fluctuations B_t.
- V_t is variance of the asset price, which start from V_0
- $\theta^t(\cdot)$ is the long-term mean parameter, which is $\mathcal{F}_t$-measurable ensuring the consistency in time
- κ is rate of mean convergence
- ν is volatility of volatility
- Γ denotes the Gamma function
- $H \in (0, \frac{1}{2})$ is the Hurst exponent, on the occasion of $H \to \frac{1}{2}$, the formula gives back to the Heston model

Another version of rHeston can be rewritten in forward variance form as [25]:

$$
\frac{dS_t}{S_t} = \sqrt{V_t}\left\{\rho dB_t + \sqrt{1-\rho^2} dB_t^\perp\right\}
$$

with

$$
V_u = \xi_t(u) + \frac{\nu}{\Gamma\left(H+\frac{1}{2}\right)} \int_t^u \frac{\sqrt{V_s}}{(u-s)^{\frac{1}{2}-H}} dB_s, \quad u \geq t
$$

which can be useful due to the reason that the observed forward variance curve ξ_t can imply the value of $\lambda\theta^t$ [26].

2.1.3 Representation of rHeston used in the project

Our work builds up on the version of rHeston proposed in [27] where the long-term mean is constant:

$$dS_t = S_t\sqrt{V_t}dW_t$$
$$V_t = V_0 + \frac{1}{\Gamma\left(H+\frac{1}{2}\right)}\int_0^t (t-s)^{H-\frac{1}{2}}\kappa\left(\theta - V_s\right)ds + \frac{1}{\Gamma\left(H+\frac{1}{2}\right)}\int_0^t (t-s)^{H-\frac{1}{2}}\kappa\nu\sqrt{V_s}dB_s$$

All the free parameters are collected in the set $\{\rho, V_0, \kappa, H, \theta, \nu\}$. The interpretation of the elements in the formula is listed below.

- $\rho \in [-1, 1]$ measures the correlation between spot W and volatility moves B
- V_t is variance of the asset price, starting from V_0
- θ describes the constant long-term mean
- κ is rate of mean convergence
- ν is volatility of volatility
- Γ denotes the Gamma function
- $H \in (0, \frac{1}{2})$ is the Hurst exponent controlling roughness of the volatility, when $H \to \frac{1}{2}$, the formula gives back to the classical Heston model

2.2 Pricing with rHeston

We aim to generate a data set of volatilities/prices with rHeston for the next step of deep calibration. Pricing here is done via reversing the Fourier transform for characteristic function. However, the characteristic function is not obtainable in closed form for rHeston, which involves solving the fractional Riccati equation. As there is no pure closed-form solution, the Adams-Bashforth-Moulton numerical scheme is implemented to get a semi-analytical solution to the characteristic function for rHeston.

2.2.1 Fourier inversion methods for pricing

Fourier transform methods are becoming more significant in asset pricing. To be more specific, Fourier inversion methods are quite useful in deriving derivative prices, especially on the occasion that the characteristic function for the price process can be computed analytically or numerically all the time.

Compared with the approach to modelling the price with the aid of an analytical process, there exists an easier way to obtain the option price for the complicated process, that is, projecting the density's characteristic function to the payout in the Fourier space. In the field of modern probability, the Fourier transform is widely applicable. In terms of implementation, the key takeaway for the inversion of the Fourier transform is integrating the option price payoff over the distribution. There is growing interest in employing characteristic functions and Fourier transforms to apply these approaches since the simple Gaussian model described by the distribution no longer exists and the models have become far more complex which are more simply derived by a characteristic function instead.

For a random variable, it is at almost all times feasible to determine the characteristic function, which contributes to the effectiveness of Fourier transform methods. Suppose we know the characteristic function (analytically or numerically). In that case, we may compute the distribution function using the Inversion theorem, which states that the characteristic function has a one-to-one correspondence to the distribution and vice versa. An inverse Fourier transform connects the two sides of the medal. Specifically, based on the representation developed by [28], the reciprocal link between the characteristic function $g_X(u)$ and the probability density function $f_X(x)$ of a real-valued random variable X can be written as:

$$g_X(u) = \mathbb{E}\left[e^{iuX}\right] = \int_{-\infty}^{\infty} e^{iux} f_X(x)dx$$

$$f_X(x) = \mathbb{F}^{-1}\left[g_X(u)\right] = \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{-iux} g_X(u) du$$

where $\mathbb{F}[\cdot]$ denotes the Fourier transform.

In literature, there are two classes of inversion methods:

- Based on the idea of Bakashi and Madan [8], the first approach develops a Black-Scholes like pricing formula using the characteristic function. Assuming no dividends and constant interest rates, the option price can be determined as :

$$C\left(S_0, T, K\right) = \Pi_1 S_0 - e^{-rT}\Pi_2 K$$

where

$$\Pi_1 = \frac{1}{2} + \frac{\int_0^{\infty} \mathrm{Re}\left(\frac{e^{-iu\log(K)} g_T(u-i)}{iu g_T(-i)}\right) du}{\pi}$$

$$\Pi_2 = \frac{1}{2} + \frac{\int_0^{\infty} \mathrm{Re}\left(\frac{e^{-iu\log(K)} g_T(u)}{iu}\right) du}{\pi}$$

Here $g_T(u)$ is characteristic function.

- Based on the idea of Carr and Madan [9], the second approach projects the Fourier transform to derivative prices immediately through the characteristic function:

$$C_T(k) = \frac{e^{-\alpha k}\int_{-\infty}^{\infty} e^{-iuk}\psi(u) du}{2\pi} = \frac{e^{-\alpha k}\int_0^{\infty} \mathrm{Re}\left(e^{-iuk}\psi(u)\right) du}{\pi}$$

$$\psi(u) = \int_{-\infty}^{\infty} e^{iuk} c(k) dk = \frac{e^{-rT} g_T(u-(\alpha+1)i)}{\alpha^2 + \alpha - u^2 + i(2\alpha+1)u}$$

$$c(k) = e^{\alpha k} C_T(k), \quad \alpha > 0$$

where $\psi(u)$ is expressed in reference to the characteristic function g_T.

The project follows the Carr-Madan route, using the Lewis'approach [29] specifically. The main idea is to represent the payoff function $\omega(S_T)$ in Fourier space:

$$\hat{w}(z) = -\frac{K^{iz+1}}{z^2 - iz}$$

with z being a complex-valued number. Based on the general payoff transform, the inverse transformation is

$$w(x) = \frac{1}{2\pi}\int_{iz_i-\infty}^{iz_i+\infty} e^{-izx}\hat{w}(z) dz$$

Assuming there exists a well-defined characteristic function $g_T(u)$ for arbitrary price dynamics and a transformed payoff $\hat{w}(z)$, the option value can be obtained:

$$\begin{aligned} V\left(S_0, K, T\right) &= \mathbb{E}^{\mathbb{Q}}[w(x)] e^{-rT} \\ &= \frac{\mathbb{E}^{\mathbb{Q}}\left[\int_{iz_i-\infty}^{iz_i+\infty} e^{-izx}\hat{w}(z) dz\right] e^{-rT}}{2\pi} \\ &= \frac{\int_{iz_i-\infty}^{iz_i+\infty} \mathbb{E}^{\mathbb{Q}}\left[e^{-izx}\right]\hat{w}(z) dz e^{-rT}}{2\pi} \\ &= \frac{\int_{iz_i-\infty}^{iz_i+\infty} g_T(-z)\hat{w}(z) dz e^{-rT}}{2\pi} \end{aligned}$$

With the transformation, the European call price is given by

$$C\left(S_0, K, T\right) = S_0 - \frac{K e^{-rT}}{2\pi}\int_{iz_i+\infty}^{iz_i-\infty} e^{-izk}\phi_T(-z)\frac{dz}{z^2 - iz}$$

with $k = \log \frac{S_0}{K} + rT$. If we could move the contour to $z_i \in (0,1)$ for integration, then by choosing $z_i = \frac{1}{2}$, we obtain a symmetric path of integration. Finally, we can change the variable $z = u + \frac{i}{2}$ and obtain these alternative formulas:

$$C\left(S_0, K, T\right) = S_0 - \frac{\sqrt{SK}e^{-rT}}{2\pi} \int_0^\infty \mathrm{Re}\left[e^{iuk} g_T\left(u - \frac{i}{2}\right)\right] \frac{du}{u^2 + \frac{1}{4}}$$

In particular, in our implementation, we will evaluate the price with the last formula for pricing.

2.2.2 Characteristic function of rHeston

Having obtained the formula for the option price via the characteristic function, we now turn to derive the characteristic function of rHeston. Our aim is to derive a rough counterpart of the Heston model. For Heston, the closed formula for the characteristic function is obtained using the property of the Markovianity and the time-homogeneity [27]. With the application of Ito's formula to the following function:

$$L\left(t, V_t, \alpha\right) = \mathbb{E}\left[e^{i\alpha \log(S_T)} \mid \mathcal{F}_t\right], \mathcal{F}_t = \sigma\left(W_s, B_s; s \leq t\right), a \in \mathbb{R},$$

then since the process L is a martingale, the Feynman-Kac partial differential equation for L can be derived:

$$-\partial_t L(t, \alpha, S, V) = \left(\gamma(\theta - V)\partial_v + \frac{1}{2}(\gamma\nu)^2 V \partial_{vv}^2 + \frac{1}{2} S^2 V \partial_{ss}^2 + \rho\nu\gamma S V \partial_{sv}^2\right) L(t, \alpha, S, V)$$

with boundary condition $L(T, \alpha, S, V) = e^{\log(S) i\alpha}$. Here the characteristic function of the log-price $X_t = \log(\frac{S_t}{S_0})$ follows

$$\mathbb{E}\left[e^{X_t i\alpha}\right] = \exp\left(h(\alpha, t)V_0 + g(\alpha, t)\right)$$

where h is the solution to the Riccati equation:

$$\partial_t h = \frac{-\alpha^2 - i\alpha}{2} + \gamma(i\alpha\rho\nu - 1)h(\alpha, s) + \frac{(\gamma\nu)^2 h^2(\alpha, s)}{2}$$

$$h(\alpha, 0) = 0$$

as well as

$$g(\alpha, t) = \theta\gamma \int_0^t h(\alpha, s) ds.$$

Then the characteristic function of X_t can be expressed in the closed form, which is derived from solving the Riccati equation.

However, rHeston, on the other hand, is not Markovian, and its variance is not semimartingale. As a result, it appears impossible to adapt the standard Heston model to our framework.

Thus, we need to turn to an alternative approach. Following the method suggested in [27], microstructural models known as Hawkes processes are used. These models can accurately mimic the stylized facts seen in high frequency markets, and over time, they exhibit a rough behaviour. In summary, an appropriate series of Hawkes processes can converge to rHeston, and their characteristic functions, particularly, resemble the one of rHeston in the limit. The key finding is the unexpected similarity in structure between the characteristic function of the log-price in rHestons and that found in the Heston. The Riccati equation is substituted by a fractional Riccati equation, in which a fractional derivative is used.

Mathematically, assume rHeston with a correlation $\rho \in (0, \frac{-1}{\sqrt{2}})$ between the two BM. We have

$$L(\alpha, t) = \exp\left(g_1(\alpha, t) + V_0 g_2(\alpha, t)\right)$$

where

$$g_1(\alpha, t) = \theta\gamma \int_0^t h(\alpha, s) ds, \qquad g_2(\alpha, t) = I^{1-\alpha} h(\alpha, t), \quad t \geq 0$$

and $h(\alpha, \cdot)$ solves the following fractional Riccati equation:

$$D^\alpha h(\alpha, t) = \frac{-\alpha^2 - i\alpha}{2} + h(\alpha, s)\gamma(i\alpha\rho\nu - 1) + \frac{(\gamma\nu)^2 h^2(\alpha, s)}{2}$$

$$I^{1-\alpha}h(\alpha,0)=0$$

with D^α and $I^{1-\alpha}$ the fractional derivative and integral operators. The fractional integral of a function f with order $r\in(0,1)$:

$$I^r f(t)=\frac{\int_0^t(t-s)^{r-1}f(s)ds}{\Gamma(r)}$$

and the fractional derivative of order $r\in(0,1)$:

$$D^r f(t)=\frac{\frac{d}{dt}\int_0^t(t-s)^{-r}f(s)ds}{\Gamma(1-r)}$$

no matter when they exist.

Note that if $\alpha=1$, we get the same result as that of Heston. Nonetheless, it should be highlighted that for $\alpha<1$, the solutions to such Riccati equations are not exact. Thus, we have to calculate it numerically.

2.2.3 Numerical scheme to solve the fractional Riccati equation

We can rewrite the above equations as:

$$D^z h(z,t)=F(zh(z,t)),\quad I^{1-z}h(z,0)=0$$

where

$$F(z,x)=\frac{-z^2-iz}{2}+\gamma(iz\rho\nu-1)x+\frac{(\gamma\nu)^2x^2}{2}$$

These equations imply the following Volterra equation:

$$h(z,t)=\frac{\int_0^t(t-s)^{z-1}F(z,h(z,s))ds}{\Gamma(z)}$$

which has the property mentioned in [30] then we can follow the pseudo-algorithm described in [30].

We write $f(z,t)=F(z,h(z,t))$. With a common time grid $(t_k)_{k\in\mathbb{R}}$, we estimate

$$h\left(z,t_{k+1}\right)=\frac{\int_0^{t_{k+1}}\left(t_{k+1}-s\right)^{z-1}f(z,s)ds}{\Gamma(z)}$$

with

$$\frac{\int_0^{t_{k+1}}\left(t_{k+1}-s\right)^{z-1}\hat{f}(z,s)ds}{\Gamma(z)}$$

where

$$\hat{f}(z,t)=\frac{t_{j+1}-t}{t_{j+1}-t_j}\hat{f}\left(z,t_j\right)+\frac{t-t_j}{t_{j+1}-t_j}\hat{f}\left(z,t_{j+1}\right),t\in\left[t_j,t_{j+1}\right),0\le j\le k,$$

which is a trapezoidal discretisation of the fractional integral. Then, the numerical scheme can be written as

$$\hat{h}\left(z,t_{k+1}\right)=\sum_{0\le j\le k}z_{jk+1}F\left(z,\hat{h}\left(z,t_j\right)\right)+z_{k+1,k+1}F\left(z,\hat{h}\left(z,t_{k+1}\right)\right)$$

with

$$z_{0,k+1}=\frac{\Delta^z}{\Gamma(z+2)}\left(k^{z+1}-(k-z)(k+1)^z\right)$$

$$z_{j,k+1}=\frac{\Delta^z}{\Gamma(z+2)}\left(\left(k-j+2\right)^{z+1}-2(k-j+1)^{z+1}\right),1\le j\le k$$

$$z_{k+1,k+1}=\frac{\Delta^z}{\Gamma(z+2)}$$

To start with, we need to give an initial estimation of $\hat{h}\left(z,t_{k+1}\right)$ in order to plug it in the trapezoidal formula. This predictor is denoted by $\hat{h}^P\left(z,t_{k+1}\right)$:

$$\hat{h}^P\left(z,t_{k+1}\right)=\frac{1}{\Gamma(z)}\int_0^t\left(t-s\right)^{z-1}\tilde{f}(z,s)ds$$

with

$$\tilde{f}(z,t) = \hat{f}(z,t_j), t \in [t_j, t_{j+1}), 0 \leq j \leq k$$

So,

$$\hat{h}^P(z,t_{k+1}) = \sum_{0\leq j\leq k} b_{j,k+1} F\left(z, \hat{h}(z,t_j)\right)$$

where

$$b_{j,k+1} = \frac{\Delta^z}{\Gamma(z+1)}\left((k-j+1)^z - (k-j)^z\right), 0 \leq j \leq k$$

To sum up, the numerical scheme can written as:

$$\hat{h}(z,t_{k+1}) = \sum_{0\leq j\leq k} z_{j,k+1} F\left(z, \hat{h}(z,t_j)\right) + z_{k+1,k+1} F\left(z, \hat{h}^P(z,t_{k+1})\right), \hat{h}(z,0) = 0.$$

2.3 Implementation

In this section, we generate the features (volatilities across a combination of strikes and maturities) and the corresponding labels (rHeston parameters $P = \{\rho, V_0, \kappa, H, \theta, \nu\}$) to build the training dataset for later use of deep calibration.

The main idea of data generation is that we set values of the six parameters of rHeston and calculate the corresponding implied volatility surface. For later deep learning training, one set of six parameters is one label and one implied volatility surface is one feature. Following the approach in [2], we set bounds for individual parameter as noted in Table 2.1, and create each parameter at random following a uniform distribution. The bounds are set in accordance with the results using the real SPX data [31].

parameter	ρ	V_0	κ	H	θ	ν
lower bound	-0.7071	0.0262	0.1206	0.1286	0.0721	0.2720
upper bound	-0.5940	0.0778	0.5041	0.1766	0.1499	0.3748

Table 2.1: rHeston parameters with narrower range

As discussed in the section of training NNs, we further generate another data set following the same procedure but with different bounds of each parameter. Specifically, we broaden the range of each parameter by 40% stated in Table 2.2. But, it should be noted that the global lower bound of ρ and H should be kept.

parameter	ρ	V_0	κ	H	θ	ν
lower bound	-0.7071	0.0157	0.0724	0	0.0433	0.1632
upper bound	-0.3564	0.1089	0.7057	0.4472	0.2099	0.5247

Table 2.2: rHeston parameters with broader range

For the out-of-sample test in the section of deep calibration, I build up another data set (Table 2.3) using the bounds with approximately 13% and 40% increase to the upper bounds of the parameters stated in Table 2.2 as its lower and upper bounds respectively.

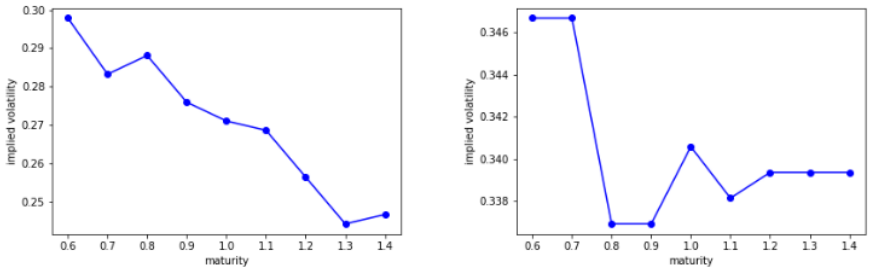
parameter	ρ	V_0	κ	H	θ	ν
lower bound	-0.31	0.12	0.8	0.47	0.24	0.6
upper bound	-0.21	0.15	1.0	0.5	0.29	0.74

Table 2.3: rHeston parameters for the our-of-sample test

Next, we use rHeston pricer we built and compute the implied volatility surface. We assume that the interest rate is zero and without divend, and the initial asset price is 1. Then, we use the most liquid strikes and maturities following [31], that is strikes $K = [0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3, 1.4]$ and maturities $T = [0.6, 0.9, 1.2, 1.5, 1.8, 2.0]$. Thus, the dimensions of the volatility surface is 9×6. Notice that strikes set here are symmetric to the centre of the ATM strike which is the spot price here.

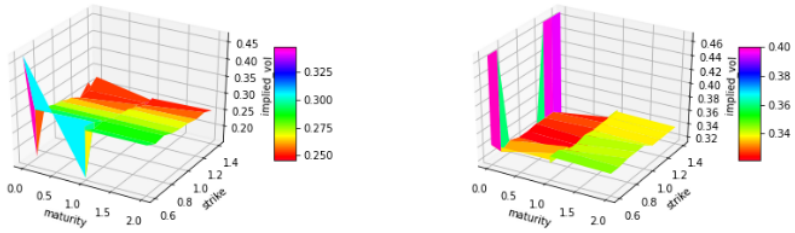
Finally, we generate a data set including 10,000 sets of features and labels. Specifically, there are a collection of 54 ($= 9 \times 6$) volatilities as the features and the labels are 6 parameters of rheston. Then following the same split ratio of the training and the test data[2], we divide the whole data set into a training set with 8,500 points and a testing set with 1,500 points.

As seen from the Figure 2.2, the data generated does not guarantee a specific shape of the volatility smile. The data also doesn't guarantee shape of the volatility surface shown in the Figure 2.3. Here I use the data set generated using the parameters with a broader range for analysis.



(a) 300th observation (b) 7014th observation

Figure 2.2: Volatility smile with maturity $T = 1.8$



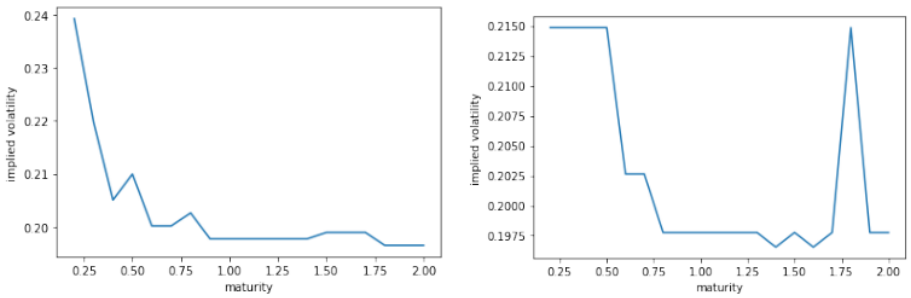
(a) 300th observation (b) 7014th observation

Figure 2.3: Volatility surface

We consider the following parameters for a further numerical example of rHeston pricing versus Heston pricing:

$$\rho = -0.5, V_0 = 0.04, \kappa = 2, H = 0.12, \theta = 0.04, \nu = 0.05.$$

We compare in the Figure 2.4 the ATM volatility skew produced by the Heston and its rough counterpart.



(a) rHeston (b) Heston

Figure 2.4: ATM volatility skew

Chapter 3

Calibration via NNs

On the occasion that the numerical computation for calibration procedures are slow, the fairly long calibration time may restrict applicability of the model for industrial production, regardless of the desired attributes of the model. This is particularly true for the generation of rough volatility models, of which the rough fractional Brownian motion in the volatility process renders it impossible to use the conventional Markovian pricing models [32]. For the family of rough volatility models, whose tremendous modelling benefits have been examined and dicussed in an increasing number of scholarly works (see, for instance, [33, 34, 35]), the calibration bottleneck have so far been a key limiting issue.

As a result, it is logical to develop deep learning approaches to speed up the calibration process. We would exclusively concentrate on the design of the FNN in this work, which is supported by evidence indicating FNNs outperform CNN in [2].

3.1 Problem setting

After using rHeston to construct the data set of volatilities and prices, we now proceed to calibrate rHeston with the goal of determining the optimum solution of parameters $s = \{\rho, V_0, \kappa, H, \theta, \nu\}$ that will best fit the data we generated. This technically is an optimisation problem with non-linear restrictions, which is to minimise the difference between the volatilities calculated by rHeston $vol(s; K, T)$ and the inferred volatilities of the market $vol^{market}(s; K, T)$(i.e., the volatilities we generated here). It's expected that the loss L with regards to the parameter set should be reduced on the occasion of tuning the parameters s to the actual market data:

$$L(s; vol^{market}) := f(vol(s; K, T), vol^{market})$$

where f is the loss function we set. The solution of the above equation s^* can be written as:

$$s^* := \arg\min_s L(s; vol^{market}).$$

We follow the direct approach of deep calibration for the volatility model in [20] where the map from the implied volatilities to the parameters of the pricing model is approximated by a FNN. This direct approach [20] also proves to outperform the two-step deep calibration procedure [32].

3.2 Input data preprocessing

3.2.1 Scaling

It is generally recognised that large scale variances in the input data can make training challenging and result in instability [2]. If the values of the features in the machine learning algorithms are more similar to one another, it is more likely that the algorithm will be trained successfully and quickly as opposed to a data set where the data points or feature values are very dissimilar from one another, which would need more time to grasp the data and result in poorer accuracy. As a result, if there exist data points far from each other, scaling is a strategy for bringing them closer together. Larger discrepancies between input data points increases the uncertainty in the model's outputs.

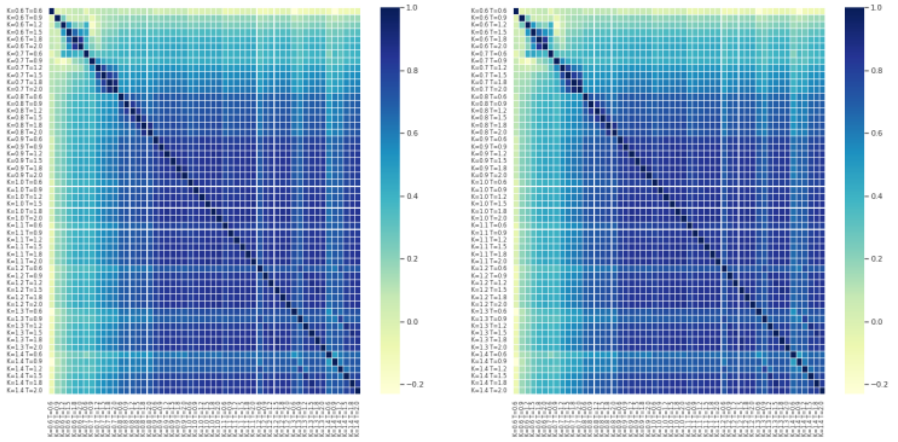
As a result, standardising the input data to numerically improve NN training is common practise. Here I use two approaches for scaling:

- Approach 1: Scale the data in a range from 0 to 1
- Approach 2: Scale the data to have a mean of 0 and a variance of 1

3.2.2 Whitening

Before designing the NN, it is critical to examine the scaled features and their intra-correlations. The correlation matrix of the input features (i.e.volatilities) of rHeston using the narrower range of parameters is displayed in the Figure 3.1 which shows the necessity of scaling:

- The volatilities are highly correlated with each other on the inferred surface.
- Relative placements of the data points on the surface influence the intre-coorelation.
- The neighbouring volatilities have larger correlations than data points far away.



(a) With the scaling Approach 1

(b) With the scaling Approach 2

Figure 3.1: Correlation matrix of the volatility surface before whitening

As a result, a whitening method is considered. The primary goal of whitening is to eliminate correlation among raw data and hence make the input features less redundant [36]. Specifically, there are two goals we are trying to accomplish with whitening:

- Make the features less correlated with one another
- Give all of the features the same variance

Technically, there are two types of whitening: PCA (principal component analysis) whitening and ZCA (zero-phase component analysis). They both can de-correlate the features and transform the correlation matrix to the identity matrix. The difference between them is that ZCA whitening keep the whitened data as close as possible to the original data (in the least squares sense). I use the ZCA whitening here as I want to keep the whitened features as close as to the original features.

ZCA whitening has the following steps [37]:

- Step 1: Zero-center the data set

  X is the initial input data. X_w is data after whitening. Then, $X_c = X - mean(X)$ is the zero-centred data.

- Step2: Calculate the covariance matrix of the zero-centered data set

  Let Σ_X be the covariance matrix of X_c. Then, Σ_X can be written as:

  $$\Sigma_X = \frac{1}{n} X_c {X_c}^T = U \Lambda U^T,$$

  where n is data size, U is the matrix of the normalised eigenvectors of X, and Λ is the diagonal matrix consisting of all eigenvalues. The second agreement comes from singular value decomposition.

- Step 3: Calculate the eigenvalue and the eigenvector

- Step 4: Apply the Eigenvalues and Eigenvectors to the data for whitening transform

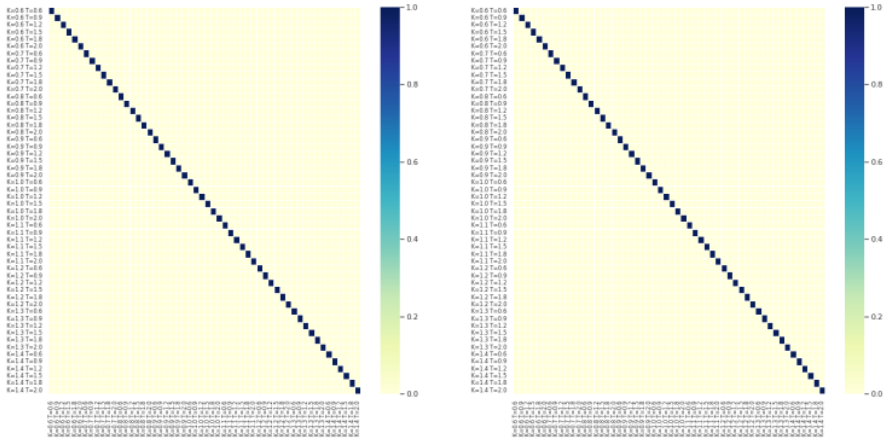
  Representation of X_w using ZCA whitening is:

  $$X_w = U \Lambda^{-\frac{1}{2}} U^T X_c.$$

  Then, the covariance matrix of Σ_w can be proved to be identity:

$$\begin{aligned}
\Sigma_w &= \frac{1}{n} X_w {X_w}^T \\
&= \frac{1}{n} U \Lambda^{-\frac{1}{2}} U^T X_c {X_c}^T U \Lambda^{-\frac{1}{2}} U^T \\
&= U \Lambda^{-\frac{1}{2}} U^T (\frac{1}{n} X_c {X_c}^T) U \Lambda^{-\frac{1}{2}} U^T \\
&= U \Lambda^{-\frac{1}{2}} U^T (U \Lambda U^T) U \Lambda^{-\frac{1}{2}} U^T \text{using singular value decomposition} \\
&= U \Lambda^{-\frac{1}{2}} (U^T U) \Lambda (U^T U) \Lambda^{-\frac{1}{2}} U^T \text{using the unitary property of } U \\
&= I
\end{aligned}$$

The Figure 3.2 displays the correlation matrix after the whitening transformation. As we can see, the intra-correlation has been removed and the whitening is indeed effective. However, it should be noted that whitening preprocessing technique does not necessarily improve the performance of NNs, as demonstrated by the Heston model [2].



(a) With the scaling Approach 1

(b) With the scaling Approach 2

Figure 3.2: Correlation matrix of the volatility surface after whitening

3.3 FNN architecture

In the next section, we will show how to build NNs that can learn the procedure from the implied volatilities straight to the model parameters. Only the FNN is examined here, via which information flows in one direction, from the input to the intermediate calculations and ultimately to the output, as shown in the Figure 3.3.

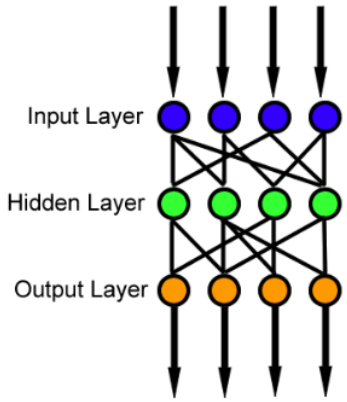


Figure 3.3: Illustration of the FNN[1]

Following the pre-processing procedure, the input features, say, 6×9 volatilities, are fed into a basic FNN that is completely connected. The main idea is to begin by calibrating the model with a simple model then gradually modifying the model until the model is reliable enough. We discovered that the simplest model (as shown in Figure 3.4) with one hidden layer and a decreasing number of neurons is adequate to get remarkable results. This architecture has a total of 372 trainable parameters.

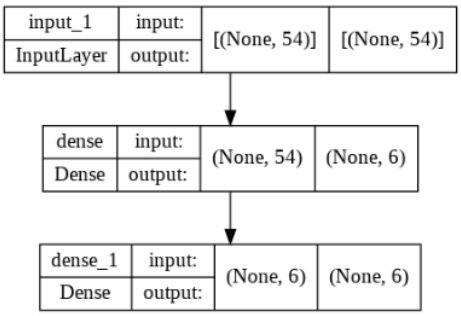


Figure 3.4: Architecture of the fitted FNN

ELU is selected as the activation function for both the hidden and output layers. The Figure 3.5 depicts three popular activation functions. In general, the ReLU function is frequently employed, in spite of the fact that it has the drawback of constantly ignoring negative values from the preceding layer, which is known as the dying ReLU problem. To address the constraint, the eLU function is implemented. The eLU and SeLU functions are put into practice in our work, however they underperform.

The output layer is made up of the six rHeston parameters $s = \{\rho, V_0, \kappa, H, \theta, \nu\}$ that need to be calibrated.

The mean squared logarithmic function is employed as the loss function. For one reason, after comparing its performance to other usual loss functions, it appears to fit the overall architecture

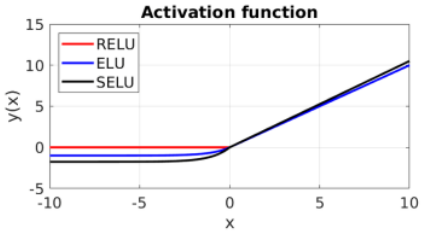


Figure 3.5: Popular activation functions

best. For another reason, it has an interpretation useful, say, a ratio of the actual and predicted values:

$$Loss(y, \hat{y}) = \frac{1}{N}\sum_{i=0}^{N}(log(y_i + 1) - log(\hat{y}_i + 1))^2$$
$$= \frac{1}{N}\sum_{i=0}^{N}(log(\frac{y_i + 1}{\hat{y}_i + 1}))^2$$

where y is the actual data, $\hat{y}$ is prediction, N is data size. This loss function can be used to compare errors.

TensorFlow/Keras is used for NN training. As a solver, the adam optimiser is utilised, which is a stochastic gradient descent approach based on adaptive estimate of first-order and second-order moments [38]. Mini-baches are used for training. The separation between the validation and training sets is 0.2.

3.4 Results

3.4.1 Training history

To start, we employ the preprocessed data created with a smaller set of parameters. Using Scaling Approach 1 or 2, the training history appears nearly identical. The Figure 3.6 depicts the training history for FNN using the Scaling Approach 1; we can infer that no overfitting happened due to the reason that training accuracy grows and loss deminishes for both training and validation data during training. Overfitting problem would have resulted in the NN's performance improving for training data but decreasing for validation data: two lines in the plot would have drifted from each other as the epochs passed. Plus, the FNN here is trained fairly quickly till it reaches its convergence. It obtains a far higher level of accuracy of more than 99% no matter which scaling approach is utilised.

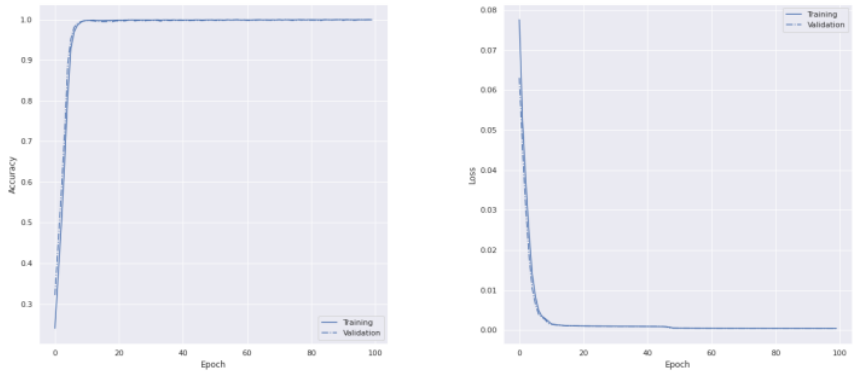
Another factor to consider is that the training history in our test is quite smooth. We look into the matter further in the following two aspects:

- FNN design

  On one hand, large mini-batch size, no noisy regularisation (such as drop out), and a small number of epochs are plausible causes from the degree of FNN design. We modify our architecture accordingly, such as experimenting with mini-batch sizes of 64, 128, and 256; adding two dropout layers with rate of 0.6; and trying with 50, 100, 150, and 200 epochs. However, none of the changes make the training any noisier, and the training history stays smooth. A smooth training history is completely acceptable if the FNN works in the intended application (in this case, predicting performance).

- data

  On the other hand, low-noise data can produce a smooth training history. This is applicable to our case since the data is created artificially with artificially generated boundaries for individual parameter. In fact, market data could be noisy. It gives hints for further research to utilise real market data.



(a) Accuracy during training

(b) Loss during training

Figure 3.6: FNN traing history with Scaling Approach 1

3.4.2 Prediction errors

The deviations between the predicted parameters of rHeston and the label for the test data with the Scaling Approach 1 and 2 are shown in the Figure 3.7 and 3.8 respectively. The mean squared logarithmic function is still used to evaluate the errors in this case.

There is strong evidence that the fitting results do not vary a lot by using different scaling approaches. It should be noticed that, while most of the predictions are very accurate, there are a few data points known as outliers that stand out from the rest of the data. This might be explained by the issue in identifying model parameters in the case of carrying out the volatility model calibration using NNs, as indicated by [20]. To be more specific, there are Heston parameterisations that differ significantly in terms of Heston parameter values yet correspond to relatively comparable implied volatility surfaces.

The Table 3.1 shows the average prediction errors for each parameter of rHeston with the test data. According to the nature of the loss function, it would be possible to allow us to carry out comparisons across the values of the errors. As can be observed, the prediction errors for parameter ν, V_0, θ, H are much lower. It is worth noting that the prediction for the parameter κ is relatively poorer than the others.

In addition, we show the prediction errors using the training data in the Table 3.1. The fact that the NN's performance is good for not only the training data but also the test data, shows that the training is successful in allowing the model to function effectively.

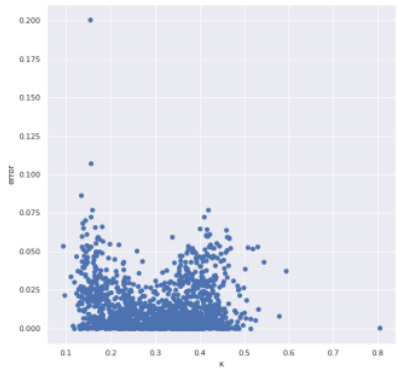
parameter	κ	ν	ρ	V_0	θ	H
training data	0.0014	0.0003	1e-07	3e-05	0.0002	0.0007
test data	0.0014	0.0003	3e-08	3e-05	0.0002	0.0007

Table 3.1: Prediction error mean for each parameter

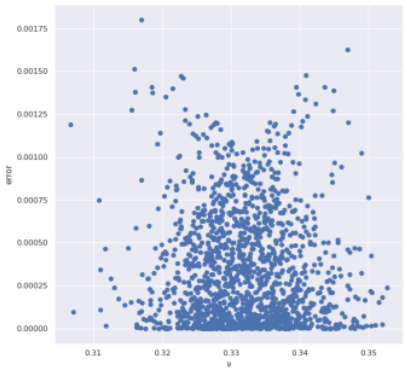
3.4.3 Robustness check

Furthermore, the out-of-sample test is crucial due to the reason that we suspect that the parameter range is too narrow and that the predictive performance of the fitted FNN is insufficiently robust. As a result, we produce a data set with parameters that are outside of the range and want to evaluate if the fitted FNN is good enough to predict with this new data set.

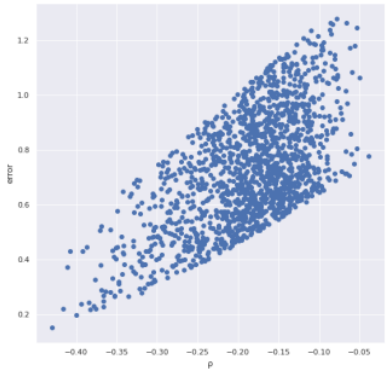
The prediction errors with test data are shown in the Figure 3.9 and 3.10 respectively. The findings are consistent no matter which scaling approach is used.



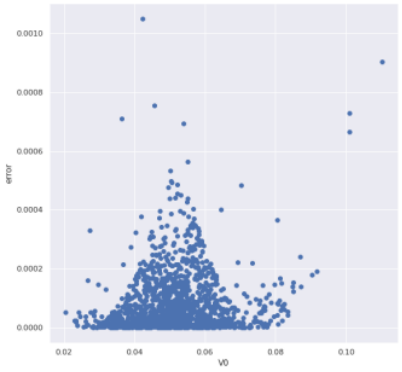
(a) Prediction error for κ



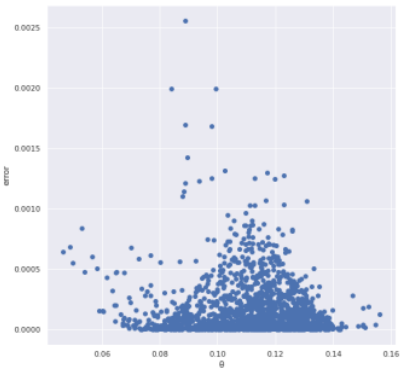
(b) Prediction error for ν



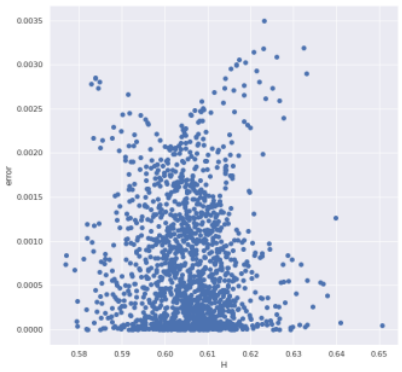
(c) Prediction error for ρ



(d) Prediction error for V_0

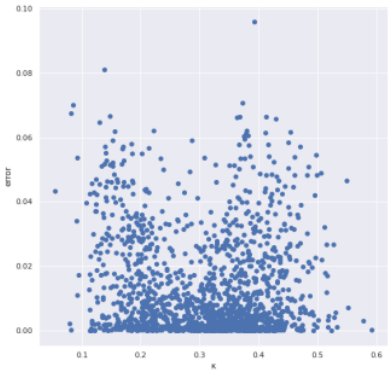


(e) Prediction error for θ

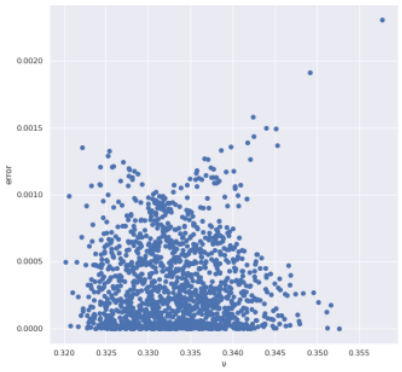


(f) Prediction error for H

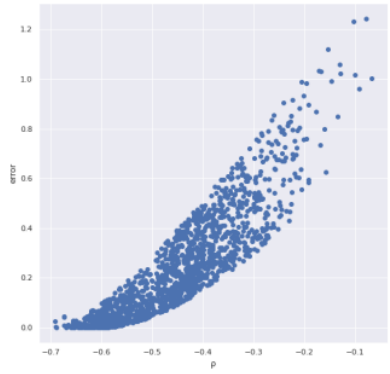
Figure 3.7: Prediction errors of each parameter with the test data (Scaling Approach 1)



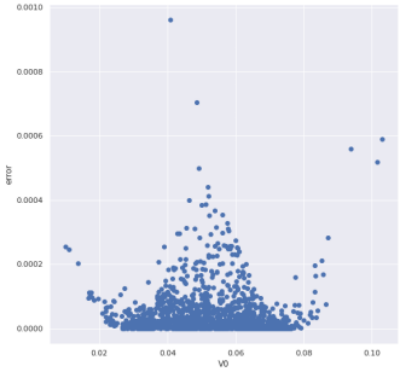
(a) Prediction error for κ



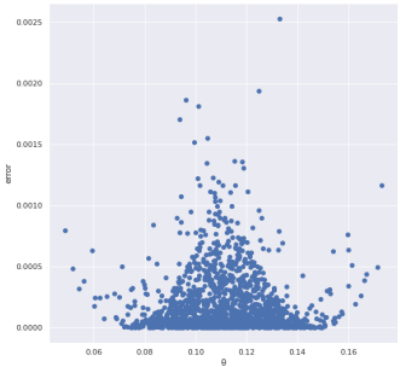
(b) Prediction error for ν



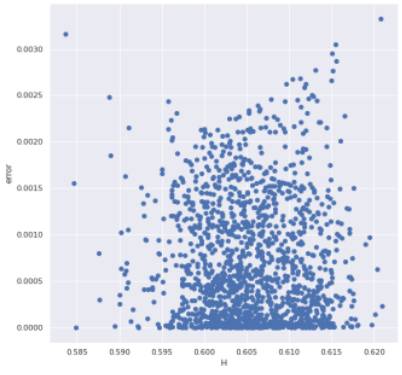
(c) Prediction error for ρ



(d) Prediction error for V_0



(e) Prediction error for θ



(f) Prediction error for H

Figure 3.8: Prediction errors of each parameter with the test data (Scaling Approach 2)

The following results are with the first scaling approach. The average prediction error is shown in the Table 3.2. The forecasts for the two parameters κ and ρ are clearly unsatisfactory, with average errors of 0.1279 and 0.5593, respectively. Overall, the forecast for the other parameters appears to be accurate. As a result of this out-of-sample test, the FNN trained with parameters with a tighter range fails to predict all six rHeston parameters well.

parameter	κ	ν	ρ	V_0	θ	H
out-of-sample data	0.1279	0.0004	0.5593	0.0003	0.0009	0.0008

Table 3.2: Prediction error mean for each parameter with the out-of-sample data

3.4.4 Implementation with the data set produced using the parameters with the wider range

Instead, as described in Section 2, we train the FNN using the data set produced with a broader range of parameters.

As seen in the Figure 3.11, the training history is still relatively smooth, the learning procedure is quick to achieve stability after around 20 epochs, and the model eventually reaches an accuracy of more than 92% regardless of the Scaling Approach used.

The problem with model parameter identifiability persists, as can be seen in the Figure 3.12, where some outliers appear to be significantly separated from the rest of the data points. The average prediction error of the fitted FNN for the training set, validation set, and test data using Scaling Approach 1 is once again presented in the Table 3.3. It illustrates that

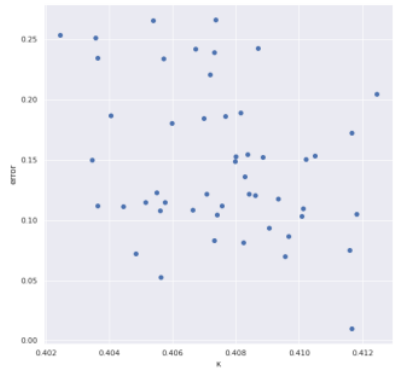
1. As expected, the prediction error of FNN increases compared with that using the parameters with a narrower range, which is applicable to all parameters with the similar extent.

2. The out-of-sample test indicates that the predictive performance for the parameter ρ improves dramatically.

We proceed with FNN trained using the data set created with the parameters with the wider range for the subsequent interpretability study due to the reason that it generates more robust predictions not only within but also beyond the data.

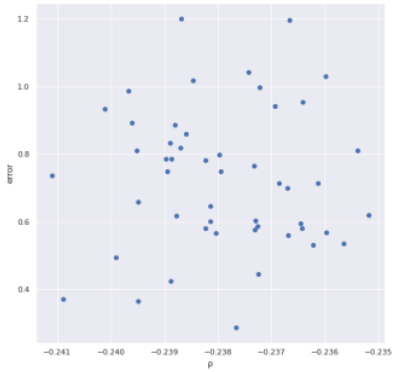
parameter	κ	ν	ρ	V_0	θ	H
training data	0.0113	0.005	0.0	0.0003	0.0008	0.005
test data	0.0111	0.005	0.0	0.0003	0.0008	0.0055
out-of-sample data	0.1167	0.0060	0.165	0.0012	0.0028	0.0051

Table 3.3: Prediction error mean for each parameter with a broader data set

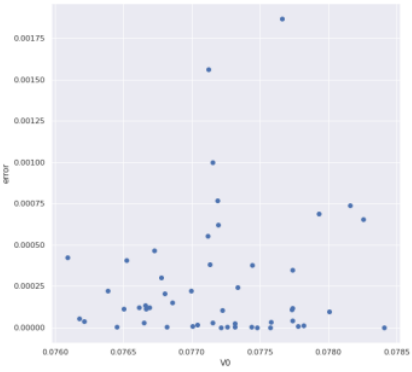


(a) Prediction error for κ

(b) Prediction error for ν

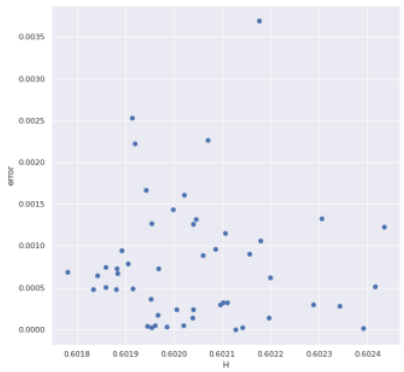


(c) Prediction error for ρ



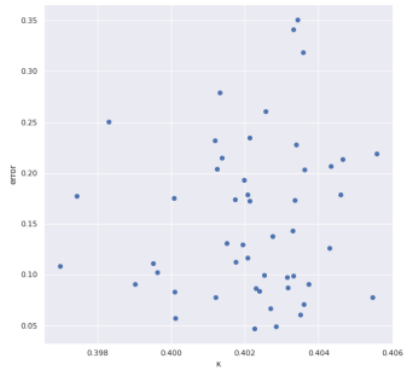
(d) Prediction error for V_0

(e) Prediction error for θ

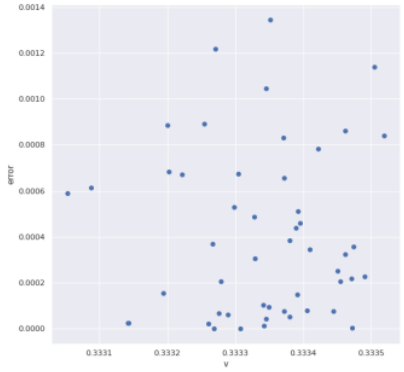


(f) Prediction error for H

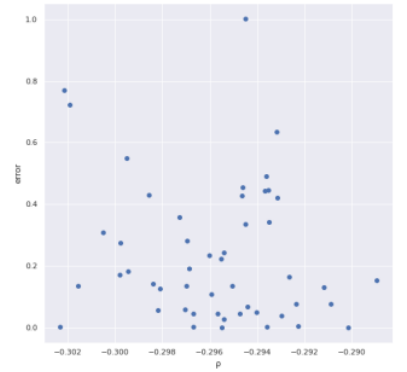
Figure 3.9: Prediction errors of each parameter with the out-of-sample data (Scaling Approach 1)



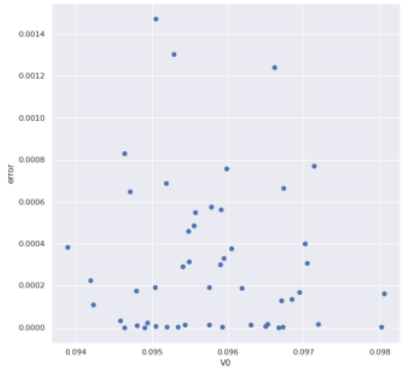
(a) Prediction error for κ



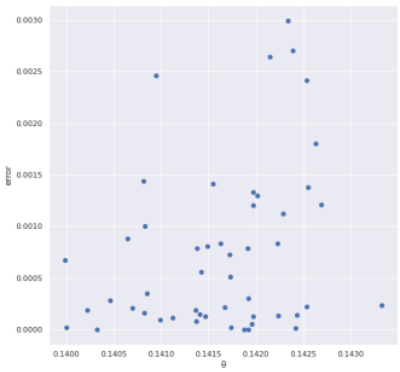
(b) Prediction error for ν



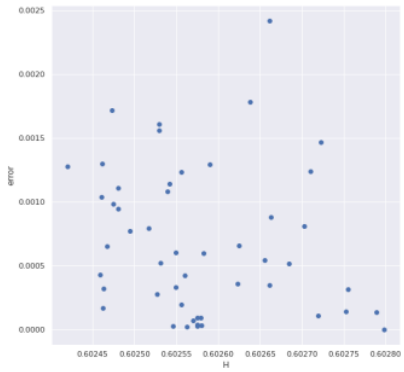
(c) Prediction error for ρ



(d) Prediction error for V_0

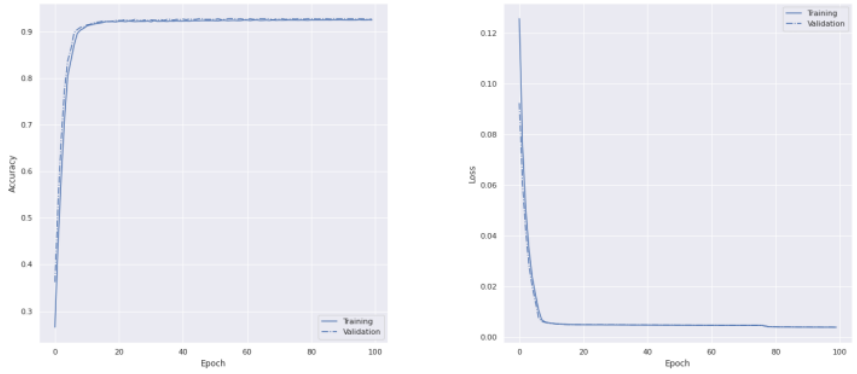


(e) Prediction error for θ



(f) Prediction error for H

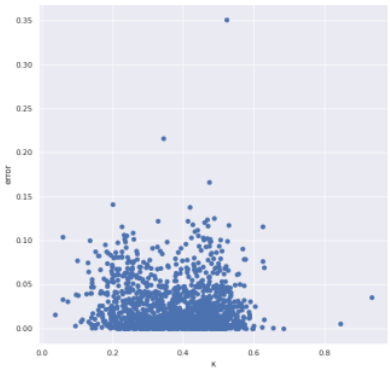
Figure 3.10: Prediction errors of each parameter with the out-of-sample data (Scaling Approach 2)



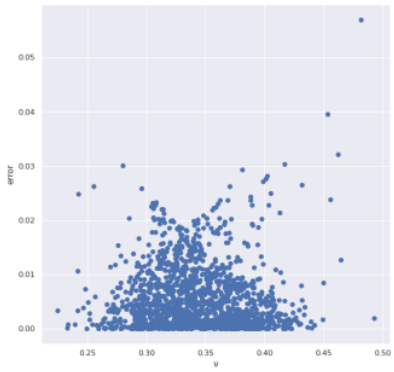
(a) Accuracy during training

(b) Loss during training

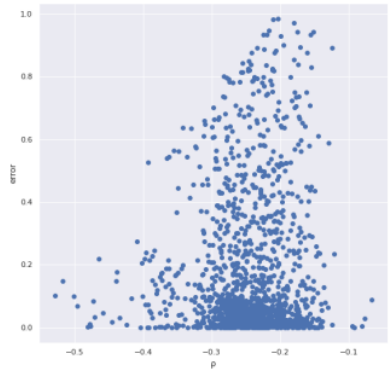
Figure 3.11: FNN traing history with a broader data set



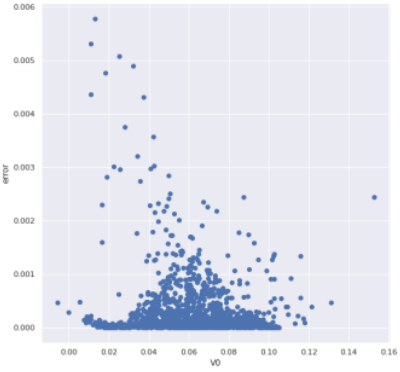
(a) Prediction error for κ



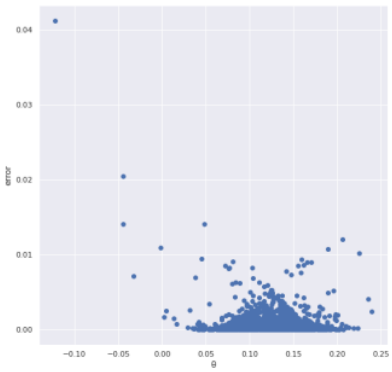
(b) Prediction error for ν



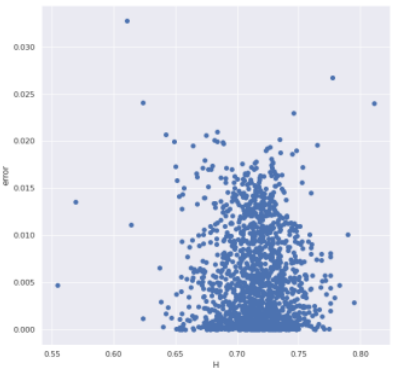
(c) Prediction error for ρ



(d) Prediction error for V_0



(e) Prediction error for θ



(f) Prediction error for H

Figure 3.12: Prediction errors of each parameter with a broader data set

Chapter 4

Interpretability of NN calibration

The black-box feature as seen in the Figure 4.1 of machine learning, which refers to the fact that its models are sufficiently complicated to not be easily understood by humans, poses an issue for its reliability even while it can speed up calibration and attain satisfactory accuracy levels.

Figure 4.1: Black-box nature of machine learning[2]

The term "interpretability" lacks a precise definition. [4] offers a non-mathematical definition: "the degree to which a human can understand the cause of a decision". The more interpretable a machine learning model is, the simpler it is to understand why particular judgments or predictions have been made.

Interpretability is a vital component in machine learning models. We are interested not just in the model's findings, but also in how the results are derived from the inputs and how trustworthy the findings are. It is useful to investigate interpretability in order to debug the model and make educated decisions about how to enhance it. In particular, interpretability can be advantageous in two scenarios. On the one hand, if we have a thorough comprehension of our model, we may test whether the mapping from inputs to outputs corresponds to our intuitive understanding of the mode. Furthermore, if we lack model expertise, we may employ interpretability models to increase our comprehension of the model.

I work on interpretability of the calibration of rHeston to study which feature (volatility) affects the majority of the label (parameters of rheston).

4.1 Local interpretability

According to [12], we may classify interpretability models into two primary groups: global and local interpretability methods. Explaining each individual forecast is typically referred to as local interpretability.

f is the machine learning model to be interpreted and $\hat{f}$ is the prediction given by FNN. To understand the prediction $y = \hat{f}(x)$ with regards to input x, we introduce the simplified input x' of which each component is binary with 0 meaning that a feature value is equal to the average of the data set and 1 otherwise. Let function h_x such that $x = h_x(x')$.

Given input x, an interpretability model is a local model g on x' such that $g(z') \approx \hat{f}(h_x(z'))$ for $z' \approx x'$. Interpretability model is an additive g

$$g(z') = \phi_0 + \sum_{n=1}^{M} \phi_i z_i^{'}$$

where the model attributes the effect ϕ_i to each feature i.

4.1.1 LIME

The technique originates from [13] where the authors perturb the initial data, enter them into the black-box model, and then track the resulting outputs. The approach then according to the distance between the initial data points and the new ones, accommodate the weights of individual data point. In the end, it uses those sample weights to fit a surrogate model to the data set with variations. Finally, we work with the newly trained interpretablity model to explain each original data point.

Mathematically, the explanation for a data point x is the model g that, while keeping the model complexity $\Omega(g)$ low, minimises the loss $L(f, g, x)$ measuring how unfaithfully g approximates the model to be explained f in its proximity $\pi_{x'}$. The objective function is as below:

$$\arg\min_{g} L(f, g, \pi_{x'}) + \Omega(g).$$

4.1.2 DeepLIFT

DeepLIFT[14] reverse recoveries the contributions of each neuron to each attribute of the input, then it provides a method for breaking down how a NN generates the output from its input. It computes contribution scores by evaluating the activation of each neuron to its "reference activation" and calculating the difference. By optionally separating out positive and negative contributions, DeepLIFT can spot dependencies that other methods miss. One backward pass can effectively compute scores.

4.1.3 LRP

LRP is based on a backward propagation method that is applied to all layers of the model successively. In this case, the model output score indicates the initial relevance, which is decomposed into values for each neuron in the underlying layers. The decomposition is determined by rules that are specific to each layer and involve its weights and activations.

The suggested propagation process fulfils a conservation property, which states that the magnitude of any output remains constant while it is backpropagated via the network's lower-level layers. From the output to the input-layer neurons, each neuron reallocates the same quantity of data to the lower layer that it got from the higher layer. The approach applies to a wide range of data.

4.2 Global interpretability

A method for determining which input properties have the greatest overall impact on a model's output across all predictions is provided by global interpretability. The major distinction between local and global interpretability is that the global method provides a conclusion based on an average of all predictions. In other words, global interpretability is utilised to determine how much each input characteristic contributes to the final outcome, utilising all data rather than just one specific instance.

4.2.1 Shapley Values

The Shapley values are a concept from game theory in economics that aims to assign profits and losses resulting from group activity to internal members. The Shapley values can be read as follows intuitively. The input feature enters a group in a random sequence to play a game. Each feature contributes differently to the game's outcome. A feature's Shapley value quantifies how, on average, the outcome of the game changes on the occasion that the feature(s) joins the game. To be more precise, we need to figure out each feature's marginal contribution before averaging it across all potential coalitions in order to get the Shapley value. It should be highlighted, nonetheless, that the Shapley value does not reflect the change in label prediction on the occasion that the feature is excluded from the model. The game serves as the prediction task in the context of machine learning, the group members serve as the features/inputs, and the Shapley value serves as a measure of interpretability for the feature attribution. Similar to this, Shapley interpretations are also given as additional feature attributions.

Mathematically, the problem can be set up as: assuming a game with n players. For the cooperation of players G, $f(G)$ is the average sum of gain/cost members of G work together to generate. Shapley Values assigns gains/costs to collaborative players. The attribution to the member k is

$$\phi_k = \sum_{G \subseteq n \setminus \{k\}} \frac{|G|!(|N| - |G| - 1)!}{|N|!} [f(G \cup \{k\}) - f(G)],$$

which can be interpreted as :

$$\phi_k = \sum_{\text{alliance without } k} \frac{\text{marginal contribution when } k \text{ joins the alliance}}{\text{number of possible alliances excluding } k}.$$

Specifically, in the context of machine learning interpretability, the feature attribution is:

$$\phi_k(\hat{f}, x) = \sum_{z' \subseteq x' \setminus k} \frac{|z'|!\,(G - |z'| - 1)!}{G!} \left[\hat{f}\left(h_x\left(z' \cup i\right)\right) - \hat{f}\left(h_x\left(z'\right)\right) \right],$$

where $\hat{f}$ is the prediction model. This fits the additive feature attribution of the interpretability model g:

$$g(z') = \phi_0 + \sum_{n=1}^{M} \phi_i z_k^{'}$$

where the model attributes the effect ϕ_k to each feature k.

SHAP (Shapley Additive exPlanations) [39] establishes a single measure of feature attribution and various practical implementation approaches, such as Kernel SHAP, Tree SHAP, and Deep SHAP. Additionally, SHAP incorporates a variety of Shapley value-based global interpretation techniques.

Deep SHAP

Making use of both Shapley values and DeepLIFT, Deep SHAP [39] puts SHAP values calculated for single part of NN together for the network as a whole. By picking a subsample as background, Deep SHAP comes close to the conditional expectation of Shapley values. By analysing how the difference between the output and the reference output relates to how the inputs differ from their reference level, DeepLIFT [14] derives feature attributions. Deep SHAP then integrates the results from DeepLIFT over different background samples.

Kernel SHAP

Kernel SHAP combines LIME with Shapley values, constructing a regression linear model similar to LIME with the coefficients calculated using Shapley values.

Tree SHAP [40]

The tree-based machine learning models are the focus of this interpretability method. By first computing the best local interpretation and then expanding it to include feature interactions, it creates an understanding of a model's overall interpretability.

The Shapley values should satisfy the following properties [16]:

- Local accuracy: For specific instance, the prediction should be fairly attributed to the feature value.
- Missingness: Attribution of 0 is assigned to the missing feature.
- Consistency: Feature attribution should follow the direction of change of simplified input's contribution.

4.3 Other methods

This section introduces gradient-based attribution techniques that calculate the slope of the prediction with reference to the features, such as Saliency Maps, Gradient * Input, and Integrated Gradients. The gradient-based attribution indicates that if I raise the volatility value, the predicted parameter will grow (for positive gradient) or decrease (for negative gradient).

4.3.1 Saliency maps

Saliency Maps [41] as a back-propagation approach first calculates the partial derivative of the output with reference to the features, then returns the absolute value. As a result, the Saliency attributions are at almost all times positive.

4.3.2 Gradient * Input

Similarly, Gradients * Input [42] attempts to improve the attribution maps. It obtains the attribution through multiplying the attributions derived using Saliency maps by the input feature itself.

Mathematically, the attribution ϕ_i is:

$$\phi_i = x_i \times \frac{\partial f(x)}{\partial x_i}.$$

4.3.3 Integrated gradients

Integrated Gradients [43] is built up from the idea of computing the gradient of the prediction output with regards to the input features.

Mathematically, the attribution ϕ_i is:

$$\phi_i = (x_i - x_i^0) \times \int_{\alpha=0}^{1} \frac{\partial f(x^0 + \alpha \times (x - x^0))}{\partial x_i} \, d\alpha,$$

where x^0 is the baseline (default:zero).

4.4 Implementation

In this section, all the local surrogate interpretability models are implemented in TensorFlow 1.15.

4.4.1 LIME

In our scenario, we should approach the NN as a regressor and utilise an explanatory model to achieve a local approximation around each individual prediction (here a linear model is used). LIME in Python could be used to implement this. The Huber Regressor in particularly is selected for its tolerance to outliers.The Figure 4.2 depicts the prediction range and provides an overview of which features contribute the most and how. The orange components have a positive impact on the forecasted values, whereas the blue ones have a negative influence. The right feature-value table highlights each feature's input value with orange for positive and blue for negative. The Figure 4.3 looks at how each input entry impacts the model output by taking the absolute value of the contribution of each feature and calculating the average of 1500 predictions. I concentrate on the impact only in terms of unsigned values and disregard whether the influence is positive or negative by taking the absolute value of all attributions. Note that the subsequent interpretability models will use the same measurement technique. The light color in the heat map indicates low attribution values and the dark color for the high.

Take the 0th κ in the test set as an example. As seen in the Figure 4.2, the prediction ranges from -0.09 to 0.74, and with 0.58 being the 0th predicted value of FNN. It is worth noting that in Section 2 κ is generated from 0.0724 to 0.7057 following the uniform distribution, which implies that the FNN covers the whole range of the model parameter set. The feature influencing the prediction most is the volatility (T = 0.6, K = 0.6) deep in the money with short maturity The second is (T = 1.2, K = 0.7) still deep in the money, and almost all of them influence the predicted value positively. It should be highlighted that the feature values are the scaled and whitened input data with a broader range of parameters.

As shown in the Figure 4.3, the most influential volatilities are concentrated in the left upper corner (fist two rows and first three columns) deep in the money with relatively short maturity (T = 1.2, K = 0.7; T = 0.9, K = 0.6; T=1.2, K=0.6; T = 0.9, K= 0.7; T = 2.0, K = 0.6). This is completely distinct from the finds with the Heston model [2].

Similarly, I use scaled and whitened data with the narrower range of parameters to implement the LIME interpretability model. The findings are quite similar:

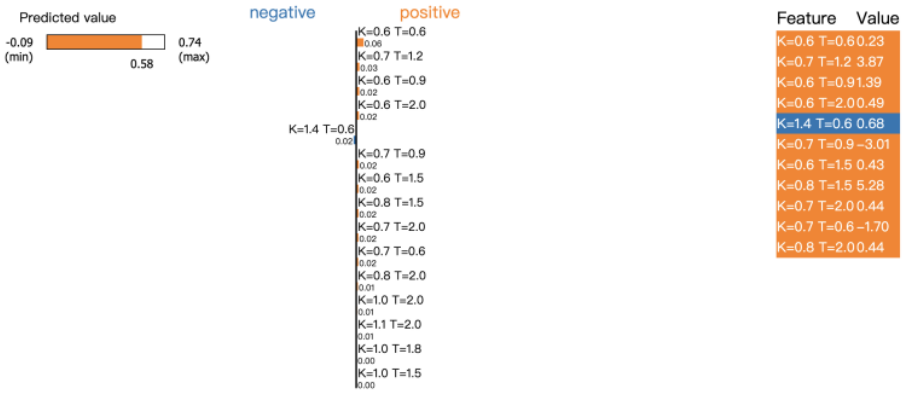
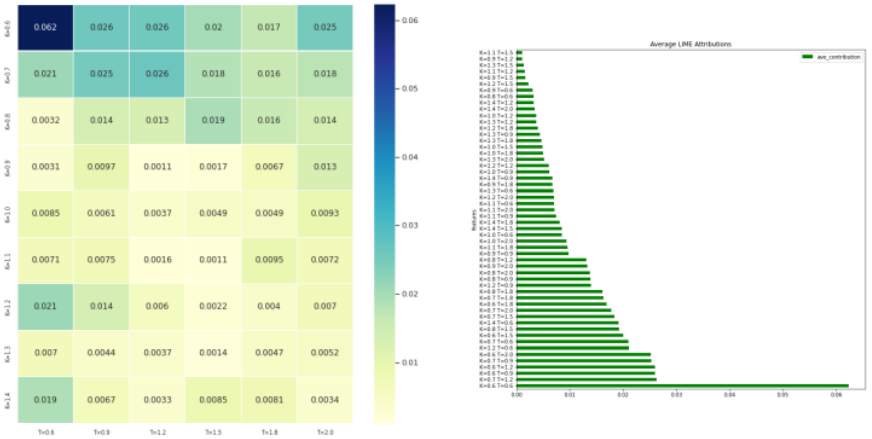


Figure 4.2: LIME attribution of κ at 0th observation (wide range of parameters)



(a) Heat map of LIME attributions

(b) LIME attributions

Figure 4.3: LIME attributions and heat map (wide range of parameters)

1. The range of the prediction covers the range of the parameter set;

2. The volatilities deep in the money contribute most to the prediction;

3. Most contributions of the features are positive;

But it's worth pointing out the difference in findings that the volatilities influencing the prediction most are not that concentrated in the upper left corner of the heatmap but locate in the upper right part, which implies that they are not all with short maturity as before.

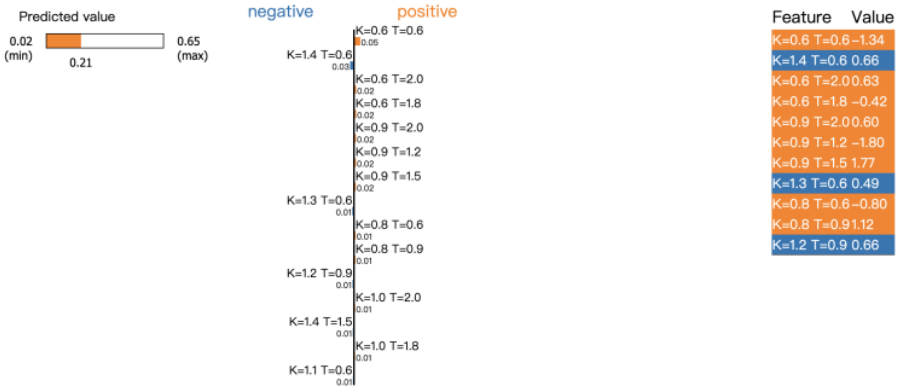
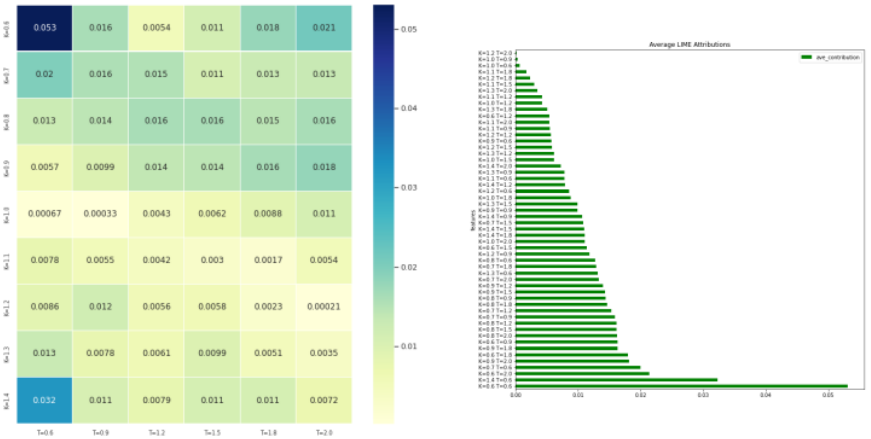


Figure 4.4: LIME attribution of κ at 0th observation (narrow range of parameters)



(a) Heat map of LIME attributions

(b) LIME attributions

Figure 4.5: LIME attributions and heat map (narrow range of parameters)

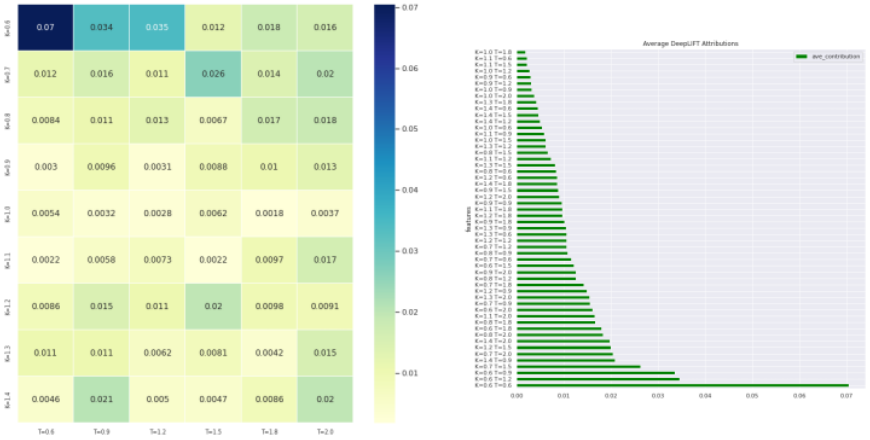
It is worth mentioning that the aforementioned attribution results are simply one of several potential LIME interpretation situations.

Specifically, the predictions will remain the same while the attributions will be likely to vary if the LIME explainer is implmented again for the same instance, such as 0th observation of κ. This demonstrates that LIME employs a local interpretability model thus assigns importance locally, resulting in a different explaination from the same input element each time.

4.4.2 DeepLIFT

Using the scaled and whitened test data, I create the (rescaled) DeepLIFT interpretability model in Python using the DeepExplain module. Here, a zero array with the size of the input is used as the baseline by default. For the FNN, these attributions take the form of (1500, 54).

The all-inclusive influence on the model output P as measured by 1500 test instances is shown in the Figure. The Figure 4.6 demonstrates that the deepest colours only arise around short maturities and extremely little strikes (deep in the money). The cases in interest are K=0.6, T=0.6, K=0.6, T=0.9, and K=0.6, T=1.2.



(a) Heat map of overall DeepLIFT attributions

(b) Overall DeepLIFT attributions

Figure 4.6: Overall DeepLIFT attributions and heat map

The influence on each individual parameter is shown in the Figure 4.7, and you can see how differently the colour blocks are distributed. In particular, κ and nu have characteristics in common with P in that the primary attributions are around short maturities and extremely tiny strikes. The most significant attributions for ρ are, roughly speaking, on the other side of the volatilities, with lengthy maturities and extremely big strikes. The most significant volatility for V_0 appears to be distributed randomly throughout the heat map and does not follow any specific pattern. Theta's most significant volatility tends to have a very lengthy maturity. The huge attributions for the extra parameter H are mostly for extremely short maturities that are far from the ATM (mainly deep in the money).

4.4.3 LRP

Once more, I use LRP in DeepExplain. In particular, I concentrate on ϵ-LRP [15], where ϵ must have a value larger than zero and is preset to 0.0001.

Figures 4.8 and 4.9 suggest findings comparable to those of DeepLIFT: The most important volatility for P, κ, and ν tends to have a very long maturity, whereas the most important volatility for Theta tends to have a short maturity and very small strikes (deep in the money). The vast majority of the attributions for the additional parameter H are for very short maturities that are not in the money (mainly deep in the money). The distinction is in the attributions for ρ and V_0: the most substantial volatility for V_0 emerges with short maturity on both sides of the wings, whereas ρ's attributions for LRP are around extreme short maturity on both sides of the wings far from the ATM.

4.4.4 Shapley values

Using Deep SHAP, we construct global interpretability analysis using Shap in Python. As a background, we pick 1000 samples at random from the test data set.

(a) DeepLIFT attributions for κ

(b) DeepLIFT attributions for ν

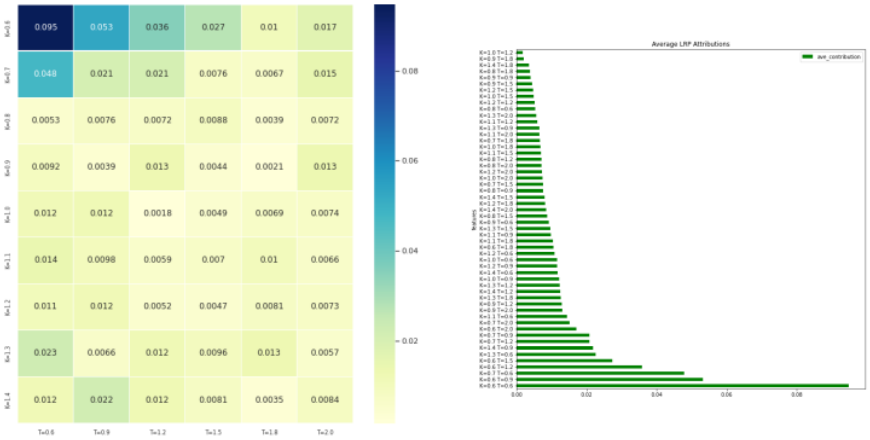
(c) DeepLIFT attributions for ρ

(d) DeepLIFT attributions for V_0

(e) DeepLIFT attributions for θ

(f) DeepLIFT attributions for H

Figure 4.7: DeepLIFT attributions for each parameter



(a) Heat map of overall LRP attributions (b) Overall LRP attributions

Figure 4.8: Overall LRP attributions and heat map

The Figure 4.10 depicts how much effect each attribute has on the overall model prediction, six rHeston parameters, in the test data. The sum of the absolute values of the Shapley values for each rHeston parameter determines the order of the features. Here we only focus on the magnitude of the feature attrication.

As we can see in the Figure 4.10, We use the two data sets generated as discussed in Section 2. For both the topmost features are (K=0.6, T=0.6), which illustrates that the volatility deep in the money with the shortest expiry makes the most contribution to the prediction of the parameters of rHeston. This is aligned with the results of the local interpretability analysis. However, it should be noted that the feature influencing the prediction most is far from ATM, which contradicts the findings from Heston [2] and our expectation. Furthermore, the other important features are not round ATM (K=1) as well. It should also be highlighted that the most important feature is significant, as about twice much as the second.

The Figures 4.11 and 4.12 give a concise breakdown of the elements that are most crucial for explaining the output as well as how the top input features affect the label and output of the model. Each instance in the data collection is represented by a single dot on each feature row. The attribution of the feature to the prediction, which corresponds to the feature's Shapley Value, is shown on the x-axis. The feature value is shown by the colour of the dots, with red denoting high values and blue denoting lesser values. It should be noticed that the density is shown by the dots representing each instance piling up vertically.

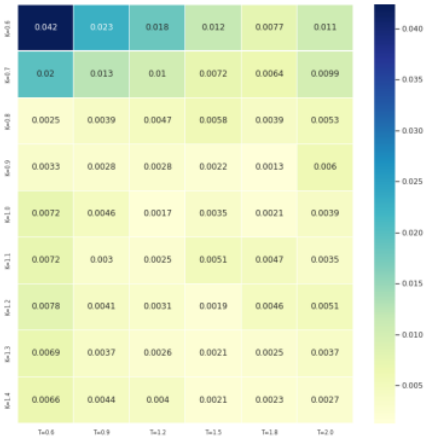
The topmost features for each parameter of rHeston can be found in the Figure 4.11. The most important features for κ, ν are volatilities deep in the money (K=0.6 or K-0.7) with medium expiry. The features that influence ρ most are with the extreme expiries (T=0.6 or T=1.8) and either deep in the money with K=0.6 or deep out of the money with K=1.4. The volatilities with the shortest maturity (K=0.6) on the wings are most important to V_0. For θ and H, the volatilities with the shortest expiry deep in the money affect the prediction most. Furthermore, the Figure 4.12 shows the results with the data set created with the narrower range of rHeston parameters.

Similarly, the Figure 4.13 shows the average of the absolute values of the features attributions for each parameter. It should be noted that there's a difference with the order of the topmost features between the Figure 4.13 and 4.11. This is because the results are not from one single implmentation of SHAP.

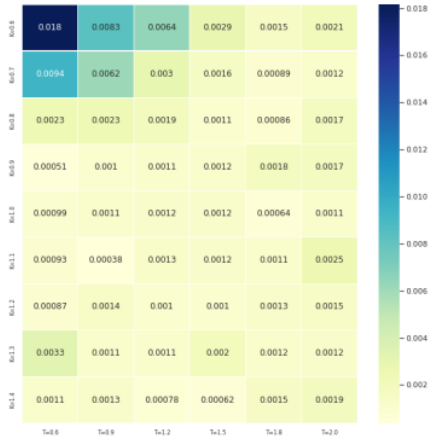
4.4.5 Other methods

I implement the following three models in DeepExplain module in Python.

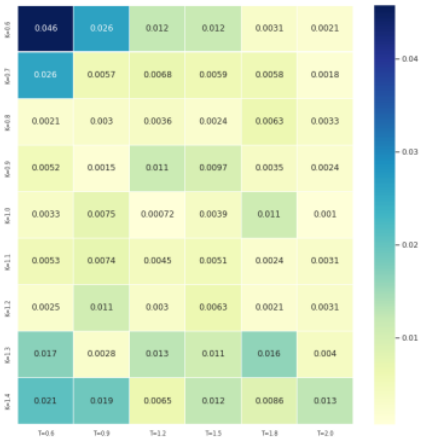
Here I only focus on the average magnitude of attribution for the overall output P.There's strong evidence as seen in the Figure 4.14, 4.15, and 4.16 that the most influential volatility



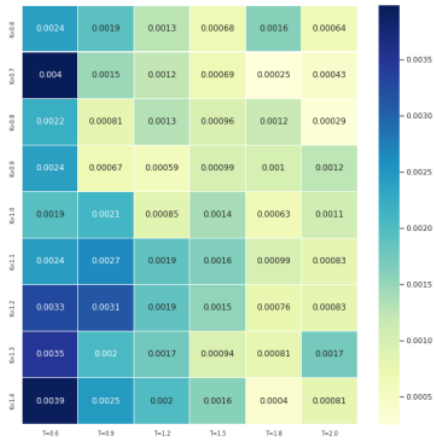
(a) LRP attributions for κ



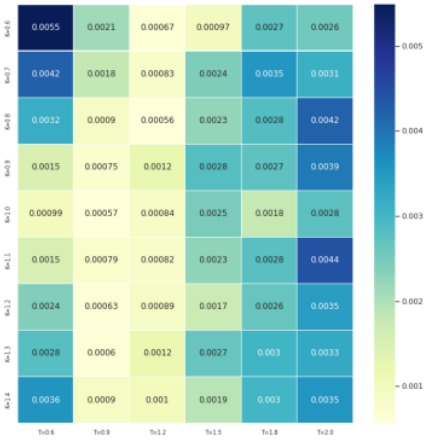
(b) LRP attributions for ν



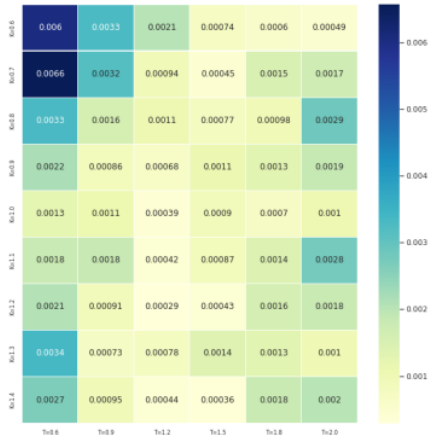
(c) LRP attributions for ρ



(d) LRP attributions for V_0

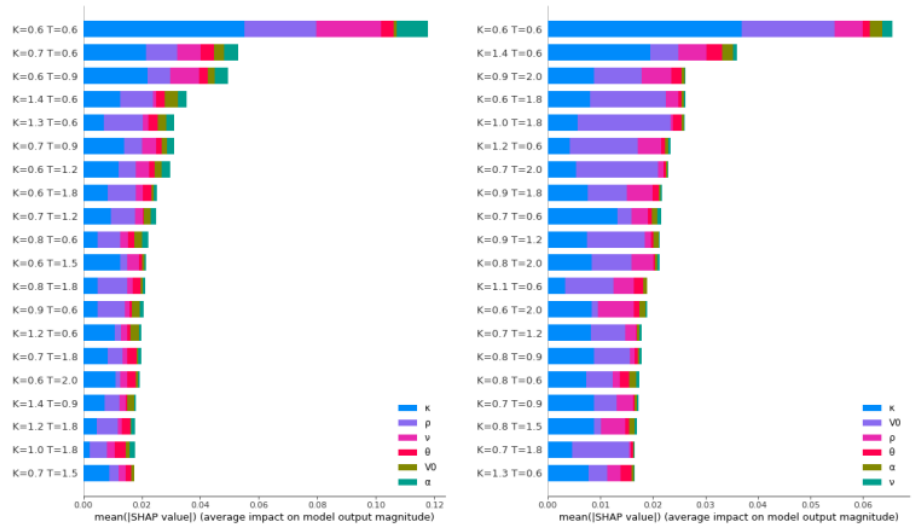


(e) LRP attributions for θ



(f) LRP attributions for H

Figure 4.9: LRP attributions for each parameter



(a) Overall Shapley values for the larger data set

(b) Overall Shapley values for the smaller data set

Figure 4.10: Overall Shapley values

is with the smallest maturity and extreme small strike (K=0.6, T=0.6). The difference is that Integrated Gradients shows only one influential volatility, while Saliency and Gradients*Input highlight topmost features are deep in the money.

4.4.6 Discussions

Now I want to compare the findings from interpretability study with the interpretation from rHeston for each parameter. It is predicted that the two will be consistent.

In reality, the reason is that ATM volatility is the most liquid, we may expect it to be the most significant in calibration. However, from both local and global interpretability analysis, we can conclude that volatilities deep in the money with short maturities contribute the most to the total model output $P = \{\rho, V_0, \kappa, H, \theta, \nu\}$. More research is required to explain this contradiction.

κ is the mean-reversion rate, which should mostly consider volatilities close to the wings of the smile. But LRP, DeepLIFT and Shapley values all illustrate that the most influential ones are only one side (in the money) and with the short maturity.

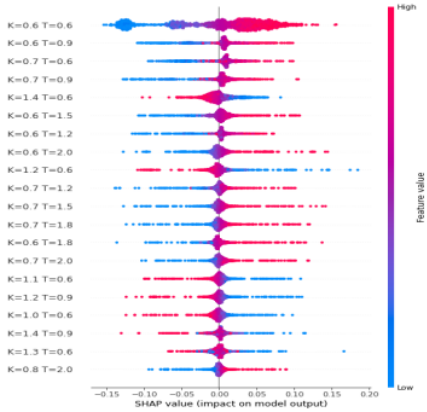
ν is the volatility of volatility. The larger the ν, the more randomness in the variance of the asset price, which implies there will be more obvious volatility smile features. LRP, DeepLIFT and SHAP provide evidence that the topmost features are deep in the money. However, it deserves further research why the topmost are not on both sides and why they are mainly with short maturity.

ρ describes the correlation, which includes information about the shape of the volatility smile. As expected, LRP, DeepLIFT and SHAP demonstrate that the most important features are on both sides of the wings.

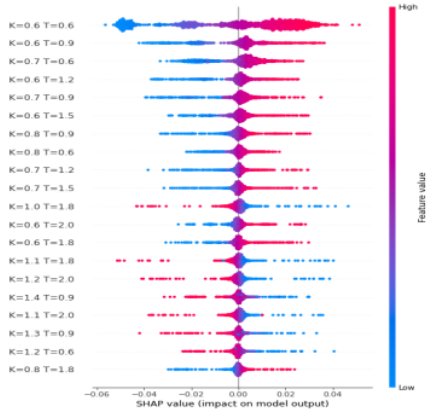
V_0 is the initial value of the variance, which contains information about the initial level of the volatility smile. It's expected that volatilities with short maturities contribute most to the prediction of V_0, which the evidence from LRP and SHAP can support.

θ is the long-term average variance of asset price, to which the volatilities with long expiry are most relevant. Both LRP and DeepLIFT support this interpretation. But the global interpretability contradicts by showing that the features deep in the money with relatively short expiry contribute most.

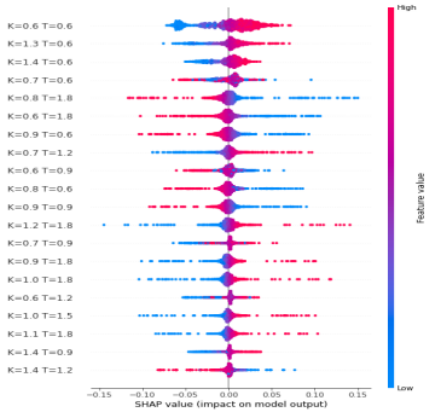
The additional parameter H measures the roughness of the volatility, which relates to the shape of the volatility smile, specifically by looking at the wings what extremes are like. LRP, DeepLIFT



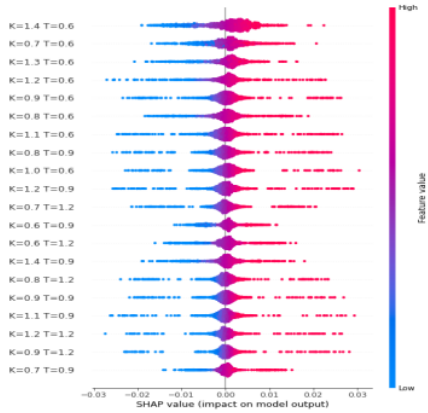
(a) Shapley values for κ



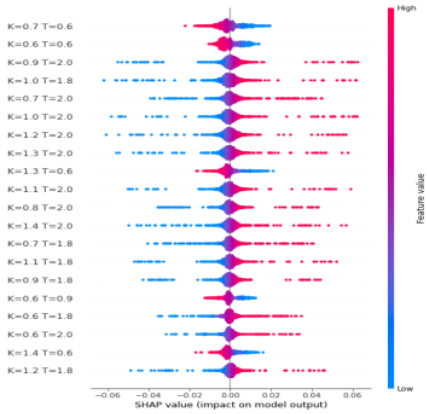
(b) Shapley values for ν



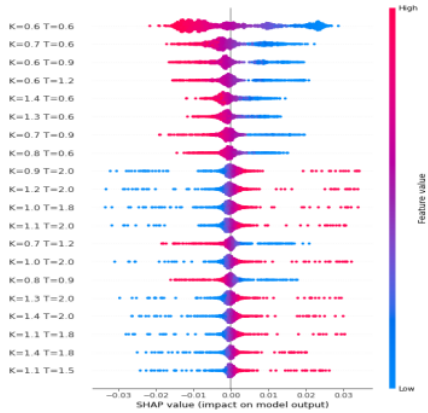
(c) Shapley values for ρ



(d) Shapley values for V_0

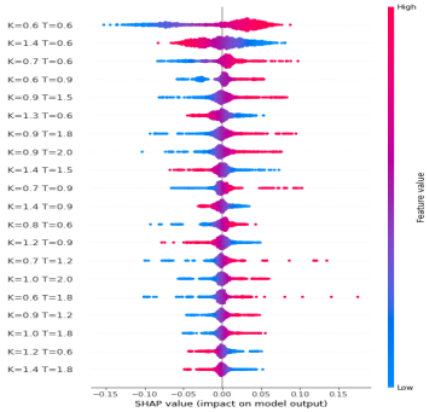


(e) Shapley values for θ

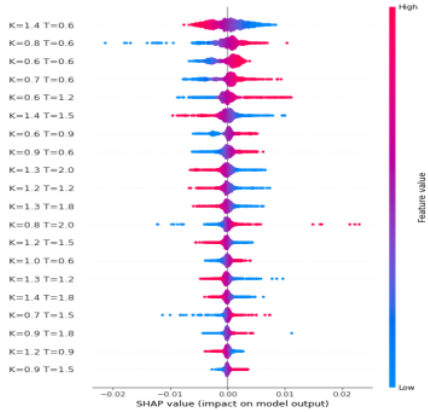


(f) Shapley values for H

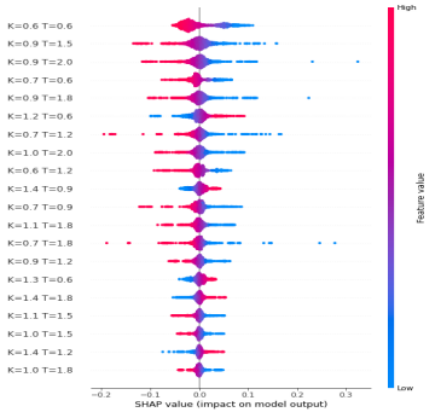
Figure 4.11: Shapley values for each parameter (wider range)



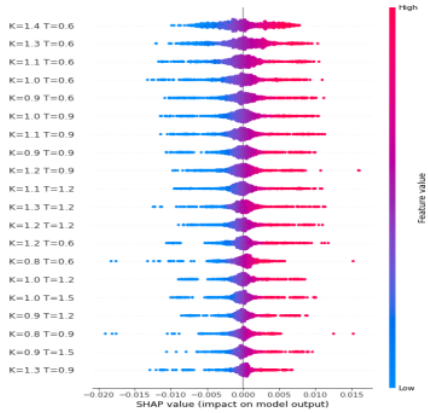
(a) Shapley values for κ



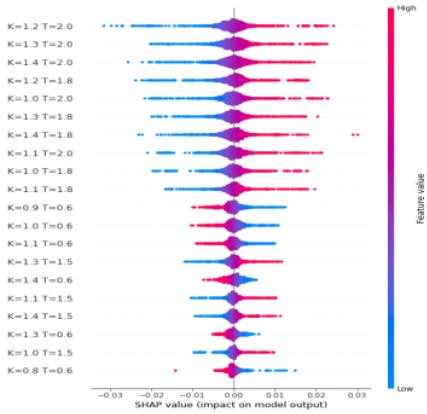
(b) Shapley values for ν



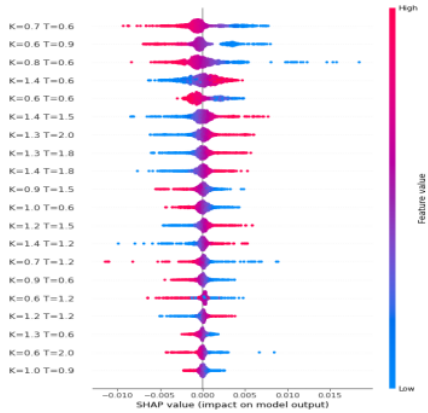
(c) Shapley values for ρ



(d) Shapley values for V_0

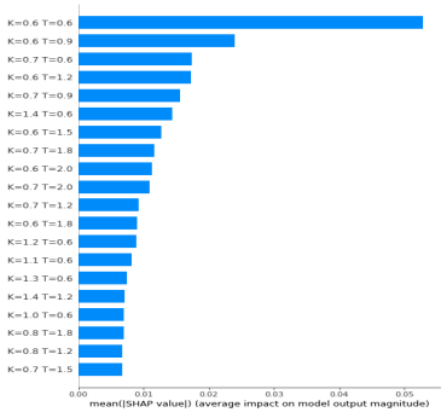


(e) Shapley values for θ

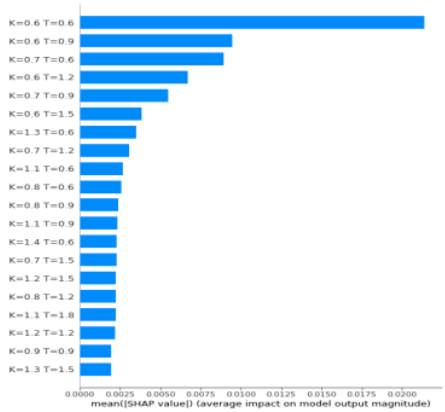


(f) Shapley values for H

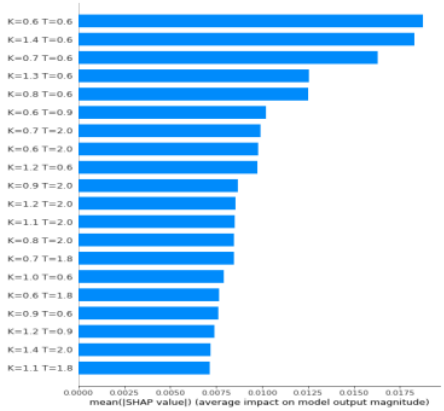
Figure 4.12: Shapley values for each parameter (narrower range)



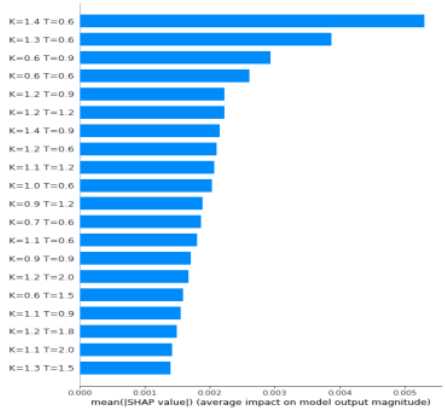
(a) Shapley values for κ



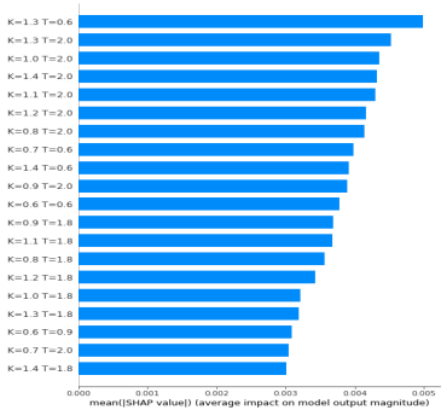
(b) Shapley values for ν



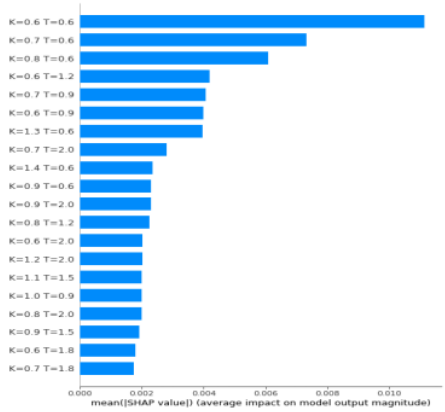
(c) Shapley values for ρ



(d) Shapley values for V_0

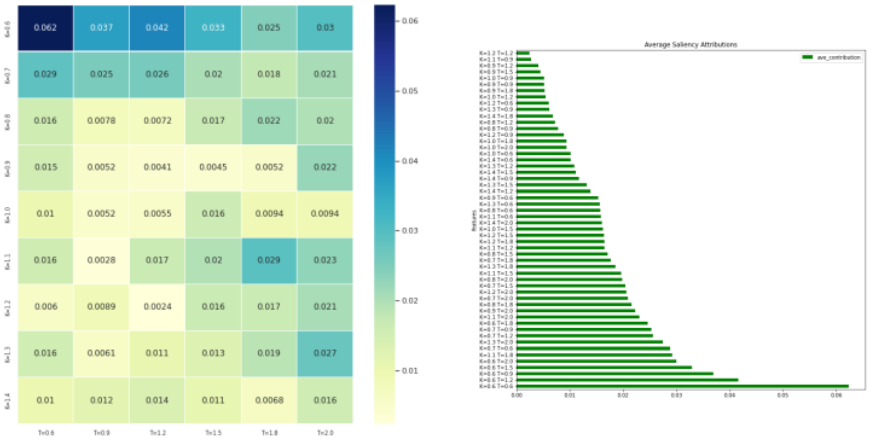


(e) Shapley values for θ



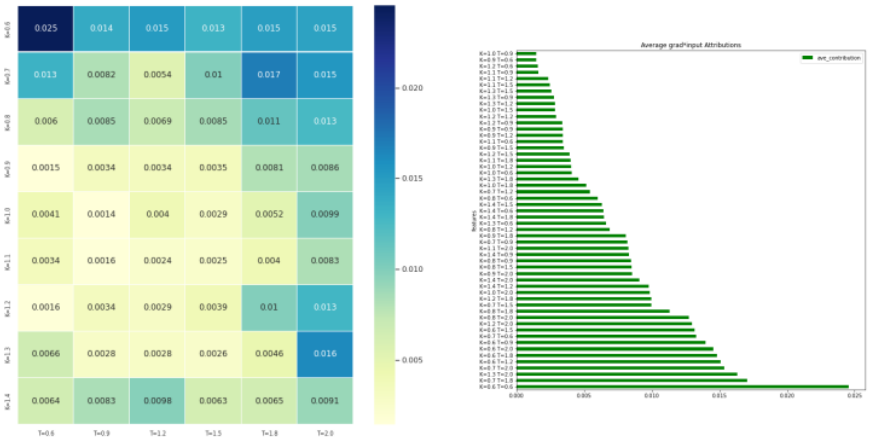
(f) Shapley values for H

Figure 4.13: Shapley values for each parameter (mean of the unsigned values)



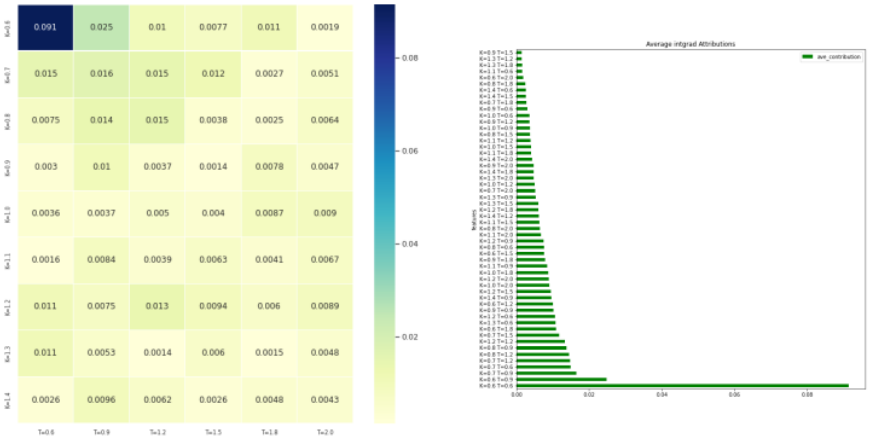
(a) Heat map of overall Saliency attributions (b) Overall Saliency attributions

Figure 4.14: Overall Saliency attributions and heat map



(a) Heat map of overall Gradients*Input attributions (b) Overall Gradients*Input LRP attributions

Figure 4.15: Overall Gradients*Input attributions and heat map



(a) Heat map of overall Integrated Gradients attributions (b) Overall Integrated Gradients attributions

Figure 4.16: Overall Integrated Gradients attributions and heat map

and SHAP all support this interpretation. However, further study should be conducted to discover why the most significant features are all with short maturities.

Chapter 5

Conclusion

The findings are broken down into three categories: insights from deep calibration, findings from interpretability analysis, and further developments.

5.1 Insights from deep calibration

By experimenting with various data preprocessing techniques, changing the number of neurons and layers, adjusting activation functions and optimisers, and accommodating learning rate and batch size, it has been discovered that the simple architecture with one hidden layer, decreasing neurons, and ELU activation function works well, achieving over 90% accuracy.

It should be noticed that while experimenting with common approaches for dealing with smooth training history, such as modifying the batch size and tweaking noisy regularisation like dropout, the training history stays smooth. According to FNN's good prediction performance, it is acceptable. The key reason for the smooth training history is that the data is intentionally manufactured and hence not noisy enough, which sheds light on future study.

We fit FNN to two data sets generated by rHeston, one with a small range of parameters and the other with a broad range of parameters. We may conclude from comparing the fitting results using two data sets plus out-of-sample data that the FNN trained with parameters with a narrower range fails to predict all six rHeston parameters well, particularly κ and ρ, but the broader data set does. It is obvious that the parameter range matters in FNN performance.

5.2 Findings from Interpretability Models

We primarily concentrate on local surrogate models (LIME, DeepLIFT, and LRP) as well as other popular gradient-based explainable approaches (Saliency Maps, Gradients * Input, and Integrated Gradients). Plus, we give a preliminary result with the global interpretability analysis, specifically with SHAP. Overall, the evidence from the local and global interpretability analysis is almost consistent.

In actuality, we may anticipate ATM volatility to be the most significant in calibration since it is the most liquid volatility. However, contradictory to the expectation, we can generally deduce from all of the results that for the overall feature importance, volatilities deep in the money with short maturities contribute the most to the overall model output $P = \{\rho, V_0, \kappa, H, \theta, \nu\}$.

The interpretation from rHeston for some parameters and the outcomes from the interpretability study are consistent, as predicted. The topmost features of the correlation as described by ρ are on both wings of the volatility. It is predicted and shown by LRP and SHAP that volatilities with short maturities contribute the most to V_0 which measures the initial value of variance. Both LRP and DeepLIFT provide evidence that theta, as average variance of asset price over the long term, is mostly driven by volatility with long expiration.

However, there is a contradiction that necessitates additional investigation. κ as the mean-reversion rate, ν as the volatility of volatility, and H as the roughness of the volatility are all expected to characterize the shape of the volatility wings. As a result, all of the top features are expected to be on both sides of the volatility. However, according to our interpretability analysis, they are only on one side of the volatility and have a short maturity.

5.3 Further developments

Our NNs are currently trained on data created with self-defined limits for individual parameter. In actuality, market data may be noisy. It would give more insights to train the networks and obtain attribution findings using real market-calibrated data. This may be used to test hypotheses based on the smooth training history. These market data should be viewed as supplementary input data for future work, which will need modifying our present NN structures.

Due to backpropagation, NNs are nonlinear and non-local. Local surrogate models are based on localization and, in certain cases, linearity, therefore they provide different interpretations for each individual prediction, whereas Shapley Values gives global interpretations for feature influence on model output. There is also evidence that Shapley values appear to capture those characteristics most well [2]. As a result, additional research into implementing the global interpretability approach should be extended. This also helps us investigate and comprehend the existing mismatch between the model interpretation and the results of the interpretability study on the parameters.

Bibliography

[1] Wikipedia. Feedforward neural network — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Feedforward%20neural%20network&oldid=1086331853, 2022. [Online; accessed 17-August-2022].

[2] Damiano Brigo, Xiaoshan Huang, Andrea Pallavicini, and Haitz Sáez de Ocáriz Borde. Interpretability in deep learning for finance: a case study for the heston model. Available at SSRN 3829947, 2021.

[3] Jim Gatheral, Thibault Jaisson, and Mathieu Rosenbaum. Volatility is rough. Quantitative finance, 18(6):933–949, 2018.

[4] Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. Artificial intelligence, 267:1–38, 2019.

[5] Been Kim, Rajiv Khanna, and Oluwasanmi O Koyejo. Examples are not enough, learn to criticize! criticism for interpretability. Advances in neural information processing systems, 29, 2016.

[6] Elias M Stein and Jeremy C Stein. Stock price distributions with stochastic volatility: an analytic approach. The review of financial studies, 4(4):727–752, 1991.

[7] Steven L Heston. A closed-form solution for options with stochastic volatility with applications to bond and currency options. The review of financial studies, 6(2):327–343, 1993.

[8] Gurdip Bakshi and Dilip Madan. Spanning and derivative-security valuation. Journal of financial economics, 55(2):205–238, 2000.

[9] Peter Carr and Dilip Madan. Option valuation using the fast fourier transform. Journal of computational finance, 2(4):61–73, 1999.

[10] Ahmet Murat Ozbayoglu, Mehmet Ugur Gudelek, and Omer Berat Sezer. Deep learning for financial applications: A survey. Applied Soft Computing, 93:106384, 2020.

[11] Damiano Brigo. Deep learning: Interpretability? imperial college. Quant Minds, 2019.

[12] Christoph Molnar. Interpretable machine learning. Lulu. com, 2020.

[13] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, pages 1135–1144, 2016.

[14] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In International conference on machine learning, pages 3145–3153. PMLR, 2017.

[15] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. PloS one, 10(7):e0130140, 2015.

[16] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. Advances in neural information processing systems, 30, 2017.

[17] Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. Towards better understanding of gradient-based attribution methods for deep neural networks. arXiv preprint arXiv:1711.06104, 2017.

[18] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In European conference on computer vision, pages 818–833. Springer, 2014.

[19] Blanka Horvath, Aitor Muguruza, and Mehdi Tomas. Deep learning volatility: a deep neural network perspective on pricing and calibration in (rough) volatility models. Quantitative Finance, 21(1):11–27, 2021.

[20] Dirk Roeder and Georgi Dimitroff. Volatility model calibration with neural networks a comparison between direct and indirect methods. arXiv preprint arXiv:2007.03494, 2020.

[21] Wikipedia. Fractional Brownian motion — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Fractional%20Brownian%20motion&oldid=1084316768, 2022. [Online; accessed 9-June-2022].

[22] Ton Dieker. Simulation of fractional Brownian motion. PhD thesis, Masters Thesis, Department of Mathematical Sciences, University of Twente . . . , 2004.

[23] Benoit B Mandelbrot and John W Van Ness. Fractional brownian motions, fractional noises and applications. SIAM review, 10(4):422–437, 1968.

[24] SIMONE MOLINARI. The rough heston model. 2020.

[25] Omar El Euch, Jim Gatheral, and Mathieu Rosenbaum. Roughening heston. Risk, pages 84–89, 2019.

[26] Omar El Euch and Mathieu Rosenbaum. Perfect hedging in rough heston models. The Annals of Applied Probability, 28(6):3813–3856, 2018.

[27] Omar El Euch and Mathieu Rosenbaum. The characteristic function of rough heston models. Mathematical Finance, 29(1):3–38, 2019.

[28] J Gil-Pelaez. Note on the inversion theorem. Biometrika, 38(3-4):481–482, 1951.

[29] Alan L Lewis. A simple option formula for general jump-diffusion and other exponential lévy processes. Available at SSRN 282110, 2001.

[30] Kai Diethelm, Neville J Ford, and Alan D Freed. A predictor-corrector approach for the numerical solution of fractional differential equations. Nonlinear Dynamics, 29(1):3–22, 2002.

[31] Siow Woon Jeng and Adem Kiliçman. Spx calibration of option approximations under rough heston model. Mathematics, 9(21):2675, 2021.

[32] Blanka Horvath, Aitor Muguruza, and Mehdi Tomas. Deep learning volatility. arXiv preprint arXiv:1901.09647, 2019.

[33] Elisa Alos, David García-Lorite, and Aitor Muguruza. On smile properties of volatility derivatives and exotic products: understanding the vix skew. arXiv preprint arXiv:1808.03610, 2018.

[34] Hamza Guennoun, Antoine Jacquier, Patrick Roome, and Fangwei Shi. Asymptotic behavior of the fractional heston model. SIAM Journal on Financial Mathematics, 9(3):1017–1045, 2018.

[35] Antoine Jacquier, Claude Martini, and Aitor Muguruza. On vix futures in the rough bergomi model. Quantitative Finance, 18(1):45–61, 2018.

[36] Deep learning tutorial - pca and whitening, Jun 2014.

[37] Posted bylearndataa. Data preprocessing: Whitening or sphering in python, Sep 2020.

[38] Keras Team. Keras documentation: Adam.

[39] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, Advances in Neural Information Processing Systems, volume 30. Curran Associates, Inc., 2017.

[40] Scott M Lundberg, Gabriel Erion, Hugh Chen, Alex DeGrave, Jordan M Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee. From local explanations to global understanding with explainable ai for trees. Nature machine intelligence, 2(1):56–67, 2020.

[41] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. arXiv preprint arXiv:1312.6034, 2013.

[42] Avanti Shrikumar, Peyton Greenside, Anna Shcherbina, and Anshul Kundaje. Not just a black box: Learning important features through propagating activation differences. arXiv preprint arXiv:1605.01713, 2016.

[43] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In International conference on machine learning, pages 3319–3328. PMLR, 2017.