A DEEP LEARNING APPROACH FOR PRICING OPTIONS BASED ON HESTON-NANDI-GARCH

SUPERVISED BY DR. LYUDMILA GRIGORYEVA

THIS PAPER WAS DEVELOPED WITHIN THE SEMINAR "DEEP LEARNING AND ECONOMETRICS"

AT THE UNIVERSITY OF KONSTANZ, SUMMER TERM 2019.

Henrik Brautmeier
Student ID: 905387
henrik.brautmeier@uni.kn

Lukas Wuertenberger
Student ID: 905804
lukas.wuertenberger@uni.kn

September 15, 2019

ABSTRACT

We present a continued development of the neural network calibration method by Horvath et al. [1]. We show that their implementation of learning option prices and calibration of model parameters works for the discrete-time Heston-Nandi-GARCH model, too. Thereby the choice of Heston-Nandi-GARCH model parameters is crucial for the performance of that approach. This seminar paper includes a great variety of performance tests with different datasets and a differentiation between option prices and implied volatilities. We also make suggestions to incorporate market data. Robustness checks cover i.a. a performance comparison to Black Scholes and therefore ensure valid results.

***K*eywords** Deep Learning, , Option Pricing, Heston Nandi, Volatility Modeling

Acknowledgments Special thanks to Lyudmila Grigoryeva for her universal support as well as her target-oriented criticism. We are grateful to Juan-Pablo Ortega for his stimulating input.

Contents

1 Introduction **3**

2 Recap: The GARCH-Model of Heston and Nandi **3**

3 Deep Learning and Calibration Approach by Horvath et al. [1] **5**

3.1 Formalisation of the Deep Learning Approach 5

3.2 The two step approach - Image-based implicit learning 6

3.3 Neural Network architecture 6

3.4 Calibration 7

4 Methodology and Model Choice **7**

4.1 Choice of scenarios 7

4.1.1 Scenarios based on Maximum Likelihood estimators 7

4.1.2 Scenarios based on low relative errors 8

4.1.3 Scenarios based on small γ^* 8

4.2 A simple approach on "How to involve real data" 9

5 Results **10**

5.1 Volatility surfaces 10

5.2 Price surfaces 12

5.2.1 Performance in Comparsion to Black-Scholes 14

5.3 Outlook: "How to involve real data" 16

5.4 Robustness 17

5.4.1 Sensitivity towards mispricing 17

5.4.2 Sensitivity towards training set size 17

6 Challenges and Discussion **18**

7 Conclusion **18**

A Further Volatility Plots **20**

B Further Robustness Test **21**

C Black Scholes Comparison **21**

1 Introduction

Since the first occurrence of modern derivatives end of last century, their importance increased significantly. Besides the usage as additional investment opportunity, options and other derivatives provide an utmost important tool for controlling market risk. Hence, researches have spend years investigating pricing and hedging of such financial products. Starting with the work of Black and Scholes (1973) and Merton (1973), tons of market models were introduced and analysed, each with it's own upsides and flaws. With increasing complexity, it's not possible to ensure the existence of closed form pricing formulas. Therefore calibrating models and estimating optimal parameters lead to the necessity of very computation-intensive simulations, which themselves add approximation errors to each calculation. This explains the attractiveness of the Black-Scholes-Merton model even after more realistic stochastic pricing models have been developed.
To tackle these problems, Horvath et al [1] introduce a new approach. The authors provide a method on separating pricing and calibration process using feed-forward neutral networks. Firstly, a neural net is trained to map model parameters directly to the pricing surface. By doing so the computational-intensive simulations are needed only once for training. Afterwards the neural net can be used as fast and accurate pricing function. Secondly, the calibration task is reduced to a "simple" constrained optimization task. The trained net is used as input function and no further computation meaningful steps are needed.
Using continuous-time stochastic volatility models often results in problems when one is asked not only to calibrate a model to observable market data, but to fit historical observations or future projections. In this paper we use the approach of [1] and apply it to discrete-time models, namely Heston-Nandi-GARCH. This allows to price in a real world scenario and not only under a risk-neutral markets. One advantage of using Heston and Nandi's model is the existence of a closed form solution. Hence, we do not deal with additional approximation errors resulting from Monte-Carlo simulation. Nonetheless, this approach is easily generalizable to any other discrete-time model. Furthermore we provide a rudimentary way on involving real data into the setting by combining the deep learning model with standard econometric theory.

2 Recap: The GARCH-Model of Heston and Nandi

This paper will use the market model developed by Heston and Nandi [2]. Therefore a short recap of the most important properties is necessary.
The development of a pricing model for European style options by Black, Scholes (1973) and Merton (1973) was crucial for the analysis and hedging of such assets. However, the simple structure does not reflect the behaviour of modern financial markets. Neither normal distributed returns nor a time invariant volatility can be observed in modern financial markets. While the Black-Scholes-Merton model (BS model) implies a flat volatility surface, real volatility surfaces tend to depend on the time to maturity as well as the strike of an option. To account for the complex structure of the underlying volatility process researchers developed a variety of different solutions. Engle's generalized autoregressive conditional heteroscedasticity (GARCH) models proved helpful and reasonable in an economic setting (cf. Engle [3]).
In their work [4] in 1997 and [2] in 2000 Heston and Nandi presented a discrete model preserving the structure of Black and Scholes, while tackling previously mentioned problems by modeling the volatility via GARCH processes (henceforth HNG model). Heston and Nandi point out that in comparison to continuous-time stochastic volatility models, which are not observable, a GARCH structure provides the possibility to use historic prices of the underlying for estimations.
The following assumptions are made in the HNG model. First, a time-discrete market on equidistant grid with $\Delta t = 1$ is considered. Secondly, the yearly risk-free rate r is constant and discount rates are continuously compounded. Third, the underlying assets price S_t follows the dynamic of (1)

$$\begin{cases} Y_t &= \ln(S_t) - \ln(S_{t-1}) = r + \lambda h_t + \sqrt{h_t} z_t \\ h_t &= \omega + \sum_{i=1}^{q} \alpha_i \left(z_{t-1} - \gamma_i \sqrt{h_{t-1}} \right)^2 + \sum_{i=1}^{p} \beta_i h_{t-1} \\ S_0 &= s \in \mathbb{R}^+ \\ h_0 &= \eta \in \mathbb{R}^+ \end{cases} \tag{1}$$

where $z_t \overset{iid}{\sim} N(0,1), \omega, \alpha_i, \beta_i, \gamma_i, \lambda \in \mathbb{R}$ and $p, q \in \mathbb{N}$.
Previous research (cf. [5]) indicate that the case of $p = q = 1$ is sufficient and therefore will be considered in this paper. Similar to Engle's GARCH(1,1) process the volatility $(h_t)_t$ is weakly stationary iff $|\alpha\gamma^2 + \beta| < 1$. Positivity can be assured by assuming $\alpha \geq 0$, $\beta \geq 0$ and $\omega > 0$. It is easy to derive the long term expectation of the volatility $(h_t)_t$ under stationarity:

$$\mathbb{E}(h_t) = \frac{\omega + \alpha}{1 - \alpha\gamma^2 - \beta}$$

In addition to Engle's GARCH Heston and Nandi account for leverage effects via the skewness parameter γ. The conditional covariance between Y_t and h_t is

$$\mathrm{Cov}(h_{t+1}, \ln(S_t)|\mathcal{F}_{t-1}) = -2\alpha\gamma h_t.$$

As usual $(\mathcal{F}_t)_t$ denotes the filtration generated by $(S_t)_t$. A positive γ leads to a leverage effect.
Heston and Nandi show that the corresponding risk-neutral process has an identical GARCH form as (1) with replacing λ by $\lambda^* = -\frac{1}{2}$ and γ by $\gamma^* = \lambda + \gamma + \frac{1}{2}$. This results in the following risk-neutral model:

$$\begin{cases} Y_t &= \ln(S_t) - \ln(S_{t-1}) = r - \frac{1}{2}h_t + \sqrt{h_t}z_t^* \\ h_t &= \omega + \alpha\left(z_{t-1}^* - \gamma^*\sqrt{h_{t-1}}\right)^2 + \beta h_{t-1} \\ S_0 &= s \in \mathbb{R}^+ \\ h_0 &= \eta \in \mathbb{R}^+ \end{cases}$$

where $z_t^* \overset{iid}{\sim} N(0,1)$ but only under the risk neutral measure $\mathbb{Q}$[1].
This change in measure has two important implications. The expected return of the underlying is, similar to BS, equal to the risk-free rate r and, given a positive λ, it holds $\gamma < \gamma^*$. This implies that in risk-neutral setting a higher average volatility and a stronger degree of leverage is observed. A more detailed analysis can be found in chapter 4.2 of [6].
Under the risk-neutral measure $\mathbb{Q}$ the price C of an European call option with Strike K at time t and expiration date T is given by

$$\begin{aligned} \mathbb{E}_{\mathbb{Q}}\left[(S_T - K)^+|\mathcal{F}_t\right] = \frac{1}{2}e^{r(T-t)}S_t + \frac{1}{\pi}\int_0^\infty \Re\left(\frac{K^{-i\phi}f_t(i\phi+1)}{i\phi}\right)d\phi \\ - K\left(\frac{1}{2} + \frac{1}{\pi}\int_0^\infty \Re\left(\frac{K^{-i\phi}f_t(i\phi)}{i\phi}\right)d\phi\right) \end{aligned}$$

and

$$C = e^{-r(T-t)}\,\mathbb{E}_{\mathbb{Q}}\left[(S_T - K)^+|\mathcal{F}_t\right] \tag{2}$$

where f_t denotes the conditional generating function of Y_t under $\mathbb{Q}$, $f_t(\phi) = \mathbb{E}_{\mathbb{Q}}\left[S_T^\phi|\mathcal{F}_t\right]$.
In the case of $p = q = 1$ the generating function can be calculated recursively with the following expressions

$$\begin{aligned} f_t(\phi) &= S_t^\phi \exp\left(A_t + B_t h_{t+1}\right) \\ A_t &= A_{t+1} + r\phi + \omega B_{t+1} - \frac{1}{2}\ln(1 - 2\alpha B_{t+1}) \\ B_t &= -\frac{1}{2}\phi + \beta B_{t+1} + \frac{\frac{1}{2}\phi^2 - 2\alpha(\gamma^*)^2 B_{t+1}\phi + \alpha(\gamma^*)^2 B_{t+1}}{1 - 2\alpha B_{t+1}} \end{aligned}$$

and terminal condition

$$A_T = B_T = 0.$$

Remark that a corresponding put price can be easily derived using put-call-parities.

[1]Even if the assumption of $z_t \overset{iid}{\sim} N(0,1)$ is released, the model still works. But the value of a call option cannot be calculated via Black-Scholes formula. In this case formula (2) does not hold. Furthermore only pseudo Maximum Likelihood estimation is possible in Section 4.2.

3 Deep Learning and Calibration Approach by Horvath et al. [1]

In their paper *Deep Learning Volatility - A deep neural network perspective on pricing and calibration in (rough) volatility models* Horvarth et al. [1] present a highly efficient and accurate deep learning approach to implement stochastic volatility models. Their approach consists of two steps: First learning a pricing function via a neural network by using model parameters as input and option price[2] surfaces as an output. The second step contains the optimization of the parameters that fit the prices best.
Using deep learning methods to price options has the goal to overcome high computational effort caused by the implementation of volatility models, especially for those which have no closed form solution available. This is because the neural network needs to learn the pricing function only once. Afterwards it is capable to evaluate new arbitrary parameter scenarios in incredibly quick time.
The paper of Horvath et al. [1] only considers stochastic volatility models in continuous time with no closed form solution available. In our analysis we deal with the discrete Heston-Nandi-GARCH model which has a closed form pricing formula derived by Heston and Nandi [2]. The reasons to do so are multifaceted. On the one hand we want to know if discrete models yield comparable results to continuous models. On the other hand a deep learning implementation of models that provide closed form solutions can be realized in more precise manner since the numerical pricing errors are negligible compared to other numerical pricing approximations like Monte Carlo based approaches.

3.1 Formalisation of the Deep Learning Approach

In the following sections we stick to the formalisation which is used in Chapter 1 of Horvath et al. [1]. We apply it to the special case of Heston-Nandi-GARCH.
Let $\mathcal{M}^{HNG}(\theta)_{\theta\in\Theta^{HNG}}$ be the Heston-Nandi-GARCH model (cf. Section 2) with parameters[3]

$$\theta = (\alpha, \beta, \gamma^*, \omega, r, \sigma_0^2) \in \Theta^{HNG} \subset \mathbb{R}^6 \,. \tag{3}$$

The parameter combination $\theta \in \Theta$ completely describes the dynamics of the Heston-Nandi-GARCH model $\mathcal{M}^{HNG}(\theta)$.
The pricing map is stated by $P : \mathcal{M}^{\text{HNG}}(\theta, \zeta) \to \mathcal{X}$, which is available in closed form for Heston-Nandi-GARCH. Thereby ζ includes the additional attributes needed to price a call option, namely strike price K and maturity T of the contract. Furthermore $\mathcal{P}^{MKT}(\zeta) \in \mathcal{X}$ are the observed market prices depending on K and T. χ denotes the set of call options prices. The aim is to find the optimal parameters $\hat{\theta}$ which solves

$$\hat{\theta} = \underset{\theta\in\Theta^{HNG}}{\operatorname{argmin}}\, \delta(P(\mathcal{M}^{\text{HNG}}(\theta, \zeta)), \mathcal{P}^{MKT}(\zeta)).$$

$\delta(\cdot, \cdot)$ is a suitable metric defined in the corresponding topological space.
There are several approaches to move forward. One possible way would be to find an inverse pricing map, which maps prices and option specifications to the model parameters (cf. Hernandez [7]). The control on that inverse pricing function seems difficult since train and test errors tend to differ significantly.
A further, more promising approach is presented in Horvath et al. [1]. It consists of two steps:

i. Learn option prices by a neural network F. The input of the network are parameters of a volatility model (e.g. equation (3)), the output is a strike-maturity determined set of option prices resp. implied volatilities.

$$F(\Theta^{HNG}, \zeta) \approx P(\mathcal{M}^{HNG}(\Theta^{HNG}, \zeta)).$$

ii. Calibrate (super fast) the deterministic learned pricing map.

$$\hat{\theta} = \underset{\theta\in\Theta^{HNG}}{\operatorname{argmin}}\, \delta(F(\theta, \zeta), \mathcal{P}^{MKT}(\zeta))$$

[2] In our paper we use the price of an option and its implied volatility (based on Black-Scholes) synonymously.
[3] We consider the risk-neutral setting and take the initial variance σ_0^2 computed by the unconditional variance $\sigma_0^2 = \frac{\omega+\alpha}{1-\alpha(\gamma^*)^2-\beta}$ as additional parameter.

3.2 The two step approach - Image-based implicit learning

As we believe the two step approach with image-based implicit learning is most suitable in the Heston-Nandi-GARCH context, we skip the inverse map approach and the two step pointwise learning approach. Instead this subsection describes the image-based concept of Horvath et al [1] in detail.
As a first step, a fixed grid of strikes and maturities, $\Delta := \{K_i, T_j\}_{i=1,\ j=1}^{n,\ m}$ is defined. This grid specifies the set of call options we are interested in, namely call options with strike K_i, $i = 1, \dots, n$, and maturity T_j, $j = 1, \dots, m$. Hereafter the following two steps are conducted.

i. The neural network $F(\theta, \hat{\nu})$ learns the set of implied volatilities $F^*(\theta) = \{\sigma_{BS}^{\mathcal{M}^{HNG}(\theta)}(T_i, K_j)\}_{i=1,\ j=1}^{n,\ m}$ (resp. prices $F^*(\theta) = \{P(\mathcal{M}^{HNG}(\theta, (T_i, K_j))\}_{i=1,\ j=1}^{n,\ m}$), where $\hat{\nu}$ are the trained network weights and

$$\begin{aligned} F^*(\theta) : \Theta^{HNG} &\to \mathbb{R}^{m \times n} \\ \theta &\mapsto F^*(\theta). \end{aligned}$$

The optimal weights are calculated via

$$\hat{\nu} = \underset{\nu \in \mathbb{R}^l}{\operatorname{argmin}} \sum_{u=1}^{N_{train}} \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{m} \left(F(\theta_u, \nu)_{ij} - F^*(\theta_u)_{ij}\right)^2}$$

ii. Find the optimal parameters $\hat{\theta}$ which solve

$$\hat{\theta} = \underset{\theta \in \Theta^{HNG}}{\operatorname{argmin}} \sum_{i=1}^{n} \sum_{j=1}^{m} (F(\theta, \hat{\nu})_{ij} - \sigma_{BS}^{MKT}(T_i, K_j))^2.$$

The optimal weights $\hat{\nu}$ implicitly depend on the structure of the grid Δ and so influence the neural network function $F(\theta, \hat{\nu})$. Hence the approximation quality of the neural network can differ across different strike-maturity grid structures. In theory the strike-maturity grid can be chosen arbitrary, however if the chosen scenarios are too extreme[4] there might be numerical issues, especially when implied volatilities are calculated. Learning and calibration can also be done pointwise (cf. Horvath et al. [1] Section 3.2) or in an inverse map procedure (cf. Hernandez [7]). However Horvath et al. [1] list advantages of the image-based implicit learning approach. Examples are

i. exploiting the structure of having a full price grid which improves the learning process since neighbouring outputs are incorporated.
ii. injectivity of the pricing map can be more easily guaranteed than in the pointwise training.
iii. dimension reduction to a finite optimisation problem. If we sample enough strike and maturity points $(k, t) \in [k_{min}, k_{max}] \times [0, T]$ the error of the neural network approximation gets smaller and smaller.
iv. fast training speed since the gridpoints stay fixed.
v. freedom of choosing the grid

3.3 Neural Network architecture

Analogous to Horvath et al. [1] we use a fully connected feed forward neural network with 3 hidden layers and 30 nodes on each layer. Although the Heston-Nandi-GARCH model has 5 input parameters[5] (the models in Horvath et al. [1] only have 4) a more complex model (e.g. adding additional layers, add dropout,...) doesn't increase the approximation quality.
The input vector contains the five variable parameters $\alpha, \beta, \gamma^*, \omega$ and σ_0. The output dimension is defined by the product of the number of strikes and the number of the maturities in the grid.

[4] E.g. if the strike price is too high and the time to maturity too low, we get negative prices with Heston-Nandi-GARCH.
[5] We used a constant interest rate r.

The activation function for the hidden layers is elu $\sigma_{elu} = \alpha(e^x - 1)$ with $\alpha = 1$ and for the output layer it is the linear function, i.e. no activation.
Similar to [1] we use 200 epochs and a batch size of 32 to train the network.
The inputs and outputs of the net are normalized according to the proposal of Horvath et al. [1]. Input parameters are normalised by

$$\frac{2\theta - (\theta_{max} + \theta_{min})}{\theta_{max} - \theta_{min}} \in [-1, 1],$$

where $\theta \in [\theta_{min}, \theta_{max}]$. The standard score is used to scale the output.

3.4 Calibration

Horvath et al. [1] implemented several gradient based optimizers and find that the Levenberg-Marquardt algorithm is the most balanced optimizer regarding speed and convergence. Therefore it is also used in order to meet the necessary first order condition

$$\nabla^\theta \delta(F(\mathcal{M}^{HNG}(\theta, \zeta), \mathcal{P}^{MKT}(\zeta))) = 0.$$

The accuracy of the optimized parameters $\hat{\theta}$ is measured in two ways. First relative errors,

$$E_R(\hat{\theta}) = \frac{|\hat{\theta} - \bar{\theta}|}{|\bar{\theta}|},$$

are considered, where $\bar{\theta}$ are the model parameters which are used to generate the data.
Another way is to measure errors in terms of the root mean square error (RMSE) for the entire surface,

$$RMSE(\hat{\theta}) = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{m}(F(\hat{\theta}, \hat{\nu})_{ij} - \sigma_{BS}^{MKT}(T_i, K_j))^2}$$

4 Methodology and Model Choice

4.1 Choice of scenarios

Before we feed our neural network presented in Section 3.3 with data we need to generate suitable datasets. We consider the following settings.

4.1.1 Scenarios based on Maximum Likelihood estimators

In Section 4.2 we present an Maximum Likelihood approach on estimating model parameters which fit real data. The Maximum Likelihood estimates form the basis for the parameter intervals in this set of scenarios. We want to compare the Heston-Nandi-GARCH model based on the Maximum Likelihood 90% confidence interval to artificial interval boundary settings below.

- Train Set: 14,450, Validation Set: 2,550 and Test Set: 3,000
- Parameter intervals: $(\alpha, \beta, \gamma^*, \omega, r, \sigma_0^2) \in \mathcal{U}[5.8e-7, 1.4e-6] \times \mathcal{U}[0.43, 0.75] \times \mathcal{U}[441, 590] \times \mathcal{U}[4.1e-7, 2.9e-6] \times 0.005/252 \times [1.3e-4, 0.001]$
- Special specifications: constant interest rate at 0.5% annually, the initial variance σ_0^2 is determined by the unconditional variance $\sigma_0^2 = \frac{\alpha+\omega}{1-\beta-\alpha\cdot\gamma^2}$.
- Strikes = $\{0.9, 0.92, 0.94, 0.96, 0.98, 1, 1.02, 1.04, 1.06, 1.08, 1.1\}$
- Maturities = $\{30, 60, 90, 120, 150, 180, 210, 240\}$ (in days)

4.1.2 Scenarios based on low relative errors

In this paragraph we introduce parameter intervals which yield the best volatility surface approximation performance by the neural network. The boundaries for the parameters are completely artificial and found by trail and error.

- Train Set: 14,450, Validation Set: 2,550 and Test Set: 3,000
- Parameter intervals: $(\alpha, \beta, \gamma^*, \omega, r, \sigma_0^2) \in \mathcal{U}[1e-6, 1.5e-6] \times \mathcal{U}[0.57, 0.7] \times \mathcal{U}[450, 500] \times \mathcal{U}[1.1e-5, 2e-5] \times 0.005/252 \times [5.8e-5, 2e-4]$
- Special specifications: constant interest rate at 0.5% annually, the initial variance σ_0^2 is determined by the unconditional variance $\sigma_0^2 = \frac{\alpha+\omega}{1-\beta-\alpha\cdot\gamma^2}$.
- Strikes = $\{0.9, 0.92, 0.94, 0.96, 0.98, 1, 1.02, 1.04, 1.06, 1.08, 1.1\}$
- Maturities = $\{30, 60, 90, 120, 150, 180, 210, 240\}$ (in days)

4.1.3 Scenarios based on small γ^*

The γ parameter in Heston-Nandi-GARCH model controls the skewness of the distribution of the log-returns. If γ is zero, the distribution will be symmetric (cf. Heston and Nandi (2000) [2]). In this set of scenarios we choose small γ^*s to generate volatility surfaces with a pronounced smile. The dataset is characterized by

- Train Set: 14,450, Validation Set: 2,550 and Test Set: 3,000
- Parameter intervals: $(\alpha, \beta, \gamma^*, \omega, r, \sigma_0^2) \in \mathcal{U}[5e-6, 7.5e-5] \times \mathcal{U}[0.85, 0.98] \times \mathcal{U}[1, 4] \times \mathcal{U}[2.75e-6, 1.5e-5] \times 0.005/252 \times [7e-5, 0.001]$
- Special specifications: constant interest rate at 0.5% annually, the initial variance σ_0^2 is determined by the unconditional variance $\sigma_0^2 = \frac{\alpha+\omega}{1-\beta-\alpha\cdot\gamma^2}$.
- Strikes = $\{0.8$ to 1.2 by 0.025 steps$\}$
- Maturities = $\{30, 60, 90, 120, 150, 180, 210, 240\}$ (in days)

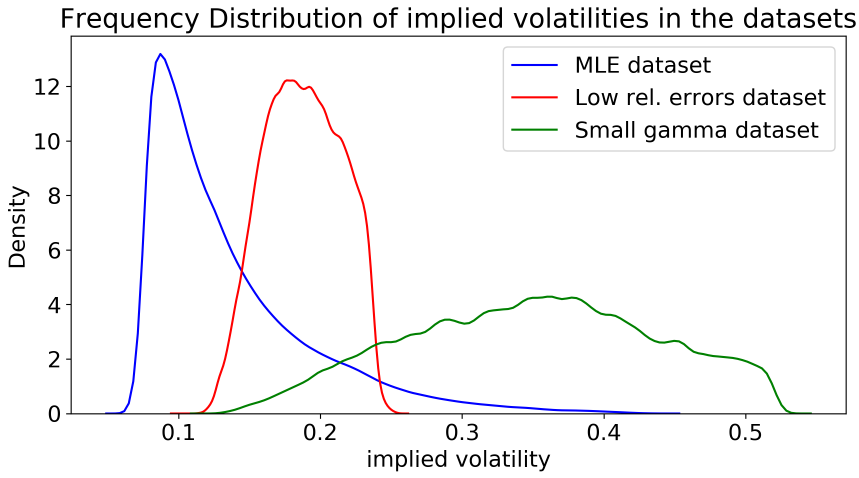

Figure 1: Comparison of frequency distributions of implied volatilities of the different datasets

Figure 1 visualizes the frequency distribution of the created datasets. The diversity among the dataset is ensured and so each dataset has its justification.

4.2 A simple approach on "How to involve real data"

Moving from continuous to a discrete model is the first step on providing a setting which can be used in practice. Nevertheless, the major part of this paper covers a model based approach and the usefulness for real application is still unclear. The goal is to find parameter boundaries of $(\omega, \alpha, \beta, \gamma, \lambda)$ for generating a training set, which fits observed S&P500 data. Similar to [2] and [6], we calibrate the HNG model by using Maximum Likelihood estimation. For data points $y_1, \ldots, y_T$, the following optimization problem needs to be solved:

$$\begin{array}{ll} \max\limits_{\omega,\alpha,\beta,\gamma,\lambda} & \mathcal{L}\left(y_1, \ldots, y_T | (\omega, \alpha, \beta, \gamma, \lambda)\right) \\ \text{subject to} & \alpha\gamma^2 + \beta < 1 \\ & \alpha \geq 0 \\ & \beta \geq 0 \\ & \omega > 0 \end{array} \tag{4}$$

$\mathcal{L}$ is the conditional Log-Likelihood function of the HNG(1,1) model:

$$\mathcal{L}\left(y_1, \ldots, y_T | (\omega, \alpha, \beta, \gamma, \lambda)\right) = -\frac{T}{2}\ln(2\pi) - \frac{1}{2}\sum_{t=1}^{T}\left(\ln(h_t) + \frac{(y_t - r - \lambda h_t)^2}{h_t}\right) \tag{5}$$

The form (5) can be calculated recursively by using identity (6), which is obtained straight from (1):

$$h_{t+1} = \omega + \beta h_t + \alpha\frac{(Y_t - r - (\lambda + \gamma)h_t)^2}{h_t} \tag{6}$$

A rolling window approach (1000 runs, window length 750 days, starting windows January 2009 - December 2011, ending window January 2013 - December 2015) is used to optimize the parameters. Note that (4) has some undesirable properties and the success of the optimization is highly dependent on the choice of starting values. We initialise the optimization with the long term expectation $h_0 = \mathbb{E}(h_t)$ as previous literature suggests. Nonetheless, including h_0 into the optimization problem does not lead to a significant increase in computational effort and might be subject of further studies. Table 4.2 and Figure2 summarize the results.

	ω	α	β	γ	λ
Average	2.894e-6	1.235e-6	0.661	473.243	10.384
Standard Deviation	1.006e-6	2.649e-7	0.087	11.489	2.871e-5
5% Quantile	1.587e-6	9.903e-7	0.527	456.564	5.052
95% Quantile	5.526e-6	1.610e-6	0.741	489.853	14.214

Table 1: Rolling estimation of Heston and Nandi between 2012 and 2015 on S&P500

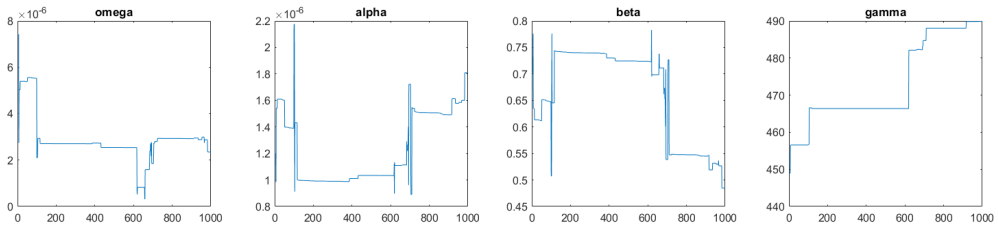

Figure 2: Rolling estimation of Heston and Nandi between 2012 and 2015 on S&P500

The symmetric 90% confidence interval of these parameters is included in one parameter set, which is used to train the neural net (cf. Section 4.1.1). This procedure should assure that the neural net is capable of pricing calls on S&P500 traded in the corresponding years.
Besides the model fit of this parameter setup (cf. Section 5.2), we want to analyse how well this rudimentary approach fits real option prices. To test this, prices of European style call options with underlying S&P500

traded between January 2010 and December 2015 are used. Options with a open interest or a trading volume of less then 100 were excluded. Additionally options with a maturity of more then 1 year and less then 30 days were excluded. The dataset contains 73.918 call options. The observed Moneyness is 0.6 between 2.8. Due to the small amount of option with extreme high/low Strikes, we only consider options with relative Strike $\frac{K}{S_0}$ between 0.9 and 1.1. Prices of options with same characteristics are averaged out to get a unique price. These differences may occur from market microstructure effects, semiefficient markets or a change in interest rate. Per year a single price surface is generated and cubic interpolation is used to smooth the surface and fit missing values. Figure 3 shows the fitted surface and real options prices in 2015. These surfaces are now used to calibrate the trained net. The results can be found in Section 5.3.

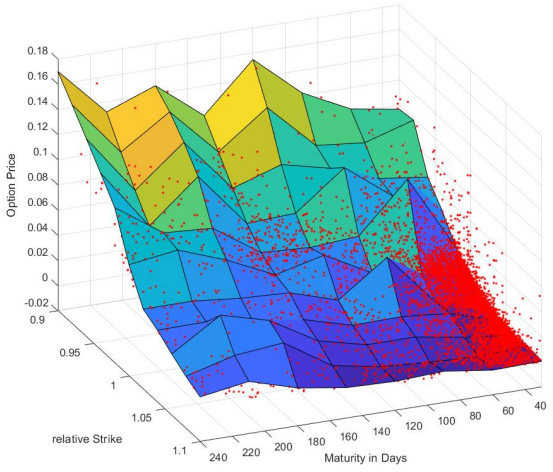

Figure 3: Call prices of 2015 and fitted price surface

5 Results

5.1 Volatility surfaces

As a starting point we analyse the results of the dataset based on the 90% Maximum Likelihood confidence interval. Figure 4 shows that the vast majority of average relative testing errors of the neural network is located significantly below 0.7% on the strike-maturity grid. Nevertheless the maximum relative errors rise up to 40% in the corners where the extreme scenarios lie (small maturity combined with high or low strikes). All in all the approximation results of the neural network are in the same order of magnitude as the findings of Horvath et al. [1] regarding the Bergomi model.
Figure 5 and 6 indicate that the Maximum Likelihood based dataset probably is not the best dataset to use for training the neural network. The generated scenarios are incapable to reproduce a volatility smile or a more complex volatility surface which would be more challenging for the neural network to approximate. Additionally we observe that the calibrated parameters, especially α, γ^* and ω, differ up to 9% on average from the true parameters in the test set.
As the dataset based on Maximum Likelihood estimation doesn't fully satisfy our high demands we generate two additional datasets.

First, we create a dataset which has the objective of minimize the neural network approximation error of the implied volatility surface. This task is very time consuming since the intervals are subject to constraints (cf. equations 4) and the neural network needs to get along with the generated dataset, too. Additionally, we observe that the shape of the implied volatility surface is very sensitive to only small changes in parameter values. Finally, we come up with a fully viable dataset which leads to mean relative errors of 0.06% and

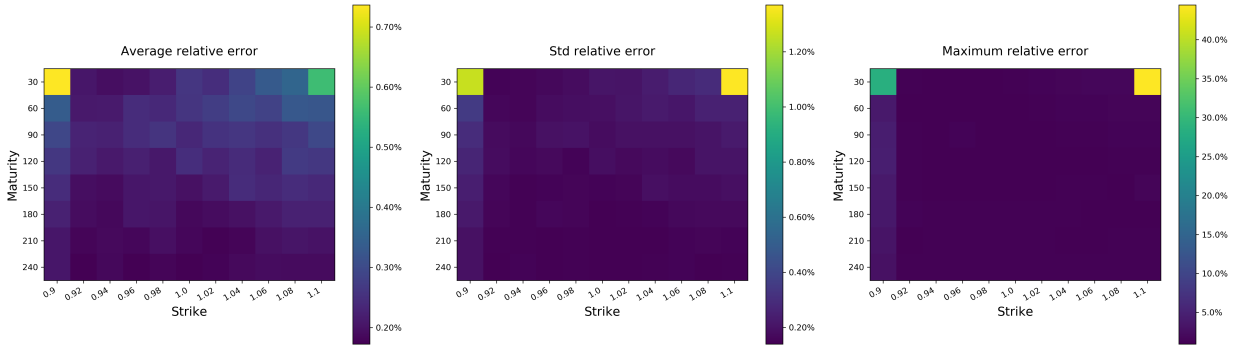

Figure 4: 90% Maximum Likelihood confidence interval - Relative errors of the neural network predicted volatilities compared to true volatilities in the test set

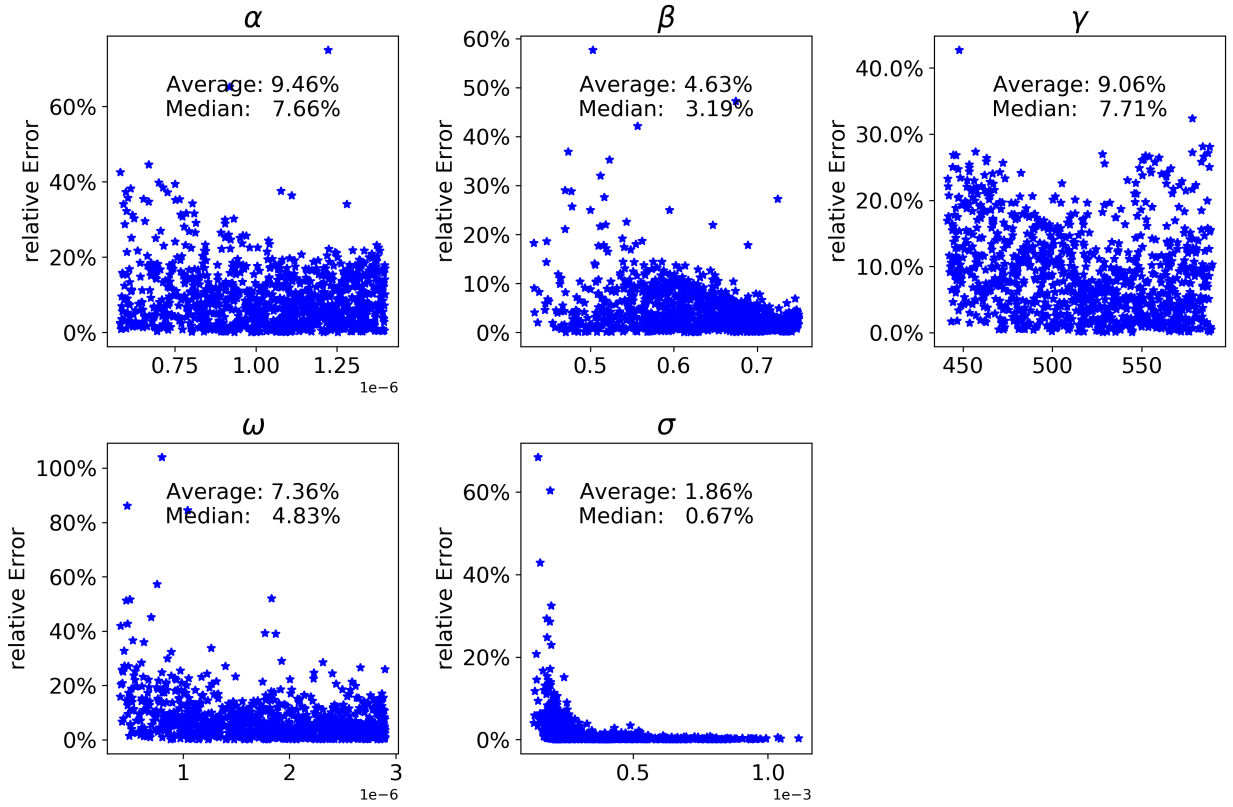

Figure 5: 90% Maximum Likelihood confidence interval - Relative errors of the parameters after calibration with Levenberg-Marquardt compared to true parameters in the test set

smaller and maximum relative errors of 2.3% and smaller (cf. Figure 7). Figure 8 visualizes that the volatilities in this set of scenarios follow a more concave process along moneyness.

The second alternative dataset contains implied volatility surfaces with a pronounced smile. This is achieved by using γ^* parameters close to zero. Figure 10 demonstrates that the network is able to replicate the modified shape of surface: The mean relative error is predominantly smaller than 0.1%, the maximum relative error smaller than 8%. Also Figure 11 and 12 illustrate the quite accurate predicted smile in particular for short maturities. It should be mentioned that the whole volatility surface is still quite flat, e.g. in Figure 12 the volatilities range from 27.2% to 28%. We tried a lot to install parameter boundaries which yield greater

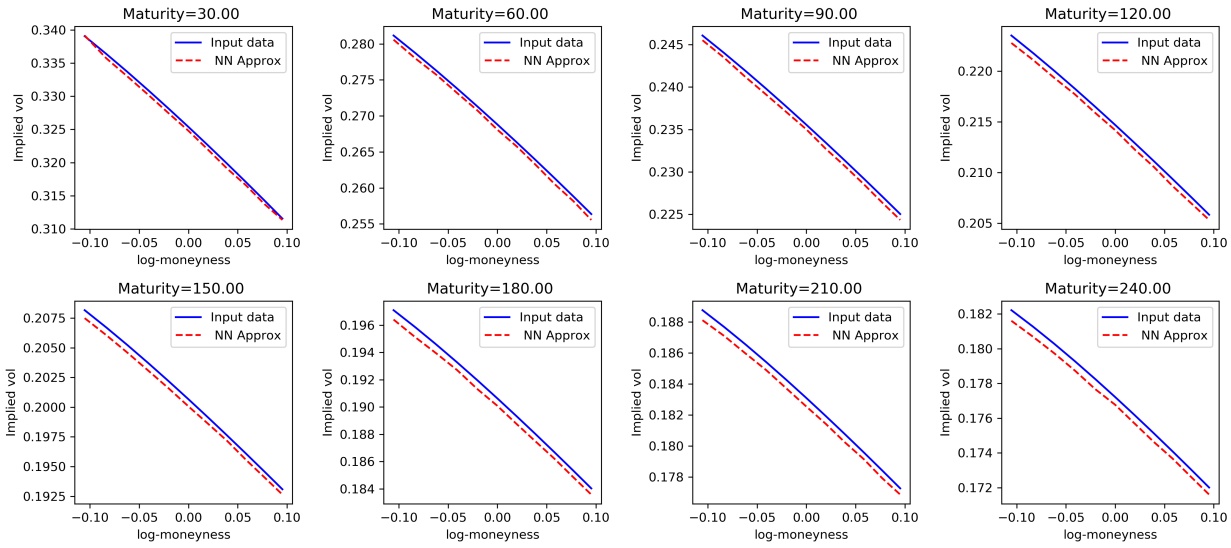

Figure 6: 90% Maximum Likelihood confidence interval - Comparison of neural network predicted volatilites and true volatilites in the test set

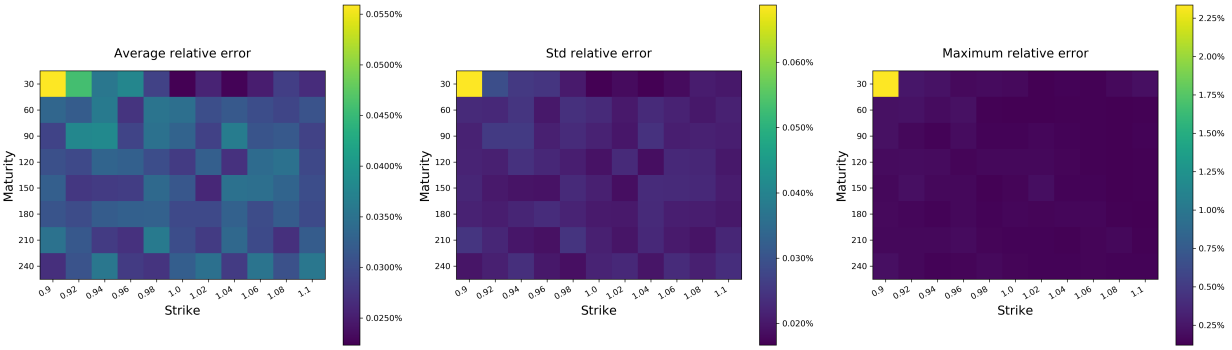

Figure 7: Best fit - Relative errors of the neural network predicted volatilities compared to true volatilities in the test set

differences in volatility height, but we don't manage to find them so far.
In Figure 13 the optimized ω parameter has a mean relative error of 19%, which is the largest among all parameters.
The plots shown in this paragraph are inspired by Horvath et al. [1]. This visualized form of well structured summaries also helps to compare our results with the result they present in their paper. To keep this section transparent we don't include all plots. For instance RMSE plots are listed in Appendix A.

5.2 Price surfaces

As the neural network performs pretty accurate at learning implied volatility surfaces we want to work out if it can also learn option prices directly. For that purpose we generate datasets according to Section 4.1 but change the output to prices instead of implied volatilities. The obtained results along all three datasets are quite similar, so we just state them for the dataset with the small γ^* parameters.
Figure 14 indicates that the neural network learns the prices on the vast majority of the grid precisely but the upper right-hand corner is problematic. There the prices are smallest on the whole grid since the call option has only a short time to maturity left while is deep out of the money. We suspect that the price surface has a more complex datastructure than the implied volatilities. For example the implied volatilites in the small γ^*

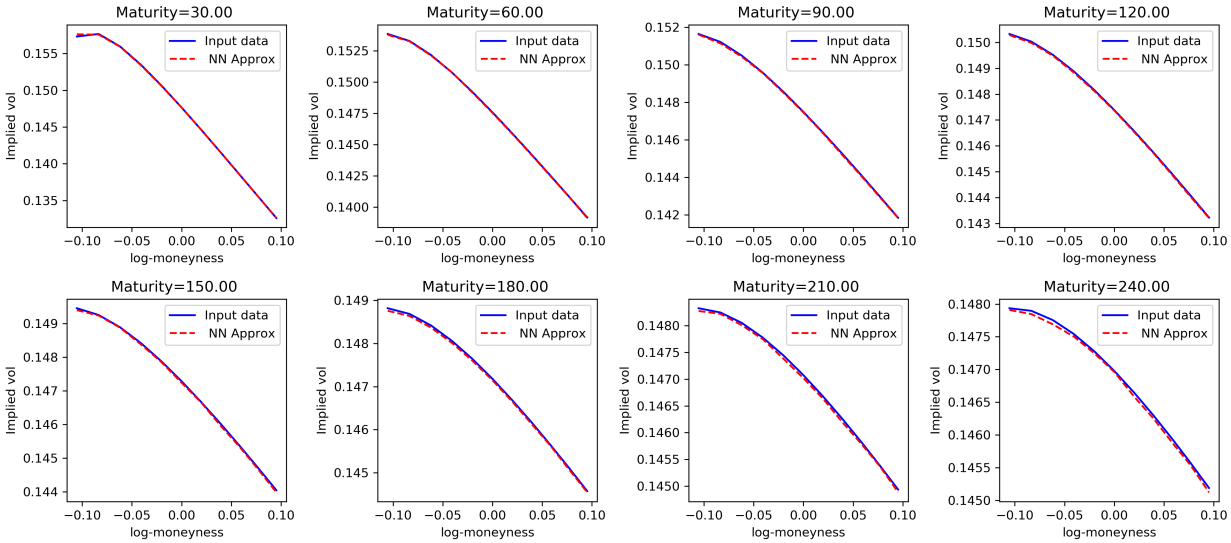

Figure 8: Best fit - Comparison of neural network predicted volatilites and true volatilites in the test set

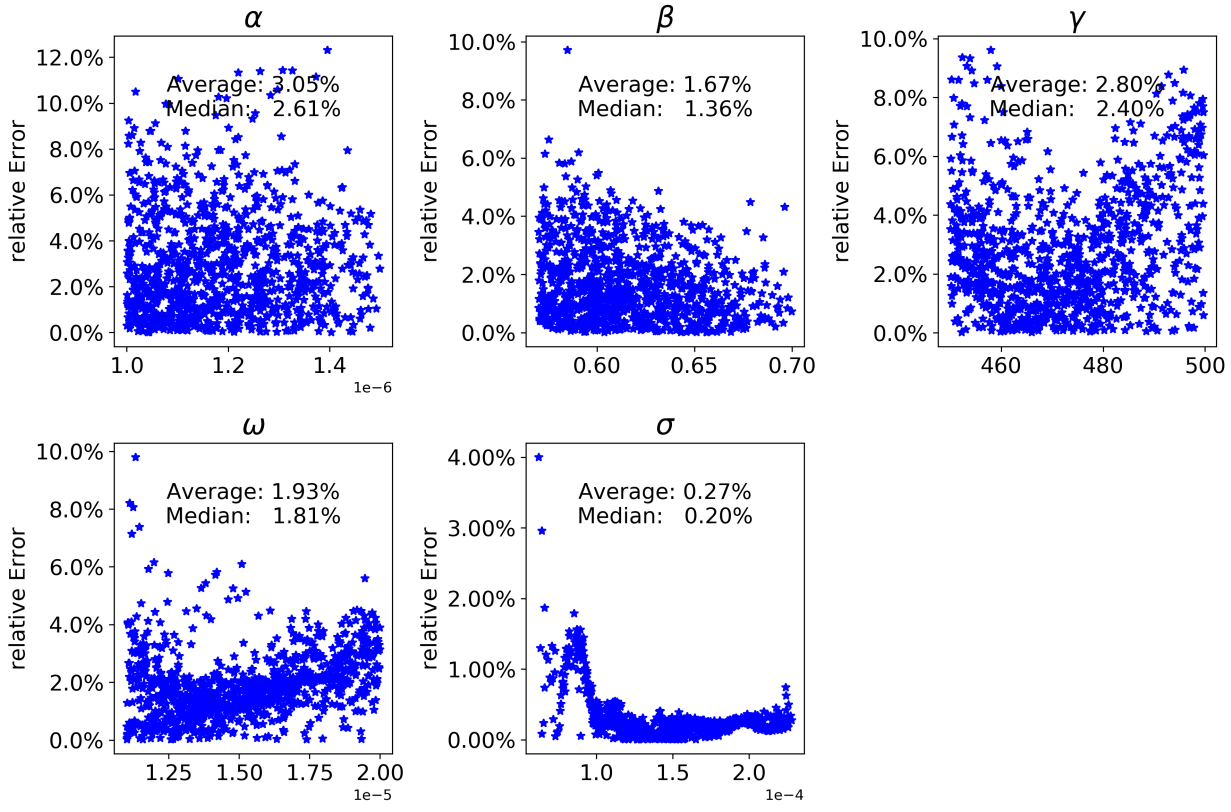

Figure 9: Best fit - Relative errors of the parameters after calibration with Levenberg-Marquardt compared to true parameters in the test set

dataset range from 0.12 up to 0.53 while the corresponding prices range from $1.1 \cdot 10^{-6}$ up to 0.3. Although we normalize these numbers we think that the much higher variability in the prices is more difficult for the net to handle.

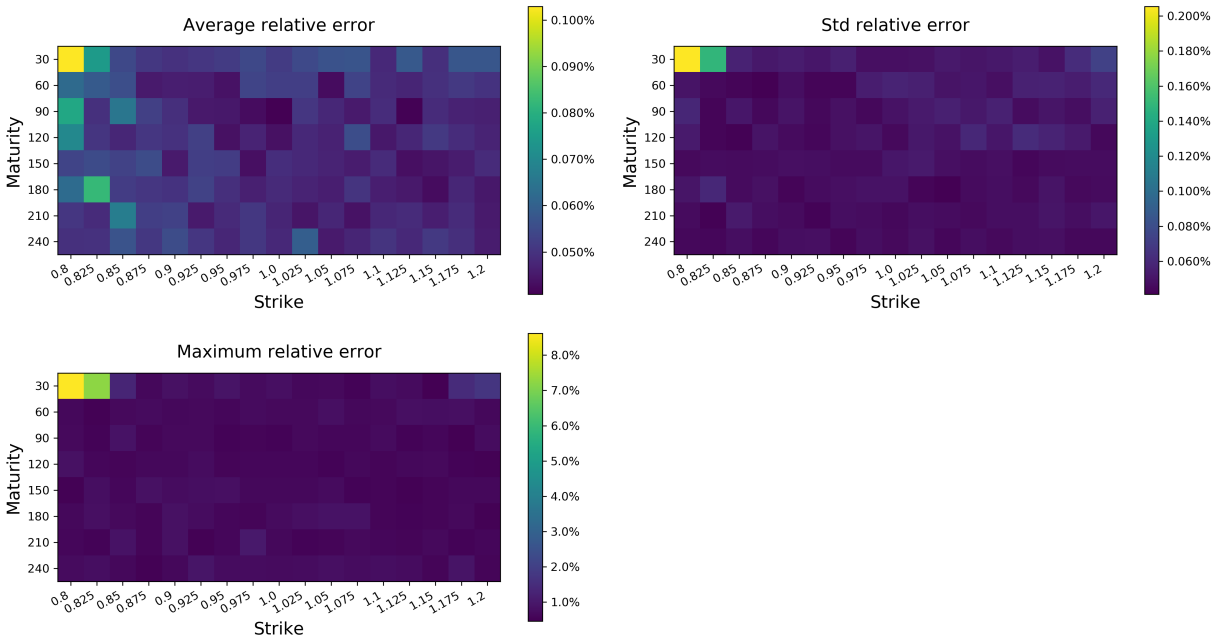

Figure 10: Small γ^* - Relative errors of the neural network predicted volatilities compared to true volatilities in the test set

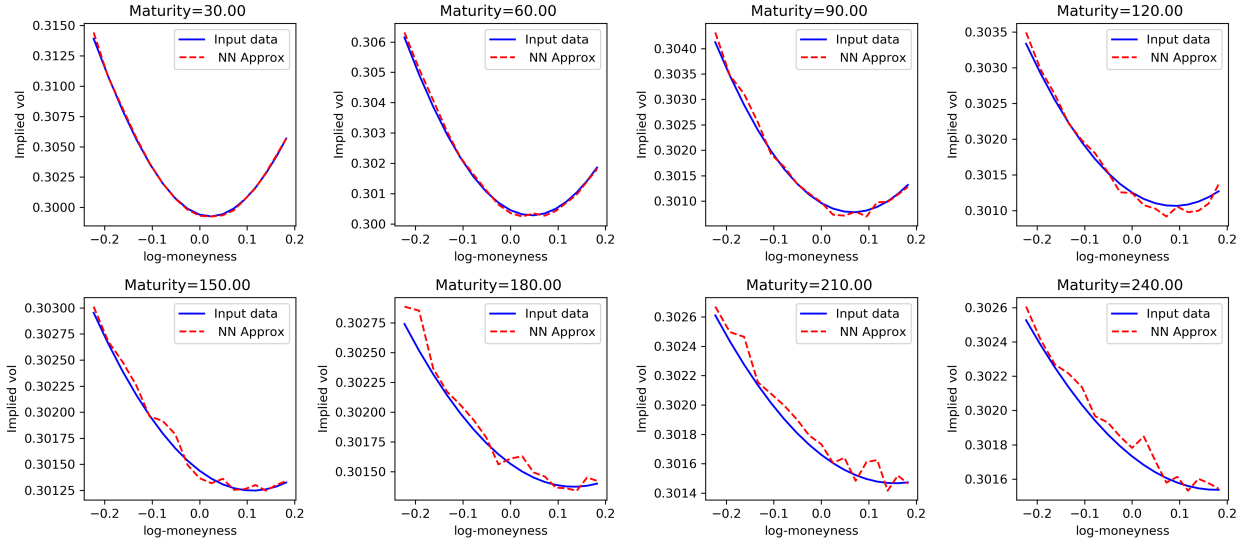

Figure 11: Small γ^* - comparison of neural network predicted volatilites and true volatilites in the test set

5.2.1 Performance in Comparsion to Black-Scholes

Even tho the fit on price surfaces looks promising, a small error itself does not guarantee a good performance of the learned network. Hence, we use the BS model as baseline. In each scenario the annualized expected volatility $\sigma_{year} = \sqrt{252\,\mathbb{E}(h_t)}$ is calculated and used as parameter for BS. The corresponding BS price surface is generated and the RMSE between BS model and HNG model as well as the RMSE between the neural net and HNG model is calculated. We find that in terms of RMSE our approach outperforms BS. The average RMSE is in the magnitude of 10^{-5} for the neural net while the average BS performance tends to be

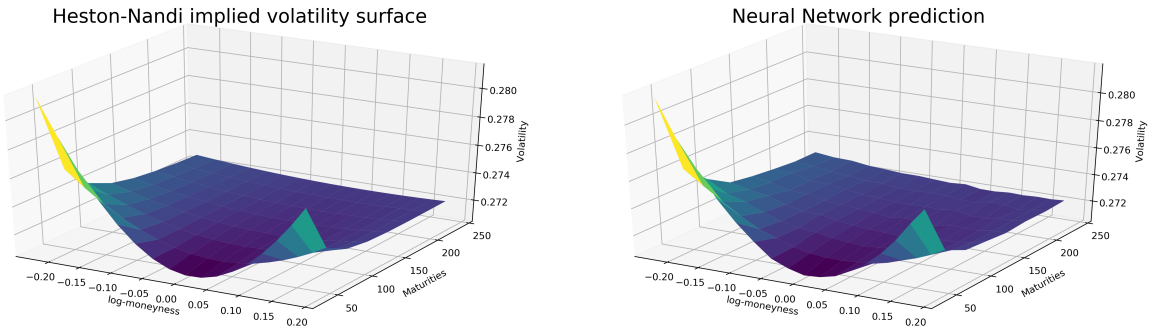

Figure 12: Small γ^* - comparison of volatility surface approximation by neural network to true surface

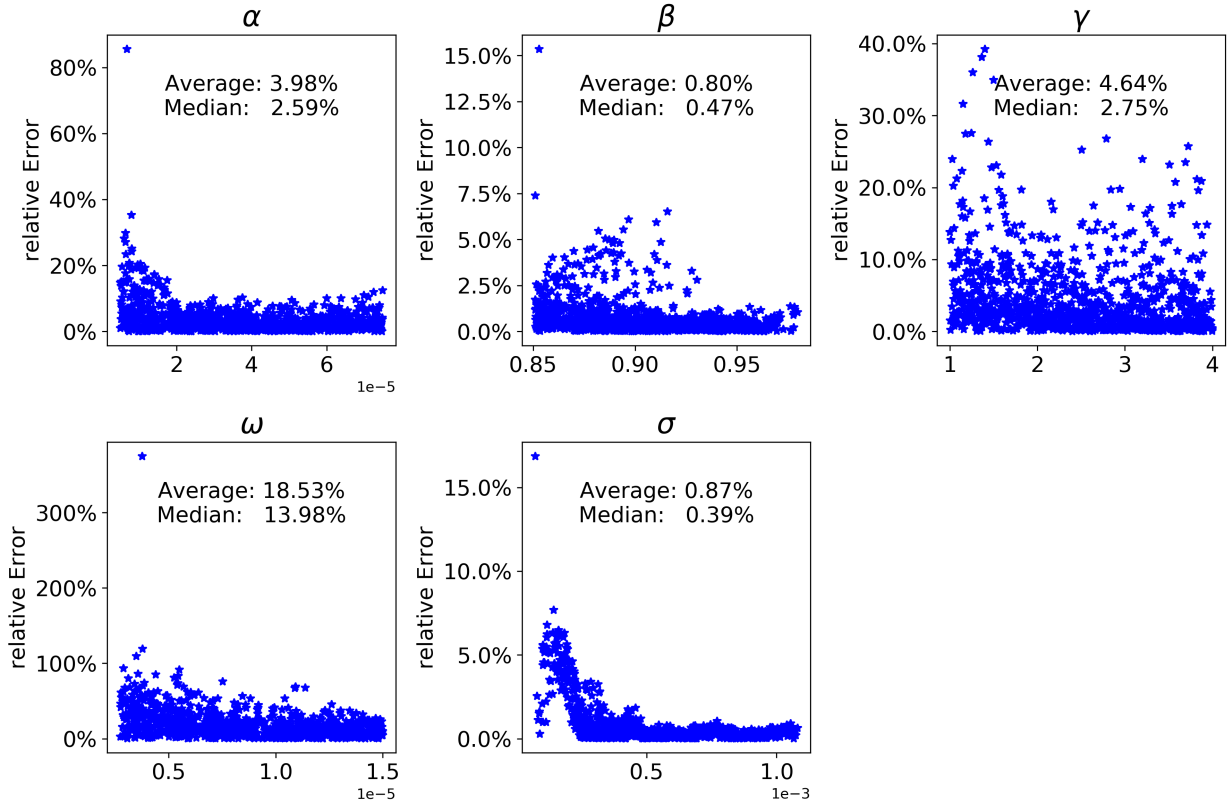

Figure 13: Small γ^* - Relative errors of the parameters after calibration with Levenberg-Marquardt compared to true parameters in the test set

around 10^{-2} (cf. Table 5.2.1). We conclude that the neural net is indeed able to fit the HNG model precisely. Additional plots can be find in Appendix C.

	1st Model [6]		2nd Model [7]		3rd Model [8]	
	NN	BS	NN	BS	NN	BS
Average	1.700e-5	6.549e-2	3.944e-6	5.412e-2	1.271e-5	9.947e-2
Standard Deviation	7.520e-6	7.578e-3	1.228e-6	1.902e-3	4.534e-6	5.001e-3

Table 2: Performance Analysis - RMSE in each setup

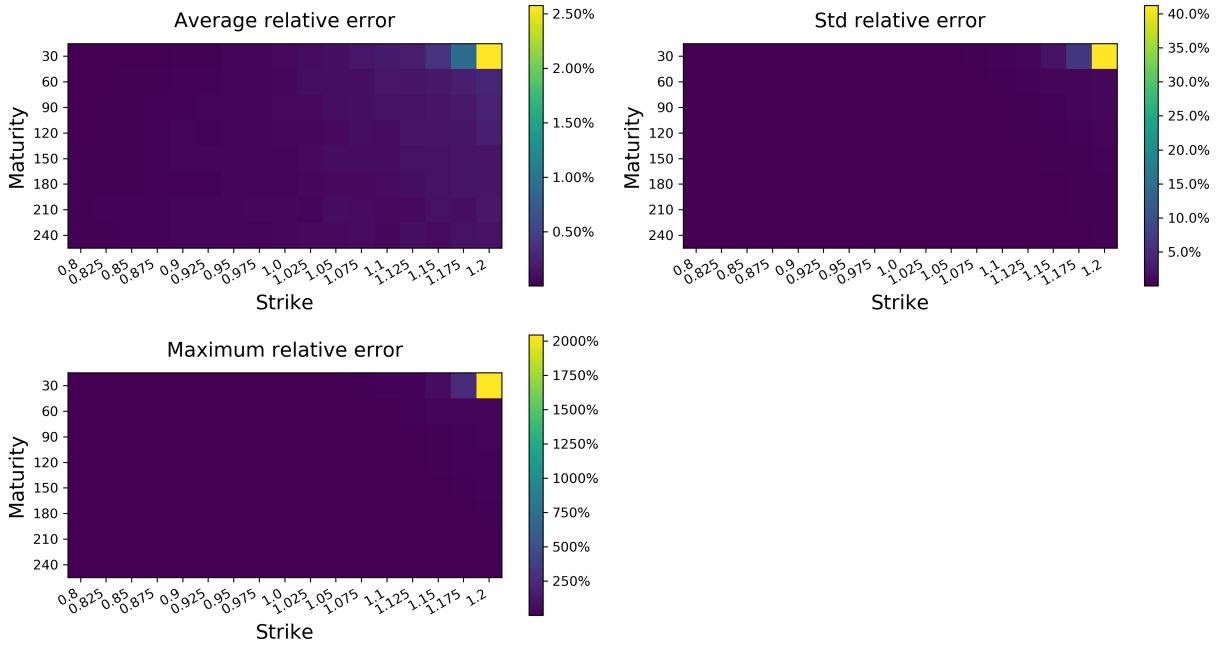

Figure 14: Small γ^* - Relative errors of the neural network predicted option prices compared to true prices in the test set

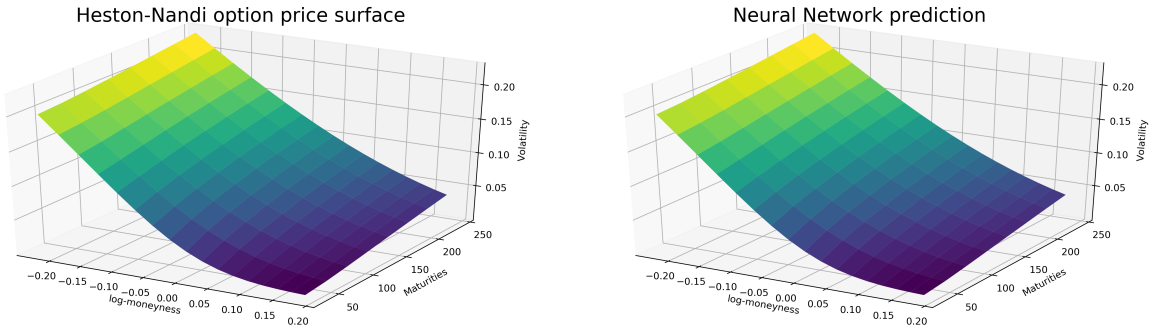

Figure 15: Small γ^* - comparison of option price surface approximation by neural network to true price surface

5.3 Outlook: "How to involve real data"

As mentioned in Section 4.2 it is desirable not only to fit a model precisely but to use the neural net for real data. We take the real call option price surface of different years and treat them as test samples. Then we test how precise the trained neural network can rebuild these real price surfaces. The neural network is trained on the 90% MLE estimator dataset. So far this is just a rough idea and not an optimized procedure. One need to keep in mind that neither the option prices are analysed on their properties (e.g. whether some of them allow arbitrage and should be excluded) nor parameters and neural net are able to adjust for a differences in real risk free rates. Additionally, the errors explode in the upper right-hand corner (cf. Figure 17) consistent with the findings in Section 5.2. So far this approach does not outperform option pricing with standard techniques like BS. We are confident that these challenges can be overcome in the future.

[5] corresponds to the setup in 4.1.1
[6] corresponds to the setup in 4.1.2
[7] corresponds to the setup in 4.1.3

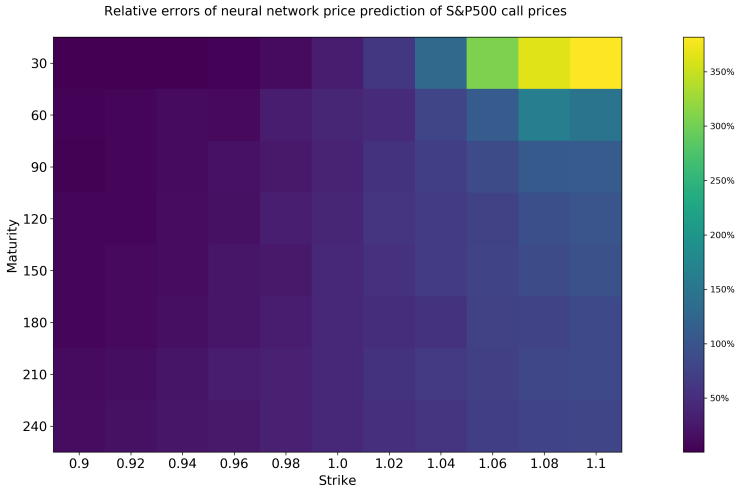

Figure 16: Relative errors of the neural network predicted option prices compared to true S&P500 prices in 2013

5.4 Robustness

5.4.1 Sensitivity towards mispricing

At a first test we want to carry out how sensitive the neural net reacts to small changes in the output data (volatilities). For this purpose we take the dataset generated according to Section 4.1.2. Thereafter we add a $\pm 1\%$ uniform distributed noise to the implied volatilities in the train and validation set and compare the performance of the neural net on the test set to the net initialized by the original train and validation data. On average the noised volatilities lead to 137% higher mean relative errors than the pure volatilities in the test set. Also the maximum relative error is 144% higher. The calibrated parameters have on average a 56% higher median relative error than the parameters calibrated based on the pure dataset.
The results indicate that a precise calculation of call option implied volatilities e.g. via a closed-form solution rather than Monte Carlo simulation significantly increase the performance of the two step calibration approach.

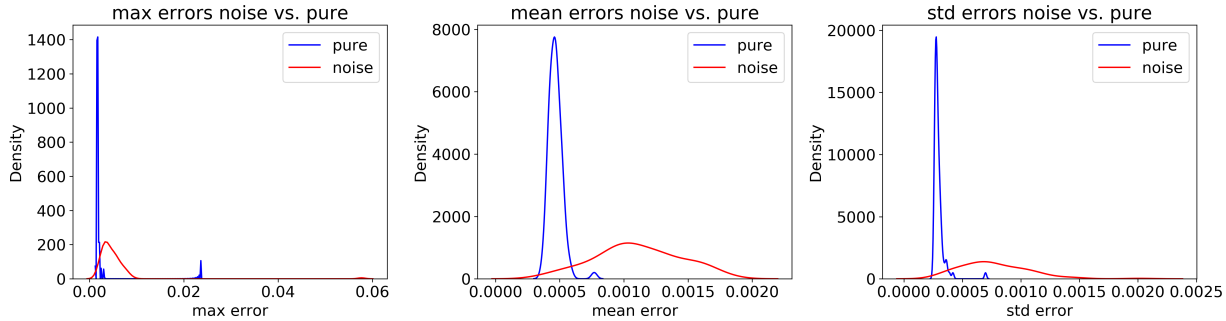

Figure 17: Robustness test for the best fit dataset (cf. Section 4.1.2) - pure dataset vs. dataset with $\pm 1\%$ noise in the train set volatilities

5.4.2 Sensitivity towards training set size

So far our findings imply that the performance is highly dependent on the quality of the generated training set. Additionally, it is important to check the sensitivity of the results towards the size of a training set. For doing so, we fix a given test set but decrease the size of training and validation set randomly by 40%. This

procedure is repeated 20 times. The change in error-distribution can be seen in Figure 22.

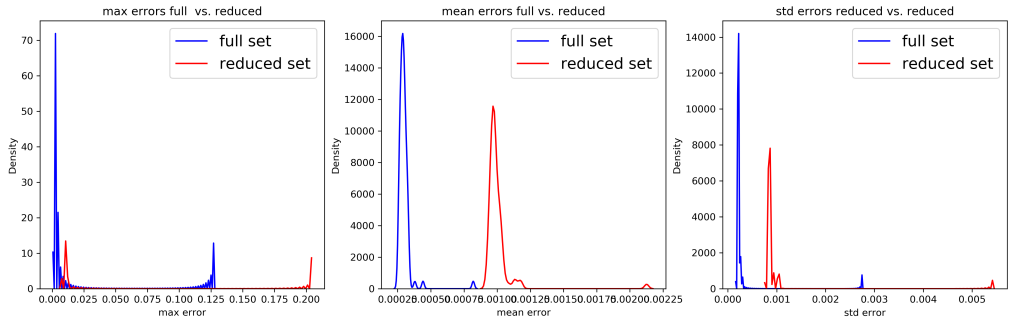

Figure 18: Robustnesstest - Change in size of training set

The size reduction leads to an increase of mean relative error of 233.9%. The maximal relative error increases by 59.8%. Furthermore the distribution of weights in the neural net is compared. 2-sample Kolmogorv-Smirnoff tests indicate significant change in distribution of weights in same layers (cf. Table 3.

	Layer 1	Layer 2	Layer 3	Output
avg. p-val	2.162e-1	1.404e-1	4.273e-1	6.975e-6***
med. p-val	1.672e-1	4.769e-2**	3.772e-1	2.708e-12***
max. p-val	6.069e-1	5.342e-1	9.653e-1	6.194e-5***

Table 3: 2-Sample-KS-test - Comparison of weights in Neural Nets

The results of a 20% size reduction are stated in Appendix B. We conclude that the neural net approach does depend on the size of the training set. Even tho the performance significantly decreases in terms of prediction accuracy, it's surprising to see only small changes in the distribution of weights.

6 Challenges and Discussion

The most time consuming part of our research was the preparation of a well-balanced artificial dataset. Particularly we tried a lot to understand the dependencies between the parameters and their influence on the shape of the volatility surface. The parameters selected out of the intervals need to fulfil certain requirements like the stationarity constraint. They shouldn't lead to numerical errors at the implied volatility calculation and the resulting surfaces need to be learnable by the neural network, too. To accomplish these task guidelines to determine meaningful parameter intervals are helpful. This might be subject of further research.
Similar to Horvath et al. [1] the neural network also struggles with the approximation of the corner points on the strike-maturity grid even though our grids don't contain strikes smaller than 0.8 and larger than 1.2. The effect is intensified if price surfaces are considered (cf. Section 5.2)
Another subject for further research would be to emphasize the topic of real data. The greatest challenge there is again the choice of parameter intervals which gives the neural network the opportunity to learn the volatility surface implied by real data. Especially, we are not able to reproduce the particularly pronounced smile with Heston-Nandi-GARCH so far. Instead our volatility smiles are too flat.

7 Conclusion

We find that the approach of Horvath et al. [1] is generalizable to discrete-time models. A neural net is able to outperform Black-Scholes in fitting the Heston-Nandi model. However, the accuracy of the trained net is highly dependent on the choice of training set. Parameter choice as well as the estimation accuracy are crucial to ensure a good performance. We provide a first idea on implementing real S&P500, but calibration and fine tuning of this approach is subject to further research.

References

[1] Horvath, B., Muguruza, A. and Tomas, M., 2019. Deep learning volatility. Available at SSRN 3322085.

[2] Heston, S.L. and Nandi, S., 2000. A closed-form GARCH option valuation model. The review of financial studies, 13(3), pp.585-625.

[3] Engle, R.F., 1982. Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. Econometrica: Journal of the Econometric Society, pp.987-1007.

[4] Heston, S.L. and Nandi, S., 1997. A closed-form GARCH option pricing model.

[5] Hansen, P.R. and Lunde, A., 2005. A forecast comparison of volatility models: does anything beat a GARCH (1, 1)?. Journal of applied econometrics, 20(7), pp.873-889.

[6] Chorro, C., Guégan, D. and Ielpo, F., 2015. A Time Series Approach to Option Pricing: Models, Methods and Empirical Performances, 2015th edn, Springer, Berlin, Heidelberg.

[7] Hernandez, A., 2017. Model calibration with neural networks. Risk.

[8] Horvath, B., Jacquier, A. and Muguruza, A., 2017. Functional central limit theorems for rough volatility. Available at SSRN 3078743.

[9] Badescu1, A., Grigoryeva, L., and Ortega, J., 2019. Option pricing and hedging with one-step unscented Kalman filtered factors in non-affine stochastic volatility models. Working Paper

[10] Podkovyrov, P., 2019. Option Pricing Using Machine Learning Techniques. Master Thesis University of Konstanz, Germany.

[11] Byun, S.J., 2011. A Study on Heston-Nandi GARCH Option Pricing Model. KAIST Business School, Korea.

Appendix

A Further Volatility Plots

This section shows furhter volatility plots and error analysis for Chapter 5.1

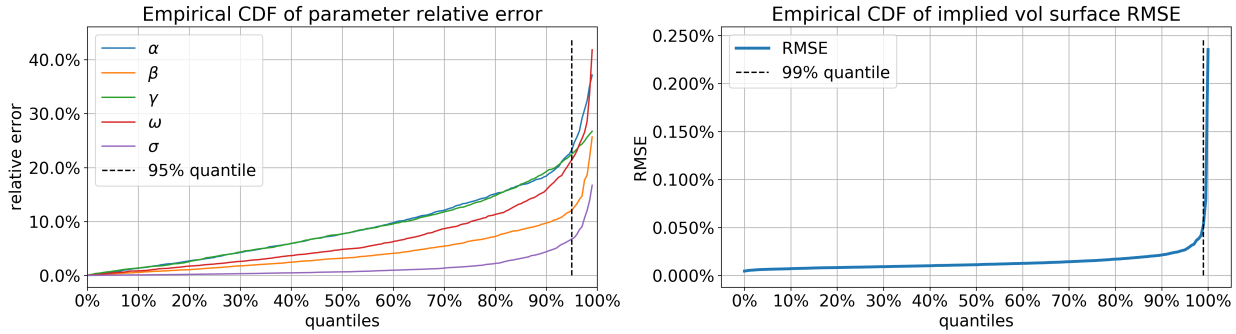

Figure 19: 90% Maximum Likelihood confidence interval - Empirical CDF of parameter relative error and empirical CDF of implied volatility surface RMSE

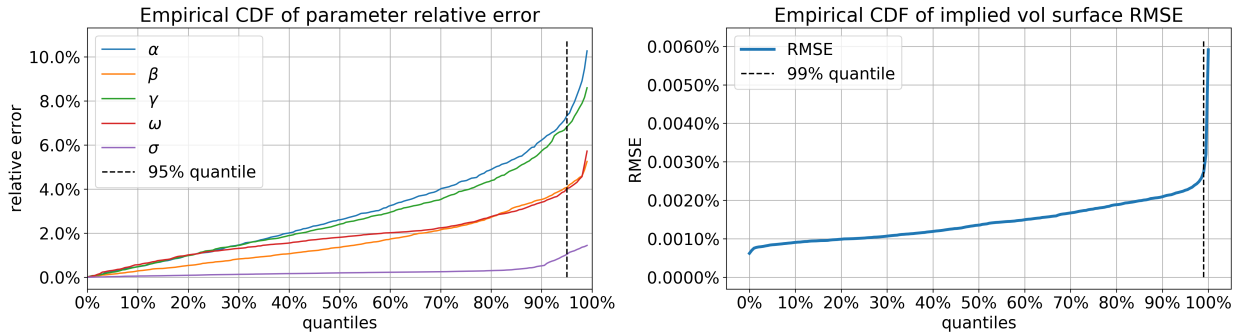

Figure 20: Best fit - Empirical CDF of parameter relative error and empirical CDF of implied volatility surface RMSE

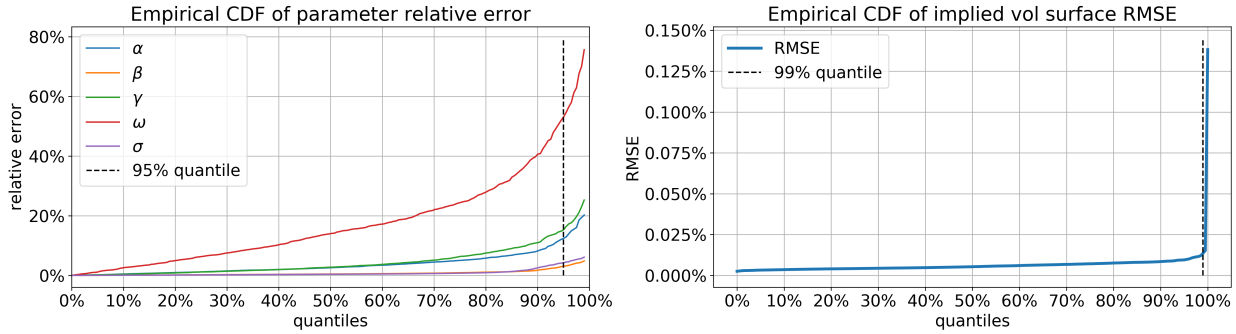

Figure 21: Best fit - Empirical CDF of parameter relative error and empirical CDF of implied volatility surface RMSE

B Further Robustness Test

This section shows the results of a 20% reduction of training set size (cf. Section 5.4.2)

	Layer 1	Layer 2	Layer 3	Output
avg. p-val	4.553e-1	2.066e-1	2.701e-1	1.184e-8***
med. p-val	4.219e-1	1.851e-1	1.951e-1	3.612e-12***
max. p-val	9.804e-1	6.125e-1	7.318e-1	2.192e-7***

Table 4: 2-Sample-KS-test - Comparison of weights in Neural Nets

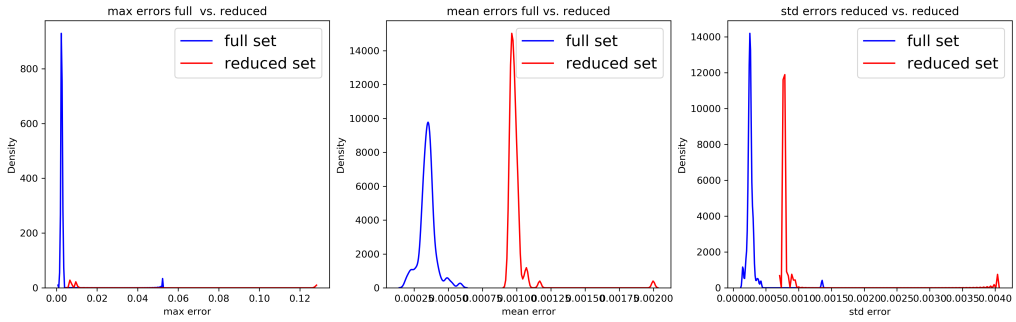

Figure 22: Robustnesstest - Change in size of training set (20%)

C Black Scholes Comparison

This section corresponds to Section 5.2.1.

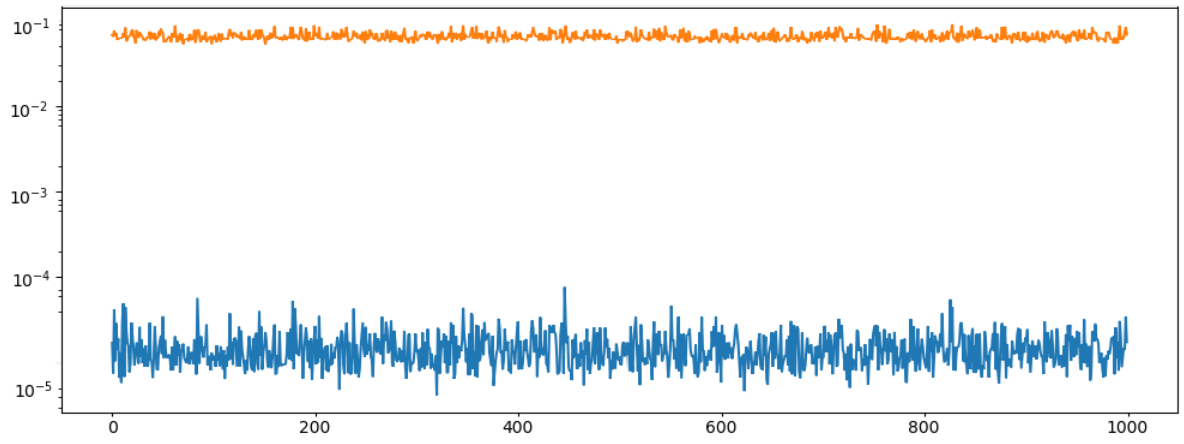

Figure 23: RMSE of Neural Net (blue) and Blacks Scholes (orange) on test set