Evaluating the Accuracy and Efficiency in the Calibration of Stochastic Volatility Models

Final Report - Undergraduate Thesis

Ross Cameron
Thesis 4490Z
Department of Computer Science
University of Western Ontario

April 11th, 2025

Project Supervisor: Prof. Ankush Agarwal, Dept. of Statistical and Actuarial Sciences
Course Instructor: Prof. Nazim Madhavji, Dept. of Computer Science

Glossary:

MLP (Multi-Layer Perceptron): A neural network created with a series of layers including an input layer, output layer, and a number of hidden layers each of which with a set number of neurons.

SVM (Stochastic Volatility Model): Refers to a model which factors the volatility of volatility into the calculation of the price. Volatility of volatility refers to the way volatility fluctuates rather than assuming it to be constant.

LHS (Latin Hypercube Sampling): A form of stratified sampling in which ranges are given and subdivided into segments to be sampled from. This type of sampling ensures strong variance among data across the entire range.

MSE (Mean Squared Error): The average of the squared difference between predicted values and true values.

Structured Abstract:

Context and Motivation:
The financial industry uses complex models and formulas to calculate and set options prices. The efficiency and accuracy of these models are critical, especially for large financial institutions, for which the calculation of option prices can heavily impact potential returns. The calibration of these complex models to the market data becomes much more efficient when applying modern machine-learning techniques. Evaluating the variations of these models becomes essential in optimizing performance.

Research Question:
What is the optimal number of hidden layers and neurons in a MLP(Multi-Layer Perceptron) for the calibration of the Heston model?

Principal ideas:
- Stochastic Volatility Models
- Heston Model
- Model Calibration
- Machine Learning

Research Methodology:
Create a program which utilizes deep learning to calibrate the Heston model, then evaluate the results with varying MLP architectures.

Anticipated Type of Result:
Based on the extensive numerical experiments, we will be able to determine the optimal architecture for an MLP when calibrating the Heston model.

Anticipated Novelty:
Within current research on the topic of Machine Learning in Finance, there exists a research gap around measuring the difference in accuracy/efficiency when applying different MLP architectures for the calibration of complex stochastic models.

Anticipated Impact of Results:
In theory, I expect this research to contribute to the understanding of machine learning's role in calibration of complex financial models.
In practice, I expect this research to enable faster, more accurate pricing models for financial institutions.

Table of Contents:

1) Introduction:

The Heston model differentiates itself from previous European option price models through its use of stochastic volatility, meaning it accounts for the change of volatility over time. With the recent advances in machine learning technology, much research has been conducted on how to apply this technology to quantitative finance or, more specifically, stochastic volatility models. One paper, in particular, describes their implementation of a program which calibrates the rough Heston model using deep learning (Rosenbaum et al., 2021). However, there are still many areas of this research which have not yet been explored. Specifically, there needs to be an optimized architecture for the MLP used in deep learning calibration of the Heston model.

This thesis focuses on the calibration of the Heston model using deep learning and compares different MLP architectures as they affect the program output. This focus is rephrased in the research question: What is the optimal number of hidden layers and neurons in an MLP for the calibration of the Heston model?

When analyzing the collected data, it was clear that the optimal MLP architecture was shallow and wide; in this case, it had two hidden layers and 1000 neurons per layer.

This thesis builds off of previous research which focused on utilizing deep learning to calibrate stochastic volatility models and fills the research gap by determining optimal MLP architectures for these deep learning model calibration programs (Rosenbaum et al., 2021).

This report is broken down into 11 sections, including the introduction, which is in section 1. Section 2 discusses the topic background and related work conducted in other research papers. Section 3 breaks down the research project into four clear objectives. Section 5 discusses the research methodology, which includes any research concepts utilized and the overall approach to conducting the research. Section 6 displays and analyzes the data collected from the research and discusses the novelty and significance of that data. Section 7 includes a thorough discussion on the limitations of the results. Section 8 summarizes the findings, and section 9 discusses what future research could expand on the lessons learned in this thesis. This is followed by sections 10 and 11, which include the acknowledgements and appendix.

2) Background and Related Work:

2.1: Option Price Models

The Black-Scholes Model is considered by some to be a foundational step in developing modern finance and option pricing theory (Bueno-Guerrero, 1993). However, The Black-Scholes model has a number of limitations, the most relevant being that it assumes volatility will remain constant across the options life (Bueno-Guerrero, 1993). This has led to the development of more advanced models, such as the Heston model. The Heston model differentiates itself from the black-scholes model using the variable sigma. In the Heston model, sigma represents the volatility of volatility, which is the measurement of how volatility changes throughout an option's life (Bueno-Guerrero, 1993). This drastically improves the black scholes model, which

assumes volatility to be constant. By accounting for the change in volatility over time, the model can predict more accurate option prices.

2.2: Heston Model Equations

Listed below is the European call price function for the Heston model (Heston, 1993). These are the base equations used in the Heston model to calculate European call option prices. P1 and P2 represent integrals involving the characteristic function of the Heston model (Heston, 1993).

$$C(S0, v0, t) = S0 * P1 - P(t, T) * P2$$

$$Pj(x, v0, T, ln[k]) = \frac{1}{2} + \frac{1}{\pi}\int_0^{\infty} Re[\frac{e^{-i\phi ln[K]} fj(x, v0, T, u)}{iu}]du$$

Calculating the characteristic function requires a number of parameters, all of which are defined in the methodology section of this paper. P1 and P2 have different values for b and u, which are shown below, along with the rest of the characteristic function equations (Heston, 1993). C and D will be very important going forward as the call price function is adjusted to factor out K (the strike price).

$$j = 1: b = \kappa - \rho\sigma, u = 0.5$$

$$j = 2: b = \kappa, u = -0.5$$

$$d = \sqrt{(\rho\sigma i\phi - b)^2 - \sigma^2(2u\phi i - \phi^2)}$$

$$g = \frac{b - \rho\sigma i\phi + d}{b - \rho\sigma i\phi - d}$$

$$C(T; \phi) = r\phi iT + \frac{\kappa\theta}{\sigma^2}[(b - \rho\sigma i\phi + d)T - 2ln(\frac{1 - ge^{dT}}{1 - g})]$$

$$D(T; \phi) = \frac{b - \rho\sigma iu + d}{\sigma^2} * (\frac{1 - e^{dT}}{1 - ge^{dt}})$$

Analysis and Research Gap:

Machine learning, as a whole, has become deeply intertwined with finance in the commercial world (Ahmed et al., 2022). One journal discusses its current applications and the future possible applications of this technology. In this journal, one author stresses the necessity of machine learning in "capturing the linear and nonlinear behaviours of finance variables"(Ahmed et al., 2022). It is clear then, both in the scientific and commercial senses, that the applications of machine learning in finance have become unavoidable. Despite this, there are still many gaps within this area that have not been researched. One such gap, which is the centre of this research, is the comparison of MLP architectures for calibrating stochastic volatility Models, specifically the Heston Model.

3) **Research Objectives:**

O1:Implement machine learning architecture to calibrate the Heston model
O2:Generate training data using LHS (Latin Hypercube Sampling)
O3:Gather metrics on performance for varying MLP architectures
O4:Perform final analysis

4) **Methodology:**

Listed below are the parameters used in the Heston model European call price calculation as well as their definitions (Sridi, 2023).

$$S0: initial\ price$$
$$K: strike\ price$$
$$\frac{S0}{K}: moneyness$$
$$r: risk\ free\ rate$$
$$T: time\ to\ maturity$$
$$v0: initial\ variance$$
$$\kappa: mean\ reversion\ rate$$
$$\theta: long\ run\ average\ variance$$
$$\sigma: volatility\ of\ volatility$$
$$\rho: \boldsymbol{correlation}\ between\ asset\ price\ and\ variance$$

Since this program utilizes moneyness instead of S0 and K separately, the basic Heston equations need to be adjusted to factor out K and S0. By setting P(t,T) to 1 and utilizing the given equality, we get the following.

$$P(t,T) = 1$$
$$\underline{C}(\frac{S0}{K}, v0, t) = \frac{S0}{K} * P1 - P2$$
$$e^{-i\phi ln[K]} fj(x, v0, T, u) = e^{C(T,u)-D(T,u)*v0+i\phi ln[\frac{S0}{K}]}$$
$$Pj(x, v0, T, ln[k]) = \frac{1}{2} + \frac{1}{\pi}\int_0^\infty Re[\frac{e^{C(T,u)-D(T,u)*v0+i\phi ln[\frac{S0}{K}]}}{iu}]du$$

When generating values for kappa, theta, v0, sigma, and rho, the ranges of possible values are listed below. The program also generates call prices for a matrix between the log-moneyness values and the maturity values. The risk-free rate is always set to 0 to simplify the training data. The log-moneyness and maturity values were collected from a paper written on a similar topic (Rosenbaum et al., 2021).

$ln(\frac{S0}{K})$ = {-0.15, -0.12, -0.1, -0.08, -0.05, -0.04, -0.03, -0.02, -0.01, 0.0, 0.01, 0.02, 0.03, 0.04, 0.05} ∪ {-0.1, -0.05, -0.03, -0.01, 0.01, 0.03, 0.05, 0.07, 0.09, 0.11, 0.13, 0.15, 0.17, 0.19, 0.21}

T = {0.03, 0.05, 0.07, 0.09, 0.2, 0.5, 1.0, 2.0, 3.0}

r = 0, $\kappa = [10^{-4}, 2]$, $\theta = [10^{-2}, 2]$, $v0 = [10^{-2}, 2]$, $\sigma = [10^{-2}, 2]$, $\rho = [-1,1]$

Calibration, as it is defined in the glossary, refers to the process by which (in the context of this research) parameters are estimated to most closely match the true (generated) values. In this case, the program is provided with a series of flattened vectors of values for moneyness, maturity, and the corresponding price, which is calculated with the generated parameter values for that sample. This means that for every sample of parameters generated, there is a matrix of price values for each pair of moneyness and maturity values. The prices are calculated using the Heston call price function, the code for which is available in the appendix. The model then attempts to learn and predict parameter values given the input described above. The process of calibration involves a series of steps, including generating/preparing data, model design, training, and evaluation.

Data samples of kappa, theta, v0, sigma, and rho were generated using Latin Hypercube Sampling. Latin Hypercube Sampling is a stratified sampling method in which each range is broken down into intervals, each of which is sampled at an even distribution (Loh, 1996). This helps to ensure the combinations of values are diverse and representative of the whole range space. These generated parameter values are used with the predetermined log-moneyness and maturity values to create the model input. Before prices can be calculated, log-moneyness must first be transformed back to moneyness. In this program, the model is trained on the log of kappa, theta, and v0 to improve learning. After training, the predicted values are transformed back to their non-log values. The number of samples generated is 10 times the desired number of samples. This is done to account for filtering due to not fulfilling the feller condition, which is $2\kappa\theta > \sigma 2$. This filter leaves the model with approximately the desired number of samples.

The machine learning aspect of the program utilizes an MLP, which is altered between executions to adjust the number of hidden layers and neurons per layer. Since this research is centred around comparing MLP architectures, data is collected on each of 3 different iterations: 5x200, 4x250, and 2x500. Each iteration, while all having varying numbers of layers and neurons per layer, has the same total number of neurons: 1000. This was done to ensure that the results were not impacted simply by the number of neurons. Data was also collected on the best-performing MLP after it was adjusted to have twice the number of total neurons. The output layer is set to 5 neurons, as this is the number of parameters the model hopes to predict.

Each MLP architecture was evaluated using every combination of the following batch sizes and learning rates: batch sizes = [32, 64, 128, 256] learning rates = [10-2, 10-3,10-4]. After analyzing the outputs, it was found that each MLP performed best using a 128 batch size and a 10-3 learning rate. All future collected data was done using these specifications. Each MLP was evaluated using 50 epochs. The model training is designed to use MSE loss; this allows the model to track how accurate its predictions are against the true values.

Evaluation

At each epoch, the program outputs the current train and validation MSE loss for the parameters. Upon completing execution, the program outputs a graph which plots the train and validation MSE loss against each epoch. This data is valuable as the shape of these plotted lines can indicate whether the model is performing as expected. The program also outputs both the

MSE and R2 score for each parameter individually and then plots the predicted values against the true values.

5) Results:

Three MLP architectures with varying depth and width were evaluated: 5 layers with 200 neurons per layer, 4 layers with 250 neurons per layer, and 2 layers with 500 neurons per layer.

The following output was collected when testing on ≈100,000 samples. Determining the model's effectiveness relies on both the MSE (Mean Squared Error) and R^2 (Coefficient of Determination) for each parameter. Ideally, the MSE should be as close to zero as possible while R^2 score should be as close to one as possible. As well, the program plotted the predicted values against the true values for each predicted parameter. These results show an effective model with acceptable MSE and R^2 scores across all five parameters.

	5x200		4x250		2x500	
	MSE	R^2	MSE	R^2	MSE	R^2
κ	0.012015	0.960596	0.010935	0.964137	0.013668	0.955172
θ	0.005061	0.983233	0.006040	0.979988	0.005093	0.983126
σ	0.002839	0.786907	0.002073	0.844389	0.003001	0.774769
ρ	0.008530	0.982916	0.007041	985897	0.011185	0.977598
v0	0.000051	0.996241	0.000039	0.997102	0.000019	0.998623

table 1. MSE & R^2 scores

The data shown above in table 1 breaks down each MLP's performance on predicting each parameter. When analyzing this data, it is clear that the 5x200 model performs the worst across all parameters except theta, which is narrowly better than the 2x500 MLP. The 2x500 MLP displays strong predictions across all parameters, especially v0 which has a higher R^2 score than any parameter predicted by any MLP in the table. It is worth noting that the 4x250 MLP not only performs well in all parameters, but also dramatically improves upon sigma's prediction by the other MLPs. Displayed below in figures 1-3 are graphs of the predicted vs true value for each parameter predicted by each MLP. These graphs are consistent with the data shown in the table.

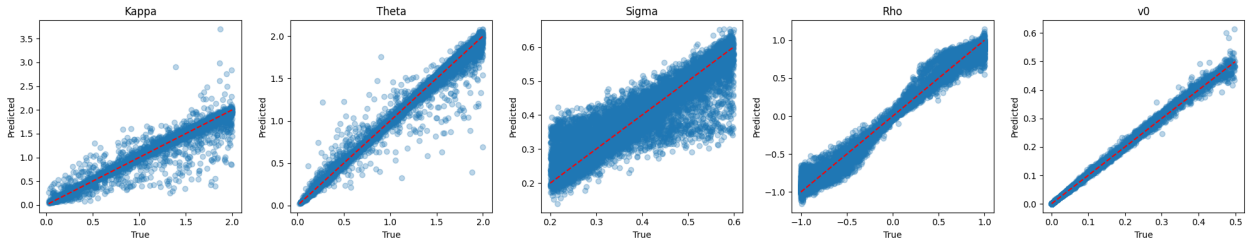

figure 1. Real vs Predicted Values (5x200)

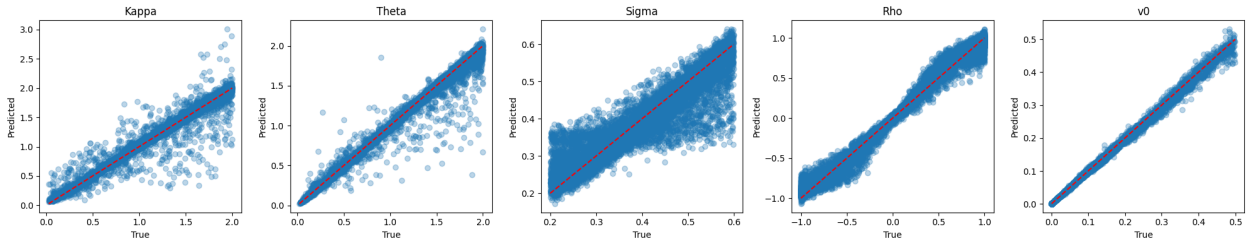

figure 2. Real vs Predicted Values (4x250)

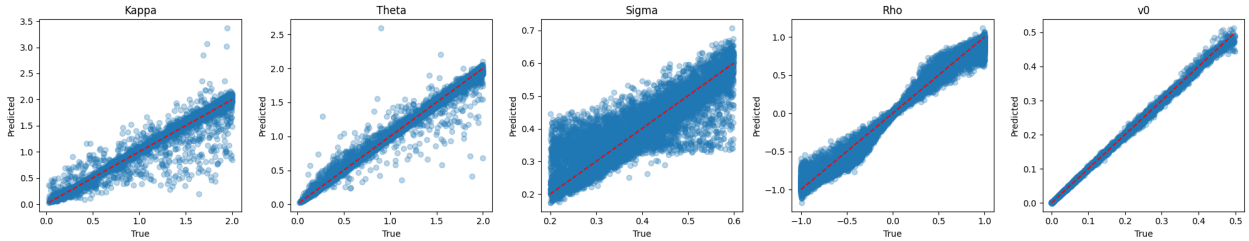

figure 3. Real vs Predicted Values (2x500)

Tracking the train and valid loss at each epoch is essential in confirming that the model continues to learn at a steady rate throughout execution. In the example below, every tenth epoch is shown to demonstrate that the model continues learning until completing execution. As well, the program plotted the log of the train and validation MSE loss against each epoch.

	5x200		4x250		2x500	
	Train Loss	Valid Loss	Train Loss	Valid Loss	Train Loss	Valid Loss
Epoch 1	0.497182	0.241213	0.433096	0.221732	0.353361	0.229631
Epoch 10	0.164321	0.157729	0.163954	0.155900	0.163571	0.154085
Epoch 20	0.157677	0.149358	0.156435	0.156597	0.135699	0.134303
Epoch 30	0.147886	0.143627	0.135563	0.129508	0.083601	0.071342
Epoch 40	0.087122	0.071270	0.070712	0.094338	0.063882	0.059574

Epoch 50	0.058652	0.057677	0.056319	0.045638	0.055818	0.061106

table 2. Train and Validation MSE Loss

The data shown in table 2 demonstrates the MLPs learning for all parameters throughout execution. These results clearly show that the 2x500 MLP produced the smallest final train loss. This is a very good indicator as to which MLP was most successful across all parameters. Shown below in figures 4-6 are graphs of the log MSE for train and validation loss against epochs. These graphs show that each MLP performs as expected despite the slight volatile behaviour with the validation loss.

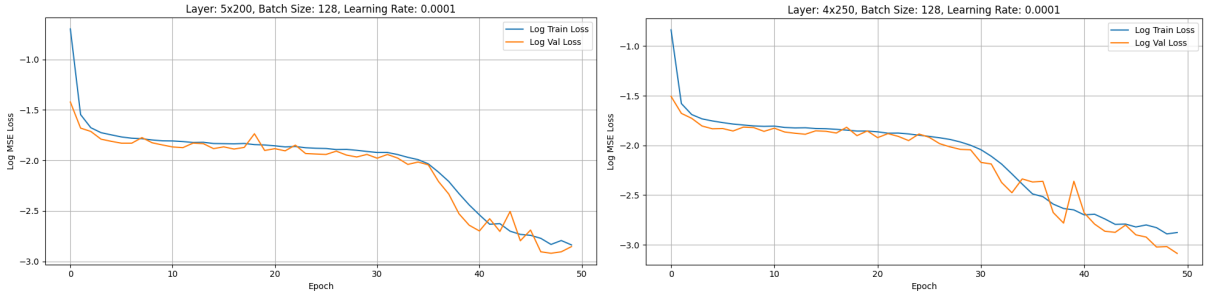

figure 4 & 5. Log of Train and Validation MSE Loss (left:5x200, right:4x250)

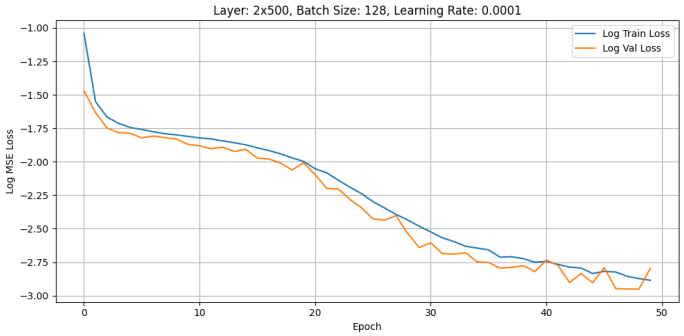

figure 6. Log of Train and Validation MSE Loss (2x500)

By analyzing all the data shown above, it is clear the 2x500 architecture performed the best out of all measured MLPs. While the 4x250 displayed impressive predictions for sigma, the train loss was simply lower for the 2x500 MLP. After coming to this conclusion, the logical next step is to measure the same architecture with an increased number of neurons. In this example, the 2x500 MLP was modified to instead be 2x1000 (2000 total neurons).

	2x1000	
	MSE	R^2
κ	0.012810	0.957176
θ	0.008094	0.973569

σ	0.001916	0.857335
ρ	0.006785	0.986522
v0	0.000021	0.998515

table 3. MSE & R^2 scores

After increasing the total neurons, the 2x1000 MLP shows similarly strong results to the previous MLPs with the exception of sigma. Sigma shows a tremendous improvement on the MSE and R^2 scores of the previously examined MLPs. This improvement can be seen in the graphs shown below in figure 7.

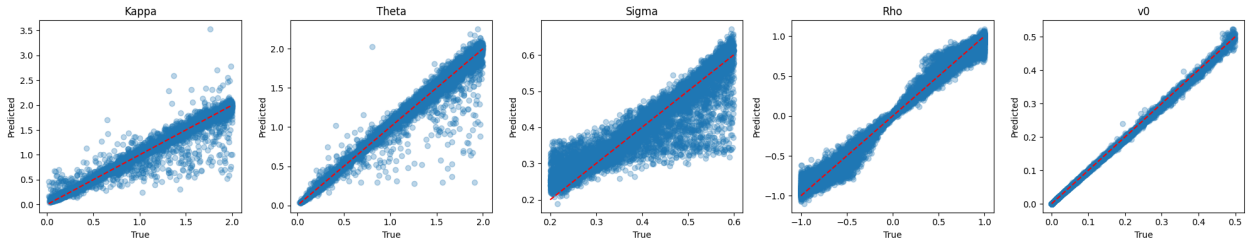

figure 7. Real vs Predicted Values (2x1000)

	2x1000	
	Train Loss	Valid Loss
Epoch 1	0.305088	0.210633
Epoch 10	0.157394	0.183154
Epoch 20	0.086233	0.077982
Epoch 30	0.060595	0.060275
Epoch 40	0.053293	0.050792
Epoch 50	0.046815	0.041301

table 4. Train and Validation MSE Loss

The 2x1000 train loss shows significant improvement among the previous MLPs. As well, it is worth noting that the graph shown below in figure 8 shows increased volatility in the validation loss. This is not ideal but could be improved upon by adjusting the batch size and learning rate.

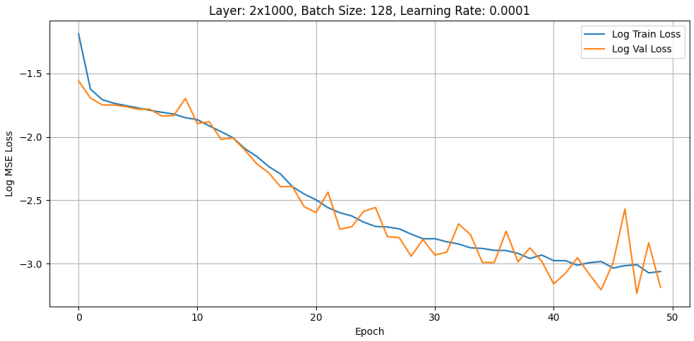

figure 8. Log of Train and Validation MSE Loss (2x1000)

Key Takeaways:

Across each MLP, the validation and train loss are convergent which is a good indicator that the model is performing as expected. It is also apparent that the 2x500 MLP outperformed both the 5x200 and 4x250 MLPs in train MSE loss. This is the most significant metric in comparing the architectures as it averages the MLPs' success across all parameters. What was very interesting was the significant improvement which came with increasing the total neuron count. By doubling the neurons per layer in the highest performing layer, the train MSE loss dropped to 0.046815. This is a significant improvement upon the previous MLPs. The important takeaways from these results are that shallow MLPs (meaning to have few layers) and wide MLPs (meaning to have many neurons) are the optimal architectures for calibrating the Heston model at ≈100,000 samples.

Novelty:

While much progress has been made in the integration of machine learning into the financial discipline, little has been done to identify the optimal approaches. This research provides a clear quantitative comparison of different MLP architectures when applied to the calibration of the Heston model. A paper sharing a similar topic discusses the calibration of the rough Heston model using deep learning (Rosenbaum et al., 2021). Some procedures included in this paper were followed to ensure accuracy in this thesis project; one specifically was the use of the same values for moneyness (S0/K) and time to maturity (T). The T values did have to be adjusted to improve learning for some parameters. This thesis project takes the previously mentioned research a step further by calibrating the Heston model using a neural network and optimizing the MLP.

6) Discussion:

One initial threat to the validity of the research was the accuracy of the Heston call price function. This function had to be verified against other Heston parameters and results to ensure that the values returned were valid call option prices. Without this validation, the data collected would be meaningless as the model would be calibrated to an irrelevant equation. Another threat was the data sample size. The biggest issue with generating data for this project is that the

program is very computationally heavy. Running the program on a large number of samples requires an even greater number of integrals, which leads to extremely long runtimes.

From a research perspective, this project can provide significant value to future researchers of this discipline. By understanding which approach is optimal, one can begin to work backwards to identify why it is optimal. This research also has an effect on practice. This research is not groundbreaking in the sense that it is creating a new approach to calibrating stochastic volatility models. However, it does provide insights into what approach is best suited to an individual's needs. By implementing the optimal approach, one could save computation time and cost.

These results were collected in a very specific situation. Firstly, the program was designed to predict the parameters for the Heston model. This means that until further research is done, the results can not be applied to programs that calibrate other models (Black Scholes, Rough Heston, etc.) or programs which solve for one value (e.g. option price). As well, all data was collected on roughly 100,000 samples, which means that the results are applicable to situations with an equal number of samples used.

At the current point of research, the results are not extremely generalizable. The data was collected in very specific circumstances and should not be generalized until future research is done. However, the results can be applied to other deep learning calibration programs which solve for the parameters of the Heston model.

7) Conclusions:

This project set out to expand on previous research into the calibration of stochastic volatility models by focusing on optimizing the MLP architecture. Over the course of conducting this research all of the predefined objectives were accomplished. A program was created to calibrate the Heston model using deep learning [O1]. The program was run on data that was generated using LHS [O2]. The program calculated and returned metrics such as MSE loss, R^2 score, train loss, and valid loss [O3]. Finally, with the metrics gathered for each MLP, the data could be analyzed to form a conclusion [O4]. When analyzing the data, the most important comparison was between the MSE losses and R^2 scores for each parameter in each MLP [Table 1&2]. As well, it was necessary to review the train loss and validation loss [Figures 5&6]to confirm that the model was learning properly throughout execution. After analyzing the data, it has become clear that between the tested MLPs, the best performing architecture contained 2 hidden layers each with 1000 neurons. This suggests that the best MLP for a program such as this one would be shallow (having few hidden layers) and wide (having many neurons).

8) Future Work:

Going forward, there are many aspects of this research which have the potential for exploration. One example of a way to progress this research would be comparing MLP architectures on larger or smaller data samples and across other stochastic volatility models. An interesting area to explore would be the comparison of other variations of MLP architectures

(e.g., using dropout). The most significant lesson learned in this research is the importance of utilizing the optimal MLP for the given program. In this research, it was found that the best combination for Heston model parameter prediction at approximately 100,000 samples was shallow and wide (2x1000).

9) Acknowledgements:

Prof. Ankush Agarwal, Dept. of Statistical and Actuarial Sciences, Western University

- Assisted in a supervisory capacity
- Provided relevant resources
- Identified logical errors in code
- Explained complex mathematical equations

10) References:

Ahmed, S., Alshater, M. M., El Ammari, A., & Hammami, H. (2022). Artificial intelligence and machine learning in finance: A bibliometric review. Research in International Business and Finance, 61, 101646. https://doi.org/10.1016/j.ribaf.2022.101646.

Sridi, A., & Bilokon, P. (2023). *Applying Deep Learning to Calibrate Stochastic Volatility Models*. https://doi.org/10.48550/arxiv.2309.07843

Heston, S. L. (1993). A Closed-Form Solution for Options with Stochastic Volatility with Applications to Bond and Currency Options. *The Review of Financial Studies*, *6*(2), 327–343. https://doi.org/10.1093/rfs/6.2.327

Bueno-Guerrero, A. (2019). Black–Scholes and Heston Models with Stochastic Interest Rates and Term Structure of Volatilities. *The Journal of Derivatives*, *27*(1), 32–48. https://doi.org/10.3905/jod.2019.1.078

Loh, W.-L. (1996). On Latin Hypercube Sampling. *The Annals of Statistics*, *24*(5), 2058–2080. https://doi.org/10.1214/aos/1069362310

Rosenbaum, M., & Zhang, J. (2021). *Deep calibration of the quadratic rough Heston model*. https://doi.org/10.48550/arxiv.2107.01611