Structural Validation of Neural Network Calibration for Stochastic Volatility Models

An Explainable AI Perspective

Shaïn Afzali

Delft University of Technology

TU Delft

Structural Validation of Neural Network Calibration for Stochastic Volatility Models

An Explainable AI Perspective

by

Shaïn Afzali

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Thursday May 21, 2026 at 10:00 AM.

Student number:	5094763	
Project duration:	June 18, 2025 – May 21, 2026	
Thesis committee:	Prof. dr. A. Papapantoleon	TU Delft, supervisor
	Dr. A. Heinlein	TU Delft
	Dr. F. Fang	TU Delft
	Dr. S. Della Corte	TU Delft, advisor

Cover: Synthetic implied volatility surface generated with the rough Heston model.

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Abstract

This thesis investigates neural network-based approaches for the inverse calibration of stochastic volatility models, with a focus on both predictive performance and structural interpretability. Calibration is formulated as an inverse problem, where model parameters are inferred from implied volatility surfaces, a setting that is inherently non-linear and potentially ill-conditioned. While neural networks offer a computationally efficient alternative to classical optimisation-based methods, existing work has primarily focused on predictive accuracy, with comparatively limited attention given to the interpretability of the learned inverse mappings. This lack of transparency raises concerns in financial applications, where understanding model behaviour is essential for validation and risk management. Recent work has begun to address this issue, but a systematic framework for validating the learned inverse mappings remains limited. To address this, we evaluate several neural network architectures, including the multilayer perceptron (MLP), Highway networks, and Generalised Highway networks, trained on synthetically generated implied volatility surfaces from the Heston and rough Heston models. We introduce a framework for the structural validation of learned calibration mappings using eXplainable Artificial Intelligence (XAI) methods, specifically SHAP and νSHAP, which capture complementary notions of feature relevance.

The results show that neural networks achieve highly accurate and fast calibration within the training domain, with the Generalised Highway architecture consistently outperforming its counterparts in terms of validation error. The XAI analysis confirms that the learned mappings rely on structurally meaningful regions of the implied volatility surface, most notably short maturities and the wings of the smile, aligning with financial intuition. At the same time, the comparison between SHAP and νSHAP highlights non-homogeneous parameter identifiability: while some parameters are associated with concentrated and stable regions of importance, others are inferred from diffuse and overlapping information, indicating redundancy in the inverse mapping.

In addition, we demonstrate that XAI can be used to inform model design. In particular, a νSHAP-guided reduction of the input grid yields comparable or improved predictive performance despite a substantial reduction in input dimensionality.

Overall, the findings show that neural network-based calibration can be both accurate and structurally interpretable when combined with appropriate analysis tools, while also revealing important limitations related to generalisation and parameter identifiability.

Preface

First and foremost, I would like to express my sincere gratitude to my supervisor, Antonis Papapantoleon, and my daily co-supervisor, Serena Della Corte, for their guidance, availability, and valuable feedback throughout this project. Despite their busy schedules, they were always willing to provide support and share their insights whenever needed.

I would also like to express my sincere appreciation to Behrouz Raftari Tangabi, who guided me during my time at ING. He introduced me to νSHAP and provided invaluable support during its implementation. His guidance and willingness to involve me as part of the team made this a particularly valuable and rewarding experience.

Furthermore, I would like to thank Alexander Heinlein and Fang Fang for their willingness to serve on my thesis committee.

Finally, I would like to thank my family and friends for their continuous support throughout my studies. I recall being advised in secondary school not to pursue the more mathematically intensive track, and it is therefore particularly meaningful to now conclude this journey with a degree in mathematics. I am also grateful for this thesis project itself, which has sparked a strong interest in option pricing, neural networks, and explainable artificial intelligence.

Shaïn Afzali
Delft, May 2026

Contents

List of Figures

List of Tables

1 Introduction

During the past decade, machine learning (ML) has become an integral component of quantitative finance, driven by the increasing availability of data, computational power, and the limitations of traditional numerical calibration methods. In particular, neural networks have demonstrated strong performance in complex, high-dimensional tasks where analytical or numerical techniques become computationally expensive or unstable. Within quantitative finance, ML models have been successfully applied to portfolio optimisation, algorithmic trading, risk management, fraud detection, and derivative pricing and calibration [1].

For the pricing of option derivatives, the relationship between model parameters and observed market prices is typically defined through a forward mapping: given a set of model parameters, option prices can be computed using analytical, semi-analytical, or numerical techniques. Model calibration, however, constitutes an inverse problem, where one seeks to recover the model parameters that best reproduce a given implied volatility surface. This inverse mapping is typically ill-conditioned, non-linear, and computationally demanding, especially when the pricing function itself is expensive to evaluate.

Classical calibration techniques therefore rely on repeated numerical evaluations of pricing functions (typically within an optimisation routine), often leading to prohibitively high computational costs when calibrating to large implied volatility surfaces or when more involved models, such as rough volatility models, are considered. Neural networks offer a promising alternative by learning an efficient surrogate of the inverse pricing map. Once trained, such models allow for near-instantaneous calibration, effectively replacing the expensive optimisation loop by a single forward pass through the network. Several contributions involving neural-network-based calibration models can be found in the literature, most notably the works by Hernandez [17], Horvath et al. [19], and Roeder and Dimitroff [38].

Despite their computational advantages, neural-network-based calibration models are often criticized for their lack of interpretability [50]. In the quantitative finance context, calibration parameters carry clear model-based interpretations, and understanding how different regions of the option surface influence the parameters resulting from calibration is essential for model validation and risk management. This creates a tension between *predictive accuracy* and *interpretability*: while complex models may achieve superior fit to market data, their behaviour can be difficult to analyse or monitor. Simpler models, even if less accurate, are sometimes preferred in practice because their parameters and limitations are easier to interpret. In contrast, neural networks offer the ability to approximate highly non-linear calibration mappings and to produce parameter estimates almost instantaneously once trained. This gain in predictive flexibility and computational efficiency comes at the cost of reduced transparency, as the influence of individual regions of the implied volatility surface on the calibrated parameters is not directly observable.

Interpretability is therefore key to promoting the use of neural-network-based calibration models further, as it provides a safety net for risk monitoring. Broadly speaking, the more one can understand a model decision or anticipate the model conclusion, the more interpretable the model is [31]. A universally accepted mathematical definition is not yet available; an overview of existing definitions is provided in [32]. This need for greater interpretability has motivated substantial research into *eXplainable Artificial Intelligence* (XAI), which aims to provide insight into the internal decision mechanisms of complex ML models.

Among the most prominent XAI methods are *additive feature attribution methods*, which decompose a model prediction into contributions of individual input features. Notable examples include *LIME* [37], *DeepLIFT* [44], and *Layer-wise Relevance Propagation* (LRP) [3]. In particular, Lundberg and Lee [25] showed that within this class there exists a unique solution satisfying a set of desirable axioms derived from cooperative game theory, leading to the *SHAP* framework (SHapley Additive exPlanations). SHAP values have since become one of the most widely used tools for explaining machine learning predictions.

More recently, alternative attribution methods have been proposed that capture different notions of feature relevance. One such method is ν*SHAP* ("nuSHAP") [28], which is based on a modified value function and places greater emphasis on feature necessity. As a result, SHAP and νSHAP provide different but potentially complementary perspectives on how a neural network uses information from its inputs.

While neural-network-based calibration methods frequently appear in the literature, comparatively little work has examined how the learned calibration mappings can be interpreted using XAI methods. A notable recent contribution in this direction is the work of Yuan et al. [49], who introduced a framework for interpreting neural-network-based calibration of rough volatility models using XAI. Their work provides initial evidence that feature attribution methods can yield insights into the learned calibration mappings, but does not address whether these explanations are structurally consistent, robust, or sufficient for model validation purposes. Building on this line of research, this thesis extends and strengthens the interpretability analysis of neural-network-based calibration of (rough) stochastic volatility models by introducing a framework for structural validation using complementary XAI methods. By combining several calibration architectures with modern XAI methods, we assess not only predictive accuracy, but also the structural information that the networks learn from implied volatility surfaces. In particular, we analyse the explanations produced by SHAP and νSHAP, and compare the extent to which these methods provide consistent and financially meaningful insights into the learned calibration mappings. This leads to the central research question of this thesis:

> *Can neural-network-based inverse calibration models for stochastic volatility be structurally validated using XAI methods, beyond traditional predictive error metrics?*

Our main contributions are as follows:

- We investigate the structural validation of neural-network-based calibration using complementary XAI methods, specifically SHAP and νSHAP, in the context of stochastic volatility models.
- We show that XAI methods can provide actionable guidance for model design, in particular by constructing an XAI-guided reduced input grid that achieves comparable or improved predictive performance with substantially fewer input features.
- We provide empirical insights into the structure and identifiability of the inverse calibration problem for the Heston and rough Heston models, highlighting non-homogeneous parameter identifiability and the role of specific regions of the implied volatility surface.

The thesis is organised as follows. Chapter 2 introduces the Heston and rough Heston stochastic volatility models and reviews their main properties. Chapter 3 presents the neural network calibration framework used in this study. In Chapter 4, we present the empirical calibration results and robustness experiments. Chapter 5 analyses the learned calibration mappings using the XAI methods as discussed above. Lastly, Chapter 6 presents the discussion and conclusions.

2

Stochastic Volatility Models & Numerical Framework

2.1. Stochastic Calculus Fundamentals

We work on a filtered probability space $(\Omega, \mathcal{F}, (\mathcal{F}_t)_{t\geq 0}, \mathbb{P})$, where Ω denotes the sample space, $\mathcal{F}$ the associated σ-algebra of events, and $\mathbb{P}$ the real-world probability measure. Alternative probability measures, in particular the risk-neutral measure, will be introduced in the context of option pricing. The filtration $(\mathcal{F}_t)_{t\geq 0}$ represents the information available up to time t, and is assumed to satisfy the usual conditions. All stochastic processes considered in this thesis are adapted to this filtration.

Asset prices and derivative payoffs are modelled as random variables on this probability space. In particular, for a given maturity $T > 0$, the payoff of a European-style derivative (option) is represented by an $\mathcal{F}_T$-measurable random variable.

2.1.1. Brownian Motion

Brownian motion is the fundamental source of randomness in continuous-time financial models and serves as the driving process in *stochastic differential equations* (SDEs).

Definition 1. *A real-valued stochastic process $\{W(t) : t \geq 0\}$ is called a (standard) Brownian motion if it satisfies the following properties:*

1. $W(0) = 0$ *almost surely;*
2. *For all $0 \leq s < t$, the increment $W(t) - W(s)$ is normally distributed with mean zero and variance $t - s$, i.e. $W(t) - W(s) \sim \mathcal{N}(0,\ t - s)$;*
3. *For any $0 \leq t_0 < t_1 < \cdots < t_n$, the increments $W(t_i) - W(t_{i-1})$, $i = 1, \ldots, n$, are independent;*
4. *The sample paths $t \mapsto W(t)$ are continuous.*

Brownian motion captures the continuous-time evolution of random fluctuations and forms the basis for modelling asset price dynamics in both the Black–Scholes model and the stochastic volatility models considered in this thesis.

2.1.2. Martingales

A central concept in stochastic calculus and mathematical finance is that of a martingale, which formalises the idea of a "fair game" in the sense that, conditional on the information currently available, the expected future value of the process equals its present value.

Definition 2. *Let $(X_t)_{t\geq 0}$ be a stochastic process adapted to the filtration $(\mathcal{F}_t)_{t\geq 0}$. Then $(X_t)_{t\geq 0}$ is called a martingale with respect to $(\mathcal{F}_t)_{t\geq 0}$ if:*

1. *X_t is integrable for all $t \geq 0$, i.e. $\mathbb{E}[|X_t|] < \infty$;*
2. *for all $0 \leq s \leq t$,*

$$\mathbb{E}[X_t \mid \mathcal{F}_s] = X_s.$$

Thus, a martingale is a process whose conditional expected future value, given the information available up to time s, coincides with its value at time s. Brownian motion is a classical example of a martingale with respect to its natural filtration. In financial applications, martingales play a central role because under the risk-neutral measure, discounted asset prices are martingales. This property forms the foundation of modern no-arbitrage pricing and will be central to the option pricing framework introduced in the next section.

2.2. Option Pricing Framework

In this section, we introduce the option pricing framework that forms the basis for the pricing models considered. We begin by defining option contracts and their associated payoff structures, after which we turn to the problem of assigning a value to such payoffs in a no-arbitrage setting. The main results in this section are stated without proof; we refer the interested reader to [43] for a rigorous and detailed treatment.

An option is a financial derivative whose value depends on an underlying asset, typically a stock with price process $(S_t)_{t\geq 0}$. An option grants its holder the right, but not the obligation, to buy or sell the underlying asset at a predetermined price K, referred to as the strike price, at or before a specified maturity date T. The counterparty, known as the option writer, is obliged to fulfil the contract if the holder chooses to exercise the option.

Two fundamental types of options are distinguished. A *call* option gives the holder the right to buy the underlying asset at price K, whereas a *put* option gives the right to sell the asset at K. Options can further be classified by their exercise style. A *European* option can only be exercised at maturity T, while an *American* option can be exercised at any time $t \leq T$. In this thesis, we restrict the attention to European-style options.

The value of an option at maturity is determined by its payoff. For a European call option, exercise is optimal only when $S_T > K$, yielding a payoff of $S_T - K$. Otherwise, the option expires worthless. This leads to the payoff function

$$H_{\text{call}}(S_T) = (S_T - K)^+, \tag{2.1}$$

where $(x)^+ = \max\{x, 0\}$. Similarly, the payoff of a European put option is given by

$$H_{\text{put}}(S_T) = (K - S_T)^+. \tag{2.2}$$

Figure 2.1 illustrates the payoff profiles of call and put options. An option is said to be *in-the-money* (ITM) if its payoff is strictly positive, *at-the-money* (ATM) if the payoff is close to zero, and *out-of-the-money* (OTM) if the payoff is zero.

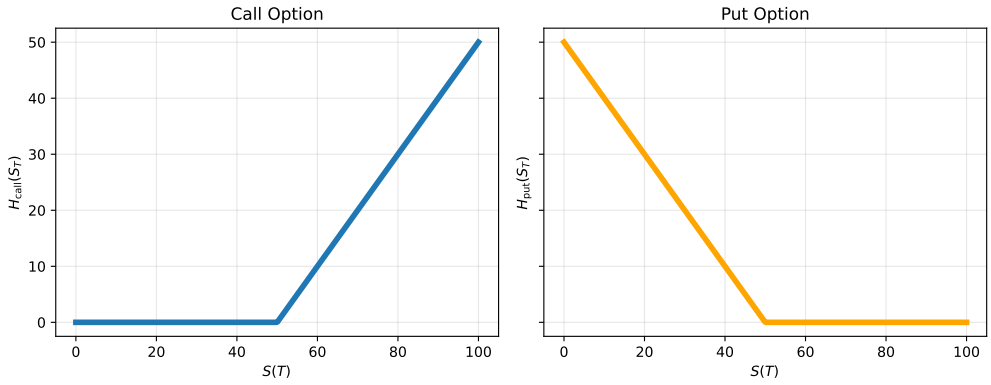


Figure 2.1: Payoff diagrams of a European call option (left) and a European put option (right), with strike price $K = 50$.

2.2.1. Stochastic Asset Dynamics

Let $(S_t)_{t\geq 0}$ denote the price process of the underlying asset, modelled as a stochastic process adapted to the filtration $(\mathcal{F}_t)_{t\geq 0}$. In continuous-time financial models, the evolution of asset prices is typically modelled to follow a geometric Brownian motion, described by an SDE of the form

$$dS_t = \mu S_t dt + \sigma S_t dW_t^{\mathbb{P}}, \tag{2.3}$$

where $\mu \in \mathbb{R}$ represents the drift, σ denotes the volatility (which is constant, but can be made time-dependent and stochastic), and $(W_t^{\mathbb{P}})_{t\geq 0}$ is a Brownian motion under the real-world measure $\mathbb{P}$.
The multiplicative structure of the dynamics ensures that the asset price remains strictly positive. Indeed, the solution to the SDE can be expressed as an exponential of a stochastic process, which is strictly positive almost surely. Different modelling assumptions on the volatility process σ_t lead to different asset pricing models. For instance, the Black–Scholes model assumes constant volatility, while stochastic volatility models such as the Heston model and its rough extension allow σ_t to evolve dynamically over time.

2.2.2. Absence of Arbitrage

A central assumption in mathematical finance is the absence of arbitrage opportunities. Informally, an arbitrage opportunity is a trading strategy that requires no initial investment, yields a non-negative payoff almost surely, and produces a strictly positive payoff with positive probability.

In well-functioning financial markets, such opportunities are assumed not to exist. This assumption imposes strong restrictions on the dynamics of asset prices and forms the foundation of modern pricing theory.

To formalise this idea, consider the discounted asset price process defined by

$$\tilde{S}_t = e^{-rt} S_t, \tag{2.4}$$

where r denotes the constant risk-free rate. A fundamental result in mathematical finance states that, in the absence of arbitrage, there exists an equivalent probability measure under which the discounted asset price process behaves as a martingale.

This result motivates the introduction of the risk-neutral measure, under which asset prices can be consistently valued in a no-arbitrage setting.

2.2.3. Risk-Neutral Pricing

The equivalent probability measure under which discounted asset prices are martingales is referred to as the *risk-neutral measure*, commonly denoted by $\mathbb{Q}$. By construction, this measure is equivalent to the real-world measure $\mathbb{P}$, in the sense that both measures assign zero probability to the same events.

Under the risk-neutral measure, the discounted asset price process

$$\tilde{S}_t = e^{-rt} S_t$$

is a martingale. In particular, while under the real-world measure $\mathbb{P}$ the asset price follows

$$dS_t = \mu S_t dt + \sigma S_t dW_t^{\mathbb{P}}, \tag{2.5}$$

under the risk-neutral measure $\mathbb{Q}$ the dynamics take the form

$$dS_t = r S_t dt + \sigma S_t dW_t^{\mathbb{Q}}, \tag{2.6}$$

where $(W_t^{\mathbb{Q}})_{t\geq 0}$ is a Brownian motion under $\mathbb{Q}$. Thus, the drift of the asset price is replaced by the risk-free rate r. This change of measure allows derivative prices to be expressed as conditional expectations under $\mathbb{Q}$.

To this end, consider a European-style derivative with maturity T and payoff given by a measurable function $H(S_T)$. The problem of option pricing consists of determining the value of this contract at time $t \leq T$.

Under the assumption of no-arbitrage, and with respect to the risk-neutral measure $\mathbb{Q}$, the value of the derivative is given by the risk-neutral pricing formula

$$V(t,S) = e^{-r(T-t)} \mathbb{E}^{\mathbb{Q}}[H(S_T) \mid \mathcal{F}_t], \tag{2.7}$$

where $V(t,S)$ denotes the option value at time t and r denotes the constant risk-free rate.

This expression states that the price of the derivative is equal to the discounted conditional expectation of its future payoff, taken under the risk-neutral measure. The conditioning on $\mathcal{F}_t$ reflects the information available at time t.
In particular, at the initial time $t = 0$, the option value simplifies to

$$V(0) = e^{-rT} \mathbb{E}^{\mathbb{Q}}[H(S_T)]. \tag{2.8}$$

This representation forms the foundation for both analytical and numerical methods in option pricing and underpins all models and pricing methods considered in this thesis.

2.3. Black-Scholes model

The Black–Scholes model is the classical framework for option pricing. It is based on the assumption of constant volatility and serves as a fundamental reference model in option pricing theory. The model is built on the following assumptions:

- The underlying asset price $(S_t)_{t\geq 0}$ follows a geometric Brownian motion, as defined in (2.3);
- The volatility $\sigma > 0$ is constant;
- The risk-free interest rate r is constant;
- Markets are frictionless (no transaction costs or taxes);
- Trading is continuous and arbitrage opportunities do not exist.

Under these assumptions, the absence of arbitrage implies that the option price must satisfy the Black–Scholes partial differential equation (PDE). Let $V(t,S)$ denote the value of a European option with payoff $H(S)$ at maturity T. The option value V must satisfy the parabolic PDE

$$\frac{\partial V}{\partial t} + rS\frac{\partial V}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} - rV = 0, \tag{2.9}$$

subject to the terminal condition

$$V(T,S) = H(S). \tag{2.10}$$

To solve this PDE and arrive at an analytical pricing formula, we make use of the Feynman–Kac theorem. This theorem establishes a connection between parabolic PDEs and conditional expectations under stochastic dynamics.

Theorem 1 (Feynman–Kac)**.** *Consider the partial differential equation*

$$\frac{\partial V}{\partial t} + rS\frac{\partial V}{\partial S} + \frac{1}{2}\sigma(t,S)^2 \frac{\partial^2 V}{\partial S^2} - rV = 0, \tag{2.11}$$

defined for $S \in \mathbb{R}_+$ and $t \in [0,T]$, subject to the terminal condition $V(T,S) = H(S)$. Here, $\sigma(t,S)$ is a given function and r is constant. The solution $V(t,S)$ admits the stochastic representation

$$V(t,S) = e^{-r(T-t)}\mathbb{E}^{\mathbb{Q}}[H(S_T) \mid \mathcal{F}_t], \tag{2.12}$$

where the underlying process S_t evolves under the risk-neutral measure $\mathbb{Q}$ according to

$$dS_t = rS_t\,dt + \sigma(t,S_t)\,dW_t^{\mathbb{Q}}. \tag{2.13}$$

A full proof of Theorem 1 can be found in [43]. The Feynman–Kac theorem plays a dual role. It not only provides a probabilistic representation of solutions to pricing PDEs, but also allows one to derive such PDEs from the underlying stochastic dynamics.

By applying Theorem 1 to the Black–Scholes PDE (2.9), the option value can be expressed as a discounted expected payoff under the risk-neutral measure $\mathbb{Q}$, where the underlying asset follows (2.13) with constant volatility. Since the coefficients r and σ are constant, the SDE admits an explicit solution, implying that S_T is lognormally distributed under $\mathbb{Q}$. Substituting this distribution into the risk-neutral expectation and evaluating the resulting integral yields the closed-form Black–Scholes formula for a European call option:

$$C(t,S_t) = S_t\Phi(d_1) - Ke^{-r(T-t)}\Phi(d_2), \tag{2.14}$$

where

$$d_1 = \frac{\log\left(\frac{S_t}{K}\right) + \left(r + \frac{1}{2}\sigma^2\right)(T-t)}{\sigma\sqrt{T-t}}, \tag{2.15}$$

$$d_2 = d_1 - \sigma\sqrt{T-t}, \tag{2.16}$$

and Φ denotes the cumulative distribution function of the standard normal distribution. The corresponding put price follows from the put–call parity [46] and is given by:

$$P(t,S_t) = Ke^{-r(T-t)}\Phi(-d_2) - S_t\Phi(-d_1). \tag{2.17}$$

While the Black–Scholes model provides a tractable and analytically convenient method for option pricing, several of its underlying assumptions are inconsistent with observed market behaviour. In particular, the assumption of constant volatility is contradicted by market data: option prices imply different volatility levels depending on the strike and maturity, a phenomenon referred to as the *volatility smile* or *volatility surface*.

More generally, empirical asset returns exhibit features such as skewness, heavy tails, and volatility clustering, none of which are captured by the Black–Scholes framework. Additionally, assumptions such as continuous trading and frictionless markets are only approximations of reality. Despite these limitations, the Black–Scholes model remains widely used in practice. This is largely due to its simplicity, computational efficiency, and the fact that it provides a common benchmark for quoting and comparing option prices via implied volatility. As such, it serves as a standard reference model, even when more sophisticated models are employed for pricing and risk management.

These limitations motivate the development of stochastic volatility models, which allow volatility to evolve dynamically over time.

2.4. Heston Model

As discussed in the previous section, the assumption of constant volatility in the Black–Scholes model is inconsistent with observed market data. A natural extension is therefore to model volatility as a stochastic process via its variance. We consider the Heston model, which introduces a stochastic variance process while retaining a favourable structure for efficient pricing. In particular, the model belongs to the class of affine stochastic volatility models, which enables semi-analytical pricing via characteristic functions.

2.4.1. Stochastic Dynamics and the Variance Process

The Heston model specifies the dynamics of the variance process $v_t = \sigma_t^2$ rather than the volatility itself. Under the risk-neutral measure $\mathbb{Q}$, the joint dynamics of the asset price S_t and its instantaneous variance v_t are governed by the following system of SDEs:

$$dS_t = rS_t\,dt + \sqrt{v_t}S_t\,dW_t^S, \quad S_0 > 0, \tag{2.18}$$

$$dv_t = \kappa(\bar{v} - v_t)\,dt + \gamma\sqrt{v_t}\,dW_t^v, \quad v_0 > 0, \tag{2.19}$$

where $(W_t^S)_{t\geq 0}$ and $(W_t^v)_{t\geq 0}$ are two standard Brownian motions under $\mathbb{Q}$ that are correlated with constant correlation $\rho \in [-1, 1]$, such that $dW_t^S dW_t^v = \rho dt$.

The parameters of the variance process have the following interpretations:

- $v_0 > 0$: initial level of the variance ($t = 0$);
- $\bar{v} > 0$: long-term mean level of the variance;
- $\kappa > 0$: rate of mean reversion towards $\bar{v}$;
- $\gamma > 0$: volatility of variance (or “volatility of volatility”).

The variance process (2.19) is a Cox–Ingersoll–Ross (CIR) process [8]. A property of the CIR process is that the presence of the square root term $\sqrt{v_t}$ ensures that the diffusion coefficient disappears as v_t approaches zero, which prevents the variance from becoming negative. However, to guarantee strict positivity, the parameters must satisfy the *Feller condition*:

$$2\kappa\bar{v} > \gamma^2. \tag{2.20}$$

When the Feller condition holds, the upward drift $\kappa\bar{v}$ at the origin pushes the process away from zero before the stochastic fluctuations $\gamma\sqrt{v_t}dW_t^v$ can drag it down to zero. If this condition is violated, the variance process can reach zero; although it remains non-negative. In practice, estimating parameters from market data often yields a violated Feller condition.

For future reference, we define the *Feller ratio* ϕ as

$$\phi = \frac{2\kappa\bar{v}}{\gamma^2}, \tag{2.21}$$

and we say the Feller condition is satisfied if $\phi > 1$, and violated if $\phi \leq 1$.

2.4.2. Derivation of the Heston PDE

We follow the derivation from [34] to construct the Heston PDE for a single asset.

Let the dynamics be given as in (2.18)-(2.19). Using Itô's lemma for multidimensional processes [34, eq 7.10], we can determine the dynamics of dV:

$$dV = \left(\frac{\partial V}{\partial t} + rS\frac{\partial V}{\partial S} + \kappa(\bar{v} - v)\frac{\partial V}{\partial v} + \frac{1}{2}vS^2\frac{\partial^2 V}{\partial S^2} + \rho\gamma vS\frac{\partial^2 V}{\partial S \partial v} + \frac{1}{2}\gamma^2 v\frac{\partial^2 V}{\partial v^2} \right) dt + \sqrt{v}S\frac{\partial V}{\partial S}\, dW_t^S + \gamma\sqrt{v}\frac{\partial V}{\partial v}\, dW_t^v. \tag{2.22}$$

To construct the Heston PDE, we use the Feynman-Kac theorem (Theorem 1) and represent the option price as a risk-neutral discounted conditional expectation

$$V(t, S, v) = e^{rt}\mathbb{E}^{\mathbb{Q}}\left[e^{-rT}V(T, S, v)|\mathcal{F}_t\right]. \tag{2.23}$$

Denote with $M_t := e^{rt}$; then $dM_t = rM_t dt$. Dividing (2.23) by M_t yields

$$\frac{V(t, S, v)}{M_t} = \mathbb{E}^{\mathbb{Q}}\left[\frac{V(T, S, v)}{M_T}\middle|\mathcal{F}_t\right]. \tag{2.24}$$

We have seen in Section 2.2.3 that, under the risk-neutral measure $\mathbb{Q}$, the right-hand side of (2.24) should be a martingale. Using the product rule for stochastic calculus, the dynamics of the left-hand side of (2.24) equal

$$d\left(\frac{V}{M}\right) = \frac{1}{M}(dV - rV dt). \tag{2.25}$$

For (2.25) to be a martingale, the drift terms must equal zero. Thus, after substitution of (2.22) and multiplication with M, we obtain

$$\frac{\partial V}{\partial t} + rS\frac{\partial V}{\partial S} + \kappa(\bar{v} - v)\frac{\partial V}{\partial v} + \frac{1}{2}vS^2\frac{\partial^2 V}{\partial S^2} + \rho\gamma vS\frac{\partial^2 V}{\partial S \partial v} + \frac{1}{2}\gamma^2 v\frac{\partial^2 V}{\partial v^2} - rV = 0, \tag{2.26}$$

which is the Heston PDE, to be solved with terminal condition $V(T, S, v) = H(S_T)$. Unlike the Black–Scholes PDE, this is a two-dimensional parabolic equation. While it does not admit a closed-form solution in terms of elementary functions, the affine structure of the model allows for semi-analytical pricing via characteristic functions. The details of this procedure are discussed in Section 2.6.

2.5. Rough Heston model

The rough Heston model extends the classical Heston framework by incorporating memory and fractional scaling into the variance dynamics, thereby capturing the empirically observed roughness of volatility. This section is structured as follows. We first discuss the empirical motivation for rough volatility models. We then introduce stochastic Volterra dynamics and the fractional kernel that underlies the model. Finally, we present the rough Heston model and its key properties.

2.5.1. Motivation

A central assumption underlying classical stochastic volatility models, such as the Heston model, is that volatility evolves as a Markovian diffusion process driven by a standard Brownian motion. As a consequence, the resulting volatility paths inherit the regularity properties of Brownian motion and are therefore relatively smooth in time. While this assumption facilitates efficient pricing, it does not align well with empirical observations.

Empirical evidence indicates that volatility exhibits a much higher degree of irregularity than predicted by diffusion-based models. In particular, volatility time series display pronounced fluctuations at very fine time scales, suggesting that their sample paths are significantly rougher than those of standard Brownian motion. This phenomenon was rigorously documented in [13], where the authors analyse high-frequency data and study the scaling behaviour of log-volatility increments. Their findings indicate that log-volatility exhibits scaling behaviour consistent with a fractional process with Hurst parameter H significantly below $1/2$, typically around $H \approx 0.1$. Contrary to standard Brownian motion, which

corresponds to $H = 1/2$, such values imply that volatility paths are much more irregular and exhibit substantial local fluctuations.

Mathematically, the Hurst parameter governs the regularity of a stochastic process. Processes with $H < 1/2$ are characterised by negatively correlated increments and possess sample paths that are $(H - \epsilon)$-Hölder continuous for any $\epsilon > 0$. As a result, they are substantially rougher than Brownian paths. Importantly, this roughness is most pronounced at short time scales, precisely where classical models tend to perform poorly.

Since diffusion-based models inherently produce paths with Brownian regularity, they are unable to reproduce the observed roughness of volatility, regardless of parameter choice. Consequently, extending classical stochastic volatility models requires a fundamental modification of the underlying stochastic dynamics. A natural approach is to move beyond diffusion processes and introduce dynamics that incorporate memory and fractional scaling [6]. This leads to rough volatility models, in which the variance process is driven by fractional (Volterra-type) dynamics. The rough Heston model extends the classical Heston model by embedding these features into the variance process.

2.5.2. Volterra Dynamics and Fractional Kernel

We consider stochastic processes with memory through a Volterra-type structure. In this setting, the evolution of the variance process depends on its entire past via a convolution with a deterministic kernel function. A general form of a *stochastic Volterra equation* [20] can be written as

$$v_t = v_0 + \int_0^t K(t-s)\, b(v_s)\, ds + \int_0^t K(t-s)\, \sigma(v_s)\, dW_s, \tag{2.27}$$

where K is a deterministic kernel and $b(\cdot)$ and $\sigma(\cdot)$ denote drift and diffusion functions, respectively.

The choice of kernel plays an important role in shaping the behaviour of the process. In particular, it determines the extent to which recent or distant past values contribute to the present. If the kernel places more weight on recent observations, the process exhibits stronger local fluctuations, whereas smoother kernels lead to more regular behaviour.

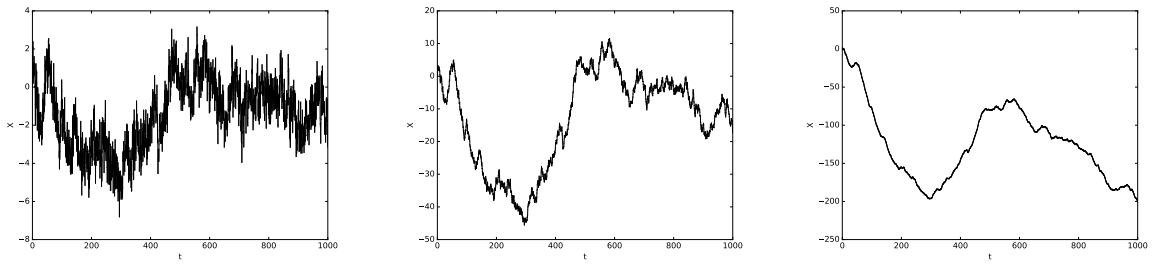
We follow the approach of [10] to construct a suitable kernel, which builds on fractional Brownian motion. To this end, we first formalise its definition [26].

Definition 3 (Fractional Brownian motion)**.** *A fractional Brownian motion $W^H = (W_t^H)_{t\geq 0}$ with Hurst parameter $H \in (0,1)$ is a centred Gaussian process with covariance function*

$$\mathbb{E}[W_t^H W_s^H] = \frac{1}{2}\left(t^{2H} + s^{2H} - |t-s|^{2H}\right), \quad \forall s,t \geq 0. \tag{2.28}$$

For $H = \frac{1}{2}$, this reduces to standard Brownian motion.

The behaviour of fractional Brownian motion is illustrated in Figure 2.2, where sample paths for different values of the Hurst parameter highlight the transition from rough to smooth trajectories.



(a) $H = 0.15$ **(b)** $H = 0.55$ **(c)** $H = 0.95$

Figure 2.2: Sample paths of fractional Brownian motion for different values of the Hurst parameter H, each based on 1000 time steps. Smaller values of H lead to rougher paths, while larger values produce smoother trajectories.

To construct a kernel that induces the rough behaviour observed in volatility, we briefly recall the Mandelbrot-van Ness representation [26]. A fractional Brownian motion $W^H = (W_t^H)_{t\geq 0}$ with Hurst

parameter $H \in (0,1)$ admits the representation

$$W_t^H = \frac{1}{\Gamma\left(H+\frac{1}{2}\right)} \int_{-\infty}^{0} \left[(t-s)^{H-\frac{1}{2}} - (-s)^{H-\frac{1}{2}}\right] dW_s + \frac{1}{\Gamma\left(H+\frac{1}{2}\right)} \int_0^t (t-s)^{H-\frac{1}{2}}\, dW_s, \tag{2.29}$$

where W is a standard Brownian motion and $\Gamma(\cdot)$ denotes the Gamma function.
This representation shows that past Brownian increments are weighted by terms involving $(t-s)^{H-\frac{1}{2}}$, so that their influence depends on how far back in time they occur. In particular, the process

$$X_t := \int_0^t (t-s)^{H-\frac{1}{2}}\, dW_s \tag{2.30}$$

captures the local regularity of the paths. One can show that for $0 \le s < t$,

$$\mathbb{E}\left[|X_t - X_s|^2\right] \le C|t-s|^{2H}, \tag{2.31}$$

for some constant $C > 0$. More generally, since the increments are Gaussian, one obtains moment estimates of the form

$$\mathbb{E}[|X_t - X_s|^p] \le C_p |t-s|^{pH}, \quad p \ge 2. \tag{2.32}$$

By Kolmogorov's continuity theorem [47], there exists a modification of the process (i.e., a version that is almost surely equal at each fixed time) whose sample paths are $(H-\epsilon)$-Hölder continuous for any $\epsilon > 0$. In particular, when $H < 1/2$, the resulting paths are rougher than Brownian motion.

Motivated by this fractional structure, a natural choice for the kernel in the Volterra formulation is given by

$$K(t) = \frac{t^{H-\frac{1}{2}}}{\Gamma(H+\frac{1}{2})}, \quad t > 0. \tag{2.33}$$

For $H < 1/2$, the exponent $H - \frac{1}{2}$ is negative, so $(t-s)^{H-\frac{1}{2}}$ becomes large as $t - s \to 0$. This places strong weight on recent increments, leading to pronounced local fluctuations and hence rough sample paths, consistent with the empirical roughness of volatility discussed earlier.

For $H = \frac{1}{2}$, the fractional kernel reduces to

$$K(t) = \frac{t^0}{\Gamma(1)} = 1.$$

Substituting this kernel into (2.27) removes the memory effect, so that the variance process depends only on its current state and the dynamics reduce to a Markovian SDE. In the rough Heston setting introduced below, this recovers the classical Heston variance dynamics.

The fractional kernel in (2.33) introduces memory and roughness into the variance dynamics. This structure underlies the rough Heston model, which is introduced in the next section.

2.5.3. Rough Heston Dynamics

We now define the rough Heston model, which extends the classical Heston model by introducing memory and roughness into the variance process via the fractional kernel.

Under the risk-neutral measure $\mathbb{Q}$, the asset price S_t and variance process v_t are given by

$$dS_t = rS_t\, dt + \sqrt{v_t} S_t\, dW_t^S, \tag{2.34}$$

$$v_t = v_0 + \int_0^t K(t-s)\,\kappa(\bar{v} - v_s)\, ds + \int_0^t K(t-s)\,\gamma\sqrt{v_s}\, dW_s^v, \tag{2.35}$$

where K is the fractional kernel defined in (2.33), and (W^S, W^v) are Brownian motions with correlation ρ, such that $dW_t^S dW_t^v = \rho dt$.

In contrast to the classical Heston model, the variance process is no longer Markovian, as its evolution depends on its entire past trajectory through the kernel K. This Volterra structure introduces memory into the model and induces rough sample paths when $H < 1/2$. Additionally, this formulation belongs to the class of affine Volterra processes, for which semi-analytical pricing via characteristic functions is available.

The parameters $\kappa > 0$, $\bar{v} > 0$, $\gamma > 0$, and $\rho \in [-1, 1]$ retain the same interpretation as in the classical Heston model: κ controls the speed of mean reversion, $\bar{v}$ the long-term variance level, γ the volatility of volatility, and ρ the correlation between the asset and variance processes. The additional parameter $H \in (0, 1)$ governs the roughness of the variance process, with smaller values of H corresponding to more irregular paths. For $H = \frac{1}{2}$, the variance dynamics simplify to

$$dv_t = \kappa(\bar{v} - v_t)\, dt + \gamma\sqrt{v_t}\, dW_t^v,$$

so that the model reduces to the classical Heston model.

2.6. Numerical Pricing of European Options

In the preceding sections, the Heston and rough Heston models were introduced as stochastic volatility models under the risk-neutral measure. In order to generate prices from these models, one requires a numerical procedure that maps a given parameter vector to a set of option prices. This section discusses the numerical methods used for that purpose. A variety of such methods are available for pricing European options; broadly speaking, these may be divided into three classes.

First, one may solve the pricing PDE numerically, for instance by finite difference or finite element methods. Such approaches are flexible and can handle a range of boundary conditions and payoff structures, but may become computationally demanding when high accuracy is required over large parameter grids.

Second, one may simulate the underlying stochastic dynamics directly by Monte Carlo methods. These are particularly attractive for path-dependent options and high-dimensional models. However, for plain European options, Monte Carlo methods are typically less efficient, since obtaining accurate prices requires a large number of simulated paths.

Third, when the characteristic function of the log-asset price is available or can be approximated efficiently, Fourier-based pricing methods provide an attractive alternative. These methods exploit the relation between option prices and transforms of the risk-neutral density, and often yield highly accurate prices at comparatively low computational cost. Since both the Heston and rough Heston models admit characteristic-function-based pricing representations, Fourier methods are especially suitable in our setting.

In this thesis, European option prices are computed using the COS method [11], which combines efficiency with high accuracy and is well suited for the repeated pricing required in synthetic data generation.

2.6.1. The COS Method

To arrive at the COS method, we follow the derivation in [11]. We start with writing the expectation in (2.23) as a truncated integral $[a, b] \subset \mathbb{R}$ over the probability density function

$$V(t, S) = e^{rt}\mathbb{E}^{\mathbb{Q}}\left[e^{-rT}H(S_T)|\mathcal{F}(t)\right] \approx e^{-r(T-t)}\int_a^b H(y)f(y|x)dy, \tag{2.36}$$

where H denotes the payoff, and x and y denote the state variables at t and T, respectively. Since the density function is not known in closed form, we replace it by its Fourier cosine expansion in y, yielding

$$f(y|x) = \sum_{k=0}^{\infty}{}' A_k(x)\cos\left(k\pi\frac{y-a}{b-a}\right) \tag{2.37}$$

where

$$A_k(x) := \frac{2}{b-a}\int_a^b f(y|x)\cos\left(k\pi\frac{y-a}{b-a}\right)dy, \tag{2.38}$$

after applying a change of variables from the interval $[0, \pi]$ to $[a, b]$. The primed sum denotes that the first term must be multiplied by $\frac{1}{2}$. Substitution into (2.36) then yields

$$V(t, S) \approx e^{-r(T-t)}\int_a^b H(y)\sum_{k=0}^{\infty}{}' A_k(x)\cos\left(k\pi\frac{y-a}{b-a}\right)dy. \tag{2.39}$$

We interchange summation and integration to obtain

$$V(t,S) \approx \frac{1}{2}(b-a)e^{-r(T-t)} \sum_{k=0}^{\infty}{}' A_k(x) V_k, \tag{2.40}$$

with

$$V_k := \frac{2}{b-a} \int_a^b H(y) \cos\left(k\pi \frac{y-a}{b-a}\right) dy. \tag{2.41}$$

Finally, the integrals in $A_k(x)$ are approximated by $F_k(x)$, which can be expressed in terms of the characteristic function φ:

$$F_k(x) = \frac{2}{b-a} \Re\left\{ \varphi\left(\frac{k\pi}{b-a}\right) \exp\left(-i\frac{ka\pi}{b-a}\right)\right\}, \tag{2.42}$$

where $\Re\{\cdot\}$ denotes taking the real part of the expression inside the brackets. After truncating the infinite sum, we obtain

$$V(t,S) \approx e^{-r(T-t)} \sum_{k=0}^{N-1}{}' \Re\left\{ \varphi\left(\frac{k\pi}{b-a}; x\right) \exp\left(-i\frac{ka\pi}{b-a}\right)\right\} V_k. \tag{2.43}$$

For the European call option, we can calculate the coefficients V_k analytically, which is done in [11, Result 3.1]. We state the result below.

$$V_k^{call} = \frac{2}{b-a} \int_0^b K(e^y - 1) \cos\left(k\pi \frac{y-a}{b-a}\right) dy = \frac{2}{b-a} K(\chi_k(0,b) - \psi_k(0,b)), \tag{2.44}$$

with

$$\begin{aligned} \chi_k(c,d) := \frac{1}{1+\left(\frac{k\pi}{b-a}\right)^2} \Bigg[&\cos\left(k\pi\frac{d-a}{b-a}\right) e^d - \cos\left(k\pi\frac{c-a}{b-a}\right) e^c \\ &+ \frac{k\pi}{b-a} \sin\left(k\pi\frac{d-a}{b-a}\right) e^d - \frac{k\pi}{b-a} \sin\left(k\pi\frac{c-a}{b-a}\right) e^c \Bigg], \end{aligned} \tag{2.45}$$

and

$$\psi_k(c,d) := \begin{cases} \left[\sin\left(k\pi\frac{d-a}{b-a}\right) - \sin\left(k\pi\frac{c-a}{b-a}\right)\right] \frac{b-a}{k\pi}, & k \neq 0, \\ (d-c), & k = 0. \end{cases} \tag{2.46}$$

The accuracy of the COS method depends on the choice of the truncation interval $[a,b]$ and the number of cosine terms N. The interval must be sufficiently wide to capture the bulk of the probability mass, while N controls the approximation accuracy of the truncated series. In [11], the truncation range for the Heston model is suggested based on the first two cumulants of $\log(S_T/K)$, given by

$$[a,b] = \left[c_1 - L\sqrt{|c_2|},\ c_1 + L\sqrt{|c_2|}\right], \tag{2.47}$$

with scaling parameter $L > 0$. The absolute value of c_2 is used, as this cumulant may become negative for sets of Heston parameters that do not satisfy the Feller condition (2.20). This truncation strategy is adopted throughout this thesis.

Thus, once the characteristic function is available, the COS method enables efficient and accurate pricing of European options under the corresponding model.

2.6.2. Characteristic Function of the Heston Model

A key advantage of the Heston model is that the characteristic function of the log-asset price admits a semi-closed-form expression due to the affine structure of the model [18]. Let $X_T = \log S_T$. Then, conditional on $X_t = x$ and $v_t = v$, the characteristic function is given by

$$\varphi_{\text{Heston}}(u; T-t, x, v) := \mathbb{E}^{\mathbb{Q}}\left[e^{iuX_T} \mid X_t = x, v_t = v\right] = \exp\left(C(u,\tau) + D(u,\tau)v + iux\right), \tag{2.48}$$

where $\tau = T - t$, and the functions C and D are defined as

$$d(u) = \sqrt{(\rho\gamma iu - \kappa)^2 + \gamma^2(iu + u^2)}, \tag{2.49}$$

$$g(u) = \frac{\kappa - \rho\gamma iu - d(u)}{\kappa - \rho\gamma iu + d(u)}, \tag{2.50}$$

$$C(u,\tau) = iru\tau + \frac{\kappa\bar{v}}{\gamma^2}\left[(\kappa - \rho\gamma iu - d(u))\tau - 2\log\left(\frac{1 - g(u)e^{-d(u)\tau}}{1 - g(u)}\right)\right], \tag{2.51}$$

$$D(u,\tau) = \frac{\kappa - \rho\gamma iu - d(u)}{\gamma^2} \cdot \frac{1 - e^{-d(u)\tau}}{1 - g(u)e^{-d(u)\tau}}. \tag{2.52}$$

From a numerical perspective, the evaluation of the Heston characteristic function is straightforward. For given u, the quantities $d(u)$, $g(u)$, $C(u,\tau)$, and $D(u,\tau)$ can be computed directly using elementary operations. In particular, the characteristic function is evaluated at the Fourier frequencies required in the COS expansion (2.43). Consequently, the characteristic function can be evaluated efficiently and with high numerical stability, enabling fast option pricing under the Heston model.

2.6.3. Characteristic Function of the Rough Heston Model

While the characteristic function of the Heston model is straightforward to evaluate, this is no longer the case for the rough Heston model. The derivation is involved, and we refer to [10] for a rigorous treatment.

In the rough Heston model, the characteristic function admits a representation in terms of the solution to a fractional Riccati equation. We state the main result below. To this end, define the fractional integral of order $r \in (0,1]$ of a function f as

$$I^r f(t) = \frac{1}{\Gamma(r)} \int_0^t (t-s)^{r-1} f(s) ds, \tag{2.53}$$

whenever the integral exists, and the fractional derivative of order $r \in [0,1)$ as

$$D^r f(t) = \frac{1}{\Gamma(1-r)} \frac{d}{dt} \int_0^t (t-s)^{-r} f(s) ds, \tag{2.54}$$

whenever it exists. The following result corresponds to Theorem 4.1 in [10].

Theorem 2. *Consider the rough Heston model* (2.34)-(2.35) *with correlation* $\rho \in (-1/\sqrt{2}, 1/\sqrt{2}]$. *For all* $t \geq 0$, *we have*

$$\varphi_{\mathrm{rHeston}}(u,t) = \exp\left(\kappa\bar{v} I^1 h(u,t) + v_0 I^{1-\alpha} h(u,t)\right), \tag{2.55}$$

where h *solves the fractional Riccati equation*

$$D^\alpha h(u,t) = \frac{1}{2}(-iu - u^2) + (\rho\gamma iu - \kappa)h(u,t) + \frac{\gamma^2}{2}h^2(u,t), \qquad I^{1-\alpha}h(u,0) = 0, \tag{2.56}$$

which admits a unique continuous solution.

Here, the fractional order is given by $\alpha = H + \frac{1}{2}$, where $H \in (0, \frac{1}{2})$ denotes the Hurst parameter. For $\alpha = 1$, the fractional operators reduce to their classical counterparts and the standard Heston characteristic function is recovered. For $\alpha < 1$, the Riccati equation becomes fractional, reflecting the non-Markovian structure of the variance process. As a consequence, no closed-form solution is available and the evaluation of the characteristic function requires the numerical solution of a fractional Riccati equation for each Fourier argument. Such equations are solved numerically using a fractional Adams–Bashforth–Moulton predictor-corrector scheme, as proposed in [9] and applied in [10]. We take a closer look at this scheme below.

The fractional Riccati equation can be written as

$$D^\alpha h(u,t) = F(u, h(u,t)), \qquad I^{1-\alpha}h(u,0) = 0, \tag{2.57}$$

where

$$F(u,x) = \frac{1}{2}(-iu - u^2) + (\rho\gamma iu - \kappa)x + \frac{\gamma^2}{2}x^2. \tag{2.58}$$

An equivalent formulation is obtained by rewriting (2.56) as the following Volterra integral equation:

$$h(u,t) = \frac{1}{\Gamma(\alpha)} \int_0^t (t-s)^{\alpha-1} F(u, h(u,s))\, ds. \tag{2.59}$$

To solve this equation numerically, we discretise time using a uniform grid $t_k = k\Delta$, $k = 0, \ldots, N$. The scheme then proceeds iteratively. Suppose that approximations $\hat{h}(u,t_j)$ have been computed for $j = 0, \ldots, k$. Then:

Predictor step: An explicit approximation $\hat{h}^P(u,t_{k+1})$ is computed using

$$\hat{h}^P(u,t_{k+1}) = \sum_{0 \le j \le k} b_{j,k+1} F(u, \hat{h}(u,t_j)), \tag{2.60}$$

where the weights $b_{j,k+1}$ are given by

$$b_{j,k+1} = \frac{\Delta^\alpha}{\Gamma(\alpha+1)} \left((k-j+1)^\alpha - (k-j)^\alpha \right), \qquad 0 \le j \le k. \tag{2.61}$$

Subsequently, this approximation is used to evaluate (2.58):

$$F(u, \hat{h}^P(u,t_{k+1})) = \frac{1}{2}(-iu - u^2) + (\rho\gamma iu - \kappa)\hat{h}^P(u,t_{k+1}) + \frac{\gamma^2}{2}(\hat{h}^P(u,t_{k+1}))^2. \tag{2.62}$$

Corrector step: The predicted value is then used to obtain a refined approximation

$$\hat{h}(u,t_{k+1}) = \sum_{0 \le j \le k} a_{j,k+1} F(u, \hat{h}(u,t_j)) + a_{k+1,k+1} F(u, \hat{h}^P(u,t_{k+1})), \tag{2.63}$$

where the weights $a_{j,k+1}$ are defined by

$$a_{0,k+1} = \frac{\Delta^\alpha}{\Gamma(\alpha+2)} \left(k^{\alpha+1} - (k-\alpha)(k+1)^\alpha \right), \qquad j = 0 \tag{2.64}$$

$$a_{j,k+1} = \frac{\Delta^\alpha}{\Gamma(\alpha+2)} \left((k-j+2)^{\alpha+1} + (k-j)^{\alpha+1} - 2(k-j+1)^{\alpha+1} \right), \qquad 1 \le j \le k \tag{2.65}$$

$$a_{k+1,k+1} = \frac{\Delta^\alpha}{\Gamma(\alpha+2)}, \qquad j = k+1. \tag{2.66}$$

This approximation is then again used to evaluate (2.58):

$$F(u, \hat{h}(u,t_{k+1})) = \frac{1}{2}(-iu - u^2) + (\rho\gamma iu - \kappa)\hat{h}(u,t_{k+1}) + \frac{\gamma^2}{2}(\hat{h}(u,t_{k+1}))^2. \tag{2.67}$$

Starting from the initial condition $\hat{h}(u,0) = 0$, this procedure yields an approximation of the solution $h(u,t)$ over the entire time grid. Theoretical guarantees for the convergence of this scheme are provided in [23]. In particular, for any fixed $t > 0$ and $u \in \mathbb{R}$,

$$\max_{t_j \in [0,t]} |\hat{h}(u,t_j) - h(u,t_j)| = o(\Delta) \tag{2.68}$$

and

$$\max_{t_j \in [\varepsilon,t]} |\hat{h}(u,t_j) - h(u,t_j)| = o(\Delta^{2-\alpha}), \tag{2.69}$$

for any $\varepsilon > 0$.

In addition to solving the fractional Riccati equation, the evaluation of the characteristic function involves computing ordinary integrals of the form (2.53). These integrals are approximated numerically using the trapezoidal rule on the same time grid:

$$I^1 h(u, t_N) \approx \int_0^{t_N} \hat{h}(u, s)\, ds \approx \Delta \left(\frac{1}{2}\hat{h}(u, t_0) + \sum_{j=1}^{N-1} \hat{h}(u, t_j) + \frac{1}{2}\hat{h}(u, t_N) \right). \tag{2.70}$$

Finally, the numerical approximation $\hat{h}(u, t)$ is substituted into the representation of the characteristic function. The resulting values are evaluated at the Fourier frequencies required in the COS method to obtain option prices.

2.7. Implied Volatility

In financial markets, option prices are commonly quoted in terms of implied volatility rather than absolute price levels. The implied volatility is defined as the value of the volatility parameter in the Black–Scholes model that reproduces a given option price.

Let V^{mkt} denote the observed market price of a European option. The implied volatility σ_{imp} is defined as the solution to

$$V^{\mathrm{BS}}(S_0, K, r, T, \sigma_{\mathrm{imp}}) = V^{\mathrm{mkt}}, \tag{2.71}$$

where V^{BS} denotes the Black–Scholes pricing function.

Equivalently, implied volatility is obtained by solving for the root in

$$f(\sigma) := V^{\mathrm{BS}}(\sigma) - V^{\mathrm{mkt}} = 0. \tag{2.72}$$

Since no closed-form solution exists, implied volatility must be computed numerically. This can be achieved either through iterative root-finding methods or through dedicated approximation techniques. In this section, we first briefly discuss standard root-finding approaches, and subsequently introduce the method of rational approximations proposed by Jäckel [21], which is used throughout this thesis. Finally, we discuss how model parameters influence the shape of the implied volatility surface.

2.7.1. Root-Finding Methods

The computation of implied volatility can be formulated as a one-dimensional root-finding problem, as in (2.72). A natural approach is therefore to apply iterative numerical methods to solve for the root of the function $f(\sigma)$.

A widely used method is the Newton–Raphson method, which constructs a sequence $(\sigma_n)_{n\geq 0}$ via

$$\sigma_{n+1} = \sigma_n - \frac{f(\sigma_n)}{f'(\sigma_n)}, \tag{2.73}$$

where

$$f'(\sigma) = \frac{\partial V^{\mathrm{BS}}(S_0, K, r, T, \sigma)}{\partial \sigma} = K e^{-r(T-t_0)} \Phi(d_2) \sqrt{T - t_0}, \tag{2.74}$$

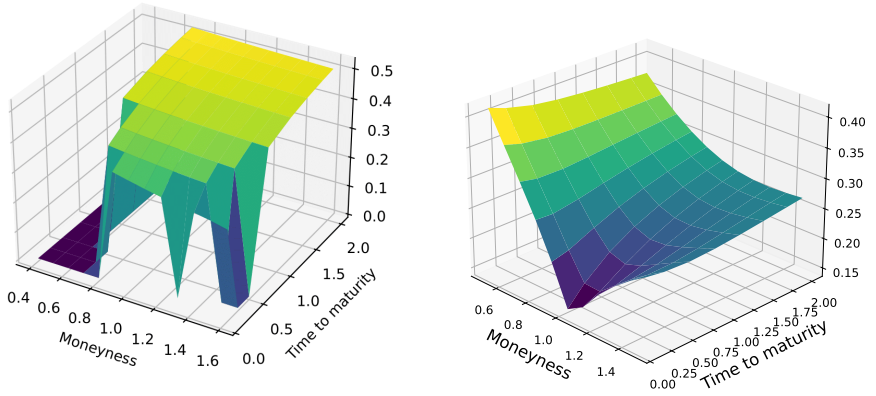
and d_2 is as in (2.16). The derivative $f'(\sigma)$ corresponds to the *Vega* of the option.

When initialised sufficiently close to the solution, Newton–Raphson exhibits quadratic convergence and is therefore highly efficient. However, the method may suffer from instability when the initial guess is poor or when the Vega is small, which can occur for deep ITM or OTM options or for very short maturities. In such cases, the iterations may diverge or converge slowly.

To improve robustness, hybrid methods such as Brent's method can be employed. These methods guarantee convergence by maintaining an interval containing the root, but typically require more function evaluations and are therefore relatively more expensive.

In practice, the performance of Newton–Raphson depends heavily on the choice of the initial guess and the stability of the iteration. In our experiments, where implied volatilities are computed across a full grid of strikes and maturities, a fixed initial guess was used across the entire grid without adaptive initialisation or safeguarding. Under this setup, the method may fail to converge or produce invalid values in regions where the Vega is small. This behaviour is illustrated in Figure 2.3, where the resulting implied volatility surface exhibits discontinuities and non-smooth regions. While more sophisticated

strategies, such as adaptive initialisation or safeguarded Newton schemes, can improve stability, they introduce additional complexity and do not fully eliminate numerical issues across all regions of the parameter space.



(a) Newton (fixed initial guess)

(b) Robust method

Figure 2.3: Comparison of implied volatility surfaces. The surface obtained using Newton's method with a fixed initial guess (left) exhibits discontinuities due to convergence failures. The surface obtained using a robust method (right) is smooth and stable.

These limitations are particularly relevant in large-scale settings, where implied volatilities must be computed over extensive grids in an efficient and robust manner. This motivates the use of an alternative method for implied volatility computation, which is discussed next.

2.7.2. Jäckel's Method of Rational Approximations

To address the limitations of standard root-finding techniques, we employ the method of rational approximations proposed by Jäckel [21], which is specifically designed for efficient and robust inversion of the Black–Scholes formula. The method is based on transforming the implied volatility problem into a normalised setting.

In particular, it introduces the log-forward moneyness

$$x := \log\left(\frac{F}{K}\right), \tag{2.75}$$

and the total standard deviation

$$\sigma := \hat{\sigma}\sqrt{T}, \tag{2.76}$$

where $F = S_0 e^{rT}$ denotes the forward price under the risk-neutral measure, and $\hat{\sigma}$ is the Black–Scholes implied volatility. In this formulation, the option price can be expressed as a function of x and σ, and the implied volatility problem reduces to inverting the normalised Black–Scholes pricing function.

Within this framework, the inversion is performed using carefully constructed rational approximations. These approximations are designed to achieve high accuracy across different regions of the parameter space, including deep ITM and OTM options and short maturities, where standard iterative methods may struggle.

In practice, the method provides highly accurate initial estimates of the implied volatility, which are subsequently refined using a small number of Newton iterations; one or two iterations are mostly enough to achieve machine precision. As a result, convergence is achieved rapidly and reliably, typically within a fixed and small number of operations.

Compared to standard root-finding techniques, Jäckel's method combines the efficiency of analytical approximations with the accuracy of iterative refinement. This yields a method that is both computationally efficient and numerically stable across a wide range of input parameters. An optimised open-source implementation of Jäckel's method is available in the Python library `pyvollib` [36], which enables fast and reliable large-scale evaluation. For this reason, it was adopted as the primary method for computing implied volatility surfaces in our experiments.

2.7.3. Parameter Influence on Implied Volatility

The parameters of stochastic volatility models determine the shape of the implied volatility surface across strikes and maturities. Understanding these relationships is essential for both model calibration and the interpretation of learned mappings in data-driven approaches. We discuss the effects of each parameter in the Heston and rough Heston model, following [34].

Initial variance v_0

The initial variance primarily affects short maturities. Since v_0 governs the instantaneous level of variance at time $t = 0$, its influence is strongest for options with small time to maturity, where there is limited time for the variance process to revert to its long-term level. An increase in v_0 raises the implied volatility level across all strikes for short maturities, with the most pronounced effect near ATM options.

Long-term mean of variance $\bar{v}$

The long-term mean of variance determines the level towards which the variance process mean-reverts. Its influence is more pronounced for longer maturities, where the effect of the initial variance v_0 diminishes. Increasing $\bar{v}$ raises the implied volatility level across the surface, particularly for long-maturity ATM options, thereby shifting the entire surface upwards at longer time horizons.

Mean reversion speed κ

The parameter κ controls the rate at which the variance process reverts to its long-term mean. A higher value of κ causes the variance process to revert more quickly to its long-term level $\bar{v}$. Consequently, the impact of v_0 is confined to short maturities, while implied volatilities at longer maturities are primarily determined by $\bar{v}$. Increasing κ flattens the term structure of implied volatility, reducing differences between short- and long-maturity ATM implied volatilities.

Volatility of volatility γ

The parameter γ determines the volatility of variance. Larger values of γ increase the uncertainty in future variance, which amplifies the curvature of the implied volatility smile. In particular, higher γ increases implied volatilities in the wings (both low and high strikes), while having a smaller relative effect near ATM. This results in a more pronounced convex shape of the smile, especially at short maturities.

Correlation ρ

The correlation between the asset price and variance processes plays a key role in shaping the asymmetry of the implied volatility smile. Negative values of ρ induce a downward-sloping skew, where implied volatilities are higher for low-strike options and lower for high-strike options. This effect arises from the leverage effect: negative price movements are associated with increases in volatility. Consequently, scenarios corresponding to low asset prices are associated with higher variance, resulting in higher implied volatilities for such options. Conversely, positive values of ρ lead to an upward-sloping skew.

Hurst parameter H (rough Heston)

In the rough Heston model, the Hurst parameter H controls the roughness of the variance process. Smaller values of H correspond to rougher paths and stronger short-term variability. This results in steeper implied volatility skews at short maturities, particularly in the wings, reflecting the increased sensitivity of short-dated options to rapid fluctuations in volatility. As H increases, the variance process becomes smoother, and the short-term skew becomes less pronounced.

Overall, the parameters of the (rough) Heston model influence distinct regions of the implied volatility surface. While v_0 and H primarily affect short maturities, $\bar{v}$ governs the long-term level, κ controls the speed of transition between these regimes, γ determines the curvature of the smile through its impact on the wings, and ρ drives the asymmetry between low- and high-strike implied volatilities. These relationships provide a useful reference for assessing whether calibration models capture economically meaningful structure in the implied volatility surface.

3

Neural Network-Based Calibration

3.1. Calibration as Inverse Problem

A central requirement for the practical use of any option pricing model is the ability to calibrate its parameters to observed market data. Two aspects are particularly relevant in this context. First, the model must be identifiable from available market information, meaning that sufficient and informative data must exist from which the model parameters can be inferred, or at least reasonably approximated. Second, the calibration procedure must be computationally efficient. Even a financially well-motivated model becomes impractical if calibration is too slow for real-world applications.

In the context of option contracts, market information is typically represented by an implied volatility surface, defined over a grid of strikes (or moneyness levels) and maturities. The calibration problem consists of determining the set of model parameters that best reproduces this observed surface.

To formalise this setting, we follow the approach as in [19]. Let $\mathcal{M} := \mathcal{M}(\mathfrak{p})_{\mathfrak{p}\in\mathcal{P}}$ denote an abstract model with parameters $\mathfrak{p}$ in the set $\mathcal{P} \subset \mathbb{R}^n$, for some $n \in \mathbb{N}$. Thus, the model $\mathcal{M}(\mathfrak{p})$ and the corresponding prices of financial contracts are fully specified by the choice of the parameter combination $\mathfrak{p} \in \mathcal{P}$.

Furthermore, let ζ denote a financial contract, such as a European call option characterised by a strike and maturity. A pricing operator P maps the model and contract to a model-implied quantity, which in our setting corresponds to the implied volatility. That is,

$$P(\mathcal{M}(\mathfrak{p}), \zeta) \tag{3.1}$$

denotes the model-implied volatility of contract ζ under parameters $\mathfrak{p}$.

Let $P^{\mathrm{mkt}}(\zeta)$ denote the corresponding observed market implied volatility. In practice, we observe a finite collection of contracts $\{\zeta_i\}_{i=1}^N$, which correspond to a discretisation of the implied volatility surface across strikes (or moneyness levels) and maturities. Model calibration consists of determining parameters $\mathfrak{p}$ such that the model-implied quantities match the observed market data as closely as possible. Given a suitable metric $\delta(\cdot,\cdot)$, this leads to the optimisation problem

$$\mathfrak{p}^* = \arg\min_{\mathfrak{p}\in\mathcal{P}} \delta\big(\{P(\mathcal{M}(\mathfrak{p}), \zeta_i)\}_{i=1}^N,\ \{P^{\mathrm{mkt}}(\zeta_i)\}_{i=1}^N\big). \tag{3.2}$$

In many practical applications, the metric δ is chosen as a weighted least-squares loss [34], yielding

$$\mathcal{L}(\mathfrak{p}) = \sum_{i=1}^{N} w_i \left(P(\mathcal{M}(\mathfrak{p}), \zeta_i) - P^{\mathrm{mkt}}(\zeta_i)\right)^2, \tag{3.3}$$

where the weights w_i may reflect, for example, liquidity considerations or the relative importance of different regions of the surface.

Rather than viewing calibration purely as an optimisation problem, it is useful to interpret it as an inverse problem. The forward map

$$\mathfrak{p} \mapsto \{P(\mathcal{M}(\mathfrak{p}), \zeta_i)\}_{i=1}^N \tag{3.4}$$

maps model parameters to observable quantities, namely implied volatilities. Calibration seeks to invert this mapping, i.e. to construct a function

$$\Theta : \mathbb{R}^N \to \mathcal{P}, \quad \{P^{\mathrm{mkt}}(\zeta_i)\}_{i=1}^N \mapsto \mathfrak{p}. \tag{3.5}$$

This inverse mapping is generally not available in closed form and must therefore be approximated numerically. In the classical approach, one solves the calibration problem by minimising the difference between model-implied and market-implied volatilities (3.3) using an iterative optimisation scheme. While this formulation is conceptually straightforward, the resulting problem is difficult for several reasons.

A first source of difficulty is the strong nonlinearity of the forward map from parameters to implied volatilities. Option prices depend on the model parameters through stochastic dynamics, and in many practically relevant models these prices are not available in closed form. Instead, they must be obtained through numerical procedures such as Fourier inversion, numerical integration, PDE-based methods, or Monte Carlo simulation. The implied volatilities are then obtained by inverting the Black–Scholes formula. Consequently, the calibration objective is the result of several nested nonlinear transformations, leading to a highly complex optimisation landscape. In particular, the loss function is generally non-convex and may contain multiple local minima, as well as flat or highly anisotropic curvature. As a result, the outcome of calibration may depend strongly on initialisation, the optimiser used, and the stopping criteria.

A second and more structural issue is that calibration is often ill-posed. Ideally, one would like the observed implied volatility surface to determine the parameter vector uniquely. In practice, however, the forward map is often far from injective: different parameter combinations may generate surfaces that are very similar, or even nearly indistinguishable on the observed grid. This means that the inverse problem may admit several parameter vectors that all provide a comparably good fit to the market data. In such a situation, the calibrated parameters are not uniquely identified by the surface alone. This is especially problematic in settings where interpretability is of interest, as non-uniqueness implies that different parameter configurations may provide equally valid explanations of the observed surface.

Closely related to the lack of identifiability is the issue of ill-conditioning. Even when a meaningful solution exists, the inverse mapping from implied volatilities to parameters can be highly sensitive to perturbations in the input. In particular, small changes in the observed surface (arising e.g. from market noise or discretisation) may lead to disproportionately large changes in the calibrated parameters. From an optimisation perspective, this manifests itself in poorly scaled loss landscapes, where certain directions are steep while others are nearly flat. As a result, gradient-based methods may struggle to provide stable updates, and calibration outcomes can become numerically unstable.

A further, conceptually distinct issue is that the information content of the implied volatility surface is not uniformly distributed. Different regions of the surface carry different amounts of information about different parameters. For example, short maturities may be highly informative for certain parameters, while other regions may contribute little to their identification. This implies that calibration is effectively driven by a subset of informative option contracts, while large parts of the surface play only a minor role. Consequently, a good overall fit to the implied volatility surface does not necessarily imply that all parameters are equally well identified.

Finally, the entire procedure is computationally expensive. Each evaluation of the calibration objective requires pricing a full set of option contracts across the grid of strikes or moneyness values and maturities. Since iterative optimisation methods require many such evaluations, the total calibration time can become substantial, particularly for models with expensive pricing methods or when global optimisation methods are used. In applications where recalibration must be performed repeatedly, this quickly becomes a practical bottleneck. As a result, there is often a trade-off between model complexity, numerical accuracy, and calibration speed.

Given these structural and computational challenges, alternative approaches have been proposed to either alleviate or bypass the classical calibration procedure. In particular, neural network-based methods have emerged as a promising direction. Broadly speaking, these approaches can be divided into two classes, depending on whether the neural network is used to approximate the forward or the inverse mapping. A first class of methods aims to accelerate the calibration procedure by approximating the *forward* pricing map. In this setting, a neural network is trained to learn the mapping as in (3.4), thereby replacing the computationally expensive pricing routine with a fast surrogate model. For instance, [19] propose a two-step approach in which the neural network is used to approximate the pricing operator. Calibration is then performed using standard optimisation techniques, but with the neural network providing rapid evaluations of the forward map. This significantly reduces the computational cost, while retaining the classical optimisation-based formulation of the calibration problem.

In contrast, a second class of methods seeks to approximate the *inverse* mapping directly. Instead of solving an optimisation problem for each observed surface, a neural network is trained to learn the

mapping as in (3.5). In this formulation, the entire calibration procedure is learned within a supervised learning framework, as explored for instance in [17] and [38]. Once trained, calibration reduces to a single forward pass through the network, eliminating the need for iterative optimisation.

While this inverse approach offers substantial computational advantages, it is inherently more challenging. In particular, the network must learn to approximate an inverse mapping that is generally ill-posed and potentially non-unique, and must generalise across a high-dimensional space of implied volatility surfaces. As discussed earlier, different parameter configurations may produce very similar surfaces, which implies that the learning problem is intrinsically ambiguous. As a result, the network is required not only to interpolate within the training data, but also to implicitly resolve ambiguities in the inverse mapping.

In this work, we adopt the latter approach. The primary motivation for this choice is that it provides direct access to the learned inverse map Θ, which enables a direct analysis of how the network extracts parameter information from the implied volatility surface. This is particularly relevant in the context of interpretability, where the goal is not only to achieve accurate calibration, but also to understand, using XAI methods, which regions of the surface are most informative for the recovery of model parameters.

3.2. Neural Network Fundamentals

This section introduces the neural network concepts that form the basis for the architectures considered later in this chapter. Since our calibration approach relies on approximating the inverse map from implied volatility surfaces to model parameters with feedforward neural networks, it is useful to first recall the main building blocks of such models. We therefore briefly discuss neural networks as function approximators, the perceptron and multilayer perceptron, the role of activation functions, and the main ingredients of training, including gradient-based optimisation and parameter initialisation. The purpose of this section is not to provide an exhaustive treatment of neural networks, but rather to establish the notation and concepts that will be used in Sections 3.3 and 4.1.

3.2.1. Neural Networks as Function Approximators

Neural networks can be viewed as parametric function approximators. Given an input vector $x \in \mathbb{R}^d$, a neural network defines a mapping

$$f_\theta : \mathbb{R}^d \to \mathbb{R}^n, \tag{3.6}$$

where θ denotes the collection of all trainable parameters of the network. The function $f_\theta(x)$ is constructed as a composition of affine transformations and nonlinear activation functions, resulting in a highly flexible class of functions capable of approximating complex nonlinear relationships.

In a supervised learning setting, the parameters θ are determined from a dataset of input-output pairs $\{(x_i, y_i)\}_{i=1}^N$, by minimising a suitable loss function that measures the discrepancy between predictions $f_\theta(x_i)$ and targets y_i. The resulting model can then be used to predict outputs for previously unseen inputs.

In the context of this work, the neural network is used to approximate the inverse mapping introduced in Section 3.1. Specifically, the input x corresponds to a discretised implied volatility surface, represented as a flattened vector

$$x = \{P^{\mathrm{mkt}}(\zeta_i)\}_{i=1}^d \in \mathbb{R}^d,$$

while the output corresponds to the model parameters

$$y = \mathfrak{p} \in \mathcal{P} \subset \mathbb{R}^n.$$

The neural network therefore provides a parametric approximation of the inverse map

$$\Theta : \mathbb{R}^d \to \mathcal{P}, \quad x \mapsto \mathfrak{p}, \tag{3.7}$$

which is learned from data.

Using this formulation, calibration can be recast as a high-dimensional nonlinear regression problem. Rather than solving an optimisation problem for each observed surface, the goal is to learn a global mapping from implied volatility surfaces to model parameters, which can be evaluated efficiently once trained.

3.2.2. Perceptron

The fundamental building block of a neural network is the perceptron. A perceptron defines a mapping from an input vector $x \in \mathbb{R}^d$ to a scalar output through an affine transformation followed by a nonlinear activation function.

Formally, a perceptron computes

$$z = w^\top x + b, \tag{3.8}$$

where $w \in \mathbb{R}^d$ is a vector of weights and $b \in \mathbb{R}$ is a bias term. The scalar z is then passed through an activation function g, yielding the output

$$y = g(z) = g(w^\top x + b). \tag{3.9}$$

The weights determine how strongly each input component contributes to the output, while the bias allows the affine transformation to be shifted. The activation function introduces nonlinearity, which is essential for the network's ability to approximate complex nonlinear mappings.

A visual representation of a perceptron is provided in Figure 3.1, where each input is multiplied by a corresponding weight, summed together and shifted by the bias, and passed through the activation function to produce the output.

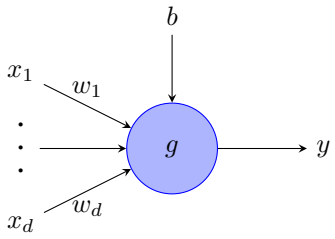


Figure 3.1: Schematic representation of a perceptron. Inputs are linearly combined using weights and a bias term, followed by a nonlinear activation function.

A perceptron thus represents a nonlinear transformation of an affine function. By combining multiple perceptrons, more complex mappings from $\mathbb{R}^d$ to $\mathbb{R}^n$ can be constructed, which forms the basis of feedforward neural networks.

3.2.3. Feedforward Networks and Layers

While a single perceptron maps an input vector to a scalar output, its representational capacity is limited due to its simple structure. By combining multiple perceptrons into layers and stacking such layers, one obtains a *feedforward neural network* capable of approximating highly complex nonlinear functions.

A layer consisting of k perceptrons (or *units*) can be written compactly in vector form as

$$h = g(Wx + b), \tag{3.10}$$

where $W \in \mathbb{R}^{k \times d}$ is a weight matrix, $b \in \mathbb{R}^k$ is a bias vector, and g is applied element-wise. The output $h \in \mathbb{R}^k$ of this transformation is referred to as a *(hidden) representation*.

By stacking multiple such layers, a feedforward neural network defines a mapping of the form

$$f_\theta(x) = g^{(L)}\left(W^{(L)} g^{(L-1)}\left(\cdots g^{(1)}(W^{(1)}x + b^{(1)})\cdots\right) + b^{(L)}\right), \tag{3.11}$$

where L denotes the number of layers, and each layer has its own parameters $(W^{(\ell)}, b^{(\ell)})$. In the case of fully connected layers, this architecture corresponds to a *multilayer perceptron* (MLP). A schematic representation of such a feedforward neural network is provided in Figure 3.2.

The number of layers L is referred to as the depth of the network, while the number of units within a layer determines its width. These two architectural dimensions play fundamentally different roles in the expressive power of the network.

Increasing the depth enables the network to represent complex functions through successive compositions of nonlinear transformations. Each additional layer applies a new transformation to the representation produced by the previous layer, allowing the network to build increasingly informative hidden

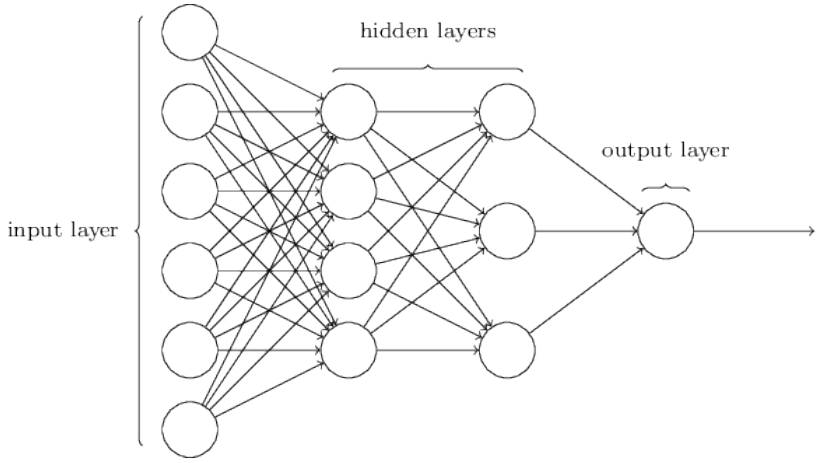


Figure 3.2: Schematic representation of a feedforward neural network with fully connected layers (multilayer perceptron). Each layer applies an affine transformation followed by a nonlinear activation function. Adapted from [4].

representations. From a functional perspective, deeper networks can express certain classes of functions more efficiently, as they naturally capture hierarchical compositions of features [33]. In contrast, increasing the width enhances the representational capacity within a single layer by allowing a larger number of features to be processed in parallel. A wider layer can approximate more complex transformations at a fixed level of abstraction, but without sufficient depth, the network remains limited in its ability to represent compositions of such transformations. Consequently, depth primarily governs the ability to model complex compositions of functions, while width determines the richness of representations at each stage. In practice, both must be balanced to achieve sufficient expressive power without introducing unnecessary redundancy. In particular, deeper architectures can represent certain functions more efficiently than shallow but wide networks, requiring significantly fewer parameters for comparable expressiveness.

In the context of this work, the input $x \in \mathbb{R}^d$ corresponds to a discretised and flattened implied volatility surface, and the network outputs a parameter vector in $\mathbb{R}^n$. The network therefore implements a nonlinear mapping between high-dimensional input and output spaces, constructed through successive compositions of affine transformations and nonlinear activation functions.

3.2.4. Activation Functions

Activation functions introduce nonlinearity into neural networks, enabling them to approximate complex nonlinear mappings. Without nonlinear activation functions, a feedforward network would reduce to a single affine transformation regardless of its depth.

An effective activation function should satisfy several desirable properties. First, it must be nonlinear to increase the expressive power of the network. Second, it should be differentiable almost everywhere to enable gradient-based optimisation. Third, it should promote stable training by avoiding issues such as vanishing or exploding gradients. During training, neural networks are typically optimised using gradient-based methods such as backpropagation, where gradients are propagated backwards through the network via repeated application of the chain rule. If the derivatives of the activation functions are small, these gradients can shrink exponentially across layers. This phenomenon, known as the *vanishing gradient problem*, can severely hinder learning in deep networks, as early layers receive negligible updates. Conversely, large derivatives may lead to exploding gradients, resulting in numerical instability.

Lastly, computational efficiency is important, as activation functions are evaluated repeatedly during both training and inference.

The choice of activation function affects gradient flow, sparsity of representations, and the overall training dynamics of the network. In particular, activation functions with saturating regions may lead to vanishing gradients, while non-saturating functions can improve optimisation but may introduce other trade-offs. Below, we present a number of activation functions which we use in our neural network configurations.

Sigmoid The sigmoid activation function is defined as

$$g(z) = \frac{1}{1 + e^{-z}}, \quad z \in \mathbb{R}, \tag{3.12}$$

with output range $(0, 1)$. It is smooth and has a natural probabilistic interpretation. However, it saturates for large positive or negative inputs, where its derivative approaches zero, leading to vanishing gradients. As a result, its use in hidden layers has largely been replaced by other activation functions. Nevertheless, sigmoid functions remain useful in gating mechanisms (discussed in Section 3.3), where outputs are naturally constrained between zero and one.

ReLU The rectified linear unit (ReLU) is defined as

$$g(z) = \max(0, z), \quad z \in \mathbb{R}, \tag{3.13}$$

with output range $[0, \infty)$. ReLU is computationally efficient and alleviates the vanishing gradient problem for positive inputs (as its derivative remains constant in this region), which facilitates training of deep networks. Additionally, it induces sparse representations, as negative inputs are mapped to zero. A major drawback is the presence of *dead neurons*, where units can become inactive and remain stuck at zero if their inputs are consistently negative. A large number of dead neurons can lead to inefficient use of network capacity.

ELU The exponential linear unit (ELU) is defined as

$$g(z) = \begin{cases} z, & z > 0, \\ \alpha(e^z - 1), & z \leq 0, \end{cases} \quad z \in \mathbb{R}, \tag{3.14}$$

where $\alpha > 0$ is a parameter controlling the saturation level for negative inputs. The output range is $(-\alpha, \infty)$. Compared to ReLU, ELU allows negative outputs and provides a smooth transition for negative inputs, which improves gradient flow and leads to more stable and centred activations. However, it is computationally more expensive due to the exponential term.

Softmax The softmax function is defined for a vector $z \in \mathbb{R}^k$ as

$$g_i(z) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}}, \quad i = 1, \ldots, k, \tag{3.15}$$

mapping $\mathbb{R}^k$ to a normalised vector. Softmax ensures that outputs are non-negative and sum to one, making it suitable for multi-class classification problems. However, it can produce highly peaked outputs for large inputs and is typically used only in the output layer rather than in hidden layers.

In this work, activation functions are selected based on their impact on training stability and their suitability for the architecture. In particular, non-saturating activation functions such as ReLU and ELU are preferred in hidden layers to facilitate gradient flow, while sigmoid functions are used in gated connections to constrain outputs. Although softmax is most commonly used in the output layer for classification tasks, we employed it in the Generalised Highway architecture (see Section 3.3) to enforce a normalised weighting between transformation and carry gates.

3.2.5. Training via Gradient-Based Optimisation

The training of a neural network consists of finding parameters θ that minimise a loss function $\mathcal{L}(\theta)$, which measures the discrepancy between the model predictions and the corresponding target values. The trainable parameters consist of the weight matrices W and bias vectors b. Formally, given a dataset $\{(x_i, y_i)\}_{i=1}^{N}$, the objective is

$$\min_{\theta} \ \mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^{N} J\big(f_\theta(x_i), y_i\big), \tag{3.16}$$

where J is a suitable error metric.

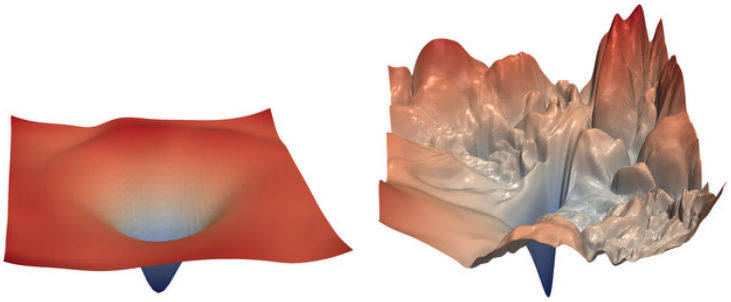


Figure 3.3: Illustration of loss landscapes encountered in neural network training, highlighting the presence of complex, non-convex structures. Adopted from [24].

The resulting optimisation problem is generally high-dimensional and non-convex, due to the composition of nonlinear activation functions across multiple layers. As a result, the loss landscape may contain multiple local minima, saddle points, and flat regions. Examples of such loss landscapes are shown in Figure 3.3, illustrating the presence of complex and non-convex structures.

To solve this optimisation problem, gradient-based methods are employed. The basic idea of *gradient descent* is to iteratively update the parameters in the direction of the negative gradient of the loss function:

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta \mathcal{L}(\theta_t), \tag{3.17}$$

where $\eta > 0$ is the learning rate.

In practice, computing the gradient over the entire dataset is computationally expensive. Therefore, stochastic gradient descent (SGD) is used, where gradients are estimated using a subset (mini-batch) of the data:

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta \mathcal{L}_{\mathcal{B}}(\theta_t), \tag{3.18}$$

where $\mathcal{B} \subset \{1, \dots, N\}$ denotes a mini-batch. This introduces stochasticity into the optimisation process, which can help escape shallow local minima and improve generalisation.

The computation of gradients $\nabla_\theta \mathcal{L}$ is performed efficiently using backpropagation. To see this more explicitly, consider a feedforward network with layerwise *pre-activations*

$$z^{(\ell)} = W^{(\ell)} a^{(\ell-1)} + b^{(\ell)}, \qquad a^{(\ell)} = g^{(\ell)}(z^{(\ell)}), \tag{3.19}$$

for layers $\ell = 1, \dots, L$, with $a^{(0)} = x$. Let $\mathcal{L}$ denote the loss function. Backpropagation computes the gradient of $\mathcal{L}$ with respect to all network parameters by recursively applying the chain rule from the output layer back to the input layer.

Defining the layerwise sensitivity of the loss with respect to the pre-activation at layer ℓ

$$\delta^{(\ell)} = \frac{\partial \mathcal{L}}{\partial z^{(\ell)}}, \tag{3.20}$$

the recursion takes the form

$$\delta^{(\ell)} = \left(W^{(\ell+1)\top} \delta^{(\ell+1)} \right) \odot g'^{(\ell)}(z^{(\ell)}), \tag{3.21}$$

where $\odot$ denotes element-wise multiplication, and $g'^{(\ell)}(z^{(\ell)})$ denotes the element-wise derivative of the activation function evaluated at $z^{(\ell)}$. Once these sensitivities are available, the gradients with respect to the parameters of layer ℓ are given by

$$\frac{\partial \mathcal{L}}{\partial W^{(\ell)}} = \delta^{(\ell)} a^{(\ell-1)\top}, \qquad \frac{\partial \mathcal{L}}{\partial b^{(\ell)}} = \delta^{(\ell)}. \tag{3.22}$$

This recursive structure makes gradient computation highly efficient, as intermediate quantities computed during the forward pass can be reused during the backward pass. As a result, the cost of computing all gradients remains of the same order as that of a single forward evaluation of the network, even for deep architectures.

While SGD provides a simple and effective optimisation scheme, it can suffer from slow convergence and sensitivity to the choice of learning rate. In our experiments, we use the *Adam* optimiser

[22], which adapts the effective step size for each parameter individually based on estimates of first and second moments of the gradients. The update rule for Adam can be written as

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\nabla_\theta \mathcal{L}_\mathcal{B}(\theta_t), \tag{3.23}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2)\big(\nabla_\theta \mathcal{L}_\mathcal{B}(\theta_t)\big)^2, \tag{3.24}$$

$$\theta_{t+1} = \theta_t - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}, \tag{3.25}$$

where m_t and v_t are estimates of the first and second moments, and $\hat{m}_t$, $\hat{v}_t$ denote their bias-corrected versions.

The use of adaptive optimisation methods such as Adam improves convergence speed and stability, particularly in high-dimensional and non-convex settings, making them well-suited for training deeper neural networks in practice.

3.2.6. Initialisation

The initialisation of a neural network refers to the choice of initial values for the trainable parameters

$$\theta = \{W^{(\ell)}, b^{(\ell)}\}_{\ell=1}^{L}$$

prior to training. While these parameters are subsequently optimised using gradient-based methods, their initial values play a crucial role in determining the stability and efficiency of the training procedure.

A central consideration in weight initialisation is the propagation of signals through the network. During the forward pass, activations are repeatedly transformed by weight matrices and nonlinear functions, while during the backward pass, gradients are propagated in the reverse direction. If the weights are not properly scaled, the variance of these signals may either grow or shrink exponentially with depth, leading to exploding or vanishing activations and gradients. This can significantly obstruct learning, particularly in deeper architectures.

To address this, initialisation schemes aim to preserve the variance of activations across layers. This is achieved by scaling the variance of the weight distribution as a function of the number of input and output units of each layer. Two widely used initialisation schemes are the *Glorot* initialisation and the *He* initialisation.

Glorot Initialisation The Glorot initialisation [14] is designed to maintain a balanced variance of activations across layers by accounting for both the number of input and output units. We employ the *Glorot Normal* variant, where weights are drawn from a normal distribution:

$$W_{ij}^{(\ell)} \sim \mathcal{N}\left(0,\ \frac{2}{n_{\text{in}} + n_{\text{out}}}\right), \tag{3.26}$$

where n_{in} and n_{out} denote the number of input and output units of the corresponding layer, respectively. The key idea behind this scaling is to ensure that the variance of the activations remains approximately constant as signals propagate through the network. By balancing the contributions of incoming and outgoing connections, Glorot initialisation stabilises both the forward pass (activations) and the backward pass (gradients). Although originally derived for symmetric activation functions such as sigmoid and $\tanh$, Glorot initialisation is often observed to perform well in relatively shallow networks employing ReLU activations, where issues related to variance propagation are less pronounced.

He Initialisation The He initialisation [16] is specifically designed for non-saturating activation functions such as ReLU and ELU, which alter the variance of activations by reducing or zeroing parts of the input signal. In this case, weights are initialised according to

$$W_{ij}^{(\ell)} \sim \mathcal{N}\left(0,\ \frac{2}{n_{\text{in}}}\right). \tag{3.27}$$

We refer to (3.27) as *He Normal* initialisation. Compared to Glorot Normal initialisation, this scaling increases the variance of the weights to compensate for the loss of signal induced by such activation functions. As a result, He Normal initialisation more effectively preserves activation variance in deeper networks and improves gradient flow during backpropagation. This becomes particularly important in architectures with many layers, where even small deviations in variance can accumulate across layers.

3.3. Network Architectures

In this section, we introduce the neural network architectures considered in this thesis. Our aim is not to study neural network design in full generality, but to compare a small set of architectures that are well suited to the inverse calibration problem considered here. In particular, we are interested in architectures that can learn a stable mapping from an implied volatility surface to the corresponding model parameters, while remaining trainable when configured with depth.

We consider three model classes: a standard multilayer perceptron (MLP), a Highway network, and a Generalised Highway network. The MLP serves as the baseline architecture. The Highway network augments the standard feedforward structure with a learned gating mechanism that regulates how much transformed information is passed on and how much of the input is carried forward unchanged. The Generalised Highway network extends this idea by introducing separate transform and carry gates. As will be discussed below, this added flexibility comes at a cost: if these gates are left unconstrained, the scale of the hidden representations may vary uncontrollably across layers. For this reason, we consider a constrained variant based on joint softmax gating.

Throughout this section, we use the notation as defined in Section 3.2; i.e., $x \in \mathbb{R}^d$ denotes the input to a layer and $y \in \mathbb{R}^k$ its output. The pre-activation is then equal to

$$z = Wx + b, \tag{3.28}$$

where $W \in \mathbb{R}^{k \times d}$ is a weight matrix and $b \in \mathbb{R}^k$ is a bias vector. The vector z will be referred to as the pre-activation. Applying a nonlinear activation function g to z yields the transformed representation

$$H = g(z). \tag{3.29}$$

In architectures with multiple nonlinear components, we distinguish between different activation functions by writing g_H, g_T and g_C for the respective branches.

3.3.1. Multilayer Perceptron

The standard MLP is the most basic architecture considered in our experiments and serves as the baseline against which the gated architectures are compared. Each hidden layer consists of an affine transformation (3.28) followed by an element-wise nonlinearity (3.29). Stacking such layers yields a standard feedforward network.

A schematic illustration of the basic feedforward structure was already given in Section 3.2 (Figure 3.2). The main point is that in a plain MLP, every layer fully transforms its input before passing it onward. There is no explicit mechanism to preserve part of the input. As a consequence, deeper MLPs may become harder to train, since relevant information and gradients must pass through a long sequence of nonlinear transformations [15]. This is one of the motivations for considering Highway-type architectures.

3.3.2. Highway Network

The Highway network [45] modifies the plain feedforward architecture by introducing a learned transform gate. Rather than forcing each layer to completely overwrite its input, the layer is allowed to interpolate between a transformed version of the input and the input itself. This provides a mechanism for adaptive information flow across depth.

Let

$$z_H = W^h x + b^h, \quad H = g_H(z_H) \tag{3.30}$$

denote the transformed branch, where $W^h \in \mathbb{R}^{k \times d}$ and $b^h \in \mathbb{R}^k$. In addition, define the transform gate by

$$z_T = W^t x + b^t, \quad T = \sigma(z_T), \tag{3.31}$$

where $\sigma(\cdot)$ denotes the sigmoid activation function. The output of a Highway layer is then

$$y = H \odot T + x \odot (1 - T), \tag{3.32}$$

where $\odot$ denotes element-wise multiplication.

A schematic representation of this architecture is shown in Figure 3.4. The gate T determines, component-wise, how much of the transformed signal H is passed through and how much of the original

input x is carried forward. If T is close to one, the layer behaves more like a standard nonlinear transformation. If T is close to zero, the layer behaves more like the identity map. In this sense, the Highway layer can be viewed as a learned interpolation between transformation and carry.

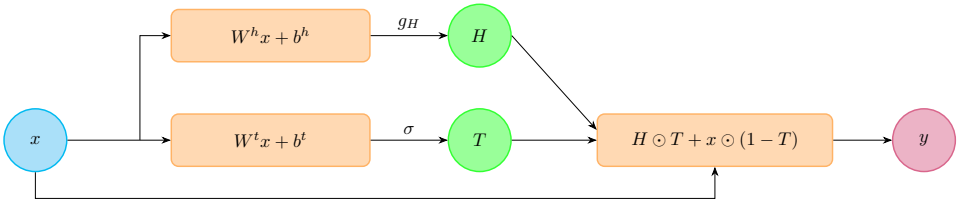


Figure 3.4: Schematic representation of a Highway layer, as defined in (3.32). The gate T regulates the flow of information between the nonlinear transformation and the input, resulting in a convex combination of both.

An important property of this formulation is that the coefficients multiplying H and x sum to one:

$$T + (1 - T) = 1.$$

Hence, the layer output is a convex combination of the transformed input and the skip connection. This prevents the layer from arbitrarily increasing the scale of the input and helps stabilise information propagation through deeper networks. Put differently, the Highway structure introduces an explicit route through which information can pass without repeatedly undergoing full nonlinear distortion.

This property is particularly relevant in the present application. The inverse calibration problem we consider involves learning a mapping from high-dimensional implied volatility surfaces to model parameters. Such a mapping may require a network that is expressive enough to capture complex nonlinear relationships, but also stable enough to avoid degradation when depth is increased. The Highway architecture addresses this by allowing the network to learn when to transform and when to preserve information.

3.3.3. Generalised Highway Network

While the standard Highway network uses a single gate T and defines the carry component implicitly through $1 - T$, one may consider a more flexible formulation in which the transform and carry paths are controlled separately. This leads to the Generalised Highway architecture. Let the transformed branch again be defined by (3.30). In addition, introduce two separate gate pre-activations

$$z_T = W^t x + b^t, \quad z_C = W^c x + b^c, \tag{3.33}$$

from which we compute a transform gate T and a carry gate C. In the most general form, the layer output is written as

$$y = H \odot T + x \odot C. \tag{3.34}$$

This formulation decouples the transform and carry paths; instead of deriving the carry coefficient directly from the transform gate, both are learned seperately. This increases the expressive capacity of the layer, since the network is no longer restricted to the specific relation $C = 1 - T$.

Independent gating A first way to realise this architecture is to let T and C be produced independently from their respective pre-activations, e.g. through separate element-wise activation functions. In abstract form, this can be written as

$$T = g_T(z_T), \quad C = g_C(z_C), \tag{3.35}$$

where g_T and g_C are activation functions applied component-wise. A schematic representation of this unconstrained Generalised Highway layer is shown in Figure 3.5.

At first sight, this appears attractive: the layer can learn not only how much transformed information to pass on, but also how much of the original input to preserve, without hard-coding a complementary relation between the two. However, this flexibility creates an important issue. Since T and C are learned independently, there is in general no constraint on their sum:

$$T + C \neq 1.$$

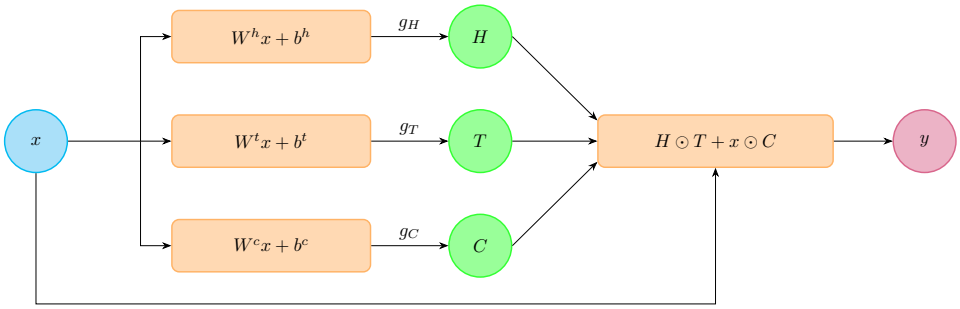


Figure 3.5: Schematic representation of a Generalised Highway layer with independent transform and carry gates. The transformed input $H = g_H(z_H)$ is combined with the input x via two separately learned gates $T = g_T(z_T)$ and $C = g_C(z_C)$, yielding (3.34). Since T and C are computed independently, no constraint is imposed on their sum, and the layer output is not necessarily a convex combination of H and x.

Consequently, the layer representation is no longer a convex combination of the transformed information and the original input. This lack of constraint can be problematic, especially in deeper architectures. If both gates become large simultaneously, the layer may amplify the signal by assigning large weight to both the transformed branch and the carried input. Conversely, if both gates become small, the signal may be reduced too strongly. In either case, the effective scale of the activations may drift as information is propagated through many layers. This undermines one of the principal advantages of Highway-type architectures, namely controlled information flow across depth.

For shallow networks, such behaviour may still be manageable, but as depth increases it becomes increasingly undesirable. A gating mechanism that does not regulate the total contribution of the transform and carry paths can make optimisation less stable and can wipe out the architectural benefit of introducing Highway-type connections in the first place.

Joint softmax gating To address the issue with independent gating, we consider a constrained variant of the Generalised Highway network in which the transform and carry gates are obtained jointly through a softmax normalisation. More precisely, given the pre-activations z_T and z_C, we define

$$(T, C) = \mathrm{softmax}(z_T, z_C), \tag{3.36}$$

meaning that for each component,

$$T_i = \frac{e^{(z_T)_i}}{e^{(z_T)_i} + e^{(z_C)_i}}, \quad C_i = \frac{e^{(z_C)_i}}{e^{(z_T)_i} + e^{(z_C)_i}}. \tag{3.37}$$

A schematic representation of this architecture is shown in Figure 3.6.

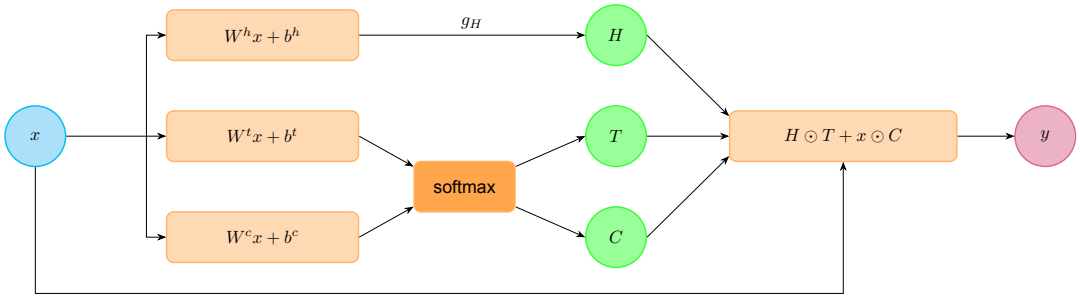


Figure 3.6: Schematic representation of a Generalised Highway layer with joint softmax gating. The transform and carry gates are obtained from pre-activations z_T and z_C via a softmax (3.36). This enforces the constraint $T + C = 1$, so that the output (3.34) is a convex combination of the transformed input and the original input, leading to more stable information propagation across layers.

By construction, this yields the constraint

$$T + C = 1$$

element-wise, such that the layer output (3.34) again becomes a convex combination of the transformed and carried inputs. In this way, the softmax-gated Generalised Highway network retains the conceptual advantage of separate transform and carry pre-activations, while avoiding the instability associated with unconstrained independent gates.

It is worth stressing that this is not equivalent to the standard Highway network. In the Highway formulation, the carry path is defined explicitly as $1 - T$, so that only a single gating quantity is learned. In the softmax-gated Generalised Highway architecture, both paths have their own pre-activations, and the final gates are obtained through *competition* between these signals. While both formulations enforce the constraint $T + C = 1$, they differ fundamentally in their parameterisation: the Highway layer learns a single gate, whereas the Generalised Highway layer learns two competing signals that are normalised jointly.

In the experiments of this thesis, we consider three architectures: a standard MLP, a Highway network as defined in (3.32), and a Generalised Highway network with joint softmax gating as defined in (3.34)–(3.37). Each of these models is trained on the inverse calibration task and their performance is compared in terms of predictive accuracy and stability.

3.4. Data Generation and Preprocessing

The performance of a neural network critically depends on the quality and structure of the training data. In our context, training data is not observed but synthetically generated. This requires careful consideration of both the sampling strategy in parameter space and the preprocessing of the resulting input features.

3.4.1. Synthetic Data Generation

A straightforward approach to sampling model parameters is independent uniform sampling over predefined ranges. While simple, this approach becomes inefficient in higher dimensions, as randomly drawn samples may cluster and leave large regions of the parameter space under-represented. This leads to poor coverage and may result in a network that generalises poorly to unseen regions.

To address this issue, we employ *Latin hypercube sampling* (LHS) [30]. LHS is a stratified sampling method that ensures a more uniform coverage of each parameter dimension. Specifically, for each parameter, its domain is divided into equally sized intervals, and samples are drawn such that each interval is represented exactly once. These samples are then randomly combined across dimensions to form parameter vectors. Compared to naive uniform sampling, LHS provides a space-filling design, reducing variance in the sampling procedure and improving coverage of the parameter domain. This is particularly advantageous in high-dimensional settings, where uniform random sampling becomes increasingly inefficient.

Given sampled parameter vectors, synthetic data is generated by evaluating the forward model to obtain implied volatility surfaces. This involves two steps: computing option prices under the model, followed by the inversion to implied volatilities.

For the Heston model, option prices are obtained via Fourier-based methods. Since the characteristic function of the log-price is known in closed form, we employ the COS method. For the rough Heston model, the characteristic function is no longer available in closed form, but can be computed by solving a fractional Riccati equation. This requires the numerical solution of a fractional integral for each evaluation of the characteristic function, resulting in a substantially higher computational cost per sample.

Once option prices have been obtained, they are converted to implied volatilities. This inversion is performed using the rational approximation method proposed by Jäckel, which provides a fast and accurate approximation of the implied volatility across a wide range of strikes and maturities.

The resulting implied volatility surface is then discretised on a fixed grid in moneyness and maturity, and subsequently flattened into a vector. This yields input-output pairs (x, y), where x represents the discretised implied volatility surface and y the corresponding model parameters.

The technical details and implementation of the pricing methods and implied volatility inversion are discussed in Chapter 2. Here, we focus on their role in constructing a consistent and efficient data generation pipeline.

3.4.2. Preprocessing

Scaling

Neural networks are sensitive to the scale of input features. Differences in scale across input features can lead to poorly conditioned optimisation, slow convergence, and unstable gradient updates. Feature scaling is therefore a crucial preprocessing step. Two commonly used scaling methods are:

- *Min-Max scaling*, which linearly rescales each feature to a fixed interval, typically $[0, 1]$:

$$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}}. \tag{3.38}$$

- *Standardisation*, which centres and scales features to have zero mean and unit variance:

$$x' = \frac{x - \mu}{\sigma}. \tag{3.39}$$

Min-Max scaling preserves the original distribution shape and bounds the input domain, which can be beneficial for neural networks with bounded activations or gating mechanisms. Standardisation, on the other hand, is often preferred when features are approximately Gaussian distributed, as it centres the data and improves numerical stability. As we prefer to keep the input features as close to original as possible for the XAI analysis, we use Min-Max scaling in our experiments.

Whitening

While scaling ensures comparable feature magnitudes, it does not address dependencies between features. In the present setting, neighbouring points (option contracts) on the implied volatility surface are highly correlated, which can lead to redundant representations and slow learning.

To mitigate this, we apply *whitening*, which transforms the input data such that it has identity covariance. Formally, given a random vector x with non-singular covariance matrix Σ and mean μ, whitening seeks a matrix W satisfying

$$W^\top W = \Sigma^{-1}, \tag{3.40}$$

such that $\mathrm{Cov}(W(x - \mu)) = I$. Note that one has

$$\Sigma = U \Lambda U^\top,$$

which denotes the eigenvalue decomposition of the covariance matrix. The orthogonal matrix U contains the eigenvectors and Λ is a diagonal matrix of eigenvalues.
Two commonly used whitening methods are:

- *Principical Component Analysis* (PCA), where $W = \Lambda^{-1/2} U^\top$.
  This amounts to the transformation given by

$$x_{\mathrm{PCA}} = \Lambda^{-1/2} U^\top (x - \mu). \tag{3.41}$$

  This decorrelates the features and aligns them with the principal component axes. However, it rotates the data into a new coordinate system, which may reduce interpretability.

- *Zero-phase Component Analysis* (ZCA), where the whitening matrix is equal to $W = \Sigma^{-1/2}$.
  The transformation can be written as

$$x_{\mathrm{ZCA}} = U \Lambda^{-1/2} U^\top (x - \mu). \tag{3.42}$$

  Unlike PCA whitening, ZCA preserves the original orientation of the data while still removing correlations. As a result, the transformed features remain as close as possible to the original inputs in a least-squares sense.

From the perspective of interpretability, whitening introduces an important trade-off. Although ZCA whitening preserves the orientation of the data in the original feature space, each transformed feature is still a linear combination of all original input features. As a result, individual coordinates of the whitened input no longer correspond directly to specific locations on the implied volatility surface. However, in contrast to PCA whitening, ZCA maintains the geometric structure of the original feature space, meaning that local neighbourhood relationships and spatial patterns are preserved. Intuitively, the transformation can be viewed as applying a global linear smoothing to the surface: each point becomes a weighted combination of surrounding points, but the overall shape and local structure of the surface remain intact.

Consequently, while whitening removes redundancy and improves numerical conditioning for learning, care must be taken when interpreting feature attributions. In particular, feature importance should be understood at the level of spatial regions rather than individual coordinates. In our framework, this issue is mitigated by consistently applying the same ZCA whitening transformation to both training data and instances used for explanation, allowing XAI methods to remain meaningful in the original surface structure.

The data generation and preprocessing pipeline therefore consists of sampling parameters using LHS, generating implied volatility surfaces via the forward model, discretising and flattening these surfaces, and applying scaling and whitening transformations. These steps ensure that the training data is well-distributed, numerically stable, and suitable for efficient neural network training.

3.5. Training Procedure and Evaluation

We train the neural networks in a supervised learning setting, where the goal is to learn a mapping from discretised implied volatility surfaces to the corresponding model parameters. Given a dataset of input-output pairs (x_i, y_i), the data is split into training, validation, and test sets. The training set is used to optimise the model parameters, while the validation set is used to monitor performance during training and to guide model selection. The test set is held out and used only for the final evaluation of model performance. Model selection is performed based on validation performance to prevent overfitting to the training data.

Training is performed using gradient-based optimisation as described in Section 3.2. As the training objective, we employ the mean squared error (MSE), which corresponds to using the squared L^2-norm as error metric J:

$$\mathrm{MSE} = \frac{1}{N} \sum_{i=1}^{N} \| f_\theta(x_i) - y_i \|_2^2, \tag{3.43}$$

where N denotes the number of samples in the considered dataset. The MSE yields smooth gradients and strongly penalises large deviations, which contributes to stable optimisation during training.

For model evaluation, we instead consider the mean absolute error (MAE), defined as

$$\mathrm{MAE} = \frac{1}{N} \sum_{i=1}^{N} \| f_\theta(x_i) - y_i \|_1. \tag{3.44}$$

In the context of calibration, MAE provides a more interpretable measure of parameter estimation accuracy, as it directly reflects the absolute deviations in each parameter. All reported results are therefore based on performance on the held-out test set, measured in terms of MAE per parameter and aggregated across parameters.

4

Empirical Calibration Results

In this chapter we present the empirical calibration results obtained using the neural network architectures introduced in Chapter 3 for the Heston and rough Heston models. Section 4.1 describes the experimental setup. The calibration results for the Heston and rough Heston models are then reported in Sections 4.2 and 4.3, respectively. Additional experiments investigating the effect of data whitening and the performance on out-of-distribution data are presented in Section 4.4. The chapter concludes with a summary of the empirical findings in Section 4.5.

4.1. Experimental Setup

4.1.1. Parameter Sampling

All experiments in this chapter are conducted on synthetically generated implied volatility surfaces. This allows full control over the parameter ranges and ensures that the ground-truth model parameters are known.

Following the calibration framework introduced in Chapter 3, we consider the supervised learning problem where the goal is to recover model parameters from an implied volatility surface. More precisely, we sample parameter vectors $\mathfrak{p} \in \mathcal{P}$, which serve as the *labels* of the learning problem. In the Heston model, the labels consist of five parameters $\mathfrak{p} = (\kappa, \gamma, \rho, v_0, \bar{v})$ living in the parameter space $\mathcal{P} = (0,\infty)^2 \times [-1,1] \times (0,\infty)^2$. For rough Heston, the Hurst parameter H is added to the labels, i.e. $\mathfrak{p} = (\kappa, \gamma, \rho, v_0, \bar{v}, H) \in \mathcal{P} = (0,\infty)^2 \times [-1,1] \times (0,\infty)^2 \times (0,1)$. For each sampled parameter set, an implied volatility surface $\sigma_{\text{imp}}(\mathfrak{p}; m, T)$ is computed on a predefined moneyness–maturity grid $\mathcal{M} \times \mathcal{T}$. The resulting surface constitutes the *feature* input to the neural network.

We define the *moneyness* m as the strike price relative to the spot price,

$$m = \frac{K}{S_0}, \tag{4.1}$$

and the *maturity* T as the time to expiration of the option expressed in years. In all experiments we set the spot price to $S_0 = 1$, such that the strike grid directly corresponds to moneyness values.

For the Heston model, implied volatility surfaces are generated on the following moneyness–maturity grid:

$$\mathcal{M} \times \mathcal{T} = \begin{matrix} \{0.5,\, 0.6,\, 0.7,\, 0.8,\, 0.9,\, 1.0,\, 1.1,\, 1.2,\, 1.3,\, 1.4,\, 1.5\} \times \\ \{0.1,\, 0.3,\, 0.6,\, 0.9,\, 1.2,\, 1.5,\, 1.8,\, 2.0\}. \end{matrix} \tag{4.2}$$

This results in a total of 88 implied volatility values per surface. Following [19], we consider a truncated version of the parameter space, i.e. $\overline{\mathcal{P}} \subset \mathcal{P}$, and we generate labels $\mathfrak{p}$ from this smaller set by means of Latin Hypercube sampling (Section 3.4). The bounds used for the truncation are detailed in Table 4.1.

For the Heston model, we sample a total of $N = 2 \cdot 10^5$ parameter vectors $\{\mathfrak{p}_i\}_{i=1}^N$ from the truncated parameter space $\overline{\mathcal{P}}$ using Latin Hypercube sampling. For each sampled parameter vector, an implied volatility surface is computed on the grid $\mathcal{M} \times \mathcal{T}$, yielding a dataset of $2 \cdot 10^5$ synthetic implied volatility surfaces.

Table 4.1: Parameter sampling ranges for the Heston model.

	κ	γ	ρ	v_0	$\bar{v}$
Lower bound	0.50	0.01	-0.95	0.0001	0.01
Upper bound	10.0	1.00	-0.10	0.0400	0.20

In addition, the sampling procedure is designed to ensure a balanced representation of different regimes of the Feller ratio

$$\phi = \frac{2\kappa\bar{v}}{\gamma^2}, \tag{4.3}$$

which determines whether the variance process satisfies the Feller condition (see Section 2.4). Specifically, we generate 40% of the labels such that $\phi \geq 1$, corresponding to labels that satisfy the Feller condition; we refer to those labels as *safe*. A further 30% of the samples are drawn from the range $0.7 \leq \phi < 1$, which we refer to as a *mild violation* of the Feller condition, while the remaining 30% are sampled from the interval $0.3 \leq \phi < 0.7$, corresponding to a *strong violation*. This stratified sampling ensures that the training data covers both regimes in which the Feller condition is satisfied and regimes in which it is violated, and it allows us to investigate how calibration performance changes across Feller regimes.

For the rough Heston model, we consider two distinct moneyness–maturity grids. The first grid contains options with intermediate to relatively long maturities, which we refer to as the *long-maturity grid*:

$$\mathcal{M} \times \mathcal{T} = \{0.6,\, 0.7,\, 0.8,\, 0.9,\, 1.0,\, 1.1,\, 1.2,\, 1.3,\, 1.4\} \times \{0.6,\, 0.9,\, 1.2,\, 1.5,\, 1.8,\, 2.0\}, \tag{4.4}$$

which contains 54 implied volatility values per surface. Parameter vectors are sampled from the truncated parameter space $\overline{\mathcal{P}}$ using Latin Hypercube sampling. The bounds used for the truncation are chosen in accordance with common practice for real SPX data [39], as reported in Table 4.2.

Table 4.2: Parameter sampling ranges for the rough Heston model.

	κ	γ	ρ	v_0	$\bar{v}$	H
Lower bound	0.1206	0.2720	-0.7071	0.0262	0.0721	0.1286
Upper bound	0.5041	0.3748	-0.5940	0.0778	0.1499	0.1766

In addition to the long-maturity grid, we consider a second grid containing shorter maturities and two additional moneyness levels. We refer to this grid as the *short-maturity grid*:

$$\begin{aligned}\mathcal{M} \times \mathcal{T} = {} & \{0.5,\, 0.6,\, 0.7,\, 0.8,\, 0.9,\, 1.0,\, 1.1,\, 1.2,\, 1.3,\, 1.4,\, 1.5\} \times \\ & \{0.02,\, 0.04,\, 0.06,\, 0.08,\, 0.10,\, 0.12,\, 0.15,\, 0.20,\, 0.30,\, 0.50\},\end{aligned} \tag{4.5}$$

This grid contains 110 implied volatility values per surface. The parameter vectors for this dataset are sampled from the same truncated parameter space $\overline{\mathcal{P}}$ used for the long-maturity grid.

For both grids we generate $N = 10^4$ parameter vectors, resulting in two datasets of 10^4 synthetic implied volatility surfaces for the rough Heston model. The smaller dataset size compared to the Heston model is primarily due to the substantially higher computational cost of generating rough Heston implied volatility surfaces.

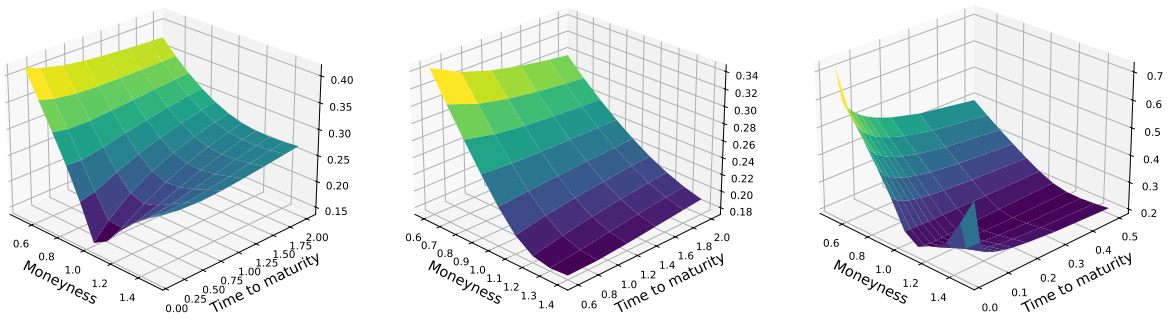
To evaluate the robustness of the calibration networks, we additionally generate an out-of-distribution (OOD) dataset for the rough Heston model on the long-maturity grid. The OOD parameter ranges are constructed by extending the original sampling intervals by 20% on both sides for each model parameter. The resulting bounds are reported in Table 4.3.

From this enlarged parameter space, we generate $2{,}000$ additional parameter vectors using Latin Hypercube sampling. The precise procedure used to construct the OOD samples is described in Section 4.4, where we introduce a mild out-of-distribution setting in which only a subset of parameters is allowed to fall outside the original training ranges.

Table 4.3: Out-of-distribution parameter sampling ranges for the rough Heston model.

	κ	γ	ρ	v_0	$\bar{v}$	H
Lower bound	0.0439	0.2514	-0.7297	0.0159	0.0565	0.1190
Upper bound	0.5808	0.3954	-0.5714	0.0881	0.1655	0.1862

Figure 4.1 shows example implied volatility surfaces generated for the Heston model and for the rough Heston model on the two grids considered in the experiments.



(a) Heston **(b)** Rough Heston (long-maturity grid) **(c)** Rough Heston (short-maturity grid)

Figure 4.1: Example implied volatility surfaces generated for the experiments.

4.1.2. Network Configurations

Using the synthetic datasets described in the previous subsection, we train the neural network architectures introduced in Section 3.3. In particular, we consider three architectures: a multilayer perceptron (MLP), a Highway network, and a Generalised Highway network. The architectural definitions follow those introduced in Chapter 3, and here we specify the configurations used in the experiments.

For the MLP architecture, we consider shallow feedforward networks with one to three hidden layers. The number of hidden units per layer is selected from the set $\{16, 32, 64, 128, 256\}$ in order to investigate the effect of model capacity on calibration performance. All hidden layers use the ReLU activation function with Glorot normal weight initialisation.

For the Highway and Generalised Highway architectures, we consider deeper networks with the number of layers selected from $\{3, 7, 15, 30\}$. Each hidden layer contains 64 units. The width of the hidden layers is fixed in order to limit the hyperparameter search space and focus the comparison on the effect of network depth, while keeping the computational cost of the experiments manageable. The transform layers use the ELU activation function with He normal weight initialisation. The transform gate employs the sigmoid activation function with Glorot normal initialisation, and the initial bias of the transform gate is set to -1.0, following common practice for Highway networks [45]. In the Generalised Highway architecture, the transform and carry gates are obtained via a joint softmax over two pre-activations, with an initial transform bias of -1.0 and initial carry bias of $+1.0$.

One may question why the MLP architecture is restricted to shallow configurations, i.e. at most three hidden layers. While deeper plain feedforward networks can in principle be trained using modern techniques such as improved initialisation, normalisation, or carefully chosen activation functions, they are generally not designed for very deep architectures and may exhibit optimisation difficulties as depth increases [2]. In this thesis, the MLP therefore serves primarily as a simple baseline architecture, whose relatively shallow structure also contributes to its conceptual simplicity and interpretability. Highway networks, in contrast, are explicitly designed to facilitate the training of deep networks through the use of gated skip connections [45]. For this reason, we explore significantly deeper configurations for the Highway and Generalised Highway architectures, with depths up to thirty layers.

A summary of all architectural and training configurations considered in the experiments is provided in Tables 4.4 and 4.5.

Table 4.4: Hyperparameter configurations considered for the Multilayer Perceptron (MLP).

	MLP
Number of Layers	$\{1, 2, 3\}$
Units per Layer	$\{16, 32, 64, 128, 256\}$
Weight Initialiser	Glorot Normal
Activation Function	ReLU
Loss Function	MSE
Learning Rate	10^{-3}
Batch Size	256

Table 4.5: Hyperparameter configurations considered for the Highway and Generalised Highway networks.

	Highway	**Generalised Highway**
Number of Layers	$\{3, 7, 15, 30\}$	$\{3, 7, 15, 30\}$
Units per Layer	64	64
Transform Weight Initialiser	He Normal	He Normal
Transform Activation Function	ELU	ELU
Gate Weight Initialiser	Glorot Normal	Glorot Normal
Transform Gate Activation	Sigmoid	Softmax
Initial Transform Gate Bias	-1.0	-1.0
Carry Gate Activation	—	Softmax
Initial Carry Gate Bias	—	$+1.0$
Loss Function	MSE	MSE
Learning Rate	10^{-3}	$\{10^{-3}, 10^{-4}\}$
Batch Size	256	256

4.1.3. Training Configuration

All neural networks are trained to learn the mapping from implied volatility surfaces to model parameters using the synthetic datasets described in Section 4.1. In order to train and evaluate the networks in a consistent manner, the datasets are split into training, validation, and test sets.

For the Heston dataset, consisting of $2 \cdot 10^5$ synthetic samples, we first perform a train–test split of 85% and 15%, respectively. The resulting training portion is subsequently split into training and validation subsets using an $80/20$ split. This results in training, validation, and test sets that are sufficiently large while still allowing reliable model selection.

For the rough Heston datasets, which consist of 10^4 samples for each grid, we adopt a slightly different splitting strategy in order to avoid excessively small training sets. In this case, we first apply a train–test split of 80% and 20%, after which the training set is further divided into training and validation subsets using a $75/25$ split. This ensures that both the training and validation sets remain sufficiently large for stable optimisation and model selection.

Prior to training, the input implied volatility surfaces are preprocessed in two steps. First, the features are scaled using Min–Max normalisation, mapping each input dimension to the interval $[0, 1]$. Subsequently, ZCA whitening is applied as described in Section 3.4. Whitening removes linear correlations between the grid points and rescales the features to have unit covariance, which can improve numerical stability and facilitate optimisation when training deep neural networks. The preprocessing transformations are fitted exclusively on the training data. In particular, the Min–Max normalisation parameters and the ZCA whitening transformation are estimated using the training set only, after which the same transformations are applied unchanged to the validation, test, and OOD datasets.

The networks are trained by minimising the mean squared error (MSE) between the predicted and

true model parameters (Section 3.6). Model selection is performed using the mean absolute error (MAE), which provides a more interpretable measure of calibration accuracy.

For visualisation purposes, we also present the *signed* relative errors for each parameter, defined as

$$\mathrm{RelativeError}_j = \frac{\hat{\theta}_j - \theta_j}{|\theta_j|}, \tag{4.6}$$

where θ_j denotes the true value of parameter j and $\hat{\theta}_j$ the corresponding prediction. The signed relative error highlights whether parameters are systematically over- or under-estimated, which is its advantage over the absolute relative error.

All networks are trained for exactly 200 epochs. During training, the model parameters obtained at each epoch are evaluated on the validation set. After training is completed, the epoch yielding the lowest validation MAE is selected. This epoch is used when comparing different architectural configurations. Training and validation loss curves for the configurations achieving the lowest error are reported in Sections 4.2 and 4.3 to verify that the networks converge stably and do not exhibit significant divergence during training.

For model selection, each configuration is trained using the training dataset and evaluated using the validation dataset. The configuration achieving the lowest validation MAE is selected as the *best-performing* configuration. As such, whenever we refer to the best-performing configuration in the remainder of this chapter, we refer to the configuration achieving the lowest validation MAE. Subsequently, this best configuration is retrained using the combined training and validation datasets for the number of epochs corresponding to the previously identified optimal epoch. The final performance of the selected model is then evaluated on the held-out test dataset.

Optimisation is performed using the Adam optimiser with mini-batch training and a fixed batch size of 256. For most architectures, a learning rate of 10^{-3} is used. However, for the Generalised Highway architecture on the Heston dataset we found that reducing the learning rate to 10^{-4} significantly improved training stability and calibration accuracy. For all other configurations, lowering the learning rate did not lead to improved performance, and therefore the default value of 10^{-3} was retained.

All neural networks are implemented using the *TensorFlow* platform [29], and experiments were conducted on a machine equipped with a 6-core Intel Core i7 processor (2.20 GHz) and 16 GB of RAM.

4.2. Calibration of the Heston Model

In this section, we present the empirical calibration results for the neural network architectures described earlier, starting with the Heston model. To maintain clarity, only the configurations achieving the lowest validation error are reported in the main text, while the complete hyperparameter sweep results are deferred to Appendix A.

4.2.1. Calibration Accuracy

First, we evaluate the performance of the considered neural network architectures on the Heston calibration problem. This includes assessing how well the different architectures recover the underlying model parameters from the implied volatility surfaces, as well as identifying the best-performing configuration within each architecture.

We begin by analysing the effect of architectural hyperparameters on validation performance. Figure 4.2a shows the mean validation MAE for the MLP architecture across different combinations of depth and width. The results indicate that increasing the number of hidden units significantly improves performance. In particular, configurations with 128 or 256 units per layer consistently outperform smaller networks. The MLP configuration with the lowest validation error consists of two hidden layers with 256 units each.

For the Highway and Generalised Highway architectures, we vary the network depth while keeping the number of hidden units fixed at 64. The results are shown in Figure 4.2b. For the Highway architecture, the best performance is achieved at a depth of 7 layers, while deeper configurations lead to a deterioration in validation accuracy. In contrast, the Generalised Highway architecture continues to improve as the depth increases, reaching its best performance at 15 layers.

Using the validation MAE as the model selection criterion, we identify the best-performing configuration for each architecture class. The comparison between these configurations is illustrated in

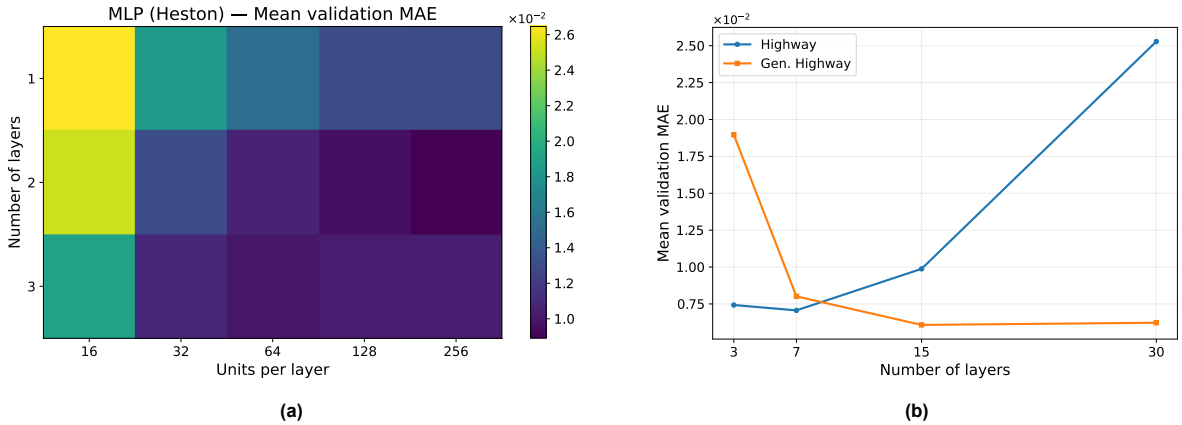


Figure 4.2: (a) Mean validation MAE for the MLP architecture on the Heston model across depths and widths. (b) Mean validation MAE as a function of depth for the Highway and Generalised Highway architectures on the Heston model.

Figure 4.3, which plots the mean validation MAE against the corresponding training time.

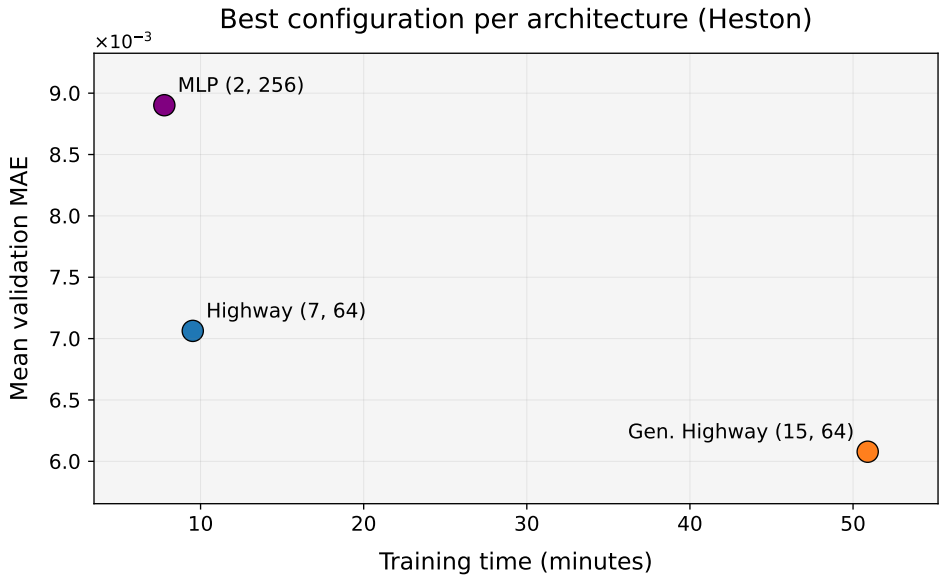


Figure 4.3: Comparison of the best-performing configuration for each architecture on the Heston model. Each point represents the configuration with the lowest validation MAE within the corresponding architecture, plotted against the training time.

The results show a clear performance hierarchy among the architectures. The MLP achieves a validation MAE of approximately 9×10^{-3}, while the Highway network improves this to roughly 7×10^{-3}. The Generalised Highway architecture yields the lowest validation error, achieving a mean validation MAE of approximately 6×10^{-3}. This improvement comes at the cost of increased training time due to the greater depth of the network.

Table 4.6: An overview of the best-performing network configuration per architecture for the Heston model. The reported values correspond to the mean test MAE.

Network	Layers	Units	Parameters	Train Time (min)	$\overline{\text{MAE}}_{\text{test}}$
MLP	2	256	89,861	8.73	0.01502
Highway	7	64	67,269	9.01	0.00907
Gen. Highway	15	64	197,765	49.18	**0.00785**

Table 4.6 reports the test performance of the selected configurations. The results confirm the validation findings: the Generalised Highway architecture achieves the lowest mean test MAE of 0.00785, followed by the Highway network (0.00907) and the MLP (0.01502). While the MLP trains slightly faster, the gated architectures achieve substantially better calibration accuracy.

To verify that the selected models exhibit stable training behaviour, Figure 4.4 shows the training and validation loss curves for the best-performing configuration of each architecture. In all cases, the training and validation losses decrease rapidly during the initial epochs and then gradually converge. The curves do not exhibit signs of divergence or unstable optimisation, confirming that the selected epoch based on the validation MAE corresponds to a well-behaved training process.

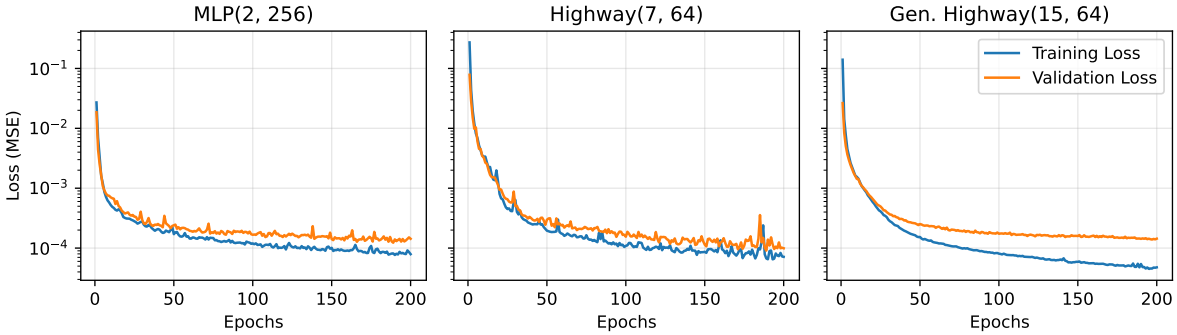


Figure 4.4: Training-validation loss curves for the best-performing configuration of each architecture on the Heston model.

Finally, we analyse the distribution of calibration errors across the different model parameters. Figure 4.5 reports the quantiles of the signed relative error for each parameter and architecture.

Overall, the errors remain concentrated around zero for the majority of the distribution, indicating that the networks are able to recover the parameters with high accuracy. The tails of the error distribution widen slightly for the MLP, particularly for the variance-related parameters v_0 and $\bar{v}$, suggesting that the simpler architecture occasionally produces larger deviations. In contrast, the Highway and Generalised Highway networks exhibit tighter error distributions across most quantiles, reflecting their improved calibration accuracy.

To summarise, all three architectures are capable of learning the inverse mapping from implied volatility surfaces to Heston model parameters. However, the results demonstrate the advantage of gated architectures for this task. Both the Highway and Generalised Highway networks outperform the shallow MLP baseline, with the Generalised Highway architecture achieving the best overall calibration accuracy on the test set.

4.2.2. Regime Comparison

For the Heston model, we perform an additional evaluation in which the test set is partitioned according to the Feller ratio ϕ of the parameter vectors. This analysis allows us to assess how calibration performance varies across regimes in which the variance process either satisfies or violates the Feller condition. The evaluation is conducted only for the best-performing architecture identified in the previous subsection; the Generalised Highway network with configuration $(15, 64)$.

Figure 4.6 reports the empirical quantiles of the signed relative error for each parameter, separated by Feller regime. Additionally, Table 4.7 presents the mean test MAE per parameter together with the average MAE across all parameters within each regime.

Overall, the results indicate that calibration accuracy varies noticeably across the different regimes. The best overall performance is obtained for the *mild violation* regime ($0.7 \leq \phi < 1$), which achieves the lowest mean MAE across all parameters. The *strong violation* regime ($0.3 \leq \phi < 0.7$) follows with slightly larger errors, while the *safe* regime ($\phi \geq 1$), where the Feller condition is satisfied, exhibits the largest overall calibration error.

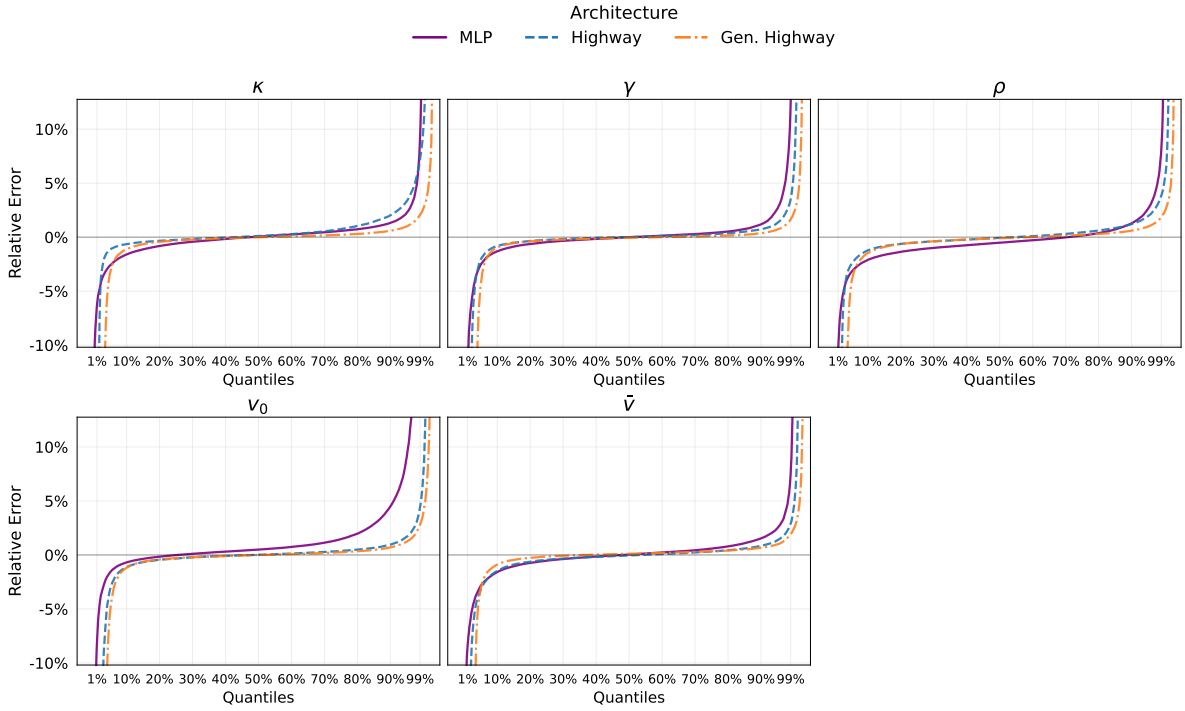


Figure 4.5: Empirical quantiles of the signed relative error for each Heston parameter on the test set for the best configuration of each architecture. For visual clarity, the plot is restricted to the 1^{st}–99^{th} quantile range.

Table 4.7: Test MAE per Heston-parameter for the best-performing configuration of the Generalised Highway network, grouped by Feller regime.

Feller violation	κ	γ	ρ	v_0	$\bar{v}$	$\overline{\text{MAE}}_{\text{test}}$
Strong ($0.3 \leq \phi < 0.7$)	0.02007	0.00265	0.00239	0.00011	0.00043	0.00513
Mild ($0.7 \leq \phi < 1.0$)	0.01379	0.00155	0.00180	0.00007	0.00026	**0.00349**
Safe ($\phi \geq 1.0$)	0.05144	0.00475	0.00843	0.00038	0.00093	0.01318

A closer inspection of Table 4.7 shows that this difference is primarily driven by the parameters κ, γ, and ρ. In particular, the mean absolute error for κ in the safe regime is substantially larger than in the other regimes; compared to the mild violation regime, the error is approximately five times higher. Similar differences are observed for γ and ρ.

The quantile plots in Figure 4.6 provide additional insight into these differences. For the safe regime, the distributions of the signed relative errors for γ and ρ exhibit noticeably heavier left tails compared to the other regimes, indicating that these parameters tend to be underestimated more frequently when the Feller condition is satisfied.

In contrast, the strong violation regime shows a tendency toward positive relative errors for several parameters. In particular, the quantile curves suggest that κ, v_0, and $\bar{v}$ are more often overestimated in this regime compared to the mild and safe regimes.

Taken together, the results indicate that calibration accuracy differs across Feller regimes, with the lowest errors observed in the mild violation regime and the largest errors in the safe regime.

4.3. Calibration of the Rough Heston Model

The presentation of the results for the calibration of the rough Heston model is analogous to Section 4.2. First, we consider the long-maturity grid, after which the results for the short-maturity grid will be presented. A comparison of the two grids is presented in Section 4.3.3.

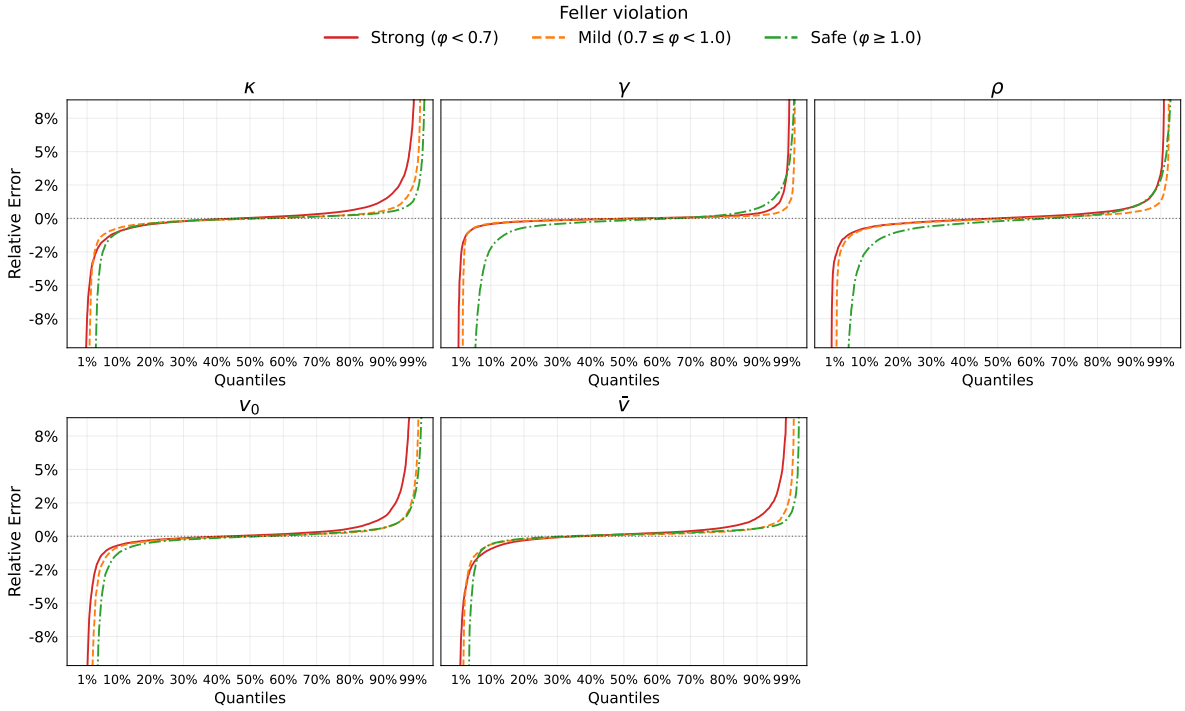


Figure 4.6: Empirical quantiles of the signed relative error for each Heston parameter on the test set for the best-performing configuration of the Generalised Highway network, split by Feller regime. For visual clarity, the plot is restricted to the 1^{st}–99^{th} quantile range.

4.3.1. Long-maturity Grid

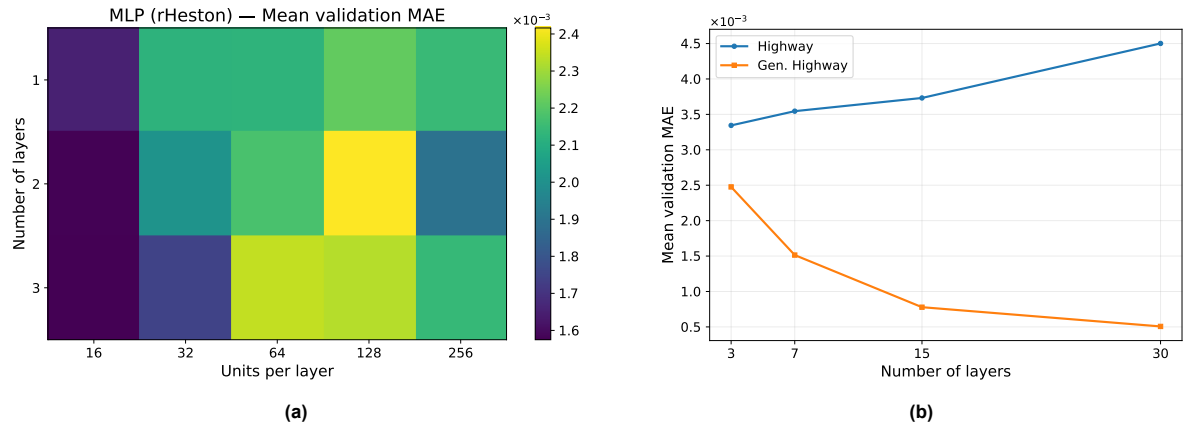


Figure 4.7: (a) Mean validation MAE for the MLP architecture on the rough Heston model (long-maturity grid) across depths and widths. (b) Mean validation MAE as a function of depth for the Highway and Generalised Highway architectures on the rough Heston model (long-maturity grid).

Figure 4.7a shows the mean validation MAE for the MLP architecture across different depths and widths for the rough Heston model on the long-maturity grid. Interestingly, contrary to what we have seen for the Heston model, the prediction performance does not seem to decrease as the width of the network increases. In fact, the configuration with the lowest mean validation MAE has 3 layers and only 16 hidden units per layer. Note that even the smallest network with 1 layer and 16 hidden units exhibits similar prediction accuracy.

When considering the mean validation MAE for the Highway and Generalised Highway networks (Figure 4.7b), once again the Generalised Highway shows strong predictive accuracy, and the network clearly benefits from increased depth. The best mean validation MAE is attained by the configuration

with 30 layers and 64 hidden units per layer. On the contrary, depth seems to have an adverse effect on the predictive accuracy of the Highway network. The lowest mean validation MAE results from the most shallow configuration of the Highway network considered, with only 3 layers and 64 hidden units per layer.

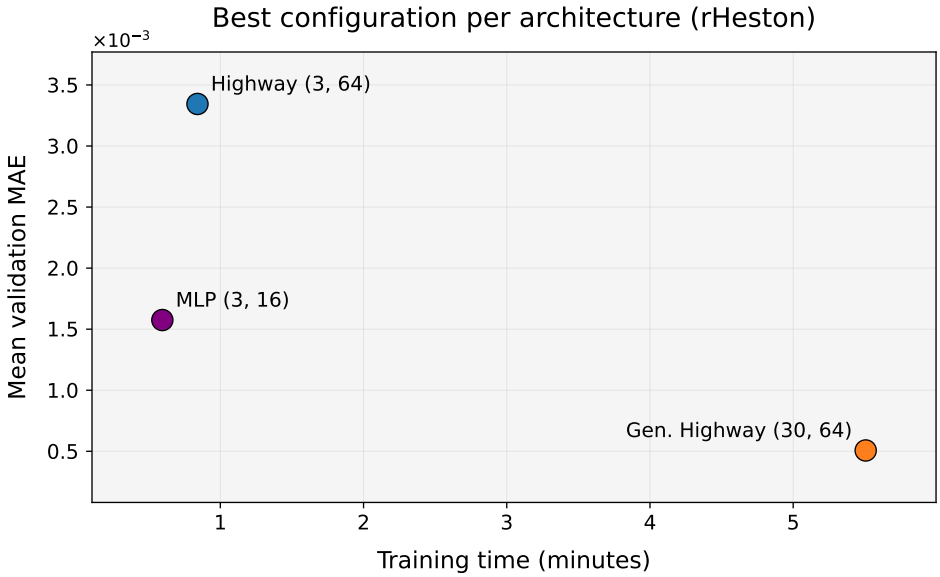


Figure 4.8: Comparison of the best-performing configuration for each architecture on the rough Heston model (long-maturity grid). Each point represents the configuration with the lowest validation MAE within the corresponding architecture, plotted against the training time.

The comparison of the best-performing configurations per architecture is visualised in Figure 4.8. Compared to the results for the Heston model, this time the Highway architecture performs the worst, with a mean validation MAE of approx. $3.5 \cdot 10^{-3}$; this is almost 7 times larger than the mean validation MAE of the Generalised Highway network (approx. $0.5 \cdot 10^{-3}$). Considering the difference in training time, the MLP-configuration exhibits very reasonable predictive performance with a mean validation MAE of $1.5 \cdot 10^{-3}$. While this is 3 times larger compared to the Generalised Highway network, the training of the network is also approx. 5 times faster.

Table 4.8: Test MAE per rough Heston-parameter for the best-performing network configuration per architecture (long-maturity grid).

Network	κ	γ	ρ	v_0	$\bar{v}$	H	$\overline{\text{MAE}}_{\text{test}}$
MLP (3, 16)	0.00264	0.00092	0.00104	0.00037	0.00068	0.00042	0.00101
Highway (3, 64)	0.00641	0.00151	0.00176	0.00087	0.00137	0.00077	0.00211
Gen. Highway (30, 64)	0.00114	0.00045	0.00037	0.00020	0.00036	0.00021	**0.00046**

In Table 4.8 the test performance of the best-performing configurations is reported. The results confirm the validation findings, with the Generalised Highway network achieving the lowest mean test MAE of $4.6 \cdot 10^{-4}$, followed by the MLP $(1.0 \cdot 10^{-3})$ and the Highway network $(2.1 \cdot 10^{-3})$.
The training behaviour of the above network configurations is visualised in Figure 4.9. In all cases, the curves do not exhibit signs of divergence or unstable optimisation, confirming that the selected epoch based on the validation MAE corresponds to a well-behaved training process. We do note that the curves for the MLP appear to be strongly decreasing until the last epoch, which might indicate that this configuration could benefit from longer training.

Figure 4.10 reports the empirical quantiles of the signed relative error for each rough Heston parameter on the test set. Overall, the Generalised Highway network consistently exhibits the smallest spread of errors across nearly all quantiles, confirming the superior predictive accuracy already observed in

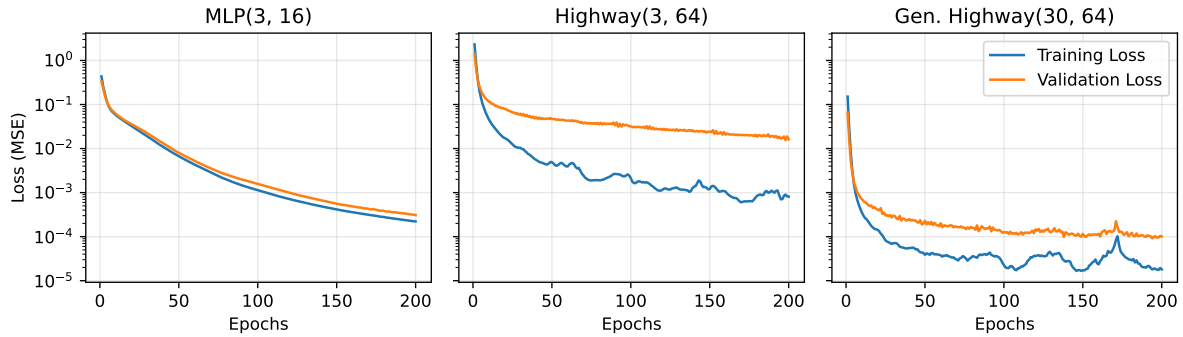


Figure 4.9: Training-validation loss curves for the best-performing configuration of each architecture on the rough Heston model (long-maturity grid).

the MAE comparison. The MLP performs reasonably well, although the error distributions are generally wider than those of the Generalised Highway network. In contrast, the Highway architecture shows noticeably larger deviations for several parameters, particularly in the tails of the distributions. Across all architectures, the majority of the errors remain concentrated close to zero, indicating that the networks are generally able to recover the parameters with high accuracy on the long-maturity grid.

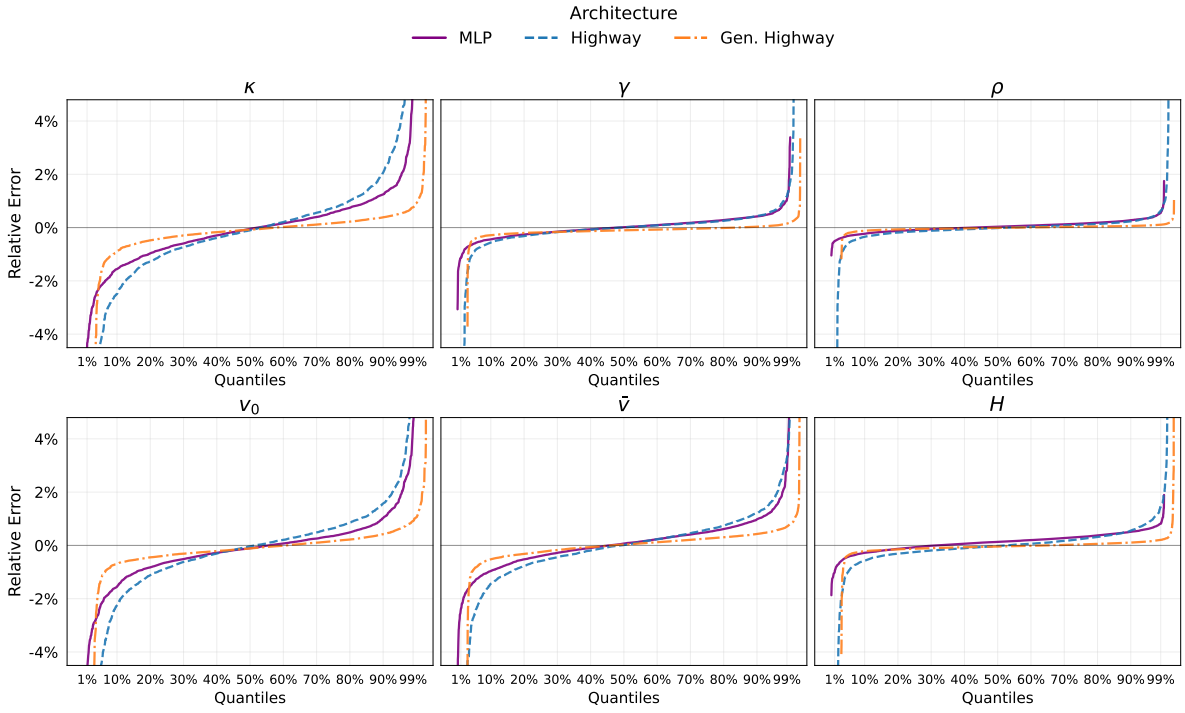


Figure 4.10: Empirical quantiles of the signed relative error for each rough Heston parameter on the test set for the best configuration of each architecture (long-maturity grid). For visual clarity, the plot is restricted to the 1^{st}–99^{th} quantile range.

In brief, all three architectures are capable of learning the inverse mapping from implied volatility surfaces to rough Heston model parameters on the long-maturity grid. While the Generalised Highway architecture outperforms the shallow MLP baseline, the Highway architecture seems to have difficulties when the depth is enlarged.

4.3.2. Short-maturity Grid

The hyperparameter search for the short-maturity grid exhibits patterns similar to those observed for the long-maturity grid. Figure 4.11a shows that the predictive performance of the MLP architecture degrades as the width of the network is increased. Adding depth does increase the performance, but

only moderately. We found that the smallest configuration considered, with 1 layer and 16 hidden units, achieves the lowest mean validation MAE for the MLP architecture.

For the Highway and Generalised Highway architectures (Figure 4.11b), the behaviour largely mirrors that observed in the long-maturity experiment. The Generalised Highway network benefits from increased depth and achieves its best performance for the deepest configuration considered. The predictive accuracy of the Highway architecture slightly deteriorates as the depth increases, with the configuration containing 7 layers and 64 hidden units providing the best results.

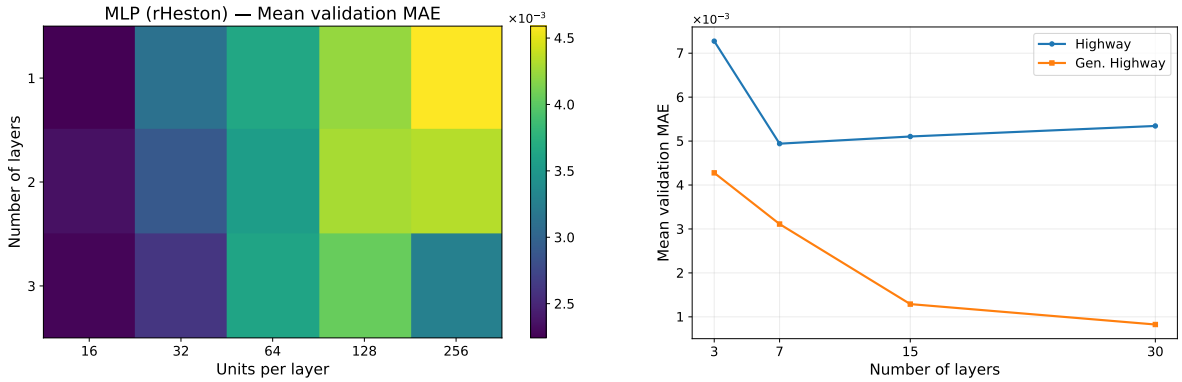


Figure 4.11: (a) Mean validation MAE for the MLP architecture on the rough Heston model (short-maturity grid) across depths and widths. (b) Mean validation MAE as a function of depth for the Highway and Generalised Highway architectures on the rough Heston model (short-maturity grid).

Figure 4.12 compares the best-performing configurations of each architecture. As in the long-maturity experiment, the Generalised Highway architecture achieves the lowest validation error, while the Highway architecture performs the worst. The MLP again provides a competitive baseline with substantially shorter training time.

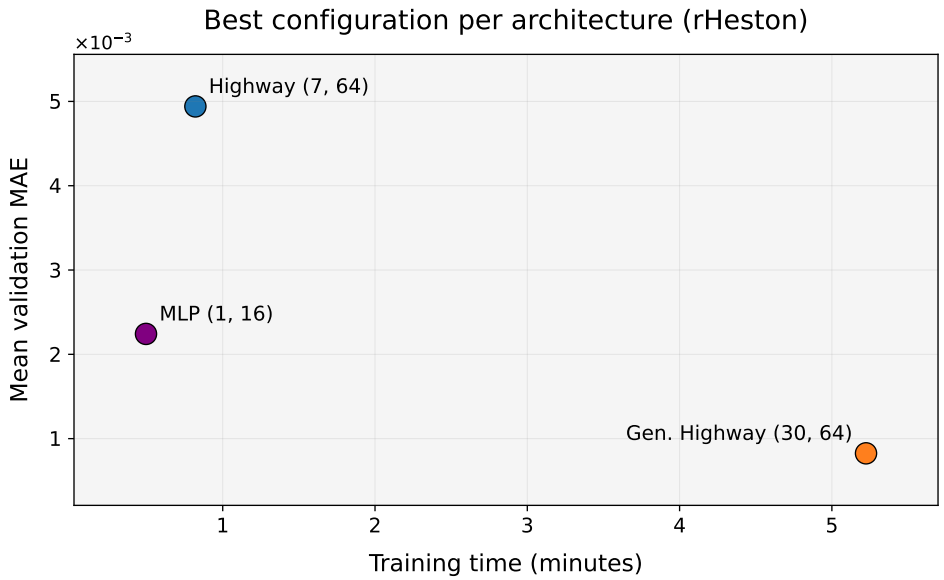


Figure 4.12: Comparison of the best-performing configuration for each architecture on the rough Heston model (short-maturity grid). Each point represents the configuration with the lowest validation MAE within the corresponding architecture, plotted against the training time.

Table 4.9: Test MAE per rough Heston-parameter for the best-performing network configuration per architecture (short-maturity grid).

Network	κ	γ	ρ	v_0	$\bar{v}$	H	$\overline{\text{MAE}}_{\text{test}}$
MLP (1, 16)	0.00964	0.00134	0.00235	0.00156	0.00324	0.00109	0.00320
Highway (7, 64)	0.00876	0.00332	0.00351	0.00181	0.00395	0.00184	0.00386
Gen. Highway (30, 64)	0.00835	0.00236	0.00220	0.00086	0.00154	0.00105	**0.00272**

Table 4.9 reports the test performance of the selected configurations. While the relative ranking of the architectures remains unchanged, the overall calibration accuracy is noticeably lower than in the long-maturity experiment. In particular, the best-performing Generalised Highway configuration achieves a mean test MAE of $2.72 \cdot 10^{-3}$, which is substantially larger than the error obtained on the long-maturity grid.

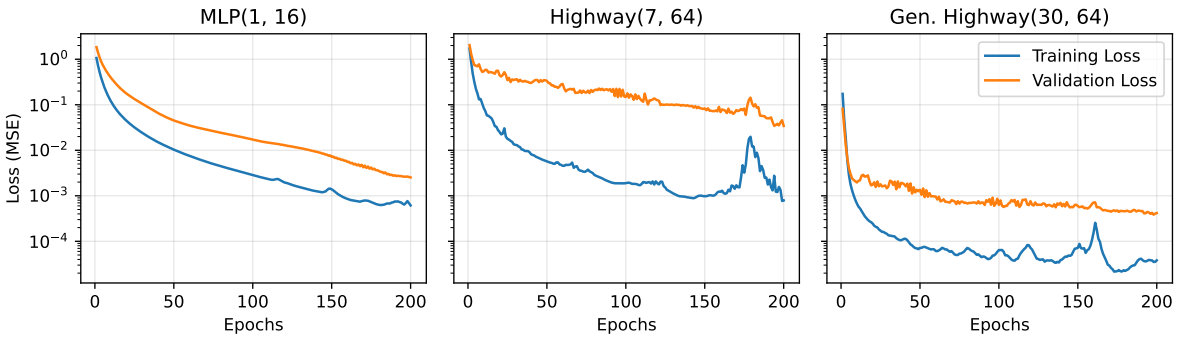


Figure 4.13: Training-validation loss curves for the best-performing configuration of each architecture on the rough Heston model (short-maturity grid).

The training behaviour of the networks is shown in Figure 4.13. Similar to the the long-maturity grid, the training and validation losses indicate stable optimisation.

Figure 4.14 reports the empirical quantiles of the signed relative error for each rough Heston parameter. The distributions are generally wider than those observed for the long-maturity grid (note the increased scale of the y-axis), confirming the increased difficulty of the calibration problem in this setting. Nevertheless, the Generalised Highway network still produces the tightest error distributions across most parameters, while the Highway architecture again shows the largest deviations.
To summarise, the results for the short-maturity grid are qualitatively consistent with those obtained for the long-maturity grid. The Generalised Highway architecture achieves the best predictive accuracy, while the MLP provides a competitive but simpler baseline and the Highway architecture performs comparatively worse. However, the calibration problem appears to be more challenging on the short-maturity grid, as reflected in the larger prediction errors across all architectures.

4.3.3. Grid Comparison

We now compare the calibration results obtained on the long-maturity and short-maturity grids. Across both experiments, the qualitative behaviour of the neural network architectures remains consistent. In particular, the Generalised Highway architecture achieves the lowest prediction error in both settings, while the MLP provides a competitive baseline and the Highway architecture exhibits the largest errors.

A clear difference between the two experiments is the overall level of calibration accuracy. For all architectures, the prediction errors are substantially larger on the short-maturity grid. For instance, the best-performing Generalised Highway configuration achieves a mean test MAE of $4.6 \cdot 10^{-4}$ on the long-maturity grid, whereas the corresponding error increases to $2.7 \cdot 10^{-3}$ on the short-maturity grid. Similar increases in prediction error are observed for the MLP and Highway architectures.

Despite the difference in absolute calibration accuracy, the relative ranking of the architectures remains unchanged across the two grid specifications.

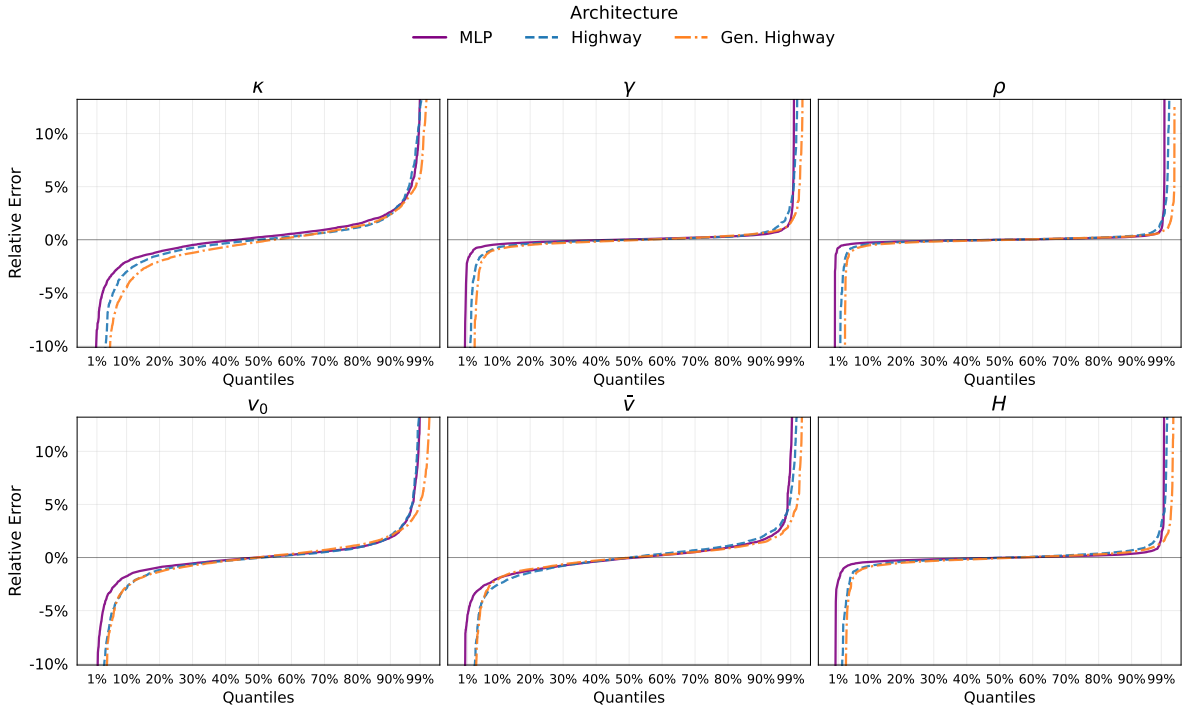


Figure 4.14: Empirical quantiles of the signed relative error for each rough Heston parameter on the test set for the best configuration of each architecture (short-maturity grid). For visual clarity, the plot is restricted to the 1^{st}–99^{th} quantile range.

4.4. Robustness Experiments

In the previous sections, we evaluated the calibration performance of the considered neural network architectures under the in-distribution settings defined in Section 4.1. We now consider two additional experiments to assess the robustness of the obtained results. First, we examine the effect of input whitening on training stability and calibration accuracy. Second, we evaluate the performance of the networks on OOD samples for the rough Heston model.

4.4.1. Effect of Whitening

To investigate whether whitening improves the calibration performance of the neural networks, we compare models trained on whitened and non-whitened implied volatility surfaces. As described in Section 4.1, the surfaces are first scaled using MinMax normalisation and subsequently transformed using ZCA whitening.

To illustrate the effect of the whitening transformation, Figure 4.15 shows the correlation matrix of the input features before and after applying ZCA whitening. Before whitening, the implied volatility surface exhibits strong correlations across strikes and maturities. After whitening, these correlations are largely removed, resulting in approximately decorrelated inputs.

Figure 4.16 shows the mean validation MAE obtained for different network depths when training the Generalised Highway architecture with whitened and non-whitened inputs. For both the Heston and rough Heston models, whitening leads to a substantial reduction in prediction error across all configurations.

The effect is particularly pronounced for the rough Heston model. Without whitening, the validation MAE remains substantially higher, whereas whitening leads to a pronounced reduction in the validation error across all depths. Similar improvements are observed for the Heston model.

The impact of whitening on the optimisation behaviour is illustrated in Figure 4.17. When training on non-whitened inputs, the training and validation losses decrease much more slowly and converge to significantly higher values. In contrast, whitening results in faster convergence and substantially lower final loss values.

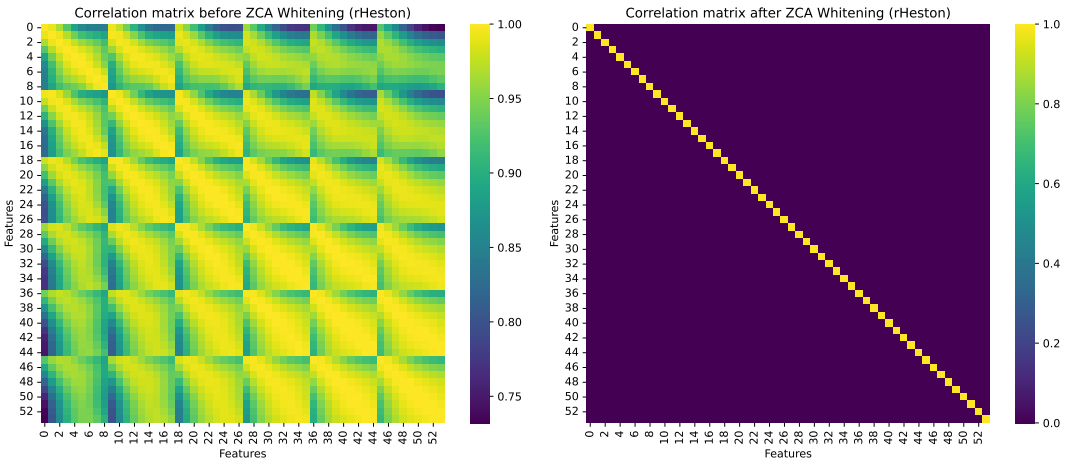


Figure 4.15: Correlation matrices of the input features (rough Heston, long-maturity grid) before and after applying ZCA whitening.

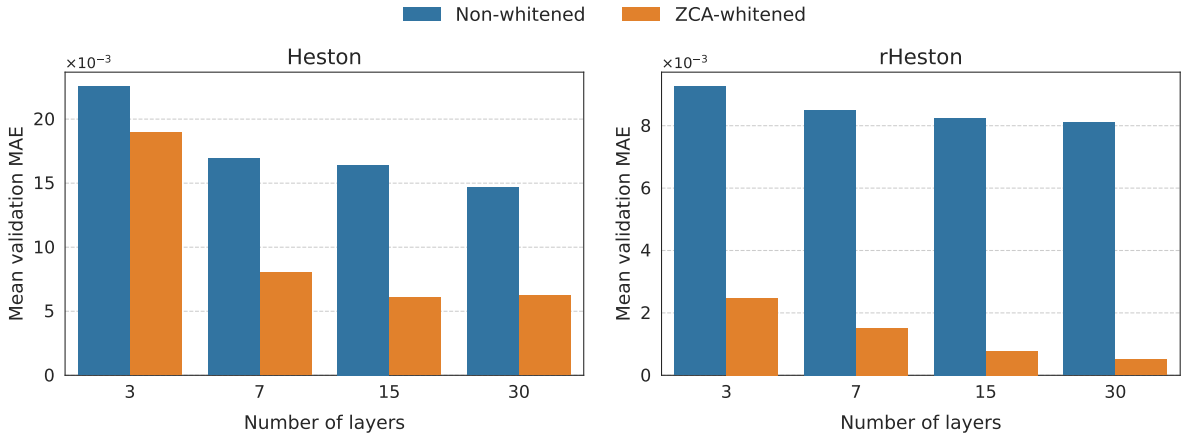


Figure 4.16: Comparison of the mean validation MAE of the Generalised Highway network trained with non-whitened and ZCA-whitened inputs, as a function of depth, for the Heston model and the rough Heston model (long-maturity grid).

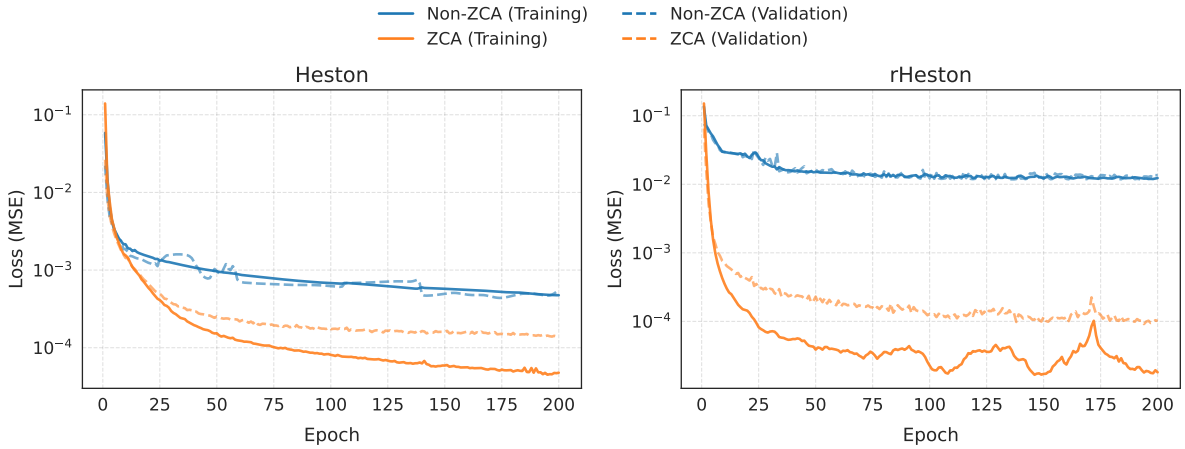


Figure 4.17: Comparison of training-validation loss curves for the best-performing configuration of the Generalised Highway network when trained with non-whitened and ZCA-whitened data. For the rough Heston model, only the curves corresponding to the long-maturity grid are shown.

Table 4.10: Comparison of test-MAE per rough Heston-parameter for the best-performing Generalised Highway configuration with and without whitened inputs (long-maturity grid).

Whitening	κ	γ	ρ	v_0	$\bar{v}$	H	$\overline{\text{MAE}}_{\text{test}}$
None	0.02860	0.00615	0.00150	0.00067	0.00487	0.00858	0.00840
ZCA	0.00114	0.00045	0.00037	0.00020	0.00036	0.00021	**0.00046**

Table 4.10 reports the test MAE per parameter for the best-performing Generalised Highway configuration trained with and without whitening on the rough Heston model. The results confirm the strong improvement in calibration accuracy; the mean test MAE decreases from $8.40 \cdot 10^{-3}$ without whitening to $4.6 \cdot 10^{-4}$ when ZCA whitening is applied.

In short, these results indicate that whitening the implied volatility surfaces is crucial for obtaining accurate and stable neural network calibration.

4.4.2. Out-of-Distribution Performance

In addition to the in-distribution experiments reported in Sections 4.2 and 4.3, we evaluate the generalisation performance of the neural networks on OOD samples. The OOD labels are generated using the extended parameter ranges introduced in Section 4.1. As described below, we consider a mild OOD setting in which only a subset of parameters is sampled outside the original training ranges.

It is well known that neural networks typically exhibit poor extrapolation behaviour outside the training distribution. For this reason, we consider a mild out-of-distribution setting rather than sampling entirely outside the training domain. In particular, each OOD parameter vector is constructed such that *at least* one and *at most* three parameters are sampled outside the ranges used for training, while the remaining parameters are drawn from the original in-distribution ranges described in Section 4.1. This setup allows us to evaluate the robustness of the calibration networks to moderate distributional shifts without constructing unrealistically severe distribution shifts.

Table 4.11: Comparison of out-of-distribution test-MAE per rough Heston parameter for the best-performing configurations of each architecture on the long-maturity and short-maturity grids.

Network	Grid Type	κ	γ	ρ	v_0	$\bar{v}$	H	$\overline{\text{MAE}}_{\text{test}}$
MLP	Long-maturity	0.02091	0.01808	0.01466	0.00285	0.00791	0.00531	**0.01162**
	Short-maturity	1.52733	0.24223	0.23029	0.29467	0.56301	0.21044	**0.51133**
Highway	Long-maturity	0.53580	0.19611	0.26225	0.07507	0.14622	0.07827	0.21562
	Short-maturity	5.65264	3.37849	2.39143	0.37305	2.36091	0.54880	2.45089
Gen. Highway	Long-maturity	0.06211	0.02784	0.04426	0.00712	0.01354	0.01112	0.02768
	Short-maturity	1.81522	0.29717	0.38267	0.12520	0.40013	0.07516	0.51592

Table 4.11 reports the test MAE per parameter for the best-performing configuration of each architecture when evaluated on the OOD datasets for both the long-maturity and short-maturity grids of the rough Heston model. We only consider the rough Heston model, as the training ranges used for the Heston model are already relatively liberal.

Across all architectures, the calibration performance deteriorates substantially when the models are evaluated on OOD samples. While the performance decrease remains relatively limited for the long-maturity grid, the decrease appears to be much more severe when considering the short-maturity grid. The Highway network exhibits particularly severe degradation in performance, with substantial mean MAE for each parameter. Notably, the MLP appears to generalise slightly better than the Generalised Highway network. Nevertheless, degradation in performance is visible across all parameters, with particularly large increases in the prediction errors for κ, ρ, and $\bar{v}$.

Overall, the results indicate that the neural networks exhibit limited extrapolation capability outside the parameter ranges used during training.

4.5. Summary of Empirical Findings

In this chapter, we evaluated the calibration performance of Multilayer Perceptron (MLP), Highway, and Generalised Highway networks on the Heston and rough Heston models.

For the Heston model, all three architectures were able to learn the inverse mapping from implied volatility surfaces to model parameters with good accuracy. The results showed a clear performance ranking, with the Generalised Highway architecture achieving the lowest validation and test errors, followed by the Highway network and the MLP. The parameter-wise error distributions were generally concentrated around zero, although the MLP exhibited somewhat wider tails for the variance-related parameters. In addition, the Feller-regime analysis revealed that calibration performance varied across regimes, with the lowest errors observed in the mild violation regime and the largest errors in the safe regime.

For the rough Heston model, the Generalised Highway architecture again achieved the best overall predictive accuracy on both the long-maturity and short-maturity grids. The MLP provided a competitive baseline, while the Highway architecture consistently performed worse. Across both grids, the qualitative ranking of the architectures remained unchanged. However, per-parameter MAE was substantially larger on the short-maturity grid than on the long-maturity grid.

Since no single network configuration simultaneously optimises predictive accuracy, training efficiency, and model complexity, we summarise the best-performing configurations under different selection objectives in Table 4.12. In particular, we consider predictive performance (mean validation MAE), total training time, and parameter efficiency, defined as the smallest network achieving a mean validation MAE of the same order of magnitude as, and comparable to, the best-performing configuration.

Table 4.12: Best-performing network configurations under different selection criteria across datasets.

Criterion	Heston	rHeston (long grid)	rHeston (short grid)
Mean validation MAE	Gen. Highway (15, 64)	Gen. Highway (30, 64)	Gen. Highway (30, 64)
Training time	MLP (2, 256)	MLP (3, 16)	MLP (1, 16)
Parameter efficiency	MLP (2, 256)	MLP (3, 16)	MLP (1, 16)

Table 4.12 reveals a clear and consistent trade-off across the 3 datasets: Generalised Highway networks achieve the lowest validation errors, MLP architectures require the least training time, and compact MLP configurations provide the most favourable balance between accuracy and model complexity. This indicates that while deeper architectures improve predictive performance, simple architectures already capture the dominant structure of the inverse calibration problem and may therefore be preferable in practice when computational efficiency is prioritised.

The robustness experiments further showed that preprocessing plays an important role in the calibration framework. In particular, ZCA whitening led to substantially lower prediction errors and more stable optimisation behaviour. Finally, the out-of-distribution (OOD) experiments demonstrated that the neural networks exhibit limited extrapolation capability outside the parameter ranges used during training, with substantially larger errors observed on OOD samples, especially for the short-maturity grid.

Altogether, the empirical results identify the Generalised Highway architecture as the strongest-performing network in terms of predictive accuracy, while also demonstrating that simpler architectures can achieve competitive performance at substantially lower computational cost. These findings provide a clear basis for the structural analysis in the next chapter.

5

Structural Validation with Explainable AI

5.1. Interpretability in Inverse Problems

Chapter 4 demonstrated that the proposed neural network architectures achieve strong predictive performance across both the Heston and rough Heston calibration tasks. Parameter-wise errors remain stable across regimes, and the networks generalise well across grid configurations. From a purely predictive standpoint, these results indicate that the learned mappings approximate the inverse calibration problem to a satisfactory degree. However, predictive accuracy alone does not ensure that the learned mapping is structurally meaningful. In particular, it remains unclear whether the network has internalised economically coherent sensitivities, or whether it merely exploits statistical regularities present in the training distribution.

Despite their strong empirical performance, neural networks are often regarded as *black-box* models: while they may accurately reproduce observed outputs, the internal mechanism by which inputs are mapped to predictions is generally opaque. Due to the layered, non-linear, and non-local structure of deep networks, tracing a prediction back to specific input features is non-trivial. Consequently, even in the presence of high predictive accuracy, it is difficult to ascertain which regions of the implied volatility surface the network relies upon and how these regions contribute to parameter recovery. In the context of model calibration, this opacity raises a fundamental question: Does the network learn a structurally meaningful inverse relationship, or does it interpolate the training data without respecting the economic structure of the underlying model?

The issue is particularly prominent in inverse problems. The problem of mapping from implied volatility surfaces to model parameters is typically ill-posed, may admit non-unique solutions, and is sensitive to noise and distributional assumptions. A neural network may therefore have high predictive accuracy while relying on diffuse or redundant surface information. In such cases, predictive performance alone provides limited insight into whether the recovered parameters reflect economically coherent sensitivities implied by the underlying stochastic volatility model.

Interpretability, although lacking a precise mathematical definition, is commonly understood as ”the degree to which a human can understand the cause of a model’s decision” [31]. The more interpretable a machine learning model is, the easier it becomes to understand why specific predictions have been made. In calibration settings, interpretability serves two complementary purposes. First, given financial intuition about how parameters influence implied volatility surfaces, one may assess whether the learned inverse mapping aligns with these structural sensitivities. Second, interpretability tools provide a means of probing the stability and structure of the learned mapping beyond aggregate error metrics, thereby providing insight into parameter identifiability in an operational sense. More generally, interpretability methods may improve comprehension when there is limited model expertise, or when the underlying mathematical model itself lacks transparency.

In this thesis, we define *structural validation* as the assessment of whether the learned inverse calibration map exhibits intuitive sensitivity patterns and supports meaningful parameter identifiability beyond predictive accuracy. In our context, a parameter is considered meaningfully identifiable if

the corresponding attribution patterns are stable across architectures, interpretable in terms of known model behaviour, and not excessively sensitive to the choice of explanation method. To conduct this validation, we adopt an explainable artificial intelligence (XAI) framework. Rather than treating the neural network as a purely predictive tool, we examine its internal logic through both established and novel XAI methods. This allows us to evaluate whether the learned calibration map relies on structurally consistent regions of the implied volatility surface and whether the resulting parameter recovery reflects stable and interpretable sensitivities.

The remainder of this chapter is structured as follows. First, the XAI methods to be employed are introduced, along with their mathematical foundations. Subsequently, the results of applying the XAI methods to the best-performing calibration models are presented and compared. The chapter concludes with a summary of the findings.

5.2. Explainable AI Framework

Before presenting the results in the next section, we discuss the background and mathematical foundations behind SHAP and νSHAP. We introduce the two methods in the notation used in the field of logic-based explainability, as in [27]. As regression-based machine learning models are our main interest, we restrict ourselves to the relevant definitions and notation for regression.

To this end, let $\mathcal{F} = \{1, \dots, M\}$ denote a set of *features*. Each feature $i \in \mathcal{F}$ takes values from a *domain* $\mathbb{D}_i$, and *feature space* is defined as $\mathbb{F} = \mathbb{D}_1 \times \mathbb{D}_2 \times \cdots \times \mathbb{D}_M$. The notation $\mathbf{x} = (x_1, \dots, x_M)$ denotes an *arbitrary point in feature space*, where each x_i is a variable taking values from $\mathbb{D}_i$. Moreover, the notation $\mathbf{v} = (v_1, \dots, v_M)$ represents a *specific point in feature space*, where each v_i is a constant representing one concrete value from $\mathbb{D}_i$. In regression models, each point in feature space is mapped to an ordinal value taken from a *set of values* $\mathbb{V}$; e.g. $\mathbb{V}$ could denote $\mathbb{Z}$ or $\mathbb{R}$.

Consequently, we may generalise a *regression model* $\mathcal{M}_R$ as being characterised by a non-constant *regression function* ξ that maps feature space $\mathbb{F}$ into the set of elements from $\mathbb{V}$, i.e.

$$\xi : \mathbb{F} \to \mathbb{V}.$$

$\mathcal{M}_R$ is then represented by a tuple $(\mathcal{F}, \mathbb{F}, \mathbb{V}, \xi)$.
An *instance* denotes a pair $(\mathbf{v}, q)$, where $\mathbf{v} \in \mathbb{F}$ and $q = \xi(\mathbf{v}) \in \mathbb{V}$. Coupled together, the regression model and the instance form an *explanation problem* $\mathcal{E} = (\mathcal{M}_R, (\mathbf{v}, q))$.

For each feature, a binary similarity operator is assumed, such that $x_i \approx v_i$ returns 1 if the values of x_i and v_i are deemed sufficiently close to each other according to a given threshold, and 0 otherwise.

Definition 4. *Let $S \subseteq \mathcal{F}$ and $\varepsilon > 0$.*
*Two points $\mathbf{x}, \mathbf{v} \in \mathbb{F}$ are said to be **input-similar** on S if*

$$\max_{i \in S} |x_i - v_i| \leq \varepsilon, \tag{5.1}$$

and we write $\mathbf{x}_S \approx \mathbf{v}_S$.
In the case where $S = \mathcal{F}$, we write $\mathbf{x} \approx \mathbf{v}$ for simplicity.

Additionally, a similarity operator is assumed for the output of the model.

Definition 5. *Let $\mathcal{M}_R : (\mathcal{F}, \mathbb{F}, \mathbb{V}, \xi)$ be a regression model and $\delta > 0$.*
*Given $\mathcal{M}_R$, two points $\mathbf{x}, \mathbf{v} \in \mathbb{F}$ are said to be **output-similar** if*

$$|\xi(\mathbf{x}) - \xi(\mathbf{v})| \leq \delta, \tag{5.2}$$

and we write $\xi(\mathbf{x}) \approx \xi(\mathbf{v})$.

Input similarity restricts feature deviations to a subset S, while output similarity ensures that predictions remain within a prescribed tolerance.

5.2.1. Local Attribution and Global Aggregation

We focus on local methods designed to explain a model's output $\xi(\mathbf{x})$ based on a single input $\mathbf{x}$. Explanation models operate on simplified inputs $\mathbf{z}' \in \{0, 1\}^M$, which are mapped to the original feature

space through a reconstruction mapping

$$h_{\mathbf{x}} : \{0,1\}^M \to \mathbb{F}.$$

A local explanation model g is constructed such that

$$g(\mathbf{z}') \approx \xi\big(h_{\mathbf{x}}(\mathbf{z}')\big)$$

whenever $\mathbf{z}'$ is sufficiently close to the reference simplified input $\mathbf{x}'$.

A broad class of local explanation methods can be characterised by the *additive feature attribution* property, which is defined as follows.

Definition 6. ***Additive feature attribution methods*** *have an explanation model that is a linear function of binary variables:*

$$g(\mathbf{z}') = \psi_0 + \sum_{i=1}^{M} \psi_i z_i' \tag{5.3}$$

where $\mathbf{z}' \in \{0,1\}^M$, M is the number of simplified input features, and $\psi_i \in \mathbb{R}$.

Explanation models matching Definition 3 attribute an effect ψ_i to each feature, and summing the effects of all feature attributions approximates the output $\xi(\mathbf{x})$ of the original model. Various explanation models can be found in the literature matching this definition, most notably [37], [44], and [3].

While local explanation methods provide feature attributions for a single input instance, structural conclusions about the learned calibration map require aggregation across instances.
Formally, let $\psi_i(\mathbf{x})$ denote the local attribution of feature i for instance $\mathbf{x}$. A global importance measure can then be obtained by aggregating these local attributions over a distribution $\mathbb{P}$ on the input space, for example via

$$\mathcal{I}_i = \mathbb{E}_{\mathbf{x}\sim\mathbb{P}}\left[|\psi_i(\mathbf{x})|\right]. \tag{5.4}$$

This aggregation yields a global measure of how strongly each feature contributes to parameter recovery across the evaluation distribution. Moreover, absolute values are used to measure the magnitude of feature contribution irrespective of direction. In our analysis, the expectation is approximated by averaging over the empirical test distribution, corresponding to a uniform probability measure on the evaluation dataset.

5.2.2. SHAP

The additive feature attribution framework can be interpreted in cooperative game-theoretic terms, where the feature attributions correspond to Shapley values of a value function induced by the model. Shapley values were introduced by L.S. Shapley in the context of cooperative game theory [40]. Consider a cooperative game $G = (N, \nu)$, where $N = \{1, ..., M\}$ denotes the set of players and ν a *value function*, which assigns a real number to each coalition of players, i.e.

$$\nu : 2^N \to \mathbb{R}. \tag{5.5}$$

In our setting, the players correspond to the feature set $\mathcal{F}$, and the cooperative game is induced by the regression function $\xi : \mathcal{F} \to \mathbb{R}$. The value function therefore becomes a mapping

$$\nu(S, \mathcal{E}) : 2^{\mathcal{F}} \to \mathbb{R}, \tag{5.6}$$

assigning a coalition value to the subset of features S with respect to the explanation problem $\mathcal{E}$.

As demonstrated below, Shapley values are the unique solution in the class of additive feature attribution methods with three desirable properties [25]. The first of these desirable properties is *local accuracy*, which requires the sum of the attributions per feature to match the output of the original model.

Property 1 (**Local accuracy**).

$$\xi(\mathbf{v}) = \psi_0(\mathbf{v}) + \sum_{i\in\mathcal{F}} \psi_i(\mathbf{v}) \tag{5.7}$$

The second property is *missingness*. If feature i has no effect on the model output, then missingness requires its attribution to be zero.

Property 2 (**Missingness**).

$$\forall S \subseteq \mathcal{F} \setminus \{i\} : \nu(S \cup \{i\}, \mathcal{E}) = \nu(S, \mathcal{E}) \implies \psi_i(\mathbf{v}) = 0. \tag{5.8}$$

The third property is *consistency*. This property states that if a model changes so that some feature contribution increases or stays the same regardless of the other inputs, the attribution of that feature should not decrease.

Property 3 (**Consistency**). *Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be two regression models on the same feature space $\mathbb{F}$, but with different regression functions ξ_1 and ξ_2, along with their respective value functions ν_1 and ν_2. Consider the explanation problems $\mathcal{E} = (\mathcal{M}_1, (\mathbf{v}, \xi_1(\mathbf{v})))$ and $\mathcal{E}' = (\mathcal{M}_2, (\mathbf{v}, \xi_2(\mathbf{v})))$. Then*

$$\forall S \subseteq \mathcal{F} \setminus \{i\} : \nu_1(S \cup \{i\}, \mathcal{E}) - \nu_1(S, \mathcal{E}) \geq \nu_2(S \cup \{i\}, \mathcal{E}') - \nu_2(S, \mathcal{E}') \implies \psi_i(\mathcal{E}) \geq \psi_i(\mathcal{E}'). \tag{5.9}$$

Theorem 3. *Only one explanation model follows Definition 3 and satisfies Properties 1, 2, and 3:*

$$\psi_i(\mathbf{v}) = \sum_{S \subseteq \mathcal{F} \setminus \{i\}} \frac{|S|!(|\mathcal{F}| - |S| - 1)!}{|\mathcal{F}|!} \left(\nu(S \cup \{i\}, \mathcal{E}) - \nu(S, \mathcal{E})\right). \tag{5.10}$$

This corresponds to the Shapley value of the cooperative game $(\mathcal{F}, \nu)$ for feature i.

Proof. The proof follows from classical results in cooperative game theory [48]. □

The Shapley value formula in Theorem 1 is independent of the specific choice of value function. Different XAI methods correspond to different definitions of ν. In the case of SHAP (*SHapley Additive exPlanations*) values, the value function is defined through conditional expectations of the model output:

$$\nu_e(S, \mathcal{E}) = \mathbb{E}\left[\xi(X) | X_S = \mathbf{v}_S\right]. \tag{5.11}$$

In practice, the conditional expectation is evaluated by averaging model outputs over a background dataset, corresponding to an empirical approximation of the underlying feature distribution.
The use of the value function ν_e for interpretability corresponds to answering the question:

How much does conditioning on feature i, in addition to a subset of other features, change the expected model output on average?

Thus, SHAP values attribute to each feature the change in the expected model output when conditioning on that feature. They explain how to get from the base output with no features to the output with all features. Here, the base output is defined as

$$\nu_e(\emptyset, \mathcal{E}) = \mathbb{E}[\xi(X)], \tag{5.12}$$

corresponding to the expected model output under the background distribution.

Numerical approximation

Exact evaluation of the Shapley formula requires $2^{|\mathcal{F}|}$ model evaluations and is therefore computationally infeasible for datasets with a large number of features. *KernelSHAP* provides a model-agnostic approximation by reformulating the Shapley value computation to a weighted linear regression problem [25]. Given a background dataset $\mathcal{D}$, the value function in (5.11) is approximated by empirical averaging over 'hybrid' instances $h_S(\mathbf{v})$, constructed by combining the instance $\mathbf{v}$ with background samples. Specifically, one solves

$$\min_{\psi} \sum_{S \subseteq \mathcal{F}} \pi(S) \left(\xi(h_S(\mathbf{v})) - \psi_0 - \sum_{i \in S} \psi_i\right)^2, \tag{5.13}$$

where:

- $h_S(\mathbf{v})$ denotes the 'hybrid' instance where features in S are fixed to $\mathbf{v}_S$;
- $\pi(S)$ is the Shapley kernel weight, defined by

$$\pi(S) = \frac{(|\mathcal{F}| - 1)}{\binom{|\mathcal{F}|}{|S|}|S|(|\mathcal{F}| - |S|)}. \tag{5.14}$$

Under this choice of weights, and without regularisation, the solution of the weighted regression accurately approximates the Shapley values.

In our experiments, the background distribution is approximated by 100 random samples from the validation set. The same background set is used for all instances to ensure consistency of attributions across explanations. The background size is chosen as a trade-off between computational efficiency and stability for the estimated attributions. For the implementation of KernelSHAP, we used the `SHAP` library available in Python [41].

5.2.3. νSHAP

While SHAP defines feature attribution via marginal changes in expected model output, νSHAP ("*nuSHAP*") adopts a sufficiency-based perspective grounded in logic-based explainability [28]. To this end, we introduce the notion of abductive explanations.

Abductive explanations are an example of formal explanations for classification and regression models, defined as follows [27].

Definition 7. *A **weak abductive explanation (WAXp)** denotes a set of features $S \subseteq \mathcal{F}$, such that for every point in feature space the model output is similar to the instance $(\mathbf{v}, \xi(\mathbf{v}))$:*

$$\textit{WAXp}(S, \mathcal{E}) := \forall \mathbf{x} \in \mathbb{F} : \mathbf{x}_S \approx \mathbf{v}_S \implies \xi(\mathbf{x}) \approx \xi(\mathbf{v}). \tag{5.15}$$

*A subset-minimal WAXp is an **abductive explanation (AXp)**.*

Note the use of input- and output-similarity in Definition 4.

To translate abductive explanations into a feature attribution framework, we construct a cooperative game in which each coalition $S \subseteq \mathcal{F}$ is assigned a value reflecting whether it is sufficient to preserve output-similarity. This yields the following binary value function, used in νSHAP:

$$\nu_s(S, \mathcal{E}) = \begin{cases} 1 & \text{if WAXp}(S, \mathcal{E}) \\ 0 & \text{otherwise} \end{cases} \tag{5.16}$$

Using ν_s for interpretability hence amounts to answering the question:

Is a subset of features S sufficient to preserve output-similarity?

The use of this value function is again inspired by the field of game theory [42]. The motivation is to assign importance to elements (voters or features) which are *critical* for changing the value of a decision of interest. In the case of voting power, importance is assigned to voters that cause coalitions to become winning when the voter is included, and that are losing when the voter is excluded. In our setting, importance is assigned to a feature that causes fixed sets of features to be sufficient for output-similarity when the feature is also fixed, and that causes fixed sets of features *not* to be sufficient for output-similarity when the feature is *not* fixed.

The value function is again inserted into the Shapley formula (5.10), yielding νSHAP-values. The SHAP- and νSHAP-values therefore only differ by the value function used in the Shapley formula. An important consequence of this value function is that νSHAP is *not* an additive feature attribution method in the sense of Definition 3, since the local accuracy property (5.7) does not hold. Indeed, the attributions sum to

$$\nu_s(\mathcal{F}) - \nu_s(\emptyset) = 1, \tag{5.17}$$

rather than to $\xi(\mathbf{v})$.

Furthermore, νSHAP-values are non-negative, taking values in $[0, 1]$. This follows from (5.17) and monotonicity of ν_s, as sufficiency of S implies that any superset $T \supset S$ is also sufficient.

Numerical approximation

A key component of rigorous approximation of νSHAP scores is the ability to efficiently decide whether a set of features S is a WAXp. The authors of the original νSHAP paper opt for a rigorous data-based approach, exploiting *sample-based explanations (sbXps)* [28] [7]. These sbXps are computed with respect to a sample of the feature space (i.e., a set of instances) and are rigorous with respect to that sample. The sample could be, e.g. the original dataset, the sampling carried out by SHAP, or an aggregation of both. As sbXps are rigorous given the sample, in cases where the sample is the entire feature space, sbXps match WAXps.

Definition 8. *Let $\mathbb{S} \subseteq \mathbb{F}$. A **weak sample-based abductive explanation (sbWAXp)** denotes a set of features $\mathcal{X} \subseteq \mathcal{F}$, such that for every point in $\mathbb{S}$ the model output is similar to the instance $(\mathbf{v}, \xi(\mathbf{v}))$:*

$$\textit{sbWAXp}(\mathcal{X}, \mathcal{E}) := \forall \mathbf{x} \in \mathbb{S} : \mathbf{x}_{\mathcal{X}} \approx \mathbf{v}_{\mathcal{X}} \implies \xi(\mathbf{x}) \approx \xi(\mathbf{v}). \tag{5.18}$$

*A subset-minimal sbWAXp is a **sample-based abductive explanation (sbAXp)**.*

Given a sample space, there exists a polynomial-time algorithm for deciding whether a set of features is a sb(W)AXp [7]. One such algorithm is shown below in Algorithm 1, which is the algorithm we used in our experiments.

Algorithm 1 Decision procedure for sbWAXp [7]

Require: Explanation problem $\mathcal{E} = (\mathcal{M}_R, (\mathbf{v}, \xi(\mathbf{v})))$; Dataset $D \subset \mathbb{F}$; Candidate sbWAXp $\mathcal{W} \subseteq \mathcal{F}$
Ensure: True if $\mathcal{W}$ is a sbWAXp; False otherwise

```
function sbWAXp(D, v, W)
    for all x ∈ D do
        if ξ(x) ≉ ξ(v) then
            if x_W ≈ v_W then
                return False
    return True
```

The procedure as described in Algorithm 1 is straightforward. Essentially, if there exists a row $\mathbf{x}$ in the dataset D which does not exhibit output-similarity with the instance as defined in (5.2), but which *does* exhibit input-similarity with the instance on the candidate set of features $\mathcal{W}$ as defined in (5.1), then $\mathcal{W}$ is *not* an sbWAXp; otherwise it is. For each row $\mathbf{x}$, the algorithm checks whether at least one feature in $\mathcal{W}$ violates the input-similarity condition. In the worst case all features in $\mathcal{W}$ must be inspected. Consequently, if n is the number of rows in D, the complexity of the algorithm can be written as $\mathcal{O}(n|\mathcal{W}|)$.

With Algorithm 1 we can numerically evaluate ν_s, which can consequently be used to compute the νSHAP-values. As discussed in the previous subsection, exact evaluation of the Shapley formula (5.10) is computationally infeasible for datasets with a large number of features. Hence, the most common approach is to approximate the Shapley values. The *ApproShapley* algorithm devised by Castro, Gómez and Tejada can be used for this task (Algorithm 2) [5].

The Shapley value admits an equivalent permutation-based representation, which forms the basis of Algorithm 2. For a uniformly sampled permutation π from $\Pi(\mathcal{F})$, the set containing all permutations of $\mathcal{F}$, let $S_\pi(i)$ denote the set of features preceding i in π. The Shapley value can then be expressed as

$$\psi_i = \mathbb{E}_\pi[\nu(S_\pi(i) \cup \{i\}) - \nu(S_\pi(i))]. \tag{5.19}$$

Algorithm 2 estimates this expectation via Monte Carlo sampling of r permutations. The resulting estimator is unbiased and converges in probability to the actual Shapley value [5]. Additionally, the ApproShapley algorithm provides a probabilistic error guarantee for the Monte Carlo estimator. For a prescribed tolerance $\epsilon > 0$ and confidence level $1-\alpha$, the number of sampled permutations r is chosen such that

$$\mathbb{P}(|\widehat{\psi}_i - \psi_i| \leq \epsilon) \geq 1 - \alpha. \tag{5.20}$$

Using the Central Limit Theorem, the required sample size satisfies

$$r \geq \frac{Z^2_{\alpha/2}\sigma_i^2}{\epsilon^2}, \tag{5.21}$$

Algorithm 2 Permutation-sampling estimator for Shapley values (*ApproShapley*) [5]

Require: Feature set $\mathcal{F} = \{1, \dots, M\}$; Value function $\nu : 2^{\mathcal{F}} \to \mathbb{R}$; Number of samples $r \in \mathbb{N}$
Ensure: Estimates $(\widehat{\psi}_1, \dots, \widehat{\psi}_M)$ of Shapley values

function ApproShapley($\mathcal{F}, \nu, r$)
  for $i \in \mathcal{F}$ **do**
    $\widehat{\psi}_i \leftarrow 0$
  for $t = 1$ **to** r **do**
    Sample a permutation π uniformly from $\Pi(\mathcal{F})$
    $S \leftarrow \emptyset$
    for $k = 1$ **to** M **do**
      $i \leftarrow \pi(k)$
      $\widehat{\psi}_i \leftarrow \widehat{\psi}_i + \big(\nu(S \cup \{i\}) - \nu(S)\big)$
      $S \leftarrow S \cup \{i\}$
  for $i \in \mathcal{F}$ **do**
    $\widehat{\psi}_i \leftarrow \widehat{\psi}_i / r$
  return $(\widehat{\psi}_1, \dots, \widehat{\psi}_M)$

where $Z_{\alpha/2}$ denotes the standard normal quantile and σ_i^2 is the variance of the marginal contribution of feature i, i.e. $\nu(S \cup \{i\}) - \nu(S)$. Since σ_i^2 is typically unknown, an upper bound is proposed in [5] based on the range of marginal contributions, yielding a conservative but polynomial-time stopping criterion.

In our experiments involving νSHAP, we used the same background set which was also used for computing the SHAP-values. This background set consists of 100 random samples of the validation set. Furthermore, we used a fixed input-similarity tolerance of 0.05. For the output-similarity tolerance δ, we adopted a data-driven calibration. This is important, as choosing δ too small leads to Algorithm 1 marking every coalition as insufficient, and choosing it too large leads to Algorithm 1 marking every coalition as sufficient (which corresponds to the trivial output). To choose an appropriate output tolerance δ, we estimate the typical scale of variation of the model output under the empirical distribution of the training (or validation) data. To this end, let $\mathcal{B}$ denote a background set of validation instances and $\mathcal{I}$ a set of randomly sampled evaluation points. For each output element j (Heston/rHeston parameter), we compute

$$\delta_j = \operatorname*{median}_{\mathbf{x} \in \mathcal{I}} (\operatorname*{median}_{\mathbf{z} \in \mathcal{B}} |\xi_j(\mathbf{x}) - \xi_j(\mathbf{z})|). \tag{5.22}$$

This choice corresponds to a robust estimate of the typical output variability under the empirical distribution and ensures that sufficiency is evaluated relative to the natural scale of the calibration problem. Subsequently, we computed the νSHAP-values using Algorithm 2 with tolerance $\epsilon = 0.0015$ at a 95% confidence level, and the number of permutations capped at $r = 3 \cdot 10^5$.

As discussed earlier, a single sbWAXp evaluation has complexity $\mathcal{O}(n|\mathcal{W}|)$, where n denotes the number of background rows and $\mathcal{W}$ is the candidate coalition of features. In Algorithm 2, the coalition size grows from 0 to $M-1$ along a permutation. Consequently, the complexity per permutation is equal to

$$\sum_{k=0}^{M-1} nk = \mathcal{O}(nM^2) = \mathcal{O}(n|\mathcal{F}|^2). \tag{5.23}$$

With r sampled permutations, the overall complexity of Algorithm 2 therefore becomes $\mathcal{O}(rn|\mathcal{F}|^2)$.
In our implementation, we introduced minor optimisations to ensure computational feasibility; however, these do not alter the statistical properties of the estimator.

5.3. Structural Validation of Learned Representations

In this section we present explanations obtained by applying the XAI methods introduced in Section 5.2 to the trained neural networks described in Chapter 4. Specifically, we analyse the best-performing configuration of the Generalised Highway architecture for each model type: the Heston model, the rough Heston model on the grid with longer maturities $(0.6-2.0)$, and the rough Heston model on the grid with short maturities $(0.02-0.50)$.

For each model, 50 instances were randomly sampled and SHAP and νSHAP values were estimated for each instance. We chose to explain 50 instances to obtain stable global explanations while keeping computational cost manageable. These local explanations were then aggregated to obtain global explanations by averaging the absolute SHAP values and the νSHAP values across instances. The resulting importance scores are normalised and visualised on the strike–maturity grid for each model type.

In addition, we report the 15 highest-ranked features on average according to both SHAP and νSHAP. After presenting the results for each model type individually, we compare the explanations across architectures and present possible implications for model design and calibration.

Throughout this section, feature importance refers to the role that input features play in the *predictions of the trained neural network*, as quantified by SHAP and νSHAP. These attributions do not directly describe the structural importance of regions in the underlying stochastic volatility model itself. Rather, they characterise how the trained Generalised Highway network uses information from the implied volatility surface to infer model parameters, where SHAP and νSHAP capture different notions of importance.

5.3.1. Heston

In Figure 5.1 the SHAP and νSHAP feature-importance heatmaps for the Generalised Highway network trained on the Heston model are presented. We describe our observations for each parameter. To avoid confusion, we repeat our notion of *moneyness*, defined as the strike price centered around 1 (corresponding to $S_0 = 1$). An option is said to be *in-the-money* (ITM) if the moneyness is less than 1, *at-the-money* (ATM) if the moneyness is exactly 1, and *out-of-the-money* (OTM) if the moneyness is greater than 1.

κ

The SHAP heatmaps show that the network assigns highest importance to medium maturities. Across the moneyness dimension, the highest importance is observed in the ITM and ATM regions, while deep OTM options receive comparatively lower attribution.

The νSHAP heatmaps display a different pattern. Here, the network assigns highest importance to short maturities. Across moneyness, the attribution is distributed more evenly, with both ITM and OTM regions receiving notable importance.

γ

For γ, the SHAP heatmaps show that the network assigns high importance primarily to short maturities. Across the moneyness dimension, importance is spread across a wide range of moneyness levels, with higher attributions observed in the ITM and ATM regions.

The νSHAP heatmaps exhibit a very similar structure. Again, the highest importance is assigned by the network to short maturities, while the attribution is distributed broadly across the moneyness grid. The highest attribution appears to be in the ITM region.

ρ

The SHAP heatmaps indicate that the highest attributions are given to ITM and OTM options. In particular, deep ITM and deep OTM receive significant attributions. In terms of maturity, the most important regions are designated by the network as the short to mid maturities.

νSHAP heatmaps show a similar pattern. The highest importance is again located at deep ITM and deep OTM options, while short maturities dominate the attribution across the maturity dimension. The highest attributions appear to be in the deep OTM region.

v_0 and $\bar{v}$

For v_0, both SHAP and νSHAP heatmaps show that the network assigns strong importance to short maturities. The highest attribution is located in the ATM region, while importance decreaes as moneyness moves further away from ATM.

A similar pattern is observed for $\bar{v}$, in the sense that both SHAP and νSHAP attribute the highest importance to deep ITM options with long maturity. The importance is distributed broadly across the grid, but short maturity options receive the lowest attributions.

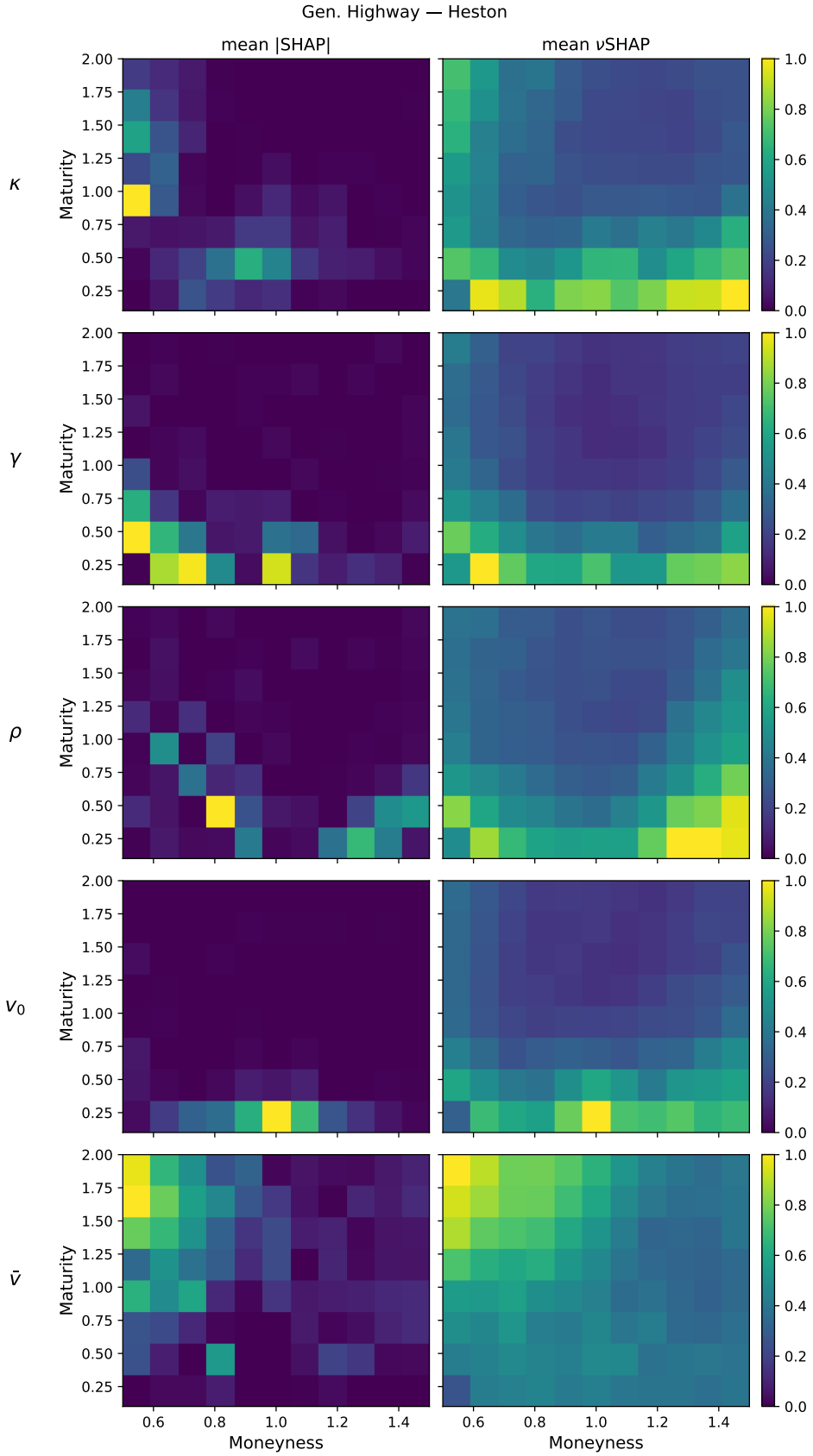


Figure 5.1: SHAP and νSHAP feature-importance heatmaps for the Generalised Highway network trained on the Heston model. Each cell corresponds to an option with a specific moneyness and maturity. Each row corresponds to a model parameter $(\kappa, \gamma, \rho, v_0, \bar{v})$; the columns show normalised mean absolute SHAP-values and mean νSHAP-values across the moneyness–maturity grid.

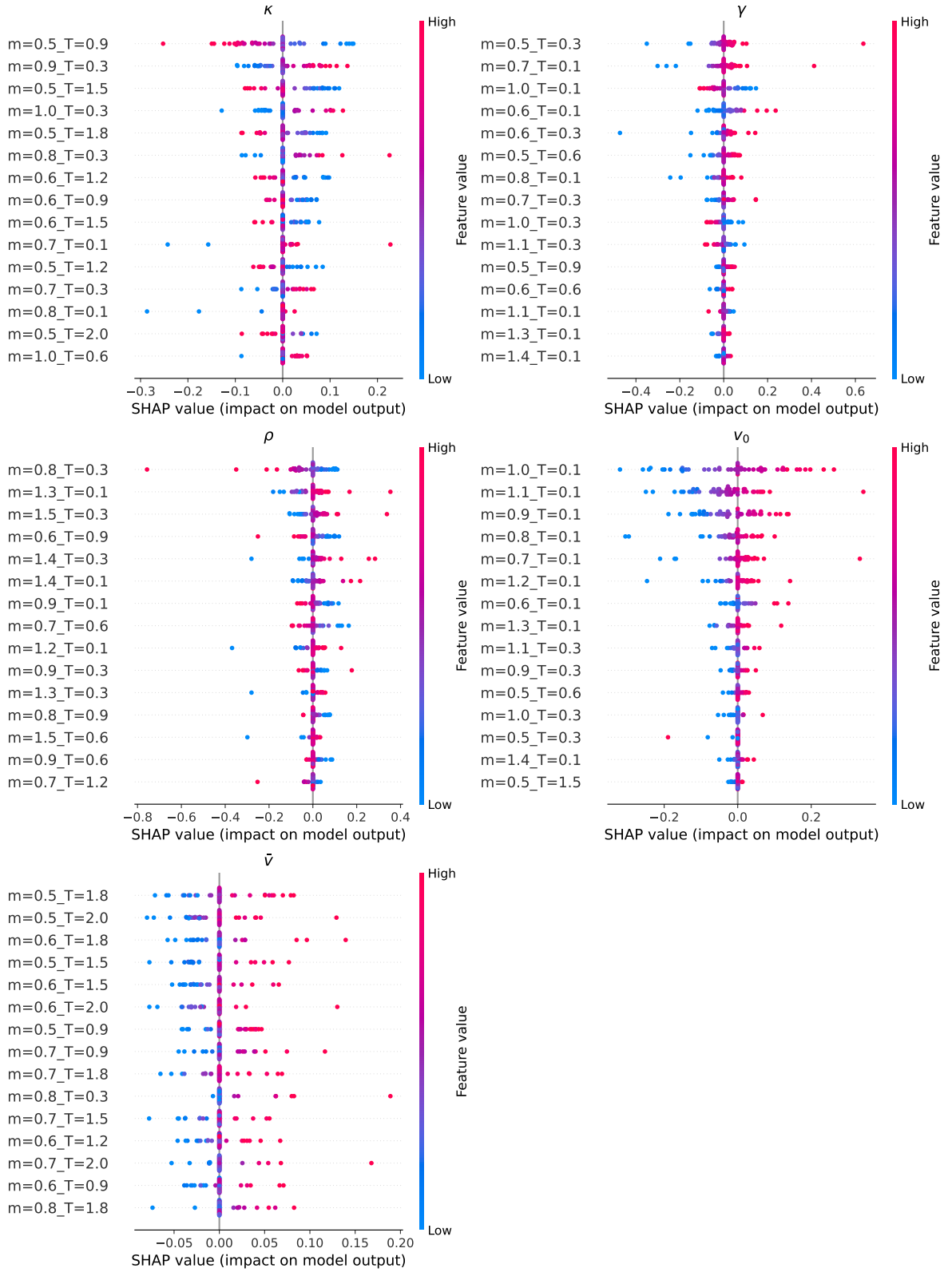


Figure 5.2: Beeswarm plots of SHAP-values for the Generalised Highway network trained on the Heston model. For each parameter ($\kappa, \gamma, \rho, v_0, \bar{v}$), the plots display the top–15 most important features ranked by mean absolute SHAP-value. Each point represents one explained instance, coloured by the corresponding feature value.

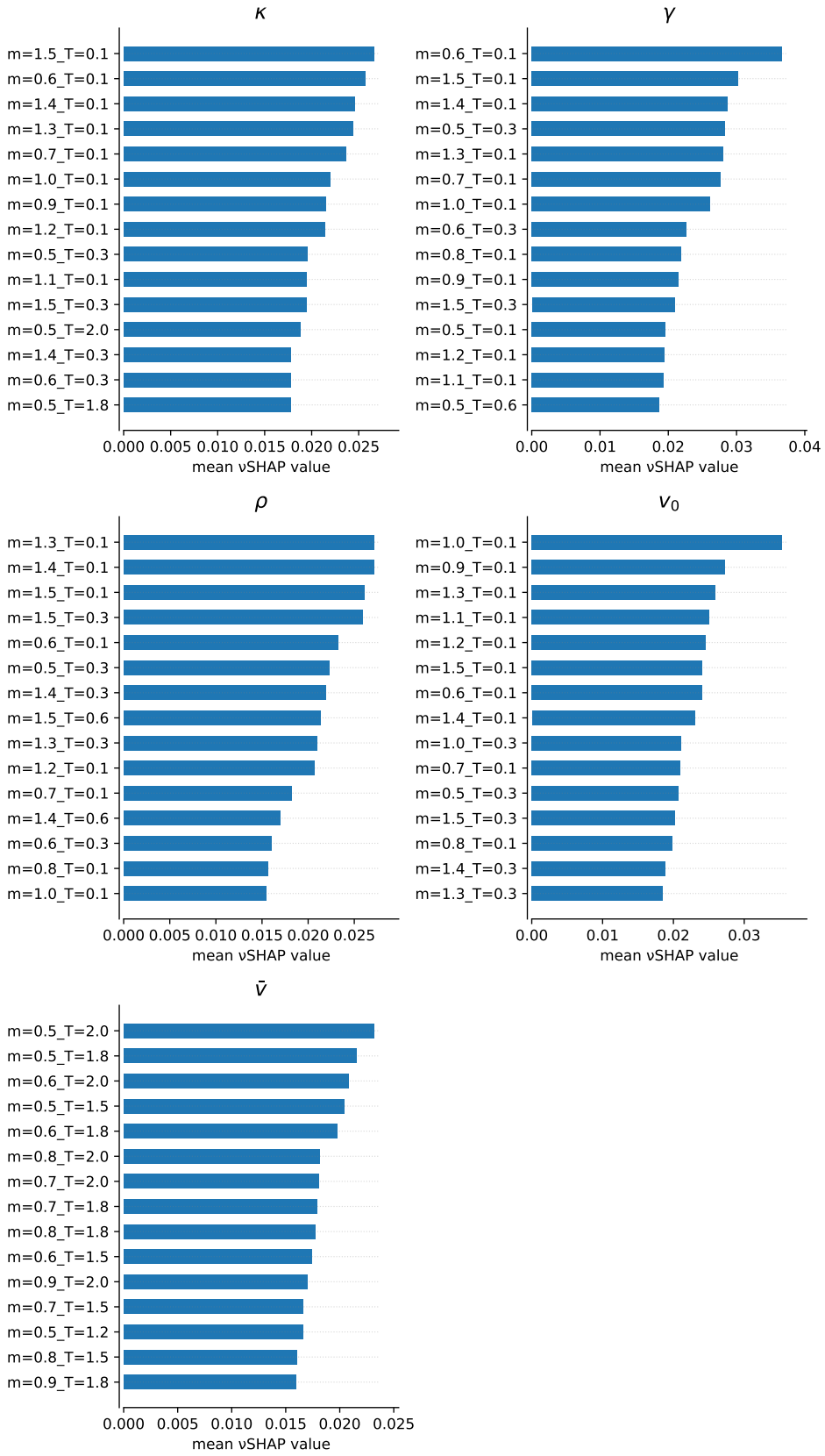


Figure 5.3: Bar charts showing the top–15 most important features for each Heston model parameter ($\kappa, \gamma, \rho, v_0, \bar{v}$) ranked by mean νSHAP-value. Explanations shown are for the Generalised Highway network. Each feature corresponds to a specific moneyness–maturity grid point.

Feature ranking analysis

To further analyse the most influential features, we inspect beeswarm plots (Figure 5.2) and bar charts (Figure 5.3) showing the top–15 features ranked by mean absolute SHAP- and νSHAP-values, respectively. While the heatmaps highlight where information is located on the implied volatility surface, these plots provide a more detailed view of the relative importance of regions on the grid.

After inspecting the beeswarm plots and the bar charts, several patterns emerge. First, the feature ranking confirm the importance of short-maturity options for most parameters. In particular, many of the highest-ranked features correspond to options with maturity $T = 0.1$, which appear consistently among the top positions for κ, γ, ρ, and v_0. This is clearly visible in the bar charts, where the majority of top-ranked features involve the shortest maturity available on the grid.

Across the moneyness dimension, the most influential features are distributed over a wide range of moneyness values. For several parameters, both deep OTM and deep ITM options appear among the highest-ranked features. This is particularly evident for ρ, where extreme moneyness values are frequently present among the top-ranked features. The beeswarm plots also illustrate that these options exhibit relatively large spreads in SHAP-values, indicating a strong influence on the model output.

For v_0, the most influential options are again concentrated at short maturities and around the ATM region. In contrast, the rankings for $\bar{v}$ show a different pattern, with the most important features primarily associated with longer maturities. In the bar charts, the highest-ranked features for $\bar{v}$ correspond predominantly to maturities between $T = 1.2$ and $T = 2.0$.

Overall, the feature rankings broadly support the patterns observed in the heatmaps. Short-maturity options dominate the importance rankings for most parameters, while the distribution across moneyness varies depending on the parameter considered. The beeswarm plots further reveal the magnitude and direction of the SHAP attributions, providing a complementary view of their influence on the model output.

5.3.2. Rough Heston (Long-maturity grid)

Figure 5.4 shows the SHAP and νSHAP feature-importance heatmaps for the Generalised Highway network trained on the rough Heston model, on the grid with relatively longer maturities ($T \in [0.6, 2.0]$). The subsequent observations follow the same structure as in Section 5.3.1.

κ

SHAP heatmaps for κ show that the network assigns importance for the speed-of-mean-reversion parameter primarily to longer maturities. The highest attributions appear at the upper part of the maturity grid, with several important features distributed across different levels of moneyness. Short maturities receive comparatively little importance in SHAP explanations.

In contrast, νSHAP explanations exhibit a different pattern. Here, the highest importance is assigned by the network to the shortest maturity available on the grid. In particular, the largest attributions occur at the extremes of the moneyness grid, where deep ITM and deep OTM options receive the highest importance. Importance is distributed mainly along the edges of the grid, implying that the ATM region with intermediate maturity receives the lowest importance.

γ

For γ, the SHAP heatmaps show that feature importance is primarily located at short to intermediate maturities on the grid. The highest attributions occur at the lower part of the maturity axis, while importance gradually decreases as maturity increases. Importance seems to be spread across the moneyness axis; nevertheless, the deep ITM region receives notably more importance compared to the ATM- and OTM regions.

νSHAP displays a structure which seems similar to what is observed for κ; importance is distributed along the edges of the grid, with the extremes (deep ITM/OTM) receiving the highest attributions.

ρ

Importance appears to be divided between ITM and OTM options on the grid for SHAP involving the correlation parameter ρ. Deep OTM options receive the highest attributions. In terms of maturity, the most prominent importance is observed at shorter maturities, although some importance extends to intermediate maturities. νSHAP shows a similar pattern, with stronger emphasis on the short maturities and ATM options also receiving notable importance. As is the case with κ and γ, the ATM intermediate maturity region seems to receive much less importance compared to the edges of the grid.

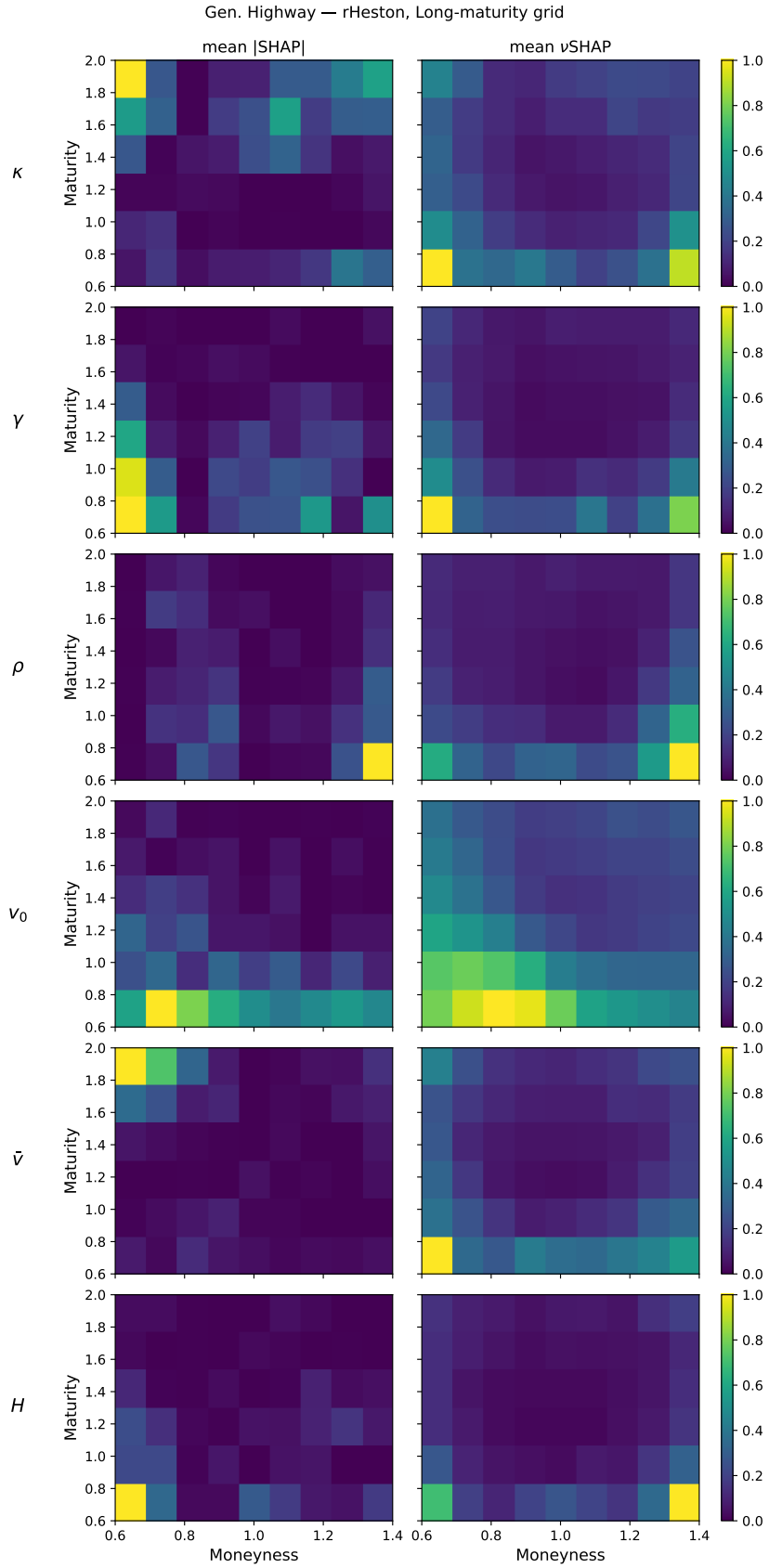


Figure 5.4: SHAP and νSHAP feature-importance heatmaps for the Generalised Highway network trained on the rough Heston model, on the grid with relatively long maturities. Each cell corresponds to an option with a specific moneyness and maturity. Each row corresponds to a model parameter $(\kappa, \gamma, \rho, v_0, \bar{v}, H)$; the columns show normalised mean absolute SHAP-values and mean νSHAP-values across the moneyness–maturity grid.

v_0

Heatmaps for the initial variance parameter v_0 show attributions concentrated along the shortest maturity on the grid, with ITM/deep ITM options receiving the highest importance. While the shortest maturities are most important, intermediate maturities also receive notable attributions. The attributions for νSHAP are distributed more evenly across the grid compared to SHAP; however, longer maturities consistently appear to receive the lowest attributions.

$\bar{v}$

Contrary to v_0, the SHAP heatmaps for $\bar{v}$ show a markedly different pattern. Here, feature attributions by the network are concentrated at longer maturities, with the highest attribution appearing at the upper part of the maturity grid. Across the moneyness dimension, the most importance is observed in the deep ITM region of the grid.

Interestingly, the νSHAP heatmaps show a different distribution of importance. We see a similar pattern as earlier observed for κ, γ, ρ, where the attributions seem to be distributed along the edges of the grid. Deep ITM short maturity options receive the highest attributions, followed by deep ITM long maturity and deep OTM short maturity options. The ATM intermediate maturity region appears to receive low attributions.

H

For the Hurst parameter H, the SHAP heatmaps show that importance assigned by the network is concentrated at the short maturity extremes on the grid. While some importance is additionally attributed to intermediate maturities, the deep ITM and deep OTM options with short maturity receive high attributions, followed by short maturity ATM. Longer maturities receive little to no attributions.

In the νSHAP heatmap, again the same pattern can be observed as with $\kappa, \gamma, \rho, \bar{v}$. It appears that the importance is distributed along the edges, albeit that for H, the deep ITM and deep OTM short maturity options clearly dominate attributions. Contrary to SHAP, deep OTM short maturity seems to receive the largest attribution in νSHAP, followed by deep ITM short maturity.

Feature ranking analysis

After inspecting the beeswarm plots for SHAP (Figure 5.5) and the νSHAP feature rankings (Figure 5.6), several patterns emerge regarding the most influential features on the implied volatility surface. Overall, the rankings are dominated by options with the shortest maturity on the grid ($T = 0.6$), which appear consistently among the most important features across nearly all parameters. In the νSHAP barcharts, the majority of the top-ranked features correspond to grid points with $T = 0.6$, indicating that the shortest maturity on the grid plays a central role for the model output.

Across the moneyness dimension, the most influential features are often located at the extremes of the grid. In particular, deep ITM and deep OTM options frequently appear among the top-ranked features. This pattern is especially visible for the parameters $\kappa, \gamma, \rho, \bar{v}$ and H. For v_0, options which are slightly ITM and around the ATM region are ranked highly.

The beeswarm plots further illustrate how these features influence the model output. For several parameters, the highest-ranked features exhibit a relatively wide spread of SHAP-values, indicating a strong influence on the model output for the respective parameter. This suggests that variations in those grid points can have a substantial impact on the model output.

While short maturities dominate the rankings for most parameters in νSHAP, some differences can also be observed. For example, several longer-maturity features appear among the top-ranked features for H; similarly, a small number of longer-maturity features appear in the ranking for κ and $\bar{v}$, while this is the complete opposite for SHAP. Overall, the feature rankings broadly support the patterns observed in the heatmaps. The most influential features are typically located either at the shortest maturity on the grid, or at the extremes of the moneyness dimension.

5.3.3. Rough Heston (Short-maturity grid)

Figure 5.7 shows the SHAP and νSHAP feature-importance heatmaps for the Generalised Highway network trained on the rough Heston model, on the short-maturity grid ($T \in [0.02, 0.50]$). Again, we state our observations for each parameter like in the previous subsections.

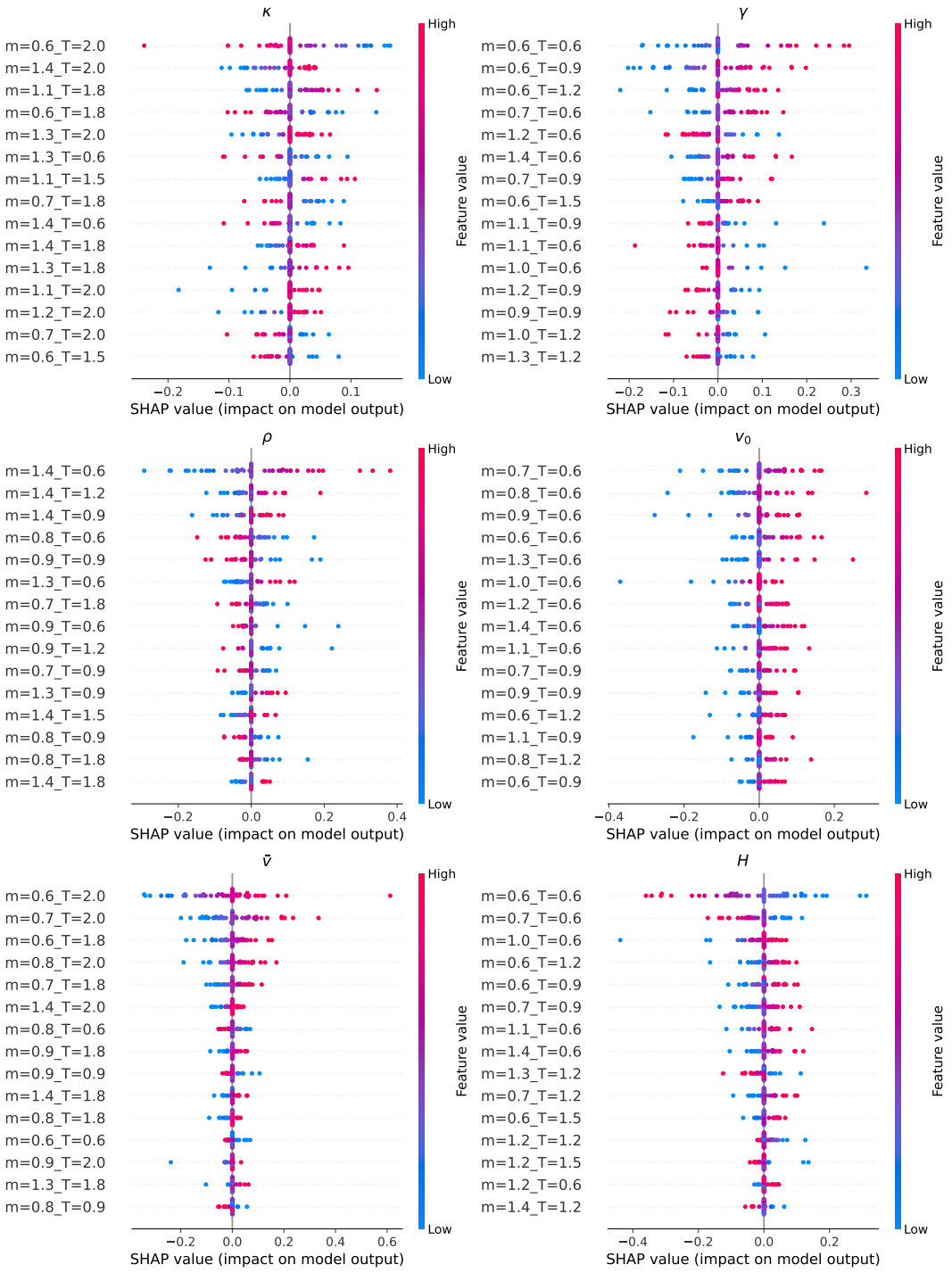


Figure 5.5: Beeswarm plots of SHAP-values for the Generalised Highway network trained on the rough Heston model on the long-maturity grid. For each parameter ($\kappa, \gamma, \rho, v_0, \bar{v}, H$), the plots display the top–15 most important features ranked by mean absolute SHAP-value. Each point represents one explained instance, coloured by the corresponding feature value.

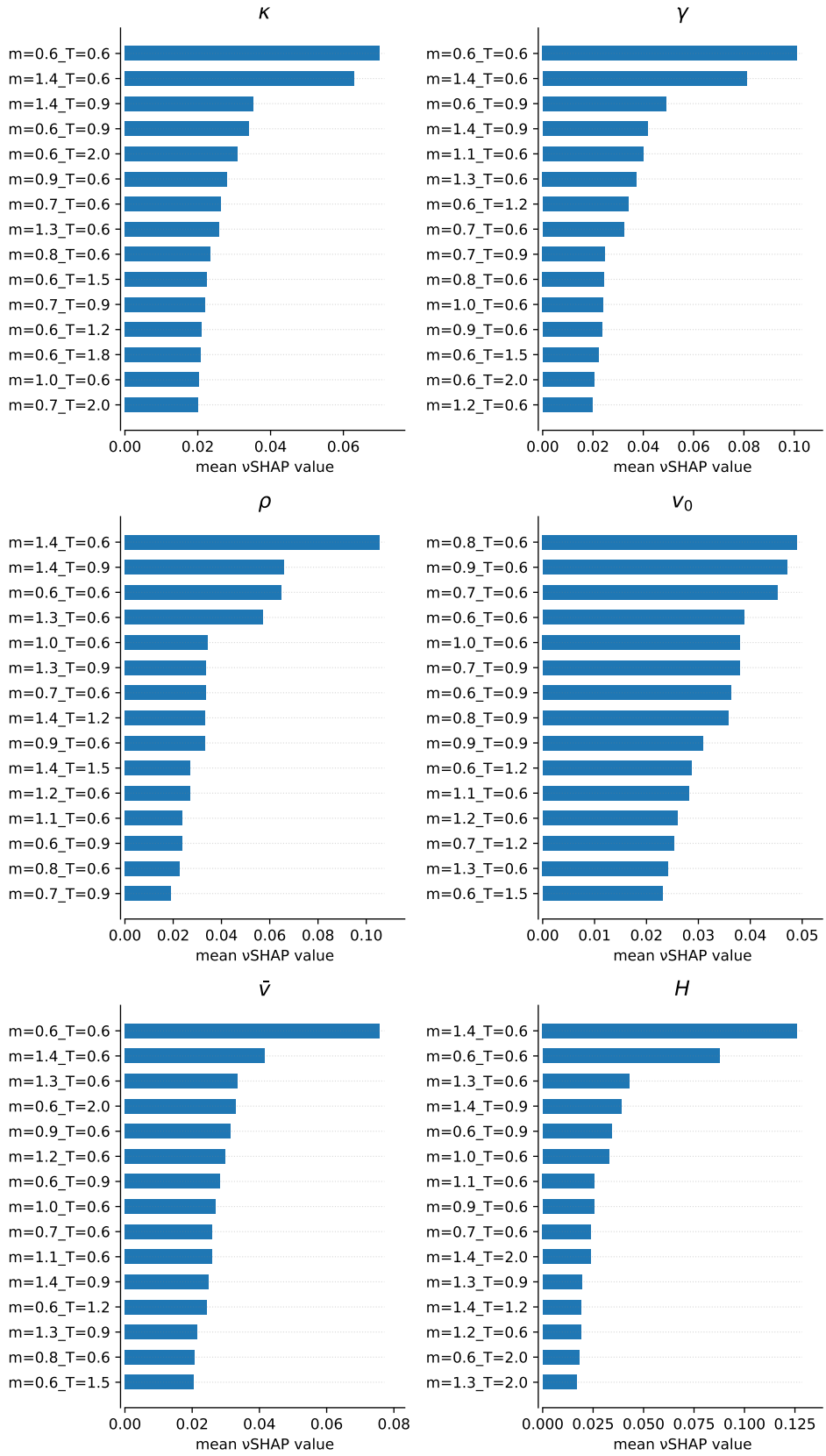


Figure 5.6: Bar charts showing the top–15 most important features for each (rHeston, Long-maturity grid) model parameter ($\kappa, \gamma, \rho, v_0, \bar{v}, H$) ranked by mean νSHAP-value. Explanations shown are for the Generalised Highway network. Each feature corresponds to a specific moneyness–maturity grid point.

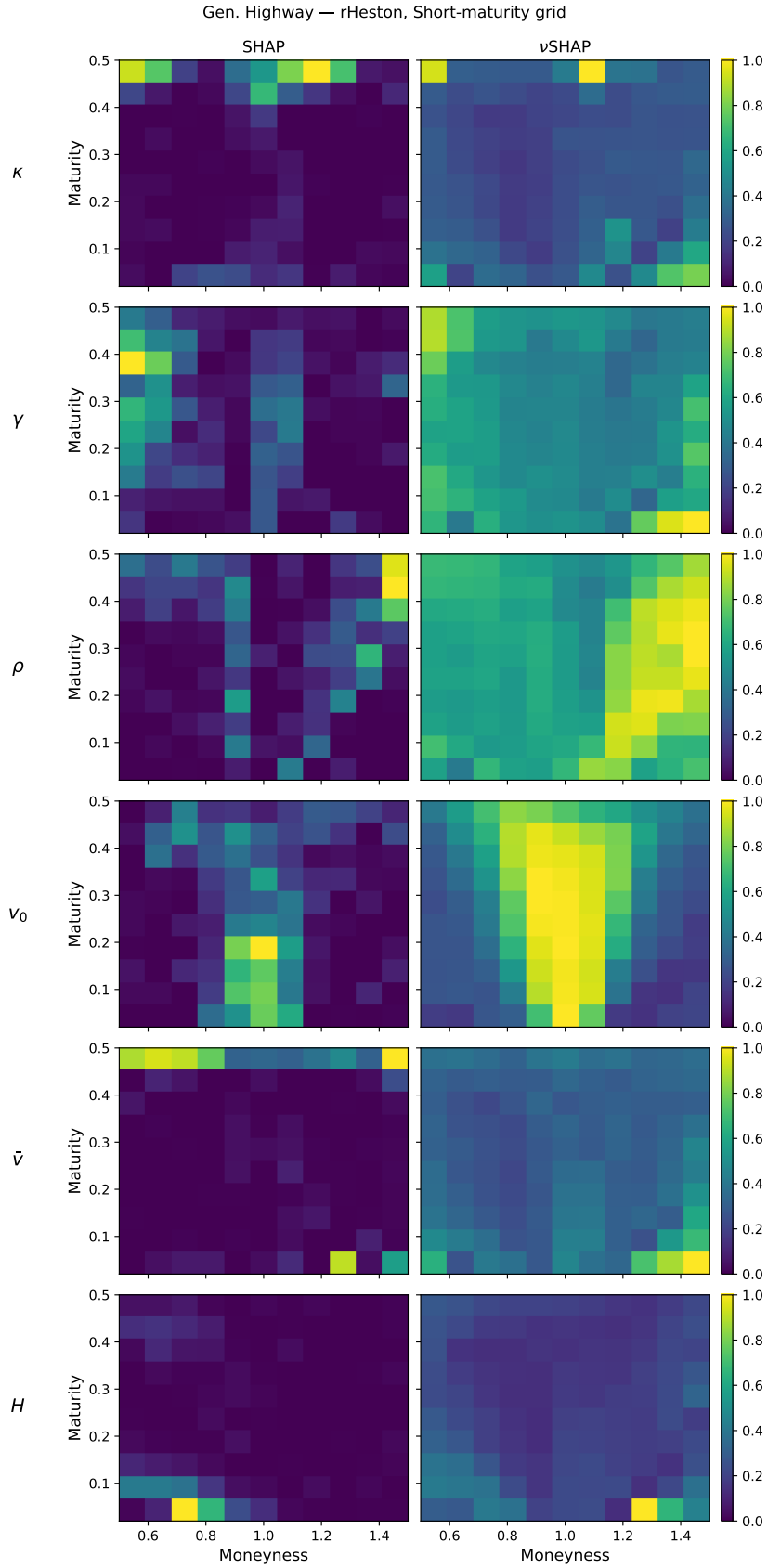


Figure 5.7: SHAP and νSHAP feature-importance heatmaps for the Generalised Highway network trained on the rough Heston model, on the grid with shorter maturities. Each cell corresponds to an option with a specific moneyness and maturity. Each row corresponds to a model parameter $(\kappa, \gamma, \rho, v_0, \bar{v}, H)$; the columns show normalised mean absolute SHAP-values and mean νSHAP-values across the moneyness–maturity grid.

κ

The SHAP heatmaps for κ show that the network assigns importance primarily to longer maturities on this grid. The highest attributions appear at the longest maturity, while shorter maturities receive considerably less importance. Across the moneyness dimension, importance is distributed across a wide range of levels without a clear concentration on a particular region.

While there are similarities, there are also differences to be observed when comparing the SHAP heatmap with the νSHAP heatmap. While attribution is more broadly spread across the surface, higher importance appears in the ATM region and the deep ITM region at the longest maturity on the grid. Considerable attributions are also visible in the short maturity deep OTM region.

γ

For γ, the SHAP heatmaps indicate that feature importance assigned by the network is mainly concentrated on the left side of the moneyness dimension, corresponding to deep ITM options. These attributions extend across a range of maturities, but higher importance is observed at intermediate and longer maturities on the grid.

Again, the νSHAP heatmap is much more diffuse across the grid when compared to SHAP. Importance is distributed broadly across both moneyness and maturity, and the highest attributions appear in the long maturity deep ITM and the short maturity deep OTM regions.

ρ

The heatmaps for SHAP and νSHAP corresponding to the correlation parameter ρ show more similarities to eachother. In both heatmaps, assigned feature importance is concentrated mainly on the right side of the moneyness dimension, corresponding to OTM options. A diagonal pattern of attributions seems to be visible, implying that options which are increasingly deeper OTM receive attributions if their maturity increases too. ATM options across all maturities receive importance, too. ITM options only seem to be slightly important with longer maturities; short maturity ITM options appear to receive little to no attributions in SHAP.

In the νSHAP heatmap, the same pattern can be observed, albeit much more diffuse. The attributions of ITM and ATM options seem to be more balanced with eachother, while OTM options clearly receive the largest attributions.

v_0

For the initial variance, again a similar pattern is observed for both SHAP and νSHAP. The ATM regions are highlighted in both heatmaps, spread across the maturity dimension. Deep ITM and deep OTM options appear to receive much lower attributions compared to ATM options, although with longer maturities some importance is still attributed to the extreme regions. In SHAP, the contrast is clearly visible, while the νSHAP heatmap is much more diffuse.

$\bar{v}$

The two methods differ greatly for the long term average variance $\bar{v}$. While SHAP mainly gives the longest maturity on the grid the highest attributions and virtually zero for the rest of the grid (aside from the short maturity OTM region), the νSHAP attributions are much more diffuse and in fact more concentrated around the OTM area ranging from very short to intermediate maturities on the grid.

H

Assigned importance by the network is concentrated at the short maturity deep ITM options in the SHAP heatmap for H; this matches what we observed earlier for the long-maturity grid. Other regions on the grid, aside from a minor attribution region for long maturity deep ITM, get virtually zero attributions.

Attributions are more diffuse for νSHAP, with the highest importance attributed to deep OTM very short maturity options. Furthermore, a notable attribution region is visible in the deep ITM short maturity region.

Feature ranking analysis

Inspection of the beeswarm plots for SHAP (Figure 5.8) and the νSHAP feature rankings (Figure 5.9) confirms several clear patterns in the most influential grid points. Across multiple parameters, the most important features are located either very close to maturity or at the longest maturity available on the grid. This is the case for both XAI methods.

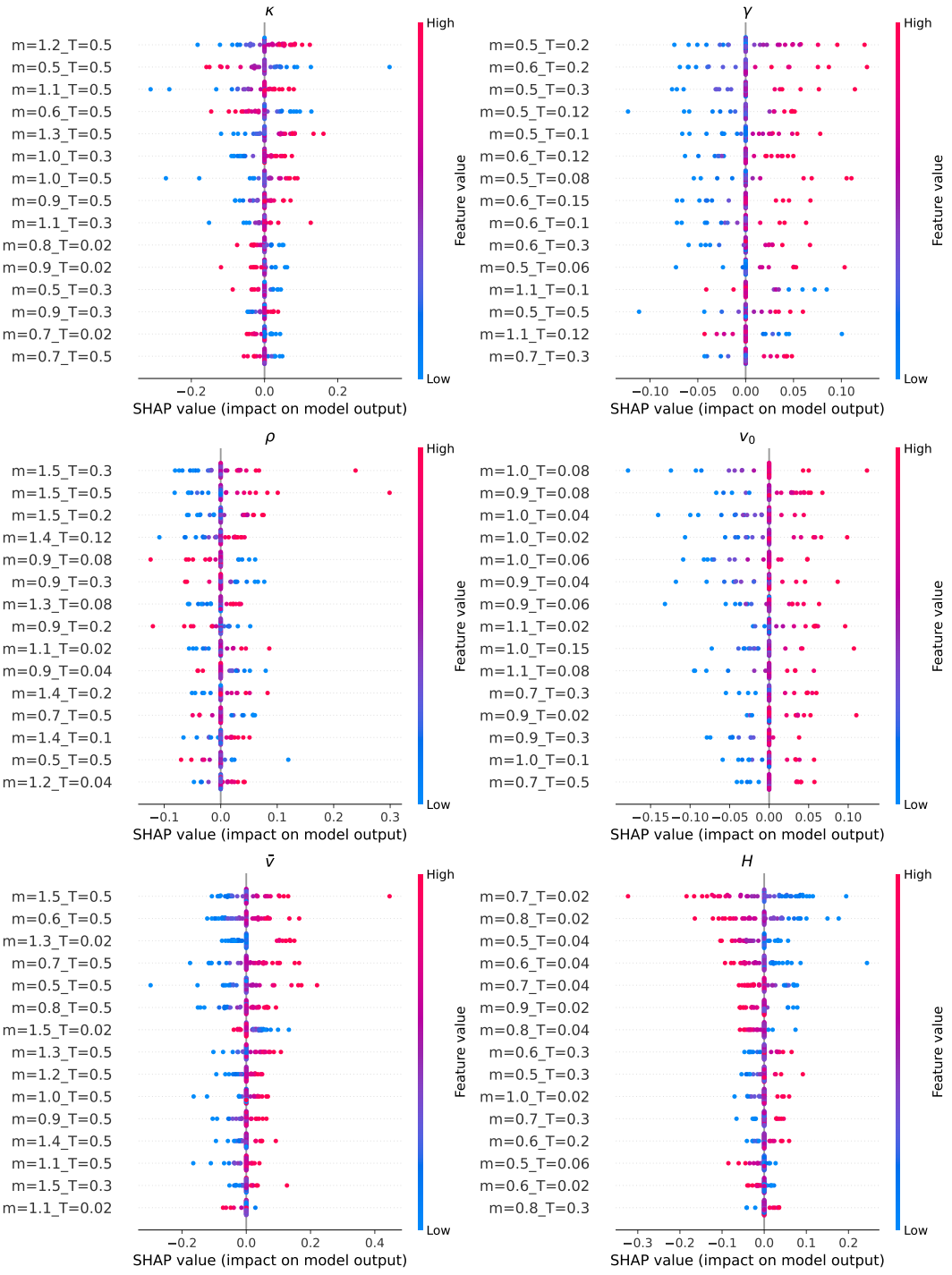


Figure 5.8: Beeswarm plots of SHAP-values for the Generalised Highway network trained on the rough Heston model, on the short-maturity grid. For each parameter ($\kappa, \gamma, \rho, v_0, \bar{v}, H$), the plots display the top–15 most important features ranked by mean absolute SHAP-value. Each point represents one explained instance, coloured by the corresponding feature value.

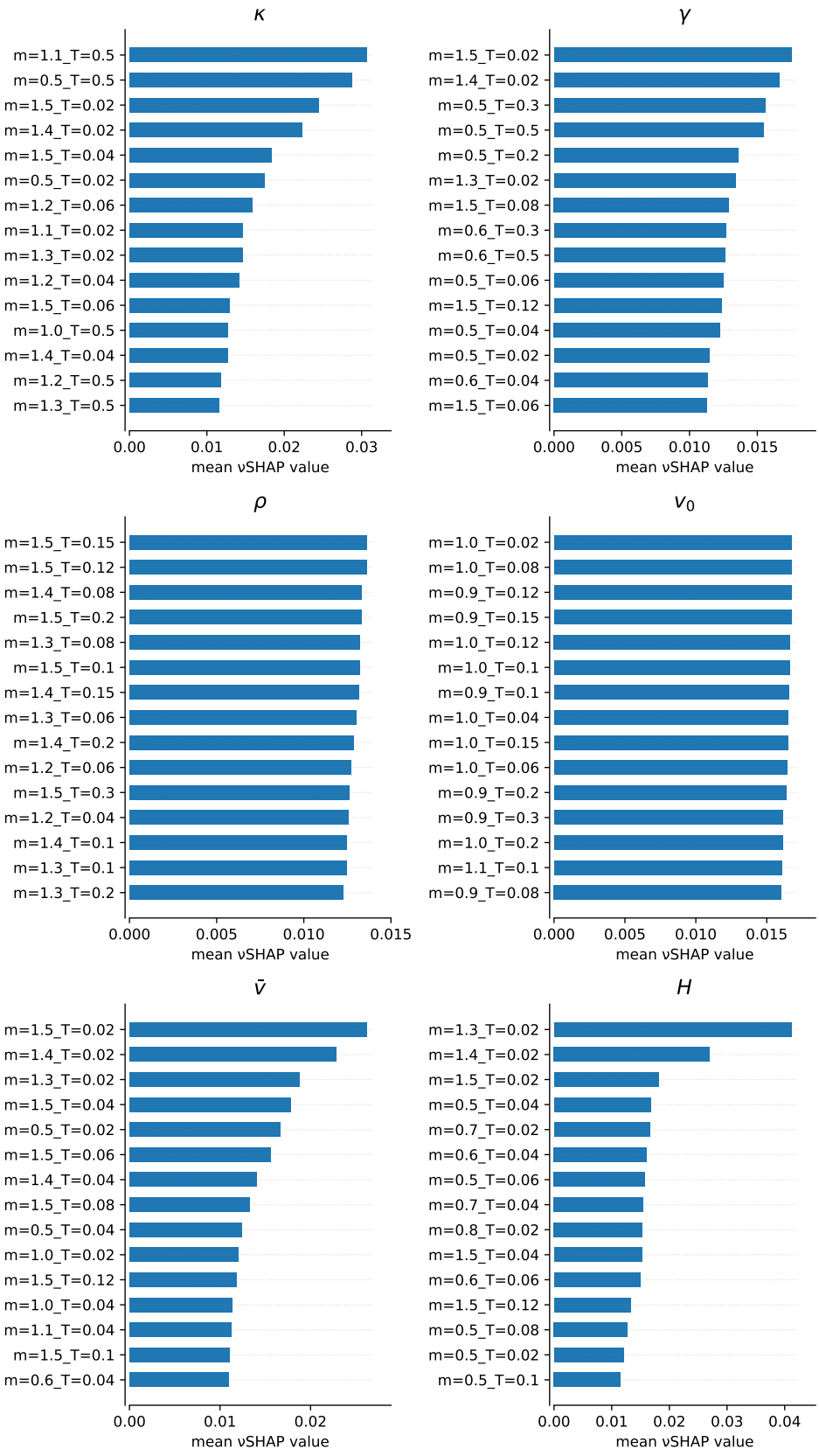


Figure 5.9: Bar charts showing the top–15 most important features for each (rHeston, Short-maturity grid) model parameter ($\kappa, \gamma, \rho, v_0, \bar{v}, H$) ranked by mean νSHAP-value. Explanations shown are for the Generalised Highway network. Each feature corresponds to a specific moneyness–maturity grid point.

Across the moneyness dimension, the influential features often lie at the extremes of the grid. Deep OTM and deep ITM options appear repeatedly among the top-ranked features for several parameters. This pattern is particularly visible for $\gamma, \bar{v}$ and H, where many of the highest-ranked features correspond to extreme moneyness levels.

The rankings further reveal that several parameters are dominated by a relatively small set of grid points. In particular, very short-maturity options frequently appear among the highest-ranked features in the νSHAP rankings. At the same time, the SHAP beeswarm plots show that these same grid points tend to exhibit the largest spread in SHAP values, indicating that they have the strongest influence on the predicted parameters.

Another observation is that the influential grid points are often located near the boundaries of the surface, either in the maturity dimension or along the wings of the smile. While interior grid points occasionally appear in the rankings, they are typically less prominent than boundary regions of the surface (except for v_0).

Overall, the feature rankings are consistent with the structural patterns observed in the heatmaps, while additionally highlighting the specific grid points that dominate the parameter predictions.

5.4. Architectural Comparison

The analyses in Section 5.3 revealed clear structural patterns in the features used by the neural networks to recover model parameters. In particular, the explanations consistently highlighted specific regions of the implied volatility surface, such as short-maturity options and the wings of the smile (deep ITM/OTM). These observations raise a natural question: whether the patterns are intrinsic to the inverse calibration problem, or whether they depend on the specific neural network architecture used to learn the mapping.

To investigate this, we compare the explanations obtained from the best-performing configuration of the MLP architecture with those obtained from the Generalised Highway architecture. Rather than analysing each architecture separately, we directly visualise the differences between their explanations on the moneyness–maturity grid. This allows us to assess whether both architectures rely on similar regions of the implied volatility surface, and to identify potential differences in the features used for parameter prediction.

Before presenting the results, we briefly describe how the difference heatmaps were constructed. For each architecture, the SHAP- and νSHAP explanations were first aggregated and normalised across the moneyness–maturity grid, as described in Section 5.3. The architectural differences were then computed as

$$\Delta_{\text{Architecture}} = \Psi_{\text{Gen.Highway(Normalised)}} - \Psi_{\text{MLP(Normalised)}}. \tag{5.24}$$

Consequently, positive values (red regions) indicate grid points where the Generalised Highway architecture assigns greater importance, whereas negative values (blue regions) correspond to regions where the MLP architecture assigns greater importance. Values close to zero indicate that both architectures attribute similar importance to the corresponding option for that parameter.

5.4.1. Heston

The difference heatmaps for the Heston model are shown in Figure 5.10. Overall, the SHAP difference heatmaps show that both architectures rely on broadly similar regions of the implied volatility surface, with most grid points exhibiting only small differences in attribution. Nevertheless, several structured differences are visible for certain parameters. We analyse the SHAP difference heatmaps first.

For κ, the MLP assigns relatively more importance to longer maturities, particularly on the left side of the moneyness grid and distributed across maturities. Light red clusters are visible in the ATM intermediate maturity region, implying that the Generalised Highway network assigns slightly more importance to that region.

The differences for γ are most pronounced at short maturities, but also present among the deep ITM options with longer maturity. The regions with short maturities show stronger attributions in Generalised Highway explanations, while the MLP has greater attributions for deep ITM options with intermediate to longer maturities.

For the correlation parameter ρ, the difference heatmap is relatively diffuse across the grid, with no single region dominating the differences. The strongest difference can be observed in the short maturity OTM region; the MLP attributions there are greater than the Generalised Highway attributions.

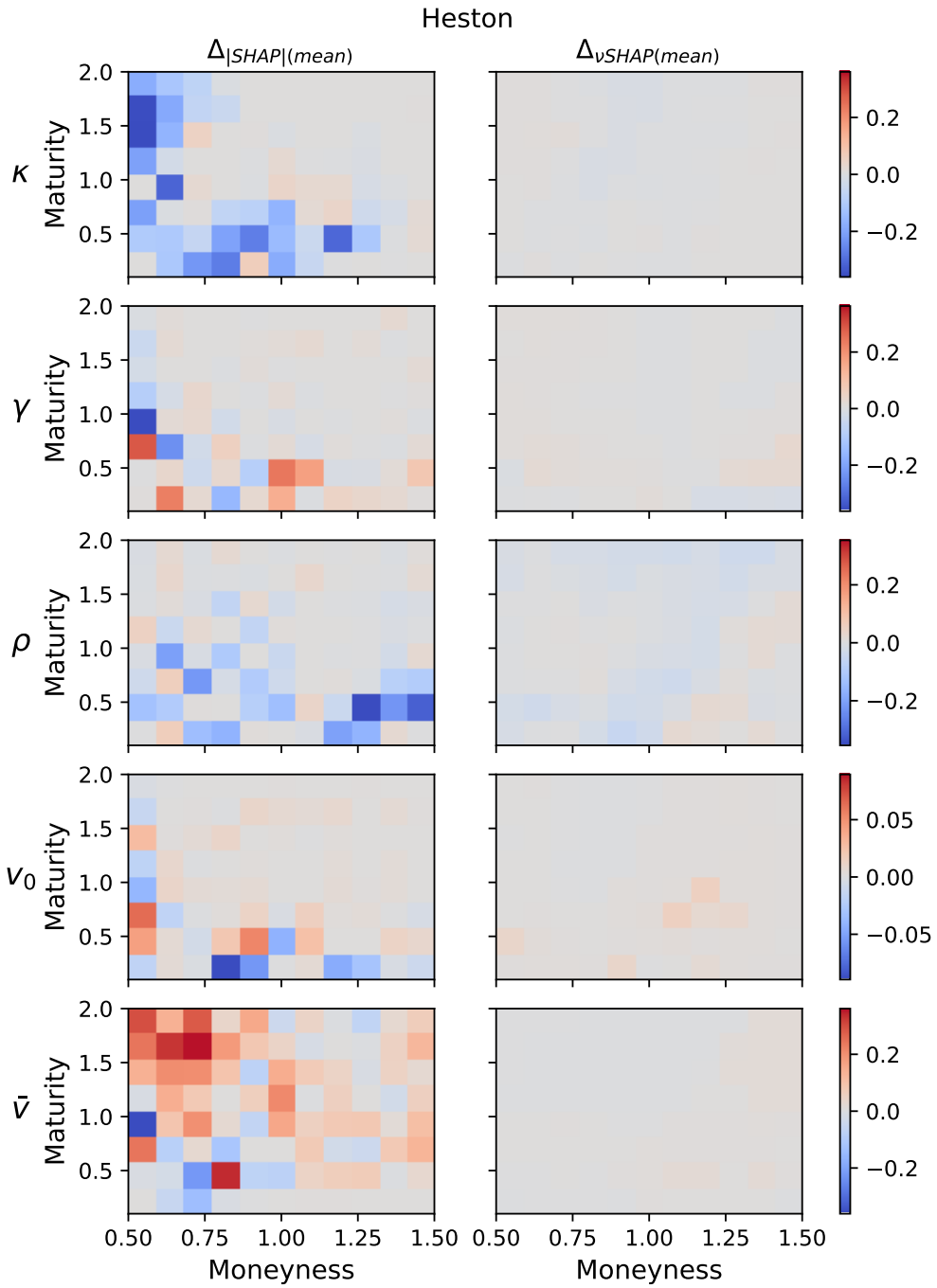


Figure 5.10: Difference heatmaps comparing explanations of the Generalised Highway and MLP architectures for the Heston model. Differences are computed as in Equation 5.24. Blue indicates higher attribution by the MLP architecture; Red indicates higher attribution by the Generalised Highway architecture.

While the differences appear prominent on the remainder of the grid, their magnitude remains limited compared to other parameters.

While the difference heatmap for v_0 does show coloured regions, the magnitude is minor when compared with the other parameters. For the long-term variance $\bar{v}$, the difference heatmap exhibits the largest structured differences between the two architectures on the Heston model. In particular, the Generalised Highway network assigns substantially more importance to longer maturities, especially in the deep ITM region. Concurrently, the MLP attributes relatively higher importance to certain shorter-maturity grid points.

In contrast to the SHAP differences, the νSHAP difference heatmaps remain close to zero across the entire grid for all parameters. This indicates that the νSHAP explanations obtained for the two architectures are highly similar across the moneyness–maturity surface.

5.4.2. Rough Heston

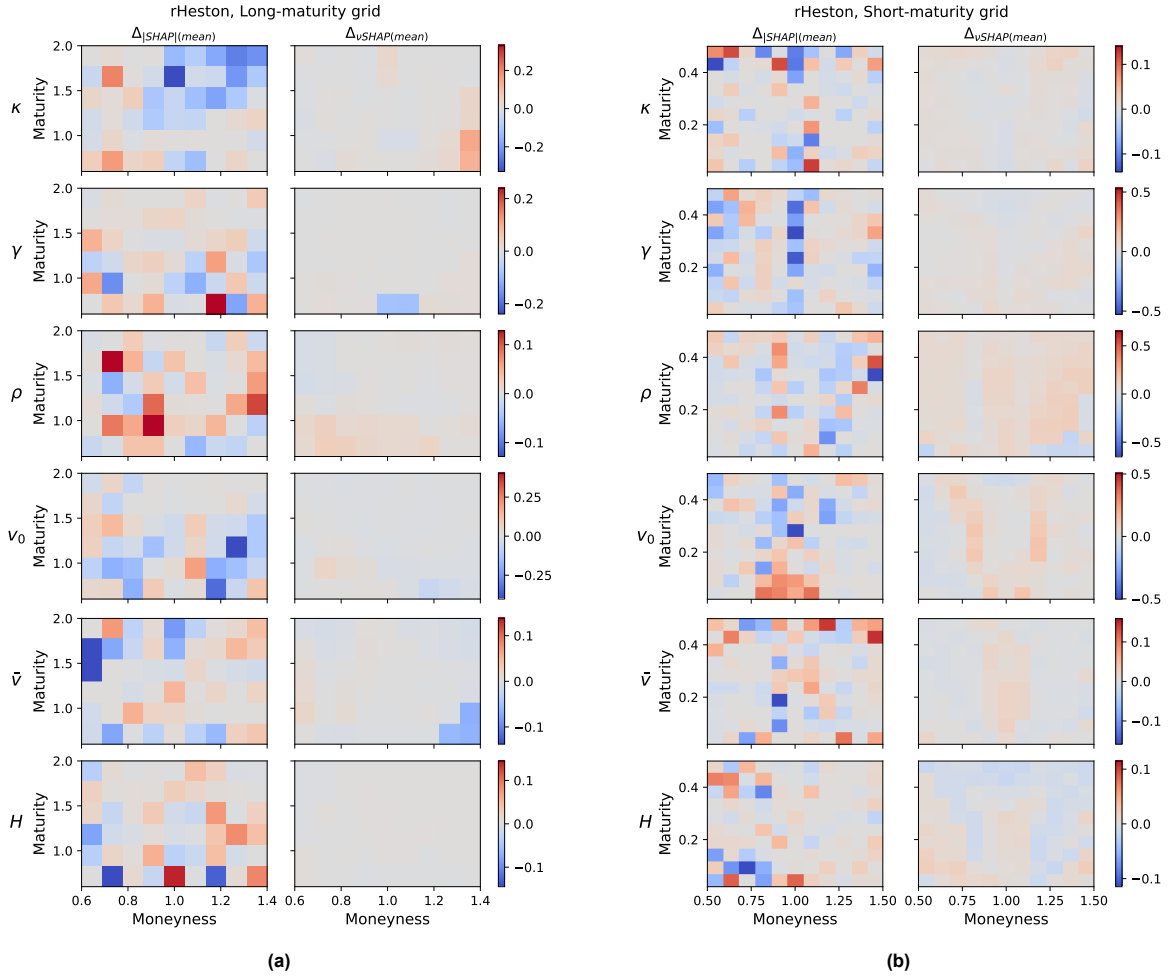


Figure 5.11: Difference heatmaps comparing explanations of the Generalised Highway and MLP architectures for the Rough Heston model. Differences are computed as in Equation 5.24. Blue indicates higher attribution by the MLP; Red indicates higher attribution by the Generalised Highway architecture.

The architectural difference heatmaps for the rough Heston model on the long-maturity grid (Figure 5.11a) and short-maturity grid (Figure 5.11b) reveal distinct patterns for SHAP and νSHAP, across both grids. For SHAP, the differences between the MLP and Generalised Highway architectures are distributed across the moneyness–maturity grid, and do not exhibit a clear spatial structure on either grid. While some local regions display larger differences, these appear scattered rather than concentrated in specific parts of the surface.

A notable difference between the two grids is the scale of the SHAP differences. On the short-

maturity grid, several parameters exhibit substantially larger differences than on the long-maturity grid. In particular, for γ, ρ and v_0, absolute differences reach values of approximately 0.5 on the short-maturity grid, whereas on the long-maturity grid the differences for the same parameters remain limited to roughly $0.2 - 0.25$. For $\kappa, \bar{v}$ and H, the magnitude of the differences appears to be comparable across both grids.

In contrast, the νSHAP difference heatmaps show only minor differences between the two architectures. On the long-maturity grid, small but visible differences appear in localised regions of the surface. For example, slightly higher νSHAP attribution by the Generalised Highway network can be observed in the deep OTM short-maturity region for κ, while the MLP assigns somewhat higher importance to the short-maturity ATM region for γ and to the deep OTM short-maturity region for $\bar{v}$. However, the magnitude of these differences remains small. On the short-maturity grid, the νSHAP differences are even less pronounced and appear diffuse across the surface without clear concentrated regions.

Overall, the architectural differences are considerably more visible in the SHAP explanations than in the νSHAP explanations, while the latter remain relatively similar across architectures for both grids.

Having compared the explanations across architectures, we next investigate whether the observed patterns remain stable across different parameter regimes of the Heston model.

5.5. Feller-Regime Analysis

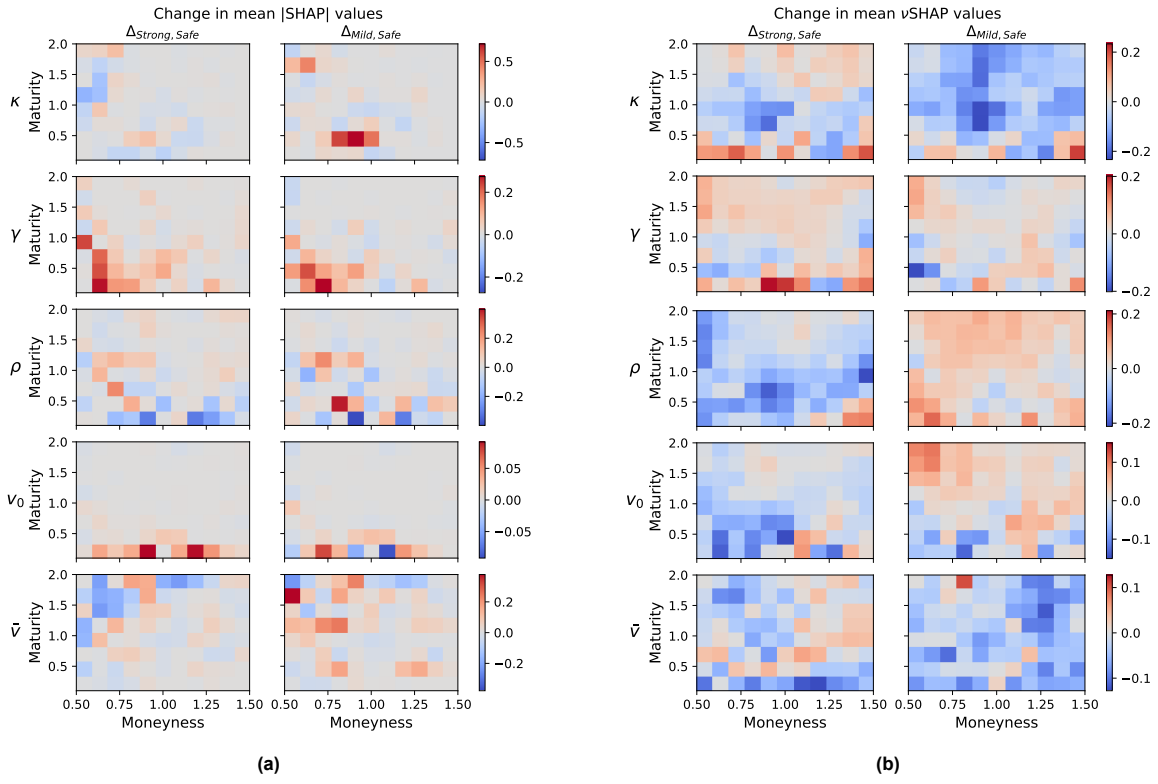


Figure 5.12: Difference heatmaps comparing explanations of the MLP architecture for the different parameter regimes of the Heston model. Differences are computed as in Equations (5.26) & (5.27). Blue indicates higher attribution by the Strong or Safe parameter regimes; Red indicates higher attribution by the Safe parameter regime.

We are interested in whether the explanations for the calibration network remain stable across different parameter regimes of the Heston model. In particular, different regimes of the variance process may lead to qualitatively different shapes of the implied volatility surface, which could affect how information about the model parameters is encoded in the option grid. To investigate this, we analyse how the explanations change across regimes characterised by the Feller ratio.

To this end, recall that the Feller ratio ϕ is defined as

$$\phi = \frac{2\kappa\bar{v}}{\gamma^2}, \tag{5.25}$$

and in our experiments, a set of parameters is in the *Safe* regime if $\phi \geq 1$; in the *Mild* or Mildly violating regime if $0.7 \leq \phi < 1$, and in the *Strong* or Strongly violating regime if $0.3 \leq \phi < 0.7$.

In a similar fashion as Section 5.4, we compute the (normalised) difference heatmaps with

$$\Delta_{\text{Strong,Safe}} = \Psi_{\text{Strong}} - \Psi_{\text{Safe}}, \tag{5.26}$$

$$\Delta_{\text{Mild,Safe}} = \Psi_{\text{Mild}} - \Psi_{\text{Safe}}. \tag{5.27}$$

For each regime, we randomly sampled 50 instances and estimated SHAP-and νSHAP-values for each instance, with the same setup as described in Section 5.2. The difference heatmaps are shown in Figure 5.12, where Figure (a) shows the difference in normalised mean absolute SHAP values, and Figure (b) shows the difference in normalised mean νSHAP values. They illustrate how the explanations change across Feller regimes, where the left column shows the difference between the Strong and Safe regimes and the right column shows the difference between the Mild and Safe regimes. Blue regions correspond to higher attribution in the Safe regime, whereas red regions indicate higher attribution in the Strong or Mild regimes.

Across both comparisons, the SHAP difference heatmaps show more structured and concentrated differences than the corresponding νSHAP heatmaps. In particular, several parameters exhibit clearly visible local differences in SHAP attribution across the moneyness–maturity grid, whereas the νSHAP differences appear more diffuse and distributed across the surface. In addition, the scale of the differences for some parameters is larger for SHAP. For example, the SHAP differences for κ reach values of approximately 0.5, while the νSHAP differences remain limited to roughly 0.2.

For some parameters, the SHAP differences reveal clear regional shifts in attribution. For κ, substantially less importance is assigned to the ATM region in the Safe regime at relatively short maturities compared to the Mild regime. For γ, the Strong and Mild regimes similarly show higher attribution in the deep ITM region at short to intermediate maturities when compared with the Safe regime. In contrast, the differences for ρ appear more diffuse across the surface, although the Safe regime displays slightly higher attribution at shorter maturities. For v_0, the differences remain negligible across the grid. In the Safe regime, somewhat higher SHAP importance seems to be attributed to long maturity deep ITM options compared to the Strong regime for $\bar{v}$, while slightly higher importance is assigned to this region in the Mild regime compared to the Safe regime.

The νSHAP difference heatmaps exhibit smaller differences across regimes. Although small variations are visible across the grid, they remain relatively diffuse and of limited magnitude. For κ, slightly higher attribution is assigned to the deep OTM short-maturity region in the Mild and Strong regimes compared to the Safe regime. For ρ, the Safe regime shows somewhat higher attribution across much of the grid relative to the Strong regime, while the Mild regime assigns slightly higher attribution across the grid compared with the Safe regime. For v_0 and $\bar{v}$, the differences remain small, although the Safe regime tends to assign slightly higher attribution across several grid points.

Overall, the regime-dependent differences are more clearly visible in the SHAP explanations, while the νSHAP differences remain comparatively small and more diffusely distributed across the grid.

5.6. νSHAP-guided Grid Reduction

While the previous analyses focused on understanding how the neural networks use information across the implied volatility surface, the explanations themselves may also provide guidance for grid design. In particular, the νSHAP heatmaps reveal systematic differences in attribution across the grid, suggesting that some regions of the surface carry more information for parameter recovery than others. This motivates a grid design experiment in which the option grid used for training the network is reduced based on the observed attribution patterns.

The experiment is performed for the rough Heston model on the long-maturity grid. The original grid consists of 9 moneyness levels and 6 maturities, yielding 54 input features per surface. Inspection of the νSHAP heatmaps (Figure 5.4) shows that the highest network attributions are consistently concentrated at the shortest maturities on the surface, while the interior of the grid and longer maturities receive

substantially lower attribution across parameters. Based on this observation, we construct a reduced grid by retaining only the two shortest maturities while preserving all moneyness levels. The resulting grid contains 18 options, corresponding to a reduction of approximately 67% in the number of input features.

This choice is also consistent with practical considerations in option markets, where short-maturity options and near-the-money strikes are typically the most liquid and reliably quoted. Consequently, restricting the grid to short maturities yields a reduced input representation that is both guided by the νSHAP explanations and aligned with commonly available market data.

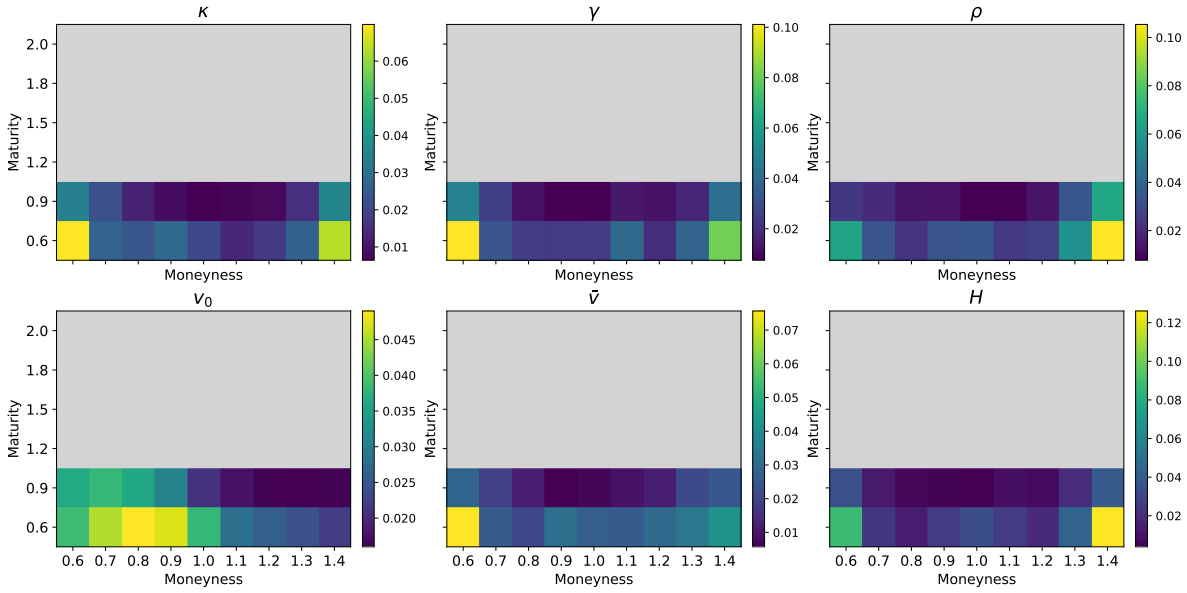


Figure 5.13: Mean νSHAP heatmaps for the Generalised Highway network, trained on the Rough Heston model on the long-maturity grid. Grey cells indicate grid points removed in the νSHAP-guided grid reduction experiment, while coloured cells correspond to the grid points retained for training the reduced-grid model.

Figure 5.13 illustrates the νSHAP heatmaps used to guide this reduction. The coloured cells correspond to grid points retained in the reduced grid, while the grey cells indicate grid points that are removed in the reduced-grid experiment. As visible in the figure, the 2 shortest maturities of the implied volatility surface comprise the retained grid points, while the intermediate and long maturity regions with low mean νSHAP values are excluded from the reduced grid.

Using this reduced grid, the Generalised Highway architecture is retrained. Only the input data is modified; the architecture and training procedure remain unchanged. The predictive performance of the resulting model is then compared to the model trained on the full grid. Table 5.1 reports the mean absolute error (MAE) obtained for each parameter under both training configurations.

Table 5.1: Mean absolute errors (MAE) for each rough Heston parameter obtained with the best-performing Generalised Highway configuration on the long-maturity grid, trained on the full option grid and on the νSHAP-guided reduced grid containing approximately 67% fewer grid points.

Grid Type	κ	γ	ρ	v_0	$\bar{v}$	H	$\overline{\text{MAE}}_{\text{test}}$
Full Grid	0.00114	0.00045	0.00037	0.00020	0.00036	0.00021	0.00046
Reduced Grid	0.00106	0.00030	0.00030	0.00012	0.00036	0.00017	**0.00039**

To further illustrate the effect of the grid reduction on the distribution of prediction errors, Figure 5.14 shows the empirical quantiles of the relative prediction errors for each parameter.
From Figure 5.14, it can be observed that the error distributions obtained using the reduced grid remain of comparable magnitude to those obtained using the full grid across all parameters. In fact, the

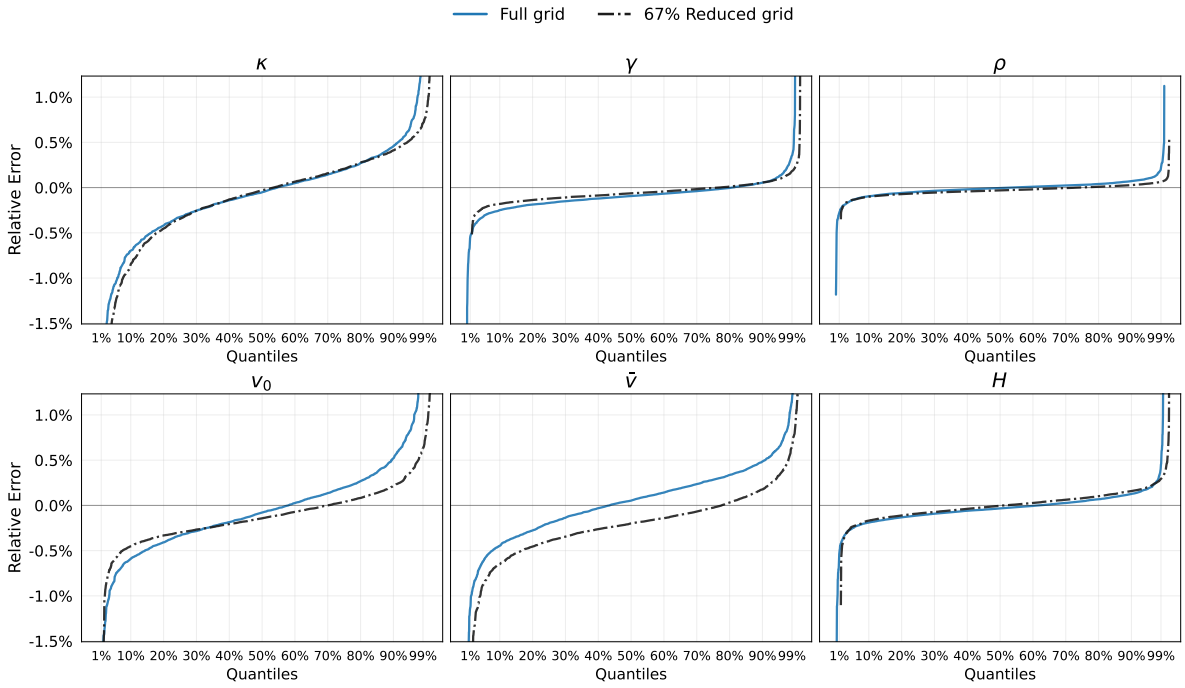


Figure 5.14: Empirical quantiles of relative prediction errors for the Generalised Highway network trained on the Rough Heston model with long-maturity grid, trained on the full grid and the 67% reduced grid. For visual clarity, the plot is restricted to the 1^{st}–99^{th} quantile range.

reduced-grid model yields slightly lower prediction errors, as reflected by the MAE values reported in Table 5.1. Overall, the prediction errors remain in the order of 10^{-4} to 10^{-3} for all parameters, despite the substantial reduction in the number of grid points used for training. This suggests that a large portion of the information required for parameter recovery is already contained in short-maturity regions of the implied volatility surface.

To conclude, these results illustrate that explainability methods such as νSHAP can provide actionable guidance for model design, enabling substantial reductions in input dimensionality while retaining comparable predictive performance.

5.7. Summary of Findings

We applied two explainable artificial intelligence (XAI) methods to analyse the structure of the learned calibration networks: SHAP and νSHAP. While SHAP attributes importance based on the sensitivity of the output with respect to each feature, νSHAP identifies which features are sufficient for the output to remain stable and attributes importance accordingly. These approaches therefore provide two complementary notions of explanation.

Across the SHAP and νSHAP analyses, several consistent patterns emerge. First, the most influential features for parameter recovery are concentrated near short maturities and at the wings of the implied volatility surface, while interior regions of the grid contribute comparatively less when considering the Rough Heston model on the long-maturity grid. Second, the explanations remain broadly consistent across different neural network architectures, suggesting that the learned inverse mappings rely on similar regions of the surface. Third, the comparison across Feller regimes for the Heston model shows that while some local differences in attribution arise, the overall explanatory patterns remain stable. Finally, the νSHAP-guided grid reduction experiment demonstrates that the dimensionality of the option grid can be substantially reduced without deteriorating predictive performance.

An important observation is that discrepancies between SHAP and νSHAP explanations are not indicative of inconsistency or model failure, but rather reflect the fundamentally different notions of feature relevance captured by the two methods. While SHAP highlights features to which the model output is sensitive, νSHAP identifies subsets of features that are sufficient to preserve the prediction. Divergence between the two therefore provides additional insight into the structure of the learned mapping,

in particular by revealing redundancy and non-uniqueness in the inverse calibration problem.

Taken together, these findings provide evidence that the calibration networks rely on structurally consistent regions of the implied volatility surface, while the complementary perspectives of SHAP and νSHAP offer additional insight into redundancy and structural properties of the inverse mapping. This illustrates how explainability methods can be used not only to interpret model predictions, but also to inform model design.

6

Discussion & Conclusions

6.1. Discussion

The preceding chapters evaluated several neural network architectures for the inverse calibration problem from both a predictive and a structural perspective. Chapter 4 established the empirical performance of the models, while Chapter 5 analysed the learned mappings using XAI methods. In this chapter, we synthesise these results, discussing calibration performance and structural insights in turn.

6.1.1. Calibration Performance and Practical Suitability

The empirical results of Chapter 4 demonstrate that neural network-based calibration provides accurate and computationally efficient approximations of the inverse mapping from implied volatility surfaces to model parameters. Across both the Heston and rough Heston models, the Generalised Highway network consistently achieves the lowest prediction errors, followed by the standard Highway network and the MLP. While this ranking is stable across most configurations, the magnitude of the performance gap depends on the model and grid choice.

A notable difference between the Highway and Generalised Highway architectures is their behaviour as the network depth increases. While the performance of the standard Highway network deteriorates when additional layers are introduced, the Generalised Highway architecture benefits from increased depth. This difference can be understood in terms of the gating mechanisms underlying both architectures.

In the standard Highway network, a single gate controls the trade-off between transforming the input and carrying it forward. As a result, the model is forced to balance preservation and transformation of information within a single degree of freedom. When stacking multiple layers, this constraint can lead either to excessive information preservation, effectively limiting the network to shallow behaviour, or to excessive transformation, which may distort relevant structural information in the input.

In contrast, the Generalised Highway architecture with joint softmax-gating introduces a flexible gating mechanism that allows the network to combine transformed and untransformed representations more freely. This additional flexibility enables the model to preserve relevant features of the implied volatility surface while simultaneously extracting higher-level representations across layers.

This distinction is particularly relevant in the context of the inverse calibration problem, where both local features, such as short-maturity behaviour and smile shape, and global structures, such as the term structure of variance, contribute to parameter recovery. The ability to propagate and transform information simultaneously allows the Generalised Highway network to exploit this multi-scale structure more effectively, which explains its improved performance with increasing depth. This highlights that the standard Highway architecture imposes a structural limitation on the representation of the inverse mapping, whereas the Generalised Highway architecture alleviates this constraint.

The MLP, while generally less accurate than the Generalised Highway architecture, provides a competitive and computationally efficient baseline. In particular, for the rough Heston model on the short-maturity grid, the performance gap between the MLP and the Generalised Highway network is relatively small. This indicates that, when the input representation already contains highly informative features, a simple feed-forward architecture may be sufficient to capture the dominant structure of the inverse mapping. However, the MLP tends to exhibit larger errors in more complex settings and

appears less robust in the tails of the parameter distributions.

From a practical perspective, these results highlight a clear trade-off between accuracy and computational cost. The Generalised Highway network offers the best predictive performance but at the expense of substantially longer training times and increased implementation complexity. The MLP, on the other hand, is easy to train and deploy, and may be preferable in applications where computational efficiency is critical and small losses in accuracy are acceptable. The standard Highway network occupies the intermediate position, but is generally dominated by the Generalised Highway architecture in both accuracy and robustness.

A particularly important finding is the strong impact of input whitening. Across all models and architectures, whitening the input features leads to substantial improvements in both convergence behaviour and predictive accuracy. This effect can be attributed to the high degree of correlation present in implied volatility surfaces, where neighbouring grid points exhibit strong linear dependencies. Whitening removes these correlations and rescales the input space, resulting in a better-conditioned optimisation problem for gradient-based training in a highly correlated input space. The magnitude of the observed improvements suggests that whitening is not merely a preprocessing refinement, but a critical component for achieving reliable calibration performance.

The OOD experiments further reveal important limitations of the proposed approach. While all networks perform well within the training domain, their accuracy deteriorates when evaluated on parameter regimes outside the training distribution. This effect is particularly pronounced for the standard Highway network, whereas the MLP and Generalised Highway architectures exhibit relatively better, but still degraded, performance. These findings indicate that the learned mappings primarily interpolate within the training distribution and should not be expected to extrapolate reliably. Consequently, practical deployment of neural network-based calibration requires careful control of the parameter domain and training data coverage.

6.1.2. Structural Insights and Identifiability

Beyond predictive performance, the central objective of this thesis is to assess whether the learned inverse calibration maps exhibit structurally meaningful behaviour. To this end, Chapter 5 employed SHAP and νSHAP to analyse the feature attributions of the trained networks. The resulting attribution patterns can be interpreted in light of the meaning behind the model parameters, providing a partial link between the learned mappings and financial intuition.

The implied volatility surface encodes different types of information across strikes and maturities, and the observed explanations largely align with theoretical expectations. Variance-related parameters exhibit the clearest structure. The initial variance v_0 is primarily identified from short-maturity ATM options, where instantaneous variance has the strongest effect on prices. In contrast, the long-term variance $\bar{v}$ is mainly associated with intermediate- to long-maturity options, reflecting its role as the asymptotic variance level.

The parameters ρ and γ, governing skew and curvature of the implied volatility surface, are predominantly identified from the wings of the smile, with a clear concentration at short maturities where these effects are most pronounced. For ρ, both SHAP and νSHAP consistently highlight deep ITM and deep OTM options, reflecting the role of correlation in determining the slope of the skew. Similarly, γ is mainly inferred from short maturities across the smile, where the curvature induced by volatility-of-volatility is most visible.

In contrast, parameters governing the dynamics of the variance process, such as the mean-reversion speed κ and the Hurst parameter H, exhibit more diffuse and heterogeneous attribution patterns. For κ, the absence of a clear and consistent structure reflects its indirect influence on the implied volatility surface through the term structure of variance, rather than through localised shape characteristics. Correspondingly, νSHAP assigns importance to short-maturity wings, indicating that the network can preserve its prediction based on local features of the short-maturity smile, rather than relying on a single dominant region. For H, both methods emphasise short maturities, consistent with the fact that roughness primarily affects near-maturity behaviour where the implied volatility skew becomes particularly steep in rough volatility models [12]. Across moneyness, the attribution patterns are concentrated in the wings of the smile. In particular, SHAP assigns the highest importance to the deep in-the-money region, while νSHAP places primary emphasis on the deep out-of-the-money side, with additional contributions from the opposite wing. This asymmetry reflects the fact that the short-maturity skew is encoded across both sides of the smile, and that the network extracts information about H from regions where the slope

of the smile is most pronounced.

Building on this financial interpretation, the degree of agreement between SHAP and νSHAP depends on the model under consideration. For the Heston model, both methods produce broadly consistent attribution patterns, whereas for the rough Heston model the agreement is weaker and the explanations exhibit greater heterogeneity. For most parameters, both methods identify similar regions of the implied volatility surface as being most informative for parameter recovery, most notably short maturities and the wings of the smile. This consistency provides evidence that the learned mappings are not driven by arbitrary statistical artifacts, but instead rely on stable and economically meaningful elements of the input. In this sense, the agreement between SHAP and νSHAP suggests a degree of *meaningful identifiability*: the parameters can be recovered from structurally interpretable, though not always uniquely defined, regions of the surface.

Despite the overall agreement, systematic differences between SHAP and νSHAP are observed. SHAP explanations tend to be highly concentrated, assigning high importance to a relatively small set of features. Conversely, νSHAP explanations are typically more diffuse, distributing importance over broader regions of the surface. This difference can be understood in terms of the underlying value functions, i.e. Equations (5.11) and (5.16). SHAP attributes importance based on marginal changes in expected model output, thereby emphasising features with strong predictive influence. In contrast, νSHAP evaluates whether a set of features is sufficient to preserve the model output, which naturally leads to more distributed attributions when multiple features jointly encode relevant information. As a result, SHAP highlights the most predictive regions, while νSHAP captures the minimal sufficient structure of the input.

The comparison across neural network architectures further reinforces the structural stability of the learned mappings. While SHAP reveals some differences in distribution of feature importance across architectures, these differences are relatively minor and largely confined to local variations. On the other hand, νSHAP explanations are nearly identical across architectures, indicating that the essential feature subsets required for parameter recovery remain stable. This suggests that different architectures may redistribute predictive influence across correlated features, but ultimately rely on the same underlying information.

The analysis across Feller regimes in the Heston model provides additional insight into the robustness of the learned mappings. For SHAP, the differences between regimes are generally localised and relatively limited in magnitude, suggesting that the main explanatory structure is preserved across regimes. For νSHAP, the difference heatmaps appear more spatially diffuse and pronounced, indicating that regime-dependent variation is also present under the sufficiency-based value function. However, these differences likewise remain limited in magnitude. Taken together, the results suggest that the learned inverse mapping is not entirely invariant across regimes, but that the overall attribution structure remains broadly stable, with regime effects appearing mainly as moderate local or distributed shifts rather than a substantial reorganisation of feature importance.

A particularly informative experiment is the νSHAP-guided grid reduction. By restricting the input to the most important features identified by νSHAP, it is possible to achieve comparable, and in some cases even improved, predictive performance relative to the original grid. This finding suggests that the original input representation contains a significant degree of redundancy, and that the essential information required for parameter recovery is concentrated in a relatively small subset of the surface. Importantly, this demonstrates that XAI methods can be used not only for post hoc interpretation, but also as a tool for improving model design. However, as this experiment is conducted under a specific configuration, these results should be interpreted as indicative rather than definitive.

Taken together, the XAI analysis indicates that the learned calibration networks rely on structurally meaningful regions of the implied volatility surface, and that these regions largely align with financial intuition for several key parameters. At the same time, the divergence between SHAP and νSHAP for parameters such as κ highlights that the inverse mapping is not uniformly well-posed and exhibits parameter-dependent levels of identifiability, with certain parameters exhibiting redundancy and weaker identifiability. Thus, while this does not constitute a formal proof of identifiability in the classical sense of an inverse problem [35], it supports the conclusion that the calibration networks capture stable and interpretable relationships between inputs and outputs. These findings further indicate that the inverse calibration problem is inherently non-homogeneous across parameters, with some parameters being strongly anchored in specific regions of the surface, while others are only indirectly identifiable through correlated features.

6.1.3. Limitations

Despite the promising results, several limitations of the proposed framework should be acknowledged. These can be broadly categorised into model, data, and methodological limitations.

Model limitations The neural networks considered in this thesis primarily act as interpolators within the training domain. As demonstrated by the out-of-distribution experiments, their predictive accuracy deteriorates when evaluated on parameter regimes not represented in the training data. This indicates that the learned inverse mappings do not generalise reliably beyond the training distribution, which limits their direct applicability in settings where the parameter space is not well covered. Furthermore, the inverse calibration problem itself is inherently ill-conditioned, and certain parameters exhibit weak identifiability, implying that multiple parameter configurations may correspond to similar implied volatility surfaces.

Data limitations All experiments in this thesis are conducted on synthetically generated data. While this allows for controlled experimentation and precise evaluation of model performance, it does not capture the characteristics of real market data. In practice, implied volatility surfaces are affected by noise, liquidity constraints, and non-uniform data availability across strikes and maturities. As a result, the patterns observed in this thesis may not transfer directly to real-world settings, where the input data may be less reliable and non-homogeneous.

Methodological limitations The structural validation analysis relies on post hoc XAI methods, namely SHAP and νSHAP, which provide complementary but indirect insights into the learned mappings. While these methods reveal meaningful patterns, they do not constitute a formal proof of identifiability or correctness of the inverse mapping. Moreover, the interpretation of attribution patterns remains partly qualitative and depends on the chosen value functions and assumptions underlying the XAI methods. Consequently, the conclusions drawn from the XAI analysis should be interpreted as indicative rather than definitive.

6.1.4. Further Research

Several directions for future research emerge from this work.

Since the experiments in this thesis are conducted on synthetically generated data, a natural extension is the application of the proposed calibration framework to real market data. In practice, implied volatility surfaces are observed with non-uniform liquidity, with the highest trading activity typically concentrated in the ATM region and shorter maturities. As a result, the reliability of observed implied volatilities varies across the surface, which may affect both model performance and the structure of the learned inverse mappings. Investigating how neural networks adapt to such heterogeneous data quality, and whether the most informative regions identified in the synthetic setting remain dominant and stable under real market conditions, constitutes an important direction for future research. In particular, it would be of interest to assess whether the strong importance of short maturities and the wings of the smile observed in this thesis persists when accounting for market liquidity and noise.

A further important direction concerns the incorporation of uncertainty quantification into the inverse calibration framework. The current networks produce point estimates of the parameters, implicitly treating the mapping from implied volatility surfaces to model parameters as if it were well-posed. However, in practice, the inverse problem is often ill-conditioned, and multiple parameter configurations may yield similar implied volatility surfaces, particularly in more complex models such as rough Heston. Incorporating uncertainty-aware predictions would allow the network to express this ambiguity explicitly, for instance by assigning confidence intervals or distributions to the estimated parameters. This is particularly relevant in practical applications, where parameter uncertainty directly translates into model risk and can significantly affect related tasks such as pricing, hedging, and risk management. Extending the framework to capture such uncertainty would therefore provide a more realistic and robust approach to neural network-based calibration. A potential approach would be to utilise Bayesian neural networks, which allow approximation of the full posterior distribution of the parameters given an observed implied volatility surface.

Moreover, the results of the νSHAP-guided grid reduction experiment suggest that XAI methods can be used to inform the design of input representations. Future work could explore this idea more system-

atically, for example by constructing adaptive or data-driven grids that focus on the most informative regions of the implied volatility surface.

Lastly, one could consider the use of XAI methods to better understand parameter identifiability in the inverse calibration problem. While the analyses in this thesis already provide qualitative evidence that identifiability varies across parameters, they also suggest that certain parameters are inherently more difficult to distinguish based on the observed surface. Future work could build on this by developing more systematic and quantitative approaches, for instance by using attribution patterns to measure the degree of overlap between parameters or the extent to which importance is distributed across the surface. Such methods could help detect ambiguities more rigorously, for example by identifying parameters that depend on similar regions of the input or exhibit highly distributed importance, thereby providing a principled way to assess parameter identifiability in inverse calibration problems.

6.2. Conclusions

This thesis investigated the use of neural networks for the inverse calibration of stochastic volatility models, with the aim of addressing the tension between predictive accuracy and interpretability in quantitative finance. Focusing on the Heston and rough Heston models, the study evaluated multiple neural network architectures and analysed their behaviour using XAI methods.

From a predictive perspective, the results demonstrate that neural networks provide an accurate and computationally efficient approach to approximating the inverse mapping from implied volatility surfaces to model parameters. In particular, the Generalised Highway architecture consistently achieves the strongest performance across a wide range of configurations. This improvement can be directly linked to its flexible gating mechanism, which allows the network to simultaneously preserve and transform information across layers. As a result, the network is able to capture the multi-scale structure of the inverse calibration problem, where both local features, such as short-maturity smile behaviour, and global characteristics, such as the term structure of variance, contribute to parameter recovery. At the same time, the results show that relatively simple architectures, such as the MLP, remain competitive when the input data is sufficiently informative, highlighting that network complexity should be aligned with the intrinsic structure of the problem.

The analysis of generalisation further indicates that neural network-based calibration primarily operates as an interpolator within the training domain. While high accuracy is achieved for parameter configurations represented in the training data, performance deteriorates outside this domain. This emphasises that careful design of the parameter space and training data coverage is essential for reliable practical deployment.

Beyond predictive performance, a central contribution of this thesis lies in the structural analysis of the learned inverse mappings. Using SHAP and νSHAP, it is shown that, for several parameters, the networks rely on regions of the implied volatility surface that are consistent with financial intuition, most notably short maturities and the wings of the smile. This provides evidence that neural networks do not merely approximate arbitrary statistical relationships, but instead capture economically meaningful structure in the data.

At the same time, the comparison between SHAP and νSHAP reveals that interpretability is inherently nuanced. While some parameters are associated with concentrated and well-defined regions of the surface, others are inferred from more diffuse and overlapping information. This indicates that the inverse calibration problem is not uniformly well-posed, and that identifiability varies across parameters. In particular, certain parameters are only indirectly identifiable through correlated features of the implied volatility surface. Consequently, neural network-based calibration should not be interpreted as yielding a unique reconstruction of model parameters, but rather as providing one of potentially multiple structurally consistent explanations of the observed surface.

Taken together, these findings show that neural networks can effectively bridge the gap between predictive accuracy and interpretability in the context of model calibration, provided that their behaviour is analysed with the appropriate tools. Rather than treating neural networks as opaque black boxes, the combination of modern architectures and XAI methods enables a more transparent understanding of how calibration is performed.

Several directions for future research follow naturally from this work. An important extension is the application of the proposed framework to real market data, where non-uniform liquidity and noise may alter the structure of the learned mappings. Furthermore, incorporating uncertainty quantification

into the calibration framework would allow the models to explicitly represent ambiguity in the inverse problem, thereby improving their robustness in practical applications. Finally, the results suggest that XAI methods can be developed into systematic tools for assessing parameter identifiability, for instance by quantifying overlap and redundancy in attribution patterns.

In conclusion, this thesis demonstrates that neural network-based calibration is a powerful and flexible approach for solving inverse problems in quantitative finance. While important challenges remain, particularly with respect to generalisation and identifiability, the integration of machine learning and explainability provides a promising framework for advancing both the practical application and theoretical understanding of model calibration.

References

[1] Shamima Ahmed et al. “Artificial Intelligence and machine learning in finance: A bibliometric review”. In: *Research in International Business and Finance* 61 (2022). doi: 10.1016/j.ribaf.2022.101646.

[2] Y. Bengio, P. Simard, and P. Frasconi. “Learning long-term dependencies with gradient descent is difficult”. In: *IEEE Transactions on Neural Networks* 5.2 (1994), pp. 157–166. doi: 10.1109/72.279181.

[3] Alexander Binder et al. *Layer-wise relevance propagation for neural networks with local renormalization layers*. 2016. arXiv: 1604.00825 [cs.CV].

[4] R. Cassani. *Multilayer perceptron example*. Nov. 2018. url: https://github.com/rcassani/mlp-example.

[5] Javier Castro, Daniel Gómez, and Juan Tejada. “Polynomial calculation of the Shapley value based on sampling”. In: *Computers Operations Research* 36.5 (2009), pp. 1726–1730. doi: 10.1016/j.cor.2008.04.004.

[6] Fabienne Comte and Eric Renault. “Long memory in continuous-time stochastic volatility models”. In: *Mathematical Finance* 8.4 (1998), pp. 291–323. doi: 10.1111/1467-9965.00057.

[7] Martin Cooper and Leila Amgoud. *Abductive explanations of classifiers under constraints: Complexity and properties*. 2024. arXiv: 2409.12154 [cs.AI].

[8] John C. Cox, Jonathan E. Ingersoll, and Stephen A. Ross. “A theory of the term structure of interest rates”. In: *Econometrica* 53.2 (1985), pp. 385–407. doi: 10.2307/1911242.

[9] Kai Diethelm, Neville J. Ford, and Alan D. Freed. “Detailed Error Analysis for a Fractional Adams Method”. In: *Numerical Algorithms* 36 (2004), pp. 31–52. doi: 10.1023/B:NUMA.0000027736.85078.be.

[10] Omar El Euch and Mathieu Rosenbaum. “The characteristic function of rough Heston models”. In: *Mathematical Finance* 29.1 (2018), pp. 3–38. doi: 10.1111/mafi.12173.

[11] F. Fang and C. W. Oosterlee. “A novel pricing method for European options based on Fourier-Cosine series expansions”. In: *SIAM Journal on Scientific Computing* 31.2 (2009), pp. 826–848. doi: 10.1137/080718061.

[12] Masaaki Fukasawa. “Short-time at-the-money skew and rough fractional volatility”. In: *Quantitative Finance* 17.2 (2016), pp. 189–198. doi: 10.1080/14697688.2016.1197410.

[13] Jim Gatheral, Thibault Jaisson, and Mathieu Rosenbaum. *Volatility is rough*. 2014. arXiv: 1410.3394 [q-fin.ST].

[14] Xavier Glorot and Yoshua Bengio. “Understanding the difficulty of training deep feedforward neural networks”. In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Vol. 9. PMLR, 2010, pp. 249–256.

[15] Kaiming He et al. *Deep residual learning for image recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[16] Kaiming He et al. *Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification*. 2015. arXiv: 1502.01852 [cs.CV].

[17] Andres Hernandez. “Model calibration with neural networks”. In: *SSRN Electronic Journal* (2016). doi: 10.2139/ssrn.2812140.

[18] Steven L. Heston. “A closed-form solution for options with stochastic volatility with applications to bond and currency options”. In: *Review of Financial Studies* 6.2 (1993), pp. 327–343. doi: 10.1093/rfs/6.2.327.

[19] Blanka Horvath, Aitor Muguruza, and Mehdi Tomas. “Deep learning volatility: A deep neural network perspective on pricing and calibration in (rough) volatility models”. In: *Quantitative Finance* 21.1 (2021), pp. 11–27. doi: 10.1080/14697688.2020.1817974.

[20] Eduardo Abi Jaber, Martin Larsson, and Sergio Pulido. *Affine Volterra processes*. 2019. arXiv: 1708.08796 [math.PR].

[21] Peter Jäckel. "Let's be rational". In: *Wilmott* 2015.75 (2015), pp. 40–53. doi: 10.1002/wilm.10395.

[22] Diederik P. Kingma and Jimmy Ba. *Adam: A method for stochastic optimization*. 2017. arXiv: 1412.6980 [cs.LG].

[23] Changpin Li and Chunxing Tao. "On the fractional Adams method". In: *Computers Mathematics with Applications* 58.8 (2009), pp. 1573–1588. doi: 10.1016/j.camwa.2009.07.050.

[24] Hao Li et al. *Visualizing the loss Landscape of neural nets*. 2018. arXiv: 1712.09913 [cs.LG].

[25] Scott Lundberg and Su-In Lee. *A unified approach to interpreting model predictions*. 2017. arXiv: 1705.07874 [cs.AI].

[26] Benoit B. Mandelbrot and John W. Van Ness. "Fractional Brownian motions, fractional noises and applications". In: *SIAM Review* 10.4 (1968), pp. 422–437. doi: 10.1137/1010093.

[27] Joao Marques-Silva. *Logic-based explainability: Past, Present Future*. 2024. arXiv: 2406.11873 [cs.AI].

[28] Joao Marques-Silva, Xuanxiang Huang, and Olivier Letoffe. *The Explanation Game – Rekindled (Extended version)*. 2025. arXiv: 2501.11429 [cs.AI].

[29] Martín Abadi et al. *TensorFlow: Large-scale machine learning on heterogeneous systems*. 2015. url: https://tensorflow.org.

[30] M. D. McKay, R. J. Beckman, and W. J. Conover. "A comparison of three methods for selecting values of input variables in the analysis of output from a computer Code". In: *Technometrics* 21.2 (1979), pp. 239–245. issn: 00401706.

[31] Tim Miller. "Explanation in artificial intelligence: Insights from the social sciences". In: *Artificial Intelligence* 267 (2019), pp. 1–38. doi: 10.1016/j.artint.2018.07.007.

[32] Christoph Molnar. *Interpretable Machine Learning. A guide for making black box models explainable*. 3rd ed. 2025. isbn: 978-3-911578-03-5.

[33] Thao Nguyen, Maithra Raghu, and Simon Kornblith. *Do wide and deep networks earn the same things? Uncovering how neural network representations vary with width and depth*. 2021. arXiv: 2010.15327 [cs.LG].

[34] Cornelis W. Oosterlee and Lech A. Grzelak. *Mathematical modeling and Computation in finance*. World Scientific, 2019. isbn: 978-1-786347-94-7.

[35] S.W. Park and D.M. Himmelblau. "Parameter estimation and unique identifiability". In: *The Chemical Engineering Journal* 25.2 (1982), pp. 163–174. doi: 10.1016/0300-9467(82)80085-3.

[36] *pyvollib*. 2023. url: https://github.com/vollib/py_vollib.

[37] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. *"Why should I trust you?": Explaining the predictions of any classifier*. 2016. arXiv: 1602.04938 [cs.LG].

[38] Dirk Röder and Georgi Dimitroff. "Volatility model calibration with neural networks a comparison between direct and indirect methods". In: *SSRN Electronic Journal* (2020). doi: 10.2139/ssrn.3645019.

[39] Sigurd Emil Rømer. "Empirical analysis of rough and classical stochastic volatility models to the SPX and VIX markets". In: *Quantitative Finance* 22.10 (2022), pp. 1805–1838. doi: 10.1080/14697688.2022.2081592.

[40] Lloyd S. Shapley. "A value for N-Person games". In: *Contributions to the Theory of Games* 28 (1953), pp. 307–317. doi: 10.1515/9781400881970-018.

[41] *SHAP*. 2023. url: https://github.com/shap/shap.

[42] L. S. Shapley and Martin Shubik. "A method for evaluating the distribution of power in a committee system". In: *American Political Science Review* 48.3 (1954), pp. 787–792. doi: 10.2307/1951053.

[43] Steven E Shreve. *Stochastic calculus for finance II: Continuous-time models*. Springer, 2010, pp. 83–209. isbn: 978-0-387401-01-0.

[44] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. *Learning important features through propagating activation differences*. 2019. arXiv: 1704.02685 [cs.CV].

[45] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. *Highway networks*. 2015. arXiv: 1505.00387 [cs.LG].

[46] Hans R. Stoll. "The relationship between put and call option prices". In: *The Journal of Finance* 24.5 (1969), pp. 801–824. doi: 10.1111/j.1540-6261.1969.tb01694.x.

[47] Daniel W Stroock and S.R.S. Varadhan. *Multidimensional diffusion processes*. Springer Science Business Media, 1997, pp. 51–52. isbn: 978-3-540903-53-6.

[48] H. P. Young. "Monotonic solutions of cooperative games". In: *International Journal of Game Theory* 14.2 (1985), pp. 65–72. doi: 10.1007/bf01769885.

[49] Bo Yuan et al. "Deep Learning Interpretability for Rough Volatility". In: *SSRN Electronic Journal* (2024). doi: 10.2139/ssrn.5037806.

[50] Yu Zhang et al. "A survey on neural network interpretability". In: *IEEE Transactions on Emerging Topics in Computational Intelligence* 5.5 (2021), pp. 726–742. doi: 10.1109/tetci.2021.3100641.

A

Hyperparameter Sweep Results per Architecture

Table A.1: Full hyperparameter sweep results for the MLP architecture (Heston model). Networks were trained for 200 epochs with a fixed learning rate of 10^{-3} and a batch size of 256. The mean validation MAE at the best epoch is reported.

Layers	Units	Parameters	Training time (min)	$\overline{\text{MAE}}$
1	16	1,509	3.43	$2.65 \cdot 10^{-2}$
2	16	1,781	3.64	$2.52 \cdot 10^{-2}$
3	16	2,053	3.81	$1.90 \cdot 10^{-2}$
1	32	3,013	3.62	$1.83 \cdot 10^{-2}$
2	32	4,069	3.78	$1.30 \cdot 10^{-2}$
3	32	5,125	3.98	$1.09 \cdot 10^{-2}$
1	64	6,021	3.77	$1.52 \cdot 10^{-2}$
2	64	10,181	4.04	$1.07 \cdot 10^{-2}$
3	64	14,341	4.35	$1.00 \cdot 10^{-2}$
1	128	12,037	4.40	$1.31 \cdot 10^{-2}$
2	128	28,549	5.49	$9.53 \cdot 10^{-3}$
3	128	45,061	6.20	$1.03 \cdot 10^{-2}$
1	256	24,069	5.29	$1.29 \cdot 10^{-2}$
2	256	89,861	7.78	$8.90 \cdot 10^{-3}$
3	256	155,653	10.35	$1.03 \cdot 10^{-2}$

Table A.2: ull hyperparameter sweep results for the MLP architecture (Rough Heston model). Networks were trained for 200 epochs with a fixed learning rate of 10^{-3} and a batch size of 256. The mean validation MAE at the best epoch is reported.

Layers	Units	Parameters	Training time (min)	$\overline{\text{MAE}}$
1	16	982	0.55	$1.65 \cdot 10^{-3}$
2	16	1,254	0.70	$1.58 \cdot 10^{-3}$
3	16	1,526	0.60	$1.57 \cdot 10^{-3}$
1	32	1,958	0.62	$2.12 \cdot 10^{-3}$
2	32	3,014	0.62	$2.01 \cdot 10^{-3}$
3	32	4,070	0.60	$1.75 \cdot 10^{-3}$
1	64	3,910	0.51	$2.12 \cdot 10^{-3}$
2	64	8,070	0.58	$2.17 \cdot 10^{-3}$
3	64	12,230	0.61	$2.34 \cdot 10^{-3}$
1	128	7,814	0.52	$2.21 \cdot 10^{-3}$
2	128	24,326	0.60	$2.42 \cdot 10^{-3}$
3	128	40,838	0.71	$2.32 \cdot 10^{-3}$
1	256	15,622	0.68	$2.15 \cdot 10^{-3}$
2	256	81,414	0.68	$1.89 \cdot 10^{-3}$
3	256	147,206	0.80	$2.14 \cdot 10^{-3}$

Table A.3: Full hyperparameter sweep results for the Highway architecture (Heston model). Networks were trained for 200 epochs with a fixed learning rate of 10^{-3} and a batch size of 256. The mean validation MAE at the best epoch is reported.

Layers	Units	Parameters	Training time (min)	$\overline{\text{MAE}}$
3	64	33,989	5.82	$7.43 \cdot 10^{-3}$
7	64	67,269	9.52	$7.06 \cdot 10^{-3}$
15	64	133,829	14.87	$9.88 \cdot 10^{-3}$
30	64	258,629	24.99	$2.53 \cdot 10^{-2}$

Table A.4: Full hyperparameter sweep results for the Highway architecture (Rough Heston model). Networks were trained for 200 epochs with a fixed learning rate of 10^{-3} and a batch size of 256. The mean validation MAE at the best epoch is reported.

Layers	Units	Parameters	Training time (min)	$\overline{\text{MAE}}$
3	64	27,526	0.84	$3.34 \cdot 10^{-3}$
7	64	60,806	1.05	$3.55 \cdot 10^{-3}$
15	64	127,366	1.42	$3.73 \cdot 10^{-3}$
30	64	252,166	2.14	$4.50 \cdot 10^{-3}$

Table A.5: Full hyperparameter sweep results for the Generalised Highway architecture (Heston model). Networks were trained for 200 epochs with a fixed learning rate of 10^{-4} and a batch size of 256. The mean validation MAE at the best epoch is reported.

Layers	Units	Parameters	Training time (min)	$\overline{\text{MAE}}$
3	64	48,005	13.61	$1.90 \cdot 10^{-2}$
7	64	97,925	24.45	$8.01 \cdot 10^{-3}$
15	64	197,765	50.90	$6.08 \cdot 10^{-3}$
30	64	384,965	87.52	$6.22 \cdot 10^{-3}$

Table A.6: Full hyperparameter sweep results for the Generalised Highway architecture (Rough Heston model). Networks were trained for 200 epochs with a fixed learning rate of 10^{-3} and a batch size of 256. The mean validation MAE at the best epoch is reported.

Layers	Units	Parameters	Training time (min)	$\overline{\text{MAE}}$
3	64	$39,366$	1.00	$2.48 \cdot 10^{-3}$
7	64	$89,286$	1.74	$1.51 \cdot 10^{-3}$
15	64	$189,126$	3.33	$7.78 \cdot 10^{-4}$
30	64	$376,326$	5.51	$5.07 \cdot 10^{-4}$