DEGREE PROJECT IN MATHEMATICS,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2018

Exotic Derivatives and Deep Learning

AXEL BROSTRÖM

RICHARD KRISTIANSSON

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ENGINEERING SCIENCES

Exotic Derivatives and Deep Learning

AXEL BROSTRÖM
RICHARD KRISTIANSSON

Degree Projects in Financial Mathematics (30 ECTS credits)
Degree Programme in Industrial Engineering and Management
KTH Royal Institute of Technology year 2018
Supervisor at Algorithmica Research: Magnus Ekdahl
Supervisor at KTH: Boualem Djehiche
Examiner at KTH: Boualem Djehiche

TRITA-SCI-GRU 2018:162
MAT-E 2018:26

Royal Institute of Technology
School of Engineering Sciences
KTH SCI
SE-100 44 Stockholm, Sweden
URL: www.kth.se/sci

Abstract

This thesis investigates the use of Artificial Neural Networks (ANNs) for calculating present values, Value-at-Risk and Expected Shortfall of options, both European call options and more complex rainbow options. The performance of the ANN is evaluated by comparing it to a second-order Taylor polynomial using pre-calculated sensitivities to certain risk-factors. A multilayer perceptron approach is chosen based on previous literature and applied to both types of options. The data is generated from a financial risk-management software for both call options and rainbow options along with the related Taylor approximations. The study shows that while the ANN outperforms the Taylor approximation in calculating present values and risk measures for certain movements in the underlying risk-factors, the general conclusion is that an ANN trained and evaluated in accordance with the method in this study does not outperform a Taylor approximation even if it is theoretically possible for the ANN to do so. The important conclusion of the study is that the ANN seems to be able to learn to calculate present values that otherwise require Monte Carlo simulation. Thus, the study is a proof of concept that requires further development for implementation.

Exotiska derivat och djupinlärning

Sammanfattning

Denna masteruppsats undersöker användningen av Artificiella Neurala Nätverk (ANN) för att beräkna nuvärdet, Value-at-Risk och Expected Shortfall för optioner, både Europeiska köpoptioner samt mer komplexa rainbow-optioner. ANN:t jämförs med ett Taylorpolynom av andra ordningen som använder känsligheter mot ett flertal riskfaktorer. En typ av ANN som kallas multilayer perceptron väljs baserat på tidigare forskning inom området och appliceras på båda typerna av optioner. Datan som används har genererats från ett finansiellt riskhanteringssystem för såväl köpoptioner som rainbow-optioner tillsammans med tillhörande Taylorapproximation. Studien visar att även om ANN slår Taylorpolynomet för vissa specifika beräkningar av nuvärdet och riskvärden så är den generella slutsatsen att ett ANN som är tränad och utvärderad enligt metoden i denna studie inte presterar bättre än ett Taylorpolynom även om det är teoretiskt möjligt att ANN:t kan göra det. Den viktigaste slutsatsen från denna studie är att ANN:t verkar kunna lära sig prissätta komplexa finansiella derivat som annars kräver Monte Carlo-simulering. Således validerar denna studie ett koncept som kräver ytterligare utveckling före det implementeras.

Acknowledgements

First and foremost, we would like to thank Dr. Magnus Ekdahl at Algorithmica Research for valuable input regarding both theory and code as well as the final report. We would also like to thank the other employees at Algorithmica Research that have contributed with productive discussions. Finally, we would like to thank our supervisor Prof. Boualem Djehiche at the Department of Mathematics at the Royal Institute of Technology for his help with the thesis.

Table of Contents

List of Tables **iii**

List of Figures **v**

1 Introduction **1**

2 Literature and Theory **4**

2.1 ANNs in Financial Economics 4

2.2 ANNs for Option Pricing 4

2.3 ANNs - A Short Overview 6

2.4 Option Pricing 15

2.5 Risk Measures 19

3 Research Design **22**

3.1 Research Design 22

3.2 Method 23

3.3 Evaluation Metrics 25

3.4 Data 26

3.5 Taylor Approximation 31
3.6 ANN Structure 32

4 Results **34**

4.1 Comparison of Run Times 34
4.2 Call Option 35
4.3 Rainbow Option 39

5 Analysis and Conclusion **47**

5.1 Analysis 47
5.2 Discussion 52
5.3 Conclusion 54
5.4 Future Research 55

6 References **56**

List of Tables

3.1 The software used to prepare data, train the model and analyze results 23

3.2 The structure of the training and validation data for call options 27

3.3 The structure of the training and validation data for rainbow options 28

3.4 A point estimate of the convergence of the Monte Carlo simulation 31

4.1 The run times for the different methods of calculating present values 34

4.2 The MSE results of the validation of different models for call options 35

4.3 The results of the model comparison for stock price movements for call options 37

4.4 The results of the model comparison for implied volatility movements for call options 37

4.5 The results of the model comparison for interest rate movements for call options 38

4.6 The results of the model comparison for VaR calculations for call options 38

4.7 The results of the model comparison for ES calculations for call options 39

4.8 The MSE results of the validation of different models for rainbow options 40

4.9 The results of the model comparison for stock price movements in equity 1 for rainbow options 42

4.10 The results of the model comparison for stock price movements in equity 2 for rainbow options 42

4.11 The results of the model comparison for stock price movements in equity 3 for rainbow options 43

4.12 The results of the model comparison for implied volatility movements in equity 1 for rainbow options 43

4.13 The results of the model comparison for implied volatility movements in equity 2 for rainbow options 44

4.14 The results of the model comparison for implied volatility movements in equity 3 for rainbow options 44

4.15 The results of the model comparison for interest rate movements for rainbow options 45

4.16 The results of the model comparison for VaR calculations for rainbow options 45

4.17 The results of the model comparison for ES calculations for rainbow options 46

List of Figures

2.1 A MLP with an input layer with four nodes, one hidden layer with five nodes and an output layer consisting of one node. . . 6

2.2 A visual representation of the bias and variance of an estimator 13

Chapter 1

Introduction

Calculating present values of financial instruments is an important part of all financial mathematics and is done by traders, risk-managers, and quantitative analysts on a daily basis. There are multiple approaches to calculating the present value of a financial instrument. One approach is using a widely-accepted mathematical expression, a classic example being the Black-Scholes model for pricing European options which derives from a perfect hedge of the option given all the assumptions of a Black-Scholes world (see [1]).

The advantage of these mathematical expressions is that they are easily computed for many different combinations of inputs. Unfortunately there are many financial instruments for which no analytic valuation expression exists or is difficult to evaluate. Thus, there is a need for a numerical method. Monte Carlo simulation is a broad class of algorithms that use random number generators to simulate random variables. Many complex financial instruments are valued using Monte Carlo simulation of potential outcomes. This approach to calculating the present value of financial derivatives was first proposed by Phelim Boyle in 1977 [2].

Since 1977 Monte Carlo simulation has become the backbone of valuation for many financial instruments. To calculate present values with Monte Carlo simulation risk-neutral paths of financial assets are analyzed, distributions estimated and models built. This allows for the creation of an arbitrary number of scenarios. The financial instruments are then valued for all of these scenarios and an approximation of the present value of the instrument is found from the average. To obtain reliable results many simulations must be run as the Monte Carlo methods use the *Law of Large Numbers* (see [3]).

This means that any changes in input variables requires the computationally intensive simulations to be run again. Thus, in a modern financial world with ever-changing spot prices, interest rates, implied volatilities, and currencies it is difficult to keep up to date with the prices of these complex instruments which require Monte Carlo simulation.

To avoid having to run time consuming simulations every time there is a change in inputs - a different approach is needed. One that preferably correctly represents the present values but avoids the computational requirements of Monte Carlo simulation. One solution may be Artificial Neural Networks (ANNs) which have been successfully applied to a variety of cases in financial economics including option pricing (see [4]). While ANNs can be computationally intensive to train they are efficient when used after the training. Thus, the question is whether an ANN can be trained to price complex financial instruments in a way that could replace Monte Carlo simulation.

The purpose of this study is to determine whether, and if so when, ANNs can adequately approximate the present values, Value-at-Risk and Expected Shortfall of complex financial instruments that would otherwise require Monte Carlo simulation.

The approach that will be used to determine whether the ANN can adequately approximate the present value, Value-at-Risk and Expected Shortfall is a comparison of the ANN-calculated value with another method of handling input moves without Monte Carlo simulation as well as with Monte Carlo simulation itself as a benchmark. One method which avoids Monte Carlo simulation is using pre-calculated sensitivities of the present value to certain risk-factors and using a second-order Taylor polynomial to handle changes in the inputs.

From the purpose of the study the following research question is specified:

- Can an ANN outperform a second-order Taylor approximation when handling moves in the inputs of financial instruments that require Monte Carlo simulation to calculate their present value, Value-at-Risk and Expected Shortfall? If so, when?

As has been mentioned, many complex financial instruments lack analytic solutions and require Monte Carlo Simulations, rainbow options are one of these instruments. Rainbow options, also called multi-asset options, correlation options, or basket options, are options whose value depends on multiple

sources of uncertainty (see [5]). The general idea behind rainbow options is that the pay-off depends on the best or worst performing asset of the basket, creating best-of rainbow options and worst-of rainbow options. The rainbow options examined in this study are best-of call options with three underlying equity assets.

There are many variations of rainbow options but a best-of call option is a good example to understand how rainbow options work. The payoff Π of a best-of call option on n underlying assets is as follows, where S_i and K_i are the respective spot- and strike prices of each underlying asset at maturity.

$$\Pi = \max_{1 \leq i \leq n} [\Pi_i, 0],$$
$$\Pi_i = \frac{S_i}{K_i} - 1.$$

For simpler rainbow options with only two underlying assets closed-form solutions for calculating the present value exist. For slightly more complex rainbow options semi-analytic solutions and analytical approximations exist but in general Monte Carlo simulation is the primary method used for calculating the present value (see [5][6][7]).

Chapter 1 has introduced the background, problematization, and purpose of the study along with the research question. In Chapter 2 the literature and theory upon which the study is based is presented giving a short overview of ANNs and their use in option pricing. In Chapter 3 the research design which allows the research question to be answered is presented. This includes the method and data used in the study. In Chapter 4 the results of the study are presented. Finally in Chapter 5, the results are analyzed and conclusions are drawn. The results indicate that an ANN can learn to price options that require Monte Carlo simulation, however further development is needed to reach adequate levels of accuracy.

Chapter 2

Literature and Theory

2.1 ANNs in Financial Economics

ANNs have a wide variety of uses from image recognition and biology to finance. Li and Ma [4] present a survey of the application of ANNs in financial economics. This survey covers many areas of finance and many research articles but in general covers ANNs and exchange rates, ANNs and stock markets, and prediction of banking and financial crisis. The most relevant aspect is ANNs and stock markets and the sub-topic option pricing and ANNs where the authors present the results of previous research regarding the topic including multiple successful applications of ANNs for option pricing.

2.2 ANNs for Option Pricing

This section will present earlier research into the use of ANNs for option pricing. The previous studies shown here are different from this study in multiple ways. Firstly, the previous studies have mostly been focused on market data meaning that the ANN is trained to price options according to a "true" market pricing formula and compared to the results of for example the Black-Scholes formula. Secondly, most of the previous studies have been on European call options. Thirdly, these studies have not utilized deep neural networks with multiple hidden layers. In spite of this there are many parts of the research that are transferable to this study.

Hutchinson et al. [8] used multiple non-parametric models including an ANN with one hidden layer with four nodes and a sigmoid function evaluated by R^2 to investigate the performance of the network when pricing S&P 500 futures options between January 1987 and December 1991. The authors used daily data and used S/K and $T-t$ as inputs and C/K as an output.

Lajbcygier and Connor [9] used an three-layer ANN with 15 hidden nodes to price option futures on the Australian SPI between January 1992 and December 1994. The authors used daily data and F/K, $T-t$ and σ as inputs and $C-C_{MB}$ as outputs, the work was evaluated using R^2.

Gencay and Qi [10] used three-layer ANNs with Bayesian regulation, early stopping and bagging to price call options on the S&P 500 Index between January 1988 and December 1993. The authors used daily data and S/K and $T-t$ as inputs and C/K as outputs, the work was evaluated using MSPE, DM test and WS test.

Amilion [11] used three-layer ANNs with 10, 12 and 14 hidden nodes evaluated by RMSE to investigate the performance of the network when pricing call options on OMXS30 between June 1997 and March 1998 as well as June 1998 and March 1999. Amilion used daily data and I/K, $T-t$ and r as inputs and C_{bid}/K and C_{ask}/K as outputs.

Gradojevic et al. [12] used modular neural networks (3-9 modules) with one hidden layer evaluated by MSPE and DM test to investigate the performance of the network when pricing call options on the S&P 500 Index between January 1987 and December 1994. The authors used daily data and S/K and $T-t$ as inputs and C/K as output.

Liang et al. [13] used three-layer ANNs and support vector machines to price options based on Hong Kong option market data (122 firms) between January 2006 and December 2007. The authors used S/K, $T-t$, $e(BT)$, $e(FD)$ and $e(MC)$ as inputs and C as output. The e-terms are the results of a binomial tree, finite difference and Monte Carlo valuation. The performance of the network was evaluated using MAPE and MRPE.

Wang [14] used three-layer ANNs with a sigmoid activation function to price options on the Taiwan Stock Index between January 2005 and December 2006. Wang used S/K, $T-t$, r and $GARCH(\sigma)$ as inputs and σ as output. The network was evaluated using RMSE, MAE, MAPE and MSPE.

2.3 ANNs - A Short Overview

ANNs are built on learning algorithms and architectures that try to resemble features of the human brain. Neurons in different constellations are connected in a network that is trained to solve different problems. The network is trained and calibrated on labeled data, known as training data. Once the model is trained new unlabeled data is presented to the model and the model outputs an answer in accordance with what it has learnt during training.

ANNs do not rely on any underlying models, there are no underlying probability distributions to be estimated or likelihoods to be maximized. The advantage of using this approach is that the algorithm determines relationships in the data itself without any assumptions. ANNs are however not a single approach. There are several different types of ANN models that all have their respective strengths in different applications, such as Convolutional Neural Networks, Recurrent Neural Networks which have their strengths in among other things image recognition and speech recognition respectively (see [15][16]). For regression a useful approach is the multilayer perceptron (MLP) since it is a theoretical universal function approximator as shown by Cybenko [17].

The MLP organizes neurons in different layers. Inputs are inserted in an input layer, then the problem solving takes place in an arbitrary number of hidden layers, and lastly the output is exhibited in the output layer. An example architecture is displayed in figure 2.1.

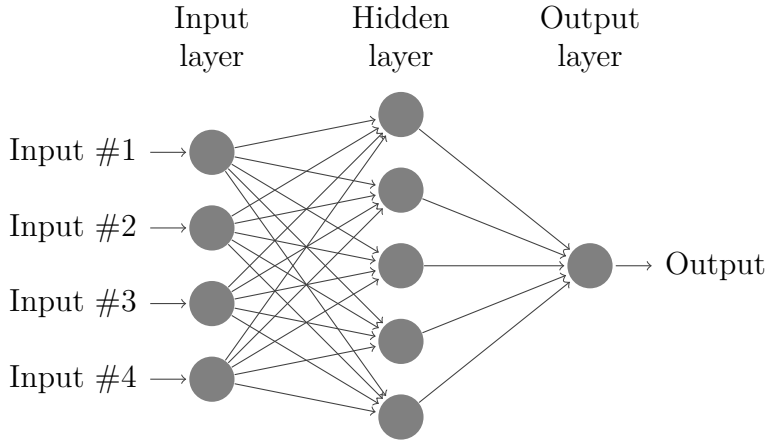

Figure 2.1: A MLP with an input layer with four nodes, one hidden layer with five nodes and an output layer consisting of one node.

An ANN with multiple hidden layers is often called a deep neural network. These deep neural networks are often of the MLP type. The extra hidden layers allows more complex relationships to be modeled with fewer neurons than a network with fewer layers that has similar performance. (see [18][19])

In general an ANN works in the following way (see [20]). Each neuron computes a weighted sum of all inputs leading to it, adds a bias term (2.1) and computes a transformation of that sum (2.2). Typically the transformation function is a sigmoid, a smooth monotonically increasing function, such as the logistic function or the hyperbolic tangent. However, it can also be a linear function such as a rectified linear unit ($max[0, x]$). The transformed sum is passed on as an input to the nodes in the next layer until the output is attained.

$$z_j^l = \sum_k w_{jk}^l a_k^{l-1} + b_j^l, \tag{2.1}$$

$$a_j^l = \sigma(z_j^l). \tag{2.2}$$

In the equation above z_j^l is the weighted input in node j in layer l, w_{jk}^l is the weight applied to the activation a_k^{l-1} from node k in the preceding layer $l-1$, b_j^l is the bias term and σ is the transformation function. In vector form this can be represented as:

$$z^l = w^l a^{l-1} + b^l, \tag{2.3}$$

$$a^l = \sigma(z^l). \tag{2.4}$$

The activations in the input layers are taken directly as the inputs without any transformation. This means that the input data must be represented in a reasonable way. The output layer can be calculated in many different ways depending on what the ANN is trying to achieve. For example in a regression a weighted sum can be calculated, while in a classification a softmax function can be applied to give probabilities to certain classifications (see [21][22]).

Lastly, when the calculations of the ANN are complete, the output is compared against the labeled values of the training data and a cost function is computed. An algorithm called backpropagation is used to understand how the weights and biases affect the cost function and another algorithm, gradient descent is used to adjust the weights and biases to minimize the cost function. This procedure is repeated until the error is minimized at which point the model is considered to be trained and ready to investigate new data.

The cost function C is a measure of difference between the output of the ANN and the correct output, and is used to train the model. To be able to use backpropagation multiple assumptions are necessary. The first assumption is that the cost function for all inputs can be written as an average over the cost function for single inputs. This is necessary since backpropagation calculates the cost function for single inputs at a time. The second assumption is that the cost function has the partial derivatives $\partial C/\partial w$ and $\partial C/\partial b$ since these are necessary part of the backpropagation calculations. The final assumption on the cost function is that it can be written as a function of the outputs from the neural network. This ensures that the cost function only responds to changes in what the network has learned. (see [23])

2.3.1 Backpropagation Algorithm

As mentioned in the previous section, ANNs can be trained using a combination of backpropagation and gradient descent to minimize the cost function. Backpropagation, made famous by Rumelhart et al. [23], calculates the gradient of the cost function with respect to the weights and biases of the ANN and gradient descent minimizes the cost function using the calculated gradient.

The backpropagation algorithm calculates how different weights and biases affect the cost function, which allows gradient descent to be applied. The following is a derivation of backpropagation for a MLP (see [20]).

The first step is to introduce the error term δ_j^l, the error in the j^{th} neuron in the l^{th} layer. Secondly, the partial derivatives for the cost function C with respect to weights w and biases b, $\partial C/\partial w_{jk}^l$ and $\partial C/\partial b_j^l$ are computed. The error term is calculated as follows:

$$\delta_j^l = \frac{\partial C}{\partial z_j^l}. \tag{2.5}$$

The procedure going forward is to calculate δ_j^l for every node and relate it to $\partial C/\partial w_{jk}^l$ and $\partial C/\partial b_j^l$. Backpropagation does this by focusing on four fundamental equations that make it possible to compute both the error δ^l and the gradient of the cost function.

The first equation (2.6), is used for calculating the error δ in the output layer L:

$$\delta_j^L = \frac{\partial C}{\partial a_j^L}\sigma'(z_j^L), \tag{2.6}$$

where the term $\partial C/\partial a_j^L$ is the partial derivative of the cost function with respect to the j^{th} output activation a_j^L. Intuitively, if a particular output neuron a_j^L has low influence on the cost function C, then the error will be small. The derivative of the activation function $\sigma'(z_j^L)$ shows the activation function's σ movements related to the weighted input z_j^L. In matrix form, the equation takes the following expression:

$$\delta^L = \nabla_a C \odot \sigma'(z^L), \tag{2.7}$$

where $\nabla_a C$ is a vector with the partial derivatives $\partial C/\partial a_j^L$, $\sigma'(z^L)$ is the vector of $\sigma'(z_j^L)$ and $\odot$ is the Hadamard product.

The second equation (2.8) is used for calculating the error δ in layer l in terms of the error in the next layer δ^{l+1}:

$$\delta^l = ((w^{l+1})^T\delta^{l+1}) \odot \sigma'(z^l), \tag{2.8}$$

where $(w^{l+1})^T$ is the weight matrix for the $(l+1)^{th}$ layer transposed. Intuitively, if the error δ^{l+1} and the weight matrix w^{l+1} is known for the $(l+1)^{th}$ layer the error can be transferred back to the l^{th} layer. The Hadamard product moves the error backwards in the network through the derivative of the activation function $\sigma'(z^l)$ in layer l. By repeating this process backwards through the network the error δ^l is calculated for all layers l.

The third equation (2.9), is used to calculate the bias term's effect on the cost function and is expressed as:

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l. \tag{2.9}$$

Which shows that the error from the bias term is equal to the rate of change $\partial C/\partial b_j^l$ which is intuitive since the bias directly affects z_j^l and thereby the error.

The fourth equation (2.10), is used to calculate the rate of change of the cost function with respect to any weight in the network, and is expressed as:

$$\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1}\delta_j^l. \tag{2.10}$$

2.3.2 Gradient Descent

Gradient descent is an optimization algorithm used to find the minimum of a differentiable function. In neural networks it is used to find the weights and biases that minimize the cost function. The iterative algorithm seeks to find local minimum by taking one step proportional to the negative gradient of the function at the current point (see [20]). In neural networks, gradient descent uses the gradient calculated in the backpropagation algorithm.

The change in the cost function can be approximated as:

$$\Delta C \approx \nabla C \cdot \Delta v, \tag{2.11}$$

where ∇C is the gradient of the cost function and Δv is the change. Δv is chosen in the following way:

$$\Delta v = -\eta \nabla C, \tag{2.12}$$

where η is a small positive parameter known as the learning rate that represents the step size.

Combining (2.11) and (2.12) gives the following expression:

$$\Delta C \approx \nabla C \cdot -\eta \nabla C = -\eta \|\nabla C\|^2 . \tag{2.13}$$

This equation shows that since $\|\nabla C\|^2 \geq 0$, the change in the cost function in guaranteed to be $\Delta C \leq 0$ if Δv is chosen as in (2.12). The algorithm keeps iterating until it finds a minimum. The learning rate η has to be small enough so that the approximation (2.11) holds, but if too small, the algorithm will be slow and inefficient. It is important to note that if the cost function is non-convex a global minimum can never be guaranteed by gradient descent.

Applied to the optimization problem of choosing weights w_k and biases b_l, the gradient descent algorithm leads to the following updating rules:

$$w_k' = w_k - \eta \frac{\partial C}{\partial w_k}, \tag{2.14}$$

$$b_l' = b_l - \eta \frac{\partial C}{\partial b_l}, \tag{2.15}$$

where w_k and b_l are the current weights and biases and w_k' and b_l' are the updated values. This update rule is repeated until the algorithm has found the weights and biases that minimize the cost function. It is important to

notice that the cost function is an average over cost for all individual training examples as per earlier assumptions:

$$C = \frac{1}{n} \sum_x C_x. \tag{2.16}$$

In practice this means that the gradient ∇C_x has to be computed for each training input x, and then averaged to get ∇C.

$$\nabla C = \frac{1}{n} \sum_x \nabla C_x. \tag{2.17}$$

This means that learning can take an extensive amount of time, since a large number of training inputs means that a large number of gradients need to be calculated.

2.3.3 Stochastic Gradient Descent

Stochastic gradient descent is an approach used to get around the time consuming problem of calculating gradients for all training inputs in order to speed up learning. The idea is to calculate the gradient only for a small sample of the training inputs and average them to get an estimate of the true gradient of the cost function ∇C. (see [24])

$$\frac{1}{m} \sum_{j=1}^{m} \nabla C_{X_j} \approx \frac{1}{n} \sum_x \nabla C_x = \nabla C. \tag{2.18}$$

The left hand side of (2.18) represents the average over the small sample m and the right hand side is the true average over the full training set n.

Applied to the optimization problem, the stochastic gradient descent algorithm works in the following way:

$$w_k' = w_k - \frac{\eta}{m} \sum_j \frac{\partial C_{X_j}}{\partial w_k}, \tag{2.19}$$

$$b_l' = b_l - \frac{\eta}{m} \sum_j \frac{\partial C_{X_j}}{\partial b_l}, \tag{2.20}$$

where m is an randomly chosen sample of the training input samples, a batch, and X_j is an input sample within the batch. This approach is repeated by

picking another randomly chosen batch from the remaining training data until all training data has been used in the training of the network. When all training inputs have been used, one epoch of training has been completed. The number of epochs used differs and is adjusted to make sure a minimum is reached while trying to avoid overfitting the model to the training data.

2.3.4 Adam Optimizer

A variation of gradient descent, the Adaptive Moment Estimation (Adam) optimization method was presented by Kingma and Ba [25]. The authors present the method with the following words:

> *The method is straightforward to implement, is computationally efficient, has little memory requirements, is invariant to diagonal rescaling of the gradients, and is well suited for problems that are large in terms of data and/or parameters. The method is also appropriate for non-stationary objectives and problems with very noisy and/or sparse gradients. The hyper-parameters have intuitive interpretations and typically require little tuning.*

The Adam optimizer works by adapting the learning rates for each parameter. The method stores an exponentially decaying average of previous gradients squared v_t and an exponentially decaying average of previous gradients m_t. The updates to the variables are calculated in the following way:

$$m'_k = \beta_1 m_k + (1 - \beta_1)\nabla C, \tag{2.21}$$

$$v'_k = \beta_2 v_k + (1 - \beta_2)(\nabla C)^2, \tag{2.22}$$

$$\hat{m}_k = \frac{m'_k}{1 - \beta_1}, \tag{2.23}$$

$$\hat{v}_k = \frac{v'_k}{1 - \beta_2}. \tag{2.24}$$

Thus the final update rule for weights and biases becomes:

$$w'_k = w_k - \eta \frac{\hat{m}_k}{\sqrt{\hat{v}_k} + \epsilon}, \tag{2.25}$$

$$b'_l = b_l - \eta \frac{\hat{m}_k}{\sqrt{\hat{v}_k} + \epsilon}, \tag{2.26}$$

where β_1 and β_2 are the exponential decay factors while ϵ is a small number to avoid zero division, and $(\nabla C)^2$ is the element-wise square of ∇C. Kingma and Ba recommend $\beta_1 = 0.9$ $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$ as default values for the parameters of the optimization method.

2.3.5 Bias-Variance Trade-Off

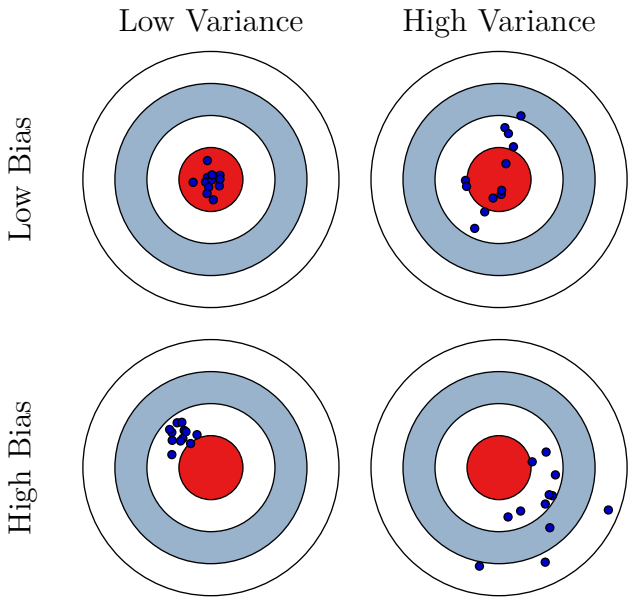

Figure 2.2: A visual representation of the bias and variance of an estimator

The bias-variance trade-off is the comparison of accuracy versus quality of an estimator by the use of bias and variance as the measurable quantities. In general the bias-variance trade-off leads to the following conclusions. If a model is too complex it is sensitive to small variations in the input data while a model that is too simple will be biased and not fit the data properly. In mathematical terms in can be explained in the following way. (see [26])

Consider a training set $x_1, .., x_n$ and real values y_i with the following relationship.

$$y_i = f(x_i) + \mathcal{E}$$

where $\mathcal{E}$ is noise with zero mean and variance σ^2. If the attempted model is represented by $\hat{f}(x)$ then the error can be decomposed in the following

manner.

$$E[(y - \hat{f}(x))^2] = \text{Bias}[\hat{f}(x)]^2 + \text{Var}[\hat{f}(x)] + \sigma^2,$$

where

$$\text{Bias}[\hat{f}(x)] = E[\hat{f}(x)] - f(x),$$
$$\text{Var}[\hat{f}(x)] = E[(\hat{f}(x) - E[\hat{f}(x)])^2].$$

Thus the total error is decomposed into three parts which form a lower bound on the expected error of the estimator on unseen samples.

- The square of the bias: The error due to overly simple models.
- The variance of the estimator: How much the estimator moves around the mean indicating more complexity.
- The irreducible error: The σ^2 error which can not be avoided due to a noisy relationship between x and y.

2.3.6 Training and Validation Data

To handle the problem of the bias-variance trade-off a common approach is to divide the input data set into two data sets, one of which is used to train the model and the other which is used for validating the model choice. This allows a comparison to be made between different models as every model is evaluated on the same validation data set. Thus a comparison can be made between simpler models and more complex models to ensure that a model of sufficient complexity for the problem is chosen without overfitting the model to the training data. (see [26])

2.3.7 Worst Case

Barron [27] investigated the approximation properties of ANNs showing that a three layer MLP with sigmoidal activation functions can achieve an integrated squared error of $O(1/n)$ where n is the number of nodes. Goodfellow et al. [28] expanded upon this by determining that in the worst case an exponential number of hidden units may be required. While a MLP with a single

hidden layer can represent any function it may become infeasibly large or fail to learn.

Montufar et al. [29] investigated similar properties for deep neural networks using ReLU activation functions. The authors showed a lower bound for the maxiamal number of linear regions that that an ANN with ReLU activation functions can approximate given input nodes of $O(1)$, the same number of nodes in all hidden layers, more nodes in the hidden layers than in the input layer and with L hidden layers. Goodfellow et al. [28] reformulated the main theorem of Montufar et al. as follows. The number of linear regions a deep ReLU network with d inputs, l hidden layers and n units per hidden layer can represent is:

$$O\left(\binom{n}{d}^{d(l-1)} n^d\right).$$

2.4 Option Pricing

2.4.1 Arbitrage-Free Pricing

Arbitrage-free pricing or valuation is a wide spread theory used in pricing models. Prices are determined in such a manner as to preclude any arbitrage opportunities.

Black-Scholes Model

The Black-Scholes model is an arbitrage-free pricing model for European options. The model calculates the price as the discounted risk-neutral expected value of the payoff of the option. This is also known as calculating the price under the risk-neutral measure Q, which is not the real world observed probability measure but a probability measure for arbitrage-free prices. The risk-neutral measure implies that there is a unique arbitrage-free price for each asset in the market (see [30]). The arbitrage-free price is realized by using dynamic hedging.

For the the Black-Scholes model to hold some main assumptions and simplifications must be applied to the underlying asset and markets (see [31]).

- Interest rate: Assumed to be known, risk-free and constant.
- Log-normal distribution of returns: This means that the stock price at maturity S_T and the stock price at time 0, S_0 has the following distribution. $\frac{S_T}{S_0} \in e^Z$ where $Z \in N((r - \frac{\sigma^2}{2})T, \sigma\sqrt{T})$.
- Volatility: Assumed to be constant over time and different strike prices.
- No dividends: A simplification, which is easily worked around by subtracting the discounted value of the dividend from the stock price or by using a dividend yield (see [31]).
- Arbitrage-free: There are no risk-free arbitrage opportunities.
- Cash: It is possible to borrow and lend any amount, even fractional at risk-free rate.
- Liquidity: It is possible to buy and sell any amount, even fractional, of the underlying without any bid-ask-spread.
- No transaction costs or taxes: A necessary assumption for the constant rebalancing in dynamic hedging.

If these assumptions hold the payoff X can be priced as follows:

$$\Pi_t(X) = e^{-r(T-t)} E^Q[X], \tag{2.27}$$

where Π_t is the price at time t of the payoff X that occurs at time T (see [32]). To find the expected value under the risk neutral probability measure, the following equation is used:

$$E^Q[X] = \int_{-\infty}^{\infty} x\psi(z)dz, \tag{2.28}$$

where $\psi(z)$ is the probability density function of $Z \in N((r - \frac{\sigma^2}{2})(T - t), \sigma\sqrt{T-t})$.

These equations give the price of a option with payoff X, which is also the price of the dynamic hedge. This is because the dynamic hedge will recreate the same cash flows as the derivative and since there is no arbitrage the price of identical cash flows must be equal. Thus by pricing the dynamic hedge the price of the option is found as well.

Problems with Arbitrage-Free Pricing

Though the Black-Scholes model is widely used in the world of finance it is not perfect. There have been many articles that are highly critical of the formula, not least an article by Haug and Taleb [33]. Some of the common criticisms of the Black-Scholes model include:

- The normality of asset returns: The normality assumption of asset returns in the Black-Scholes model has been criticized for underestimating extreme movements of assets. As Hull [31] states returns are leptokurtic meaning that there are far too many outliers for a normal distribution to be correct.
- Constant volatility: As noted by Yalincak [34] asset volatility is often clustered over time. In practice volatility is also non-constant for different strike prices and times to maturity, leading to so called volatility smiles (see [35]).
- Instant and cost-less trading: In the real world there are fees for trading options and stocks as well as barriers to trading. The model also assumes perfect liquidity in the market which has been proven false on multiple occasions not least during the global financial crisis and other times of financial distress.

Why Monte Carlo?

There are multiple problems with the Black-Scholes framework, some of which can be handled in different ways, but a problem which it can not handle is increasing complexity. Monte Carlo simulation is one option for handling increases in complexity when calculating the present value of options.

2.4.2 Monte Carlo Pricing

Monte Carlo pricing is a commonly used technique for calculating option prices with complicated features that are difficult if not impossible to price using analytic expressions. Examples of options that are usually priced with

Monte Carlo simulations are rainbow options as well as path dependent options such as look-back options and options with Asian tails, as no analytic solution exists for these derivatives.

The Monte Carlo method relies on risk neutral valuation where the price of the option is the discounted expected value. The first step is to generate a large sample of random possible risk-neutral price paths for the underlying asset(s) by simulation. Secondly, the option payoff of each price path is calculated. Finally, the value of the option is calculated as the discounted average of all payoffs. (see [2])

The benefits of the Monte Carlo approach is that it allows for compounding in the source of uncertainty. This opens up the possibility to price options with multiple sources of uncertainty such as rainbow options with multiple underlying assets. For rainbow options correlation plays an important role and is therefore incorporated in the simulations. (see [36])

Furthermore, the Monte Carlo pricing approach is not limited to any type of probability distribution, which makes it a flexible approach for pricing. It is also possible to specify the stochastic process of the underlying asset(s) so that it exhibits jumps or mean reversion. (see [36])

The main drawback with the Monte Carlo method is that it is computationally intensive. If an analytic technique for valuing the option exists the Monte Carlo method will usually be too slow to be competitive (see [3]). This is mainly due to the fact that the convergence of a Monte Carlo simulation is inversely proportional to the square root of the number of samples.

Monte Carlo simulation is carried out by generating random numbers X_i from the probability density function $f_X(x)$ and computing the objective function for each case and estimating the average $\hat{\mu}$ (see [3]). If $Y = h(X)$ then by the *Law of the Unconscious Statistician*:

$$E[Y] = E[h(X)] = \int_{-\infty}^{\infty} h(x) f_X(x) \mathrm{d}x, \tag{2.29}$$

$$\hat{\mu} = \frac{1}{N} \sum_{i=1}^{N} h(x_i), \tag{2.30}$$

where x_i is an independent sample of the random variable X and $\hat{\mu}$ converges

almost surely to $E[Y]$ by the *Strong Law of Large Numbers*.

$$\hat{\mu} = \lim_{N\to\infty} \frac{1}{N}\sum_{i=1}^{N} h(x_i) \xrightarrow{a.s.} \int_{-\infty}^{\infty} h(x) f_X(x)\mathrm{d}x = E[Y]. \tag{2.31}$$

The standard error $\sigma_{\hat{\mu}}$ (2.34) of the Monte Carlo simulation is proportional to $1/\sqrt{N}$ where N is the number of samples.

$$\mathrm{Var}(\frac{1}{N}\sum_{i=1}^{N} h(x_i)) = \frac{\sigma^2}{N}, \text{ where } \sigma^2 = \mathrm{Var}(x_i), \tag{2.32}$$

$$\hat{\sigma}^2 = \frac{1}{N-1}\sum_{i=1}^{N}(h(x_i) - \hat{\mu})^2, \tag{2.33}$$

$$\sigma_{\hat{\mu}} \propto \frac{\hat{\sigma}}{\sqrt{N}}. \tag{2.34}$$

2.5 Risk Measures

2.5.1 Value-at-Risk

The Value-at-Risk (VaR) is a measure of risk in a portfolio. The VaR estimates the potential losses in a portfolio for a certain amount of time. The VaR of a portfolio X is given as:

$$\begin{aligned} \mathrm{VaR}_p(X) &= min\{m : P(m \cdot R_0 + X < 0) \leq p\} \\ &= min\{m : P(-X/R_0 \leq m) \geq 1 - p\}, \end{aligned} \tag{2.35}$$

where R_0 is the return of the risk-free rate and $p \in (0, 1)$ is the confidence level. (see [37])

If X is given as $V_1 - V_0 R_0$, the net gain of the portfolio, then the discounted loss can be represented as:

$$L = -\frac{X}{R_0} = V_0 - \frac{V_1}{R_0}, \tag{2.36}$$

where V_0 and V_1 are the values of the portfolio at time 0 and 1. Using this notation the VaR can be expressed as:

$$\mathrm{VaR}_p(X) = min\{m : P(L \leq m) \geq 1 - p\}. \tag{2.37}$$

In statistical terms this is the $(1-p)$-quantile of L and thus is follows that:

$$\mathrm{VaR}_p(X) = F_L^{-1}(1-p). \tag{2.38}$$

2.5.2 Expected Shortfall

Expected Shortfall is an extension of the VaR concept which takes into account the shape of the tail of the loss distribution (see [37]). The Expected Shortfall of a portfolio X can be calculated as:

$$\mathrm{ES}_p(X) = \frac{1}{p}\int_0^p \mathrm{VaR}_u(X)du. \tag{2.39}$$

2.5.3 Empirical Distribution

The empirical distribution of a sample $X_1, \ldots, X_n$ is given as:

$$F_{n,X}(x) = \frac{1}{n}\sum_{k=1}^{n} I\{X_k \leq x\}, \tag{2.40}$$

where n is the number of samples and I is an indicator function. This representation can be justified by the *Law of Large Numbers*. (see [37])

If $Z_1, \ldots, Z_n$ are independent copies of Z and if $E[Z]$ is finite, then the *Law of Large Numbers* states:

$$\lim_{n\to\infty} \frac{1}{n}\sum_{k=1}^{n} Z_k \xrightarrow{a.s.} E[Z]. \tag{2.41}$$

Setting $Z_k = I\{X_k \leq x\}$ implies that $E[Z_k] = P(X_k \leq x) = F(x)$. Thus, the *Law of Large Numbers* implies that $\lim_{n\to\infty} F_{n,X}(x) = F(x)$ almost surely.

2.5.4 Empirical Value-at-Risk

With X and L as in section 2.5.1, using independent samples $L_1, \ldots, L_n$ of L the empirical $\mathrm{VaR}_p(X)$ can be calculated as:

$$\widehat{\mathrm{VaR}}_p(X) = L_{[np]+1,n}, \tag{2.42}$$

where the sample is ordered, $L_{1,n} \geq, \ldots, \geq L_{n,n}$ (see [37]).

2.5.5 Empirical Expected Shortfall

Using the empirical VaR the empirical Expected Shortfall is estimated by simply replacing $\mathrm{VaR}_p(X)$ by its empirical estimator $\widehat{\mathrm{VaR}}_p(X)$ (see [37]).

$$\begin{aligned}\widehat{\mathrm{ES}}_p(X) &= \frac{1}{p}\int_0^p \widehat{\mathrm{VaR}}_u(X)du = \frac{1}{p}\int_0^p L_{[nu]+1,n}du \\ &= \frac{1}{p}\left(\sum_{k=1}^{[np]}\frac{L_{k,n}}{n} + \left(p - \frac{[np]}{n}\right)L_{[np]+1,n}\right).\end{aligned} \tag{2.43}$$

Chapter 3

Research Design

3.1 Research Design

The design of a study is what allows the research question to be answered. Using ANNs to predict option prices was a well-motivated choice as their usefulness in the area has been proven on multiple occasions as seen in section 2.2. While those studies mostly try to estimate prices from market data there is an obvious parallel to pricing options using risk-factor simulation leading to the hypothesis that ANNs may be able to price options which require Monte Carlo simulation as well. To investigate the research question and hypothesis the following research design was used.

1. Evaluate performance for European call options to investigate performance for simpler options that do not require Monte Carlo simulation.
2. Evaluate performance for rainbow options to investigate performance for more complex options that do require Monte Carlo simulation.

For both the call option and the rainbow option the following plan was used.

1. Collect data containing option present values and inputs.
2. Clean and format the data as necessary.
3. Train and validate ANNs on the data.

4. Collect new data using current approximations.

5. Compare the results of the ANN with current approximations for calculations of present value, Value-at-Risk and Expected Shortfall.

3.2 Method

To execute the study in accordance with the research design the workflow shown in Table 3.1 was used.

Risk-Management Software	Excel	Python/Tensorflow
1. Data Gathering		
	2. Data Formatting	
	3. Data Transfer	
		4. Data Split
		5. Model Construction
		6. Model Training
		7. Model Validation
8. Comparison Data		
	9. Data Formatting	
	10. Data Transfer	
		11. Model Use
	12. Evaluation	

Table 3.1: The software used to prepare data, train the model and analyze results

3.2.1 Data Gathering

The present values were collected from a risk-management software. The risk-management software does not use market data explicitly but rather uses market data as an input to create risk-factors which are then used to evaluate pre-determined pricing formulas or simulate outcomes and calculate present values. This means that the options and present values used in the study were not live market prices therefore no cleaning of the data is necessary to handle for example bid-ask-spreads.

3.2.2 Data Cleaning, Formatting, Transfer, and Split

As Python could not directly interact with the risk-management software Excel was used as an intermediary step. The data was written into an Excel file where the data was formatted in such a way that was easy for Python to read. The Python script read from the Excel file to load the data. In Python the rows of the data were randomly arranged and split into a training set and a validation set.

3.2.3 Model Construction, Training, and Validation

The construction, training and validation of the model was completed using the open-source library for machine learning in Python, Tensorflow with the various native commands that are offered. Tensorflow's native commands allows for easy execution of backpropagation, gradient descent, stochastic gradient descent and the use of the Adam optimizer mentioned in sections 2.3.1-2.3.4.

The validation was done by evaluating the different ANNs on the same validation data set and comparing the results. Tensorflow allows for easy changes of network architecture by adding another matrix multiplication and transform to add a layer or simply changing a variable to change the number of nodes in a layer.

3.2.4 Comparison Data, Formatting, and Transfer

Comparison data was generated by creating an option position in the risk-management software, calculating the position's sensitivities to different risk-factors using the centered finite difference method and using these sensitivities to calculate a present value for different types of moves in the risk-factors affecting the position. The new values were recorded, formatted and transferred to Python via Excel.

3.2.5 Model Use and Evaluation

The comparison data was loaded into Python and run through the trained ANN of choice. This generated predicted option values for all the different cases. These predicted option values were saved along with the other comparison data.

To evaluate the performance of the ANN against the Taylor approximation the data was moved to Excel. In Excel the MSE and the MAPE of the predicted present value and the Taylor approximated present value were compared for the different types of moves in the risk-factors to see when and if the ANN outperformed the Taylor approximation. Once present values had been compared the ANN's performance on Value-at-Risk and Expected Shortfall was evaluated and compared by calculating the MSE, the MAPE and the MPE.

3.3 Evaluation Metrics

Three different metrics were used to evaluate the performance of the ANN and Taylor approximation. The mean squared error (MSE) and the mean absolute percentage error (MAPE) were used to investigate the performance of the two valuation methods for moves in equity spot prices, interest rates and implied volatility. In addition the mean percentage error (MPE) was used when evaluating performance on Value-at-Risk and Expected Shortfall calculations to investigate whether the method over- or underestimates the risk measures on average.

In the following expressions P represents the predicted values, A is the actual value and n is the number of investigated cases.

$$\text{MSE}(P) = \frac{1}{n}\sum_{i=1}^{n}(A_i - P_i)^2, \tag{3.1}$$

$$\text{MAPE}(P) = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{A_i - P_i}{A_i}\right|, \tag{3.2}$$

$$\text{MPE}(P) = \frac{1}{n}\sum_{i=1}^{n}\frac{A_i - P_i}{A_i}. \tag{3.3}$$

3.4 Data

This section briefly describes the data used in the study. It specifies which data was used to train and validate the ANN as well as the data used to compare the results of the ANN with the Taylor approximations.

3.4.1 Description

Call Option

In the case of the European call option the following inputs were used: Time to maturity, risk-free rate, spot/strike and implied volatility for the underlying equity as seen in the most of the previous research. Here:

- Time to maturity: The time between the date on which the present value is being calculated and the maturity of the option in years.
- Risk-free rate: The risk-free rate corresponding to the time to maturity.
- Spot/strike: The spot/strike was calculated to remove the effect of different strikes such that the ANN interprets the input as how much the equity must move in percent terms and not in absolute terms which would require training the ANN for all different combinations of spot and strike prices.
- Implied volatility: The individual volatility of the underlying equity. As in the Black-Scholes model this is not a historical volatility but rather a market-implied volatility.

Rainbow Option

In the case of a rainbow option with three underlying equities quoted in a common currency the following inputs were used: time to maturity, risk-free rate, correlations between the equities as well as strike-level and implied volatility for each underlying equity which closely resembles the approach used for the European call options. To further clarify:

- Time to maturity: As for the call option.
- Risk-free rate: As for the call option.
- Correlations: The correlations between the returns of the equities.
- Strike-level: For each equity the strike-level is calculated in the following manner, K/S where S is the spot price of the equity and K is the strike price for that equity. Thus, the strike-level is a number representing the strike price as a percentage of the spot price.
- Implied volatility: The individual volatility of each underlying equity. As in the Black-Scholes model this is not a historical volatility but rather a market-implied volatility.

3.4.2 Input to the ANN

Call Option

The input data used to train and validate the ANN took the form shown in Table 3.2. Note that Table 3.2 is an extraction from the original data set.

S/K	T	r	σ
1.47	0.15	0.02	0.34
0.91	0.98	0.01	0.10
0.55	0.74	0.05	0.14
⋮	⋮	⋮	⋮

Table 3.2: The structure of the training and validation data for call options

In the Table 3.2 S/K is the spot/strike, T is the time to maturity of the option, r is the risk-free rate and σ is the implied volatility for the underlying equity.

Rainbow Option

The input data used to train and validate the ANN took the form shown in Table 3.3. Note that Table 3.3 is an extraction from the original data set.

T	r	s_1	σ_1	s_2	σ_2	s_3	σ_3	ρ_{12}	ρ_{13}	ρ_{23}
0.81	0.02	0.87	0.26	1.62	0.40	0.90	0.06	0.26	-0.89	0.53
0.32	0.05	0.68	0.43	0.72	0.72	0.69	0.57	-0.02	0.56	0.75
0.98	0.02	1.10	0.16	1.77	0.61	0.96	0.50	0.89	-0.68	-0.68
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Table 3.3: The structure of the training and validation data for rainbow options

In the Table 3.3 T is the time to maturity of the option, r is the risk-free rate, s_i is the strike-level for underlying i, σ_i is the implied volatility for underlying i and ρ_{ij} is the correlation between underlying i and j.

Training and Validation Data Split

For both the call option and the rainbow option the generated data set consisted of 1 million samples with different combinations of all inputs in random order. This data set was split into two different data sets. 80% of the data was moved into the training data set while the remaining 20% was used as a validation data set by which different models could be compared.

Call Option: The 1 million call options generated for training and validation used the following parameters.

- Time to maturity: Randomly selected from a uniform distribution between 1 day to 1 year.
- Risk-free rate: Randomly selected from a uniform distribution between 0% and 5%.
- Spot/strike: Randomly selected from a uniform distribution between 0.5 and 1.5.
- Implied volatility: Randomly selected from a uniform distribution between 5% and 80%.

Rainbow Option: The 1 million rainbow options generated for training and validation used the following parameters.

- Time to maturity: As for the call option.
- Risk-free rate: As for the call option.
- Correlations: Randomly selected from a uniform distribution between -1 and 1 independently for each equity pair.
- Strike-level: Randomly selected from a uniform distribution between 2/3 and 2 independently for each equity which represents a spot price at 50% of the strike price up to a spot price at 150% of the strike price.
- Implied volatility: Randomly from a uniform distribution selected between 5% and 80% independently for each equity.

3.4.3 Data for Present Values

In order to evaluate the performance of the ANN with the current approximation methods new data needed to be generated as the comparison would be unfair if the ANN was evaluated with the same data upon which it had been trained. This data was generated and handled in the same way as the training and validation data. The difference being that this time the ANN was not going to be trained or validated but rather tested, thus the ANN only received the inputs and generated predictions.

Call Option

To compare performance, 10000 call options were generated and priced using the same parameters as the training and validation data while allowing for the 20% moves in either direction without moving outside the input space for which the ANN was trained.

After the first- and second-order sensitivities were calculated each of the inputs were moved ceteris paribus between -20% and +20% and a new true present value was calculated along with a Taylor approximation of the present value as well as an ANN prediction of the present value. 10000 options with 18 seperate moves for these options gives 180000 individual triplets of true present value, Taylor approximated present value and ANN predicted present value to compare.

Rainbow Option

To compare performance, 10000 rainbow options were generated and priced using the same parameters as the training and validation data while allowing for the 20% moves in either direction without moving outside the input space for which the ANN was trained.

After the first- and second-order sensitivities were calculated each of the inputs were moved ceteris paribus between -20% and +20% and a new true present value was calculated with Monte Carlo simulations along with a Taylor approximation of the present value as well as an ANN prediction of the present value. 10000 options with 42 seperate moves for these options gives 420000 individual triplets of true present value, Taylor approximated present value and ANN predicted present value to compare.

3.4.4 Data for Risk Measurement

In order to be able to evaluate the ANNs' performance on risk measures new data needed to be generated. Once again 10000 options were generated, for both the call and rainbow option, and priced using the same parameters as the training and validation data while allowing for the 20% moves in either direction without moving outside the input space for which the ANN was trained. For each option 100 random end of day market states were generated, where all inputs could move up to 20% in either positive or negative direction. Once the data set was complete the true empirical VaR and ES was calculated as well as approximated with the ANN and the Taylor polynomial.

3.4.5 Monte Carlo Convergence

As Monte Carlo simulations are a stochastic method it is important to understand that if the same option is evaluated twice with Monte Carlo simulation two different present values will be calculated. This means that there is an uncertainty in what the ANN interprets as the true answer. Thus, it is important to examine the convergence of the Monte Carlo method used in the study. The following table shows a point estimate of the convergence of the Monte Carlo method for 10000 calculations of the same rainbow option with a different number of samples per calculation.

Samples	Mean PV	$\hat{\sigma}^2$	$\hat{\sigma}$	$\hat{\sigma}$/Mean PV
1000	$1.665 \cdot 10^{-1}$	$1.66 \cdot 10^{-5}$	$4.07 \cdot 10^{-3}$	2.44%
5000	$1.666 \cdot 10^{-1}$	$3.90 \cdot 10^{-6}$	$1.98 \cdot 10^{-3}$	1.19%
10000	$1.666 \cdot 10^{-1}$	$2.01 \cdot 10^{-6}$	$1.42 \cdot 10^{-3}$	0.85%

Table 3.4: A point estimate of the convergence of the Monte Carlo simulation

It is important to note that this is only a point estimate for one set of inputs. Thus, the irreducible error for the general data set may be larger but Table 3.4 gives an indication as to the order of magnitude of the irreducible error in the Monte Carlo simulations.

As mentioned in section 2.4.2 the convergence of the Monte Carlo simulation is inversely proportional to the square root of the number of samples. In theory this means that an extraordinarily large amount of samples are needed to converge to an accurate rainbow option price.

In practice, a simple fit of an exponential curve shows that reducing the variance in the Monte Carlo simulations to an order of magnitude of 10^{-8} approximately 40000 paths per option are needed. Using the personal computer available to the authors this would require days of Monte Carlo simulation to produce the training data. Thus, 10000 paths were used as this allowed for generation of the training data in approximately six hours while still reducing the error from 5000 paths by a factor of two. Parallelization and micro-architecture optimization of the Monte Carlo simulation could allow for more samples to be used in the same time frame but this is deemed out of scope for the purpose of this study.

3.5 Taylor Approximation

The Taylor approximation used in this study is a second-order Taylor polynomial with the following sensitivities:

- Delta: The sensitivity of the present value with respect to a move in the spot price of the underlying equity.
- Gamma: The second-order sensitivity of the present value with respect to a move in the spot price of the underlying equity.

- Vega: The sensitivity of the present value with respect to a move in the implied volatility of the underlying equity.
- Volga: The second-order sensitivity of the present value with respect to a move in the implied volatility of the underlying equity.
- Rho: The sensitivity of the present value with respect to a move in the interest rate.
- Second-order Rho: The second-order sensitivity of the present value with respect to a move in the interest rate.

These sensitivities were calculated using the centered finite difference method for all options in the comparison data. For the rainbow options each sensitivity was calculated for each underlying equity individually.

3.6 ANN Structure

3.6.1 Choice of Cost Function

As shown in section 2.2 the MSE has a proven history as a cost function when using ANNs for option pricing.

$$C = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y_i})^2.$$

3.6.2 Choice of Activation Function

The Rectified Linear Unit (ReLU) ($max[0, x]$) was chosen as the activation function for multiple reasons. One reason is the sparse activation of the network meaning that training is faster. Another reason is that the ReLU avoids the vanishing gradient problem that sigmoid activation functions suffer from (see [38]). The final and most important reason for choosing the ReLU activation function is it's success and popularity in recent ANN applications (see [39]). An important point to note is that the ANN is still a universal function approximator while using the ReLU activation function as shown by Leshno et al. [40].

3.6.3 Training Parameters

All ANNs were trained using a batch size of 200 as this resulted in the lowest validation errors. The batch size was varied between 10 and 1000 with 200 yielding the lowest validation error thus those are the results presented in this study.

The number of epochs used was 50 as all networks had reached a stable validation error which did not change after 50 epochs. The number of epochs was varied between 10 and 500 but as mentioned the validation error did not improve after 50 in any of the cases and thus those are the results presented in this study.

Chapter 4

Results

4.1 Comparison of Run Times

Since the basis of the research question is the fact that an ANN can outperform Monte Carlo computationally it is important to ensure that this is the case. The following table shows the run times for calculating the present value of 10000 options with the different methods as a percentage of the slowest method. The rainbow options were evaluated with three different amounts of samples for the Monte Carlo simulation. The ANN run times for both the call option and the rainbow option are the ANNs with the lowest validation errors as shown in the sections that follow. In the table RMS stands for risk-management software.

Method	Run Time
Call Option RMS	0.01%
Call Option ANN	4.03%
Rainbow Option RMS 1000 samples	10.18%
Rainbow Option RMS 5000 samples	50.80%
Rainbow Option RMS 10000 samples	100.00%
Rainbow Option ANN	4.24%

Table 4.1: The run times for the different methods of calculating present values

4.2 Call Option

4.2.1 Model Selection

This section will present the results of the training of the model with different network architectures.

Validation Results

In Table 4.2 the results of the MSE evaluation of the validation data set are shown. The MSE is calculated as the MSE between the ANN's predicted C/K and the true C/K.

Inputs	Layer 1	Layer 2	Layer 3	Output	MSE
4	8	0	0	1	$7.12 \cdot 10^{-5}$
4	16	0	0	1	$2.84 \cdot 10^{-5}$
4	32	0	0	1	$1.18 \cdot 10^{-5}$
4	64	0	0	1	$3.63 \cdot 10^{-6}$
4	128	0	0	1	$1.39 \cdot 10^{-6}$
4	256	0	0	1	$2.99 \cdot 10^{-7}$
4	512	0	0	1	$4.34 \cdot 10^{-7}$
4	8	8	0	1	$3.24 \cdot 10^{-5}$
4	16	16	0	1	$2.58 \cdot 10^{-6}$
4	32	32	0	1	$1.61 \cdot 10^{-6}$
4	64	64	0	1	$3.52 \cdot 10^{-7}$
4	128	128	0	1	$8.03 \cdot 10^{-8}$
4	**256**	**256**	**0**	**1**	$\mathbf{7.27 \cdot 10^{-8}}$
4	512	512	0	1	$9.43 \cdot 10^{-8}$
4	8	8	8	1	$7.92 \cdot 10^{-6}$
4	16	16	16	1	$1.31 \cdot 10^{-6}$
4	32	32	32	1	$2.13 \cdot 10^{-7}$
4	64	64	64	1	$1.39 \cdot 10^{-6}$
4	128	128	128	1	$8.15 \cdot 10^{-8}$
4	256	256	256	1	$9.26 \cdot 10^{-8}$
4	512	512	512	1	$9.74 \cdot 10^{-8}$

Table 4.2: The MSE results of the validation of different models for call options

Choice of Model

The model that was chosen has two hidden layers with 256 nodes in each. This is the model with the lowest MSE, as seen in Table 4.2. This model was chosen with the bias-variance trade-off in mind and therefore the model with the lowest validation MSE was chosen rather than the model with the lowest training MSE.

4.2.2 Evaluation of Present Value Calculations

The chosen ANN's performance was compared with Taylor approximation for three different cases; movements in stock price, movement in implied volatility and movement in interest rates. For each case six sub cases were investigated, both positive and negative moves of 1%, 10%, and 20%.

In all of these evaluations the MSE was calculated as the MSE of the ANN's predicted C/K times $K = 1649$ (predicted present value) and the true present value as well as the Taylor approximated present value and the true present value. This explains why the ANN MSE is no longer in the order of magnitude 10^{-8} as it was in the earlier example.

When calculating the MAPE all cases with a true PV lower than 1 have been excluded from the calculations as this value is only approximately 0.06% of the underlyings value while at the same time leading to extreme and misleading errors for both the Taylor approximation and the ANN.

Movement in Stock Price

Move	Mean PV	MSE Taylor	MSE ANN	MAPE Taylor	MAPE ANN
-20%	46.27	$1.26 \cdot 10^{3}$	$8.75 \cdot 10^{-2}$	188.40%	1.81%
-10%	87.56	$4.01 \cdot 10^{1}$	$1.65 \cdot 10^{-1}$	19.62%	1.32%
-1%	145.17	$9.86 \cdot 10^{-5}$	$1.89 \cdot 10^{-1}$	0.02%	0.95%
+1%	160.53	$9.22 \cdot 10^{-5}$	$1.91 \cdot 10^{-1}$	0.02%	0.89%
+10%	239.80	$2.97 \cdot 10^{1}$	$2.19 \cdot 10^{-1}$	4.34%	0.65%
+20%	344.54	$9.14 \cdot 10^{2}$	$2.48 \cdot 10^{-1}$	11.52%	0.48%

Table 4.3: The results of the model comparison for stock price movements for call options

Movement in Implied Volatility

Move	Mean PV	MSE Taylor	MSE ANN	MAPE Taylor	MAPE ANN
-20%	129.33	$2.34 \cdot 10^{-2}$	$1.74 \cdot 10^{-1}$	0.44%	1.11%
-10%	140.85	$2.84 \cdot 10^{-4}$	$1.79 \cdot 10^{-1}$	0.04%	1.04%
-1%	151.54	$6.34 \cdot 10^{-8}$	$1.89 \cdot 10^{-1}$	0.00%	0.95%
+1%	153.95	$6.22 \cdot 10^{-8}$	$1.91 \cdot 10^{-1}$	0.00%	0.95%
+10%	164.93	$2.19 \cdot 10^{-4}$	$2.01 \cdot 10^{-1}$	0.03%	0.86%
+20%	177.35	$1.39 \cdot 10^{-2}$	$2.14 \cdot 10^{-1}$	0.22%	0.81%

Table 4.4: The results of the model comparison for implied volatility movements for call options

Movement in Interest Rates

Move	Mean PV	MSE Taylor	MSE ANN	MAPE Taylor	MAPE ANN
-20%	151.47	$7.22 \cdot 10^{-8}$	$1.88 \cdot 10^{-1}$	0.00%	0.96%
-10%	152.10	$3.65 \cdot 10^{-8}$	$3.62 \cdot 10^{-1}$	0.00%	0.97%
-1%	152.68	$4.47 \cdot 10^{-10}$	$2.26 \cdot 10^{-1}$	0.00%	0.97%
+1%	152.80	$4.47 \cdot 10^{-10}$	$2.01 \cdot 10^{-1}$	0.00%	0.97%
+10%	153.38	$3.66 \cdot 10^{-8}$	$1.27 \cdot 10^{-1}$	0.00%	0.97%
+20%	154.02	$7.22 \cdot 10^{-8}$	$8.71 \cdot 10^{-2}$	0.00%	0.99%

Table 4.5: The results of the model comparison for interest rate movements for call options

4.2.3 Evaluation of Risk Measurement

Value-at-Risk

The ANN's performance on Value-at-Risk was compared with Taylor approximation by investigating the MSE, the MAPE and the MPE. When calculating the MAPE and MPE all cases with a true VaR/ES lower than 1 have been excluded from the calculations as this value is only approximately 0.06% of the underlyings value while at the same time leading to extreme and misleading errors for both the Taylor approximation and the ANN.

Mean VaR	MSE Taylor	MSE ANN	MAPE Taylor	MAPE ANN	MPE Taylor	MPE ANN
109.08	$7.80 \cdot 10^{2}$	$1.01 \cdot 10^{-1}$	13.45%	0.46%	−4.08%	0.09%

Table 4.6: The results of the model comparison for VaR calculations for call options

Expected Shortfall

Mean ES	MSE Taylor	MSE ANN	MAPE Taylor	MAPE ANN	MPE Taylor	MPE ANN
161.64	$1.58 \cdot 10^3$	$8.06 \cdot 10^2$	13.92%	13.57%	−9.68%	13.55%

Table 4.7: The results of the model comparison for ES calculations for call options

4.3 Rainbow Option

4.3.1 Model Selection

This section will present the results of the training of the model with different network architectures.

Validation Results

In Table 4.8 the results of the MSE evaluation of the validation data set are shown. The MSE is calculated as the MSE between the ANN's predicted PV and the true PV. The validation data set was calculated using 10000 samples per option.

Inputs	Layer 1	Layer 2	Layer 3	Output	MSE 1000	MSE 5000	MSE 10000 samples
11	8	0	0	1	$1.89 \cdot 10^{-3}$	$1.69 \cdot 10^{-3}$	$9.30 \cdot 10^{-3}$
11	16	0	0	1	$7.71 \cdot 10^{-4}$	$6.75 \cdot 10^{-4}$	$6.72 \cdot 10^{-4}$
11	32	0	0	1	$4.76 \cdot 10^{-4}$	$2.88 \cdot 10^{-4}$	$3.99 \cdot 10^{-4}$
11	64	0	0	1	$3.39 \cdot 10^{-4}$	$1.55 \cdot 10^{-4}$	$1.34 \cdot 10^{-4}$
11	128	0	0	1	$1.94 \cdot 10^{-4}$	$6.85 \cdot 10^{-5}$	$6.66 \cdot 10^{-5}$
11	256	0	0	1	$1.75 \cdot 10^{-4}$	$5.39 \cdot 10^{-5}$	$5.90 \cdot 10^{-5}$
11	512	0	0	1	$1.42 \cdot 10^{-4}$	$4.33 \cdot 10^{-5}$	$3.80 \cdot 10^{-5}$
11	8	8	0	1	$8.80 \cdot 10^{-4}$	$6.31 \cdot 10^{-4}$	$7.17 \cdot 10^{-4}$
11	16	16	0	1	$2.62 \cdot 10^{-4}$	$2.31 \cdot 10^{-4}$	$1.78 \cdot 10^{-4}$
11	32	32	0	1	$1.60 \cdot 10^{-4}$	$6.59 \cdot 10^{-5}$	$5.36 \cdot 10^{-5}$
11	64	64	0	1	$1.52 \cdot 10^{-4}$	$4.89 \cdot 10^{-5}$	$1.68 \cdot 10^{-5}$
11	128	128	0	1	$1.41 \cdot 10^{-4}$	$1.66 \cdot 10^{-5}$	$7.61 \cdot 10^{-6}$
11	256	256	0	1	$1.29 \cdot 10^{-4}$	$8.81 \cdot 10^{-6}$	$5.98 \cdot 10^{-6}$
11	**512**	**512**	**0**	**1**	$1.39 \cdot 10^{-4}$	$1.42 \cdot 10^{-5}$	$\mathbf{3.54 \cdot 10^{-6}}$
11	8	8	8	1	$5.35 \cdot 10^{-4}$	$6.46 \cdot 10^{-4}$	$7.08 \cdot 10^{-4}$
11	16	16	16	1	$2.13 \cdot 10^{-4}$	$1.34 \cdot 10^{-4}$	$9.39 \cdot 10^{-5}$
11	32	32	32	1	$1.46 \cdot 10^{-4}$	$5.74 \cdot 10^{-5}$	$3.68 \cdot 10^{-5}$
11	64	64	64	1	$1.58 \cdot 10^{-4}$	$1.43 \cdot 10^{-5}$	$1.18 \cdot 10^{-5}$
11	128	128	128	1	$1.22 \cdot 10^{-4}$	$1.31 \cdot 10^{-5}$	$7.69 \cdot 10^{-6}$
11	256	256	256	1	$1.25 \cdot 10^{-4}$	$1.32 \cdot 10^{-5}$	$4.50 \cdot 10^{-6}$
11	512	512	512	1	$1.25 \cdot 10^{-4}$	$9.96 \cdot 10^{-6}$	$3.96 \cdot 10^{-6}$

Table 4.8: The MSE results of the validation of different models for rainbow options

Choice of Model

Since the present value which the ANN uses as an answer contains an error due to the fact that it is the product of a Monte Carlo simulation it is important to remember σ^2 and $\mathcal{E}$ from section 2.3.5. As shown in section 3.4.5 the $\hat{\sigma}^2$ for the Monte Carlo simulations used with 10000 samples are of the order of magnitude 10^{-6} meaning that there is an irreducible error of that size which the ANN cannot avoid.

As can be seen in Table 4.2 the general trend is that the more samples used in the Monte Carlo simulation, the lower the MSE for a model of a given complexity. This indicates that the error from the Monte Carlo simulations is carrying forwards into the ANN. The lowest validation error for each sample

amount is close to being the same size as the irreducible error for that sample amount meaning that the ANN is close to learning as much as it can from the Monte Carlo simulations. This indicates that more rigorous Monte Carlo simulations of the present value could allow for even better MSE results for the ANN but as mentioned in section 3.4.5 there are certain computational boundaries.

The model that was chosen has two hidden layers with 512 nodes in each. The choice was based on the performance with 10000 samples in the Monte Carlo simulations. This is the model with the lowest MSE, as seen in Table 4.8. This model was chosen with the bias-variance trade-off in mind and therefore the model with the lowest validation MSE was chosen rather than the model with the lowest training MSE.

4.3.2 Evaluation of Present Value Calculations

The chosen ANN's performance was compared with Taylor approximation for seven different cases; movements in stock price for each equity, movement in implied volatility for each equity and movement in interest rates. For each case six sub cases were investigated, both positive and negative moves of 1%, 10%, and 20%.

In all of these evaluations the MSE was calculated as the MSE of the ANN's predicted present value and the true present value as well as the Taylor approximated present value and the true present value.

When calculating the MAPE all cases with a true PV lower than 0.001 have been excluded from the calculations as this value is only approximately 0.1% of the underlyings value while at the same time leading to extreme and misleading errors for both the Taylor approximation and the ANN.

Movement in Stock Price for Equity 1

Move	Mean PV	MSE Taylor	MSE ANN	MAPE Taylor	MAPE ANN
-20%	0.21	$2.27 \cdot 10^{-4}$	$4.82 \cdot 10^{-5}$	20.58%	4.19%
-10%	0.23	$1.68 \cdot 10^{-5}$	$4.63 \cdot 10^{-5}$	3.40%	3.46%
-1%	0.25	$1.10 \cdot 10^{-5}$	$4.44 \cdot 10^{-5}$	1.03%	2.93%
+1%	0.26	$1.13 \cdot 10^{-5}$	$4.33 \cdot 10^{-5}$	1.01%	2.81%
+10%	0.29	$1.81 \cdot 10^{-5}$	$4.15 \cdot 10^{-5}$	1.42%	2.36%
+20%	0.34	$2.05 \cdot 10^{-4}$	$3.94 \cdot 10^{-5}$	3.31%	1.91%

Table 4.9: The results of the model comparison for stock price movements in equity 1 for rainbow options

Movement in Stock Price for Equity 2

Move	Mean PV	MSE Taylor	MSE ANN	MAPE Taylor	MAPE ANN
-20%	0.20	$3.14 \cdot 10^{-4}$	$3.76 \cdot 10^{-5}$	21.57%	4.07%
-10%	0.22	$2.13 \cdot 10^{-5}$	$4.10 \cdot 10^{-5}$	4.27%	3.42%
-1%	0.25	$1.05 \cdot 10^{-5}$	$4.30 \cdot 10^{-5}$	1.02%	2.93%
+1%	0.26	$1.09 \cdot 10^{-5}$	$4.31 \cdot 10^{-5}$	1.01%	2.84%
+10%	0.29	$2.01 \cdot 10^{-5}$	$4.52 \cdot 10^{-5}$	1.52%	2.38%
+20%	0.35	$2.39 \cdot 10^{-4}$	$4.74 \cdot 10^{-5}$	3.61%	1.97%

Table 4.10: The results of the model comparison for stock price movements in equity 2 for rainbow options

Movement in Stock Price for Equity 3

Move	Mean PV	MSE Taylor	MSE ANN	MAPE Taylor	MAPE ANN
-20%	0.21	$2.35 \cdot 10^{-4}$	$2.83 \cdot 10^{-5}$	22.26%	3.75%
-10%	0.23	$1.67 \cdot 10^{-5}$	$3.53 \cdot 10^{-5}$	3.31%	3.39%
-1%	0.25	$1.09 \cdot 10^{-5}$	$4.23 \cdot 10^{-5}$	1.03%	2.97%
+1%	0.26	$1.13 \cdot 10^{-5}$	$4.44 \cdot 10^{-5}$	1.02%	2.92%
+10%	0.29	$1.77 \cdot 10^{-5}$	$5.39 \cdot 10^{-5}$	1.41%	2.54%
+20%	0.34	$1.86 \cdot 10^{-4}$	$6.37 \cdot 10^{-5}$	3.28%	2.24%

Table 4.11: The results of the model comparison for stock price movements in equity 3 for rainbow options

Movement in Implied Volatility in Equity 1

Move	Mean PV	MSE Taylor	MSE ANN	MAPE Taylor	MAPE ANN
-20%	0.24	$9.62 \cdot 10^{-6}$	$4.44 \cdot 10^{-5}$	1.01%	3.14%
-10%	0.25	$9.71 \cdot 10^{-6}$	$4.33 \cdot 10^{-5}$	0.98%	2.96%
-1%	0.25	$1.07 \cdot 10^{-5}$	$4.34 \cdot 10^{-5}$	0.99%	2.89%
+1%	0.25	$1.12 \cdot 10^{-5}$	$4.24 \cdot 10^{-5}$	1.02%	2.89%
+10%	0.26	$1.23 \cdot 10^{-5}$	$4.23 \cdot 10^{-5}$	1.03%	2.75%
+20%	0.27	$1.35 \cdot 10^{-5}$	$4.29 \cdot 10^{-5}$	1.04%	2.68%

Table 4.12: The results of the model comparison for implied volatility movements in equity 1 for rainbow options

Movement in Implied Volatility in Equity 2

Move	Mean PV	MSE Taylor	MSE ANN	MAPE Taylor	MAPE ANN
-20%	0.24	$9.73 \cdot 10^{-6}$	$3.69 \cdot 10^{-5}$	1.02%	2.94%
-10%	0.25	$1.01 \cdot 10^{-5}$	$3.92 \cdot 10^{-5}$	1.01%	2.93%
-1%	0.25	$1.09 \cdot 10^{-5}$	$4.30 \cdot 10^{-5}$	1.01%	2.93%
+1%	0.25	$1.11 \cdot 10^{-5}$	$4.31 \cdot 10^{-5}$	1.02%	2.90%
+10%	0.26	$1.21 \cdot 10^{-5}$	$4.83 \cdot 10^{-5}$	1.01%	2.89%
+20%	0.27	$1.32 \cdot 10^{-5}$	$5.42 \cdot 10^{-5}$	1.02%	2.81%

Table 4.13: The results of the model comparison for implied volatility movements in equity 2 for rainbow options

Movement in Implied Volatility in Equity 3

Move	Mean PV	MSE Taylor	MSE ANN	MAPE Taylor	MAPE ANN
-20%	0.24	$9.27 \cdot 10^{-6}$	$2.88 \cdot 10^{-5}$	1.02%	2.78%
-10%	0.25	$9.85 \cdot 10^{-6}$	$3.53 \cdot 10^{-5}$	1.00%	2.87%
-1%	0.25	$1.07 \cdot 10^{-5}$	$4.27 \cdot 10^{-5}$	1.01%	2.93%
+1%	0.25	$1.09 \cdot 10^{-5}$	$4.47 \cdot 10^{-5}$	1.01%	2.97%
+10%	0.26	$1.22 \cdot 10^{-5}$	$5.30 \cdot 10^{-5}$	1.02%	3.01%
+20%	0.27	$1.39 \cdot 10^{-5}$	$6.53 \cdot 10^{-5}$	1.04%	3.06%

Table 4.14: The results of the model comparison for implied volatility movements in equity 3 for rainbow options

Movement in Interest Rates

Move	Mean PV	MSE Taylor	MSE ANN	MAPE Taylor	MAPE ANN
-20%	0.25	$3.64 \cdot 10^{-5}$	$4.31 \cdot 10^{-5}$	1.95%	2.93%
-10%	0.25	$1.89 \cdot 10^{-5}$	$4.93 \cdot 10^{-5}$	1.38%	2.88%
-1%	0.25	$1.13 \cdot 10^{-5}$	$4.29 \cdot 10^{-5}$	1.04%	2.84%
+1%	0.25	$1.02 \cdot 10^{-5}$	$4.35 \cdot 10^{-5}$	1.00%	2.86%
+10%	0.25	$1.16 \cdot 10^{-5}$	$4.29 \cdot 10^{-5}$	1.09%	2.92%
+20%	0.25	$2.19 \cdot 10^{-5}$	$4.36 \cdot 10^{-5}$	1.49%	2.92%

Table 4.15: The results of the model comparison for interest rate movements for rainbow options

4.3.3 Evaluation of Risk Measurement

Value-at-Risk

The ANN's performance on Value-at-Risk was compared with Taylor approximation by investigating the MSE, the MAPE and the MPE. When calculating the MAPE and MPE all cases with a true VaR/ES lower than 0.001 have been excluded from the calculations as this value is only approximately 0.1% of the underlyings value while at the same time leading to extreme and misleading errors for both the Taylor approximation and the ANN.

Mean VaR	MSE Taylor	MSE ANN	MAPE Taylor	MAPE ANN	MPE Taylor	MPE ANN
0.11	$4.02 \cdot 10^{-4}$	$6.03 \cdot 10^{-5}$	12.55%	4.78%	1.92%	−2.56%

Table 4.16: The results of the model comparison for VaR calculations for rainbow options

Expected Shortfall

Mean ES	MSE Taylor	MSE ANN	MAPE Taylor	MAPE ANN	MPE Taylor	MPE ANN
0.19	$9.49 \cdot 10^{-4}$	$8.49 \cdot 10^{-4}$	11.25%	14.64%	3.30%	14.29%

Table 4.17: The results of the model comparison for ES calculations for rainbow options

Chapter 5

Analysis and Conclusion

5.1 Analysis

5.1.1 Run Times

The run times of the different methods are interesting, beginning with the call option. Since the risk-management software uses a pre-determined pricing formula it executes extremely quickly. The ANN on the other hand must execute multiple matrix multiplications and transformations that take time. Even though matrix multiplication is often quite fast it is not faster than evaluating a single formula.

The rainbow option on the other hand is the most interesting as this is the use case being studied. The ANN is much more computationally efficient running between two and twenty-five times faster than the Monte Carlo simulation depending on the number of samples per option. It is also interesting to note that the run time for the ANN is the same for both call and rainbow options since approximately the same matrix multiplications and transformations must be executed.

5.1.2 Call Option

The present value evaluation in section 4.2.2 shows that the ANN outperforms Taylor approximation for large moves in the stock price when evaluating both MSE and MAPE, while the Taylor approximation performs better than the ANN for small 1% moves. For moves in implied volatility the Taylor approximation performs extremely well with a small MSE and a low MAPE, with the largest error for movements of 20% while the ANN performs worse but with a constant MAPE around 1%. For moves in interest rates the Taylor approximation is the better performing approach with a very low MSE and a MAPE close to 0%. For interest rate moves the ANN MAPE is stable around 1% as well.

An important observation is that for all samples and cases the total MSE for the ANN is $1.88 \cdot 10^{-1}$ and the median squared error is $6.55 \cdot 10^{-2}$ while the total MSE for the Taylor approximation is $1.25 \cdot 10^{2}$ with a median squared error of $2.38 \cdot 10^{-7}$. This points to the fact that the Taylor approximation does very poorly in a few cases and very well in all others leading to a misleading total MSE.

Equity Moves

It is interesting to note that the ANN outperforms the Taylor approximation in four sub-cases within the equity movements. The ANN outperforms the Taylor approximation for both positive and negative moves of 10% and 20%, especially negative 20%. This can be traced to the large movements of the price of the option with regards to theses moves meaning that the Taylor approximation fails. This is quite natural since the Taylor approximation fits a second degree polynomial to the delta function which is a third degree polynomial, thus it fits the tails of the delta function poorly and is more suitable for approximating small moves. These large errors are the probable root cause of the misleading MSE of the Taylor approximation in comparison with the median squared error as referenced above.

Implied Volatility Moves

The Taylor approximation beats the ANN in all the different types of implied volatility moves examined in this study. This is mainly due to the fact that

the even the largest moves in implied volatility of plus or minus 20% do not affect the present value as much as for example and equity move of plus or minus 20% as seen in the movements of the mean present value. This means that the assumptions in the Taylor approximation are closer to being true.

The MAPE for ANN is as for equity moves approximately 1%, thus if one believes the Taylor error will be around 1% using an ANN could still be a useful approach.

Interest Rate Moves

The Taylor approximation once again outperforms the ANN with regards to handling movements in rates. The rate moves are similar to the volatility moves since the effect of a plus or minus 20% move on the present value of the option is small in comparison to a similar equity move. Once again this means that the assumptions in the Taylor model are more closely followed leading to good results using the Taylor approximation.

The MAPE for ANN is as for equity- and implied volatility moves approximately 1%, thus if one believes the Taylor error will be around 1% using an ANN could still be a useful approach.

Risk Measures

It is interesting to note that the ANN outperforms Taylor approximation for Value-at-Risk calculations both when looking at the MSE and the MAPE and MPE. As for the present value calculations the ANN has a MSE in the order of magnitude 10^{-1}, while Taylor performs much worse with an error in the order of magnitude of 10^2. The ANN's MAPE of 0.46% and MPE of 0.09% shows that the ANN both over- and underestimates the present value of the call option. The Taylor approximation overestimates the Value-at-Risk which is better than underestimating the risk but the ANN only slightly underestimates the risk.

When calculating Expected Shortfall the Taylor approximation performs better than the ANN. In this case both methods have a large MSE in the order of magnitude of $10^2 - 10^3$ and approximately the same MAPE. However, the MPE shows that the ANN underestimates the Expected Shortfall by

roughly 10% while the Taylor approximation overestimates it by a similar amount which is a better mistake.

5.1.3 Rainbow Option

The present value evaluation in section 4.3.2 shows that as for the call option the ANN performs better than the Taylor approximation for large equity moves of 20% when evaluating with MSE and MAPE. However, for small 1% moves the Taylor approximation outperforms the ANN with a MSE and a MAPE close to the irreducible error from the Monte Carlo simulation. For moves in implied volatility the Taylor approximation performs very well with a MSE close to the irreducible error regardless of the size of the move, and the MAPE is very low. The ANN performs worse, but with a stable MSE in the order of magnitude 10^{-5} for all cases and a MAPE close to 3%. For changes in the interest rate the Taylor approximation performs better when evaluated with MAPE and MSE.

An interesting observation is that the ANN exhibits an MSE in the order of magnitude 10^{-5} for all types of moves and cases. This is in line with what can be expected since it is approximately equal to the irreducible error and validation error from the Monte Carlo simulations as shown in section 3.4.5.

Equity Moves

As for the call option the ANN performs better than the Taylor approximation for larger moves in the equity. However, it is only in the case of a negative move of 20% that the ANN substantially outperforms. The reason that the ANN does not outperform the Taylor approximation as much as it does for call options can be traced to the movements of the option price with regards to the moves in the equity spot price. For the call option a large move in equity affects the option price more than in the case when one equity out of three moves in the rainbow option.

Implied Volatility Moves

For all types of moves in implied volatility the Taylor approximation is close to the irreducible error meaning that the Taylor approximation itself has

an extremely small error. The strong performance can be traced to what is discussed for the equity case, small movements in the option price. For the largest moves in implied volatility of plus or minus 20% the option price moves similar to a 1% move in equity. This means that the assumptions in the Taylor approximation are closer to being true, and thus it performs well.

The MAPE for the Taylor approximation is close to 1% and for the ANN is it is stable around 3%.

Interest Rates Moves

The Taylor approximation once again outperforms the ANN with regards to handling movements in rates. The reason for this is as for the implied volatility, the moves in interest rates do not affect the present value of the option much and thus the assumptions in the Taylor approximation are closer to being true. Once again the Taylor approximation is very close to the irreducible error meaning that the Taylor approximation has a very small error in comparison to the Monte Carlo simulations.

The MAPE for ANN is as for equity- and implied volatility moves approximately 3%, thus if the error from a Taylor approximation is expected to be larger than than 3% using an ANN could still be a useful approach.

Risk Measures

For calculation of Value-at-Risk the ANN outperforms the Taylor approximation in all categories except MPE. It is important to note that while the ANN has a larger MPE in absolute terms the ANN overestimates the risk while the Taylor approximation underestimates the risk.

When calculating Expected Shortfall the Taylor approximation outperforms the ANN in all categories. It is interesting to note the difference between the MAPE and MPE between the two methods. The Taylor MAPE is approximately three times larger than the MPE indicating that there are large errors in both direct directions that cancel out when calculating the MPE. Meanwhile, the ANN consistently underestimates the risk by approximately 14%.

5.2 Discussion

5.2.1 Reflection on the Study

In the ideal case the ANN would learn the exact pricing function used by the risk-management software for both the call option and the rainbow option. This is the universal approximation theorem proven by Cybenko [17]. The theorem states that a MLP with a single hidden layer can represent continuous functions on compact subsets. An important point to note is that the theorem does not comment upon the learnability of the correct parameters to create this approximation.

Regarding the underperforming ANNs for both call options and rainbow options it does not look like the ANN has learned the underlying function it is trying to approximate well enough to outperform the Taylor approximation. In the rainbow option case both the Monte Carlo simulation and an imperfect ANN cause errors while in the call option case it is only an imperfect ANN that leads to the errors seen in the results. In the ideal case, the errors due to an imperfect ANN would be zero but that is not the case in this study. There may be multiple reasons for this which can be grouped into four major categories: data, algorithms, algorithm tuning, and learning difficulty.

One possible explanation for the underperformance of the ANN is a lack of qualitative training data. A lack of qualitative training data would lead to the ANN not learning the underlying function in an adequate manner. In the case of ANNs more data is always better and this is a possible explanation of the underperformance of the ANN. This can be shown by looking at an example from the call option case. Using only 10% of the training data and the model of choice for call options gives an MSE in the order of magnitude 10^{-6} which is much larger than the original 10^{-8}. This shows that adding more data improves the MSE results for the ANN.

Another explanation for the performance of the ANN in the rainbow option case is the quality of the training data. Since the rainbow option is a product of a Monte Carlo simulation there is an error term in the "correct answer" creating a lower bound for the approximation error as shown in section 2.3.5. As seen in Table 4.8 this became a bottleneck regarding the training of the ANN. As mentioned, this study has been bounded by the performance and memory of a personal computer and thus equipment allowing for more and better data along with more training could yield better results.

Another possible explanation is the choice of algorithm. A poor choice of algorithm could lead to difficulties for the ANN in learning the underlying mechanics that are supposed to be modeled. The choices of algorithm made in this study were based upon established methods found in previous literature.

Another possible explanation is poor algorithm tuning. Poor choice of network architecture, activation function and optimization could inhibit the ANNs learning capabilities. This study has tried to counteract these effects by choosing established methods from previous research as well as diligent use of a validation data set to evaluate network architecture. Unfortunately there is no guarantee that any network architecture or learning algorithm can find the correct parameters especially if working with an error function with many local minima.

As stated above the universal approximation theorem does not comment upon the learnability of the correct parameters for universal function approximation. This along with the fact that gradient descent does not guarantee a global minimum for non-convex surface means that there is a theoretical possibility that the ANN could outperform the Taylor approximation but this study could not reach the correct parameters to approximate the underlying pricing function well enough.

5.2.2 Comparison with Other Studies

The ANN used in this study did not manage to replicate the pricing function used by the risk-management software well enough to beat the Taylor approximation. However, when comparing the results to other studies the ANN presented in this study performs well. It is important to note that most studies conducted in the field of option pricing with ANNs differ from this study, both in terms of the purpose of the studies and how the ANN has been designed and used. This makes it hard to relate the performance of these ANNs to the ANN used in this study. The study that most strongly resembles the ANN used in this study is Amilon's [11] comparison of ANN performance on pricing and hedging versus the Black-Scholes model for options on the Swedish stock index. As mentioned earlier, this study compares the ANN to the Black-Scholes model applied to market data meaning that the ANN is trying to replicate a "true" market pricing formula which is quite different to what this study is trying to achieve but it is still closest to what

this study has attempted.

In spite of the differences, Amilon's best performing ANN had an average MSE of 1.82 when evaluated using implied volatility and the Black-Scholes model had an average MSE of 2.48 when evaluated with the same data. To be able to compare to this study the average value for bid and ask MSE was calculated for Amilon's study.

The ANN used in this study for pricing call options has a validation error of $1.20 \cdot 10^{-4}$ when calculated in terms of the option price meaning that this study clearly outperforms the ANN Amilon presents. This is not a surprise as the network used in this study is more complex and has an easier task at hand with clean data.

5.3 Conclusion

While the idea of using an ANN to approximate the present value function for European call options and rainbow options is theoretically sound, the method used in this study does not give adequate enough approximations though there are many positive signs. While there are some cases in the equity movements when the ANN beats the Taylor approximation the general answer to the research question is that the ANN does not outperform the Taylor approximation in general for either call options or rainbow options when trained and evaluated according to this study's method. While the ANNs examined in this report did not outperform the Taylor approximation it is important to remember that if the correct weight and bias parameters can be learnt an ANN is a universal function approximator. This means that a method which can find those correct weights and biases will outperform any Taylor approximation since the ANN will approximate the underlying function arbitrarily well.

In the case of the call option the bottleneck for ANN performance seems to be the training, both in terms of data and time. One possible solution is to increase the training by orders of magnitude. This includes training data that may include billions of samples along with running the ANN on a/multiple dedicated and optimized machine(s). While this is no guarantee for success both theory and the learning trend when increasing data in this study point to it being a road forward.

For the rainbow option there are two bottlenecks. The first and the limiting factor in this study is the quality of the Monte Carlo simulations. The ANN will never be able to outperform the irreducible error from the Monte Carlo simulation. Once the error from the Monte Carlo simulation has been reduced the same bottleneck as for the call option is still in place regarding training. This study shows positive signs that the ANN seems to be able to learn how to price rainbow options but once again to be able to compete with the Taylor approximation an order of magnitude shift is needed in training. To reach the level of precision needed for risk measurement once again billions of samples may be needed to train an ANN running on a/multiple dedicated and optimized machine(s).

The most interesting part of the study is the fact that there are positive signs that the ANN can learn the pricing function for rainbow options that require Monte Carlo simulation. This is interesting since there are no analytic solutions for the price of a rainbow option as has been mentioned multiple times. Thus, the study is a proof of concept that an ANN can learn these types of pricing functions. The ANNs in this study were able to minimize the validation error to approximately the irreducible error meaning that approaches which have a smaller irreducible errors, such as more accurate Monte Carlo simulations, could yield better results. While the ANNs trained and evaluated in this study could not outperform the Taylor approximation there are no indications that it could not done in the right setting.

5.4 Future Research

One potential road to improved performance of the ANN is to create a committee machine known as an ensemble. An ensemble utilizes multiple ANNs and averages their results to create a final result. Ensembles rely on two facts. The first is the fact that an ANN can reduce bias while increasing variance and the second is that a group of networks can reduce variance without affecting bias (see [41]).

The creation of an ensemble of neural networks has multiple benefits. Firstly, the ensemble is less complex than a similarly performing single network in almost every case (see [42]). Secondly, the ensemble often outperforms any single network (see [43]). Finally, as there are fewer parameters in every ANN the risk of overfitting is reduced (see [44]). These benefits could help improve the performance of an ANN trying to calculate present values of options.

Chapter 6

References

[1] Black F, Scholes M. The pricing of options and corporate liabilities. Journal of political economy. 1973;81(3):637–654.

[2] Boyle P. Options: A monte carlo approach. Journal of financial economics. 1977;4(3):323–338.

[3] Glasserman P. Monte Carlo methods in financial engineering. vol. 53. Springer Science & Business Media; 2013.

[4] Li Y, Ma W. Applications of artificial neural networks in financial economics: a survey. In: Computational Intelligence and Design (ISCID), 2010 International Symposium on. vol. 1. IEEE; 2010. p. 211–214.

[5] Rubinstein M. Somewhere over the rainbow. Risk. 1991;4(11):61–63.

[6] Austing P. Smile pricing explained. Springer; 2014.

[7] Alexander C, Venkatramanan A. Analytic Approximations for Multi-Asset Option Pricing. Mathematical Finance. 2012;22(4):667–689.

[8] Hutchinson J, Lo A, Poggio T. A nonparametric approach to pricing and hedging derivative securities via learning networks. The Journal of Finance. 1994;49(3):851–889.

[9] Lajbcygier P, Connor J. Improved option pricing using artificial neural networks and bootstrap methods. International journal of neural systems. 1997;8(04):457–471.

[10] Gençay R, Qi M. Pricing and hedging derivative securities with neural networks: Bayesian regularization, early stopping, and bagging. IEEE Transactions on Neural Networks. 2001;12(4):726–734.

[11] Amilon H. A neural network versus Black–Scholes: a comparison of pricing and hedging performances. Journal of Forecasting. 2003;22(4):317–335.

[12] Gradojevic N, Gençay R, Kukolj D. Option pricing with modular neural networks. IEEE transactions on neural networks. 2009;20(4):626–637.

[13] Liang X, Zhang H, Xiao J, Chen Y. Improving option price forecasts with neural networks and support vector regressions. Neurocomputing. 2009;72(13-15):3055–3065.

[14] Wang YH. Nonlinear neural network forecasting model for stock index option price: Hybrid GJR–GARCH approach. Expert Systems with Applications. 2009;36(1):564–570.

[15] Ciregan D, Meier U, Schmidhuber J. Multi-column deep neural networks for image classification. In: Computer vision and pattern recognition (CVPR), 2012 IEEE conference on. IEEE; 2012. p. 3642–3649. Available at: `http://arxiv.org/abs/1202.2745`.

[16] Sak H, Senior A, Beaufays F. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In: Fifteenth annual conference of the international speech communication association; 2014. p. 1–5. Available at: `https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/43905.pdf`.

[17] Cybenko G. Approximation by superpositions of a sigmoidal function. Mathematics of control, signals and systems. 1989;2(4):303–314.

[18] Schmidhuber J. Deep learning in neural networks: An overview. Neural networks. 2015;61:85–117.

[19] Bengio Y, et al. Learning deep architectures for AI. Foundations and trends in Machine Learning. 2009;2(1):1–127.

[20] Gurney K. An introduction to neural networks. CRC press; 2014.

[21] White H. Connectionist nonparametric regression: Multilayer feedforward networks can learn arbitrary mappings. Neural networks. 1990;3(5):535–549.

[22] Bridle J. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In: Neurocomputing. Springer; 1990. p. 227–236.

[23] Rumelhart D, Hinton G, Williams R. Learning representations by back-propagating errors. nature. 1986;323(6088):533.

[24] Bottou L. Stochastic gradient learning in neural networks. Proceedings of Neuro-Nımes. 1991;91(8):12.

[25] Kingma D, Ba J. Adam: A method for stochastic optimization. arXiv preprint arXiv:14126980. 2014;.

[26] Tibshirani R, James G, Witten D, Hastie T. An introduction to statistical learning-with applications in R. New York, NY: Springer; 2013.

[27] Barron A. Universal approximation bounds for superpositions of a sigmoidal function. IEEE Transactions on Information theory. 1993;39(3):930–945.

[28] Goodfellow I, Bengio Y, Courville A, Bengio Y. Deep learning. vol. 1. MIT press Cambridge; 2016.

[29] Montufar G, Pascanu R, Cho K, Bengio Y. On the number of linear regions of deep neural networks. In: Advances in neural information processing systems; 2014. p. 2924–2932.

[30] Bingham N, Kiesel R. Risk-Neutral Valuation; Pricing and Hedging of Financial Derivatives. Springer-Verlag London Ltd.; 2004. ISBN: 978-1-84996-873-7.

[31] Hull J. Options, futures, and other derivatives. Pearson Education; 2016.

[32] Harrison M, Pliska S. Martingales and stochastic integrals in the theory of continuous trading. Stochastic processes and their applications. 1981;11(3):215–260.

[33] Haug E, Taleb N. Why we have never used the Black-Scholes-Merton option pricing formula. Social Science Research Network Working Paper Series. 2008;1(4).

[34] Yalincak OH. Criticism of the Black-Scholes Model: But Why is It Still Used?:(The Answer is Simpler than the Formula); 2012. Available at SSRN: `https://ssrn.com/abstract=2115141` or `http://dx.doi.org/10.2139/ssrn.2115141`.

[35] Derman E. Laughter in the dark-the problem of the volatility smile. In: Euronext Options Conference (Amsterdam); 2003. p. 1–13. Available at: https://pdfs.semanticscholar.org/3d2a/0c34c71c069c736f07a44da0c5a11203c87b.pdf.

[36] Cortazar G, Gravet M, Urzua J. The valuation of multidimensional American real options using the LSM simulation method. Computers & Operations Research. 2008;35(1):113–129.

[37] Hult H, Lindskog F, Hammarlid O, Rehn CJ. Risk and portfolio analysis: Principles and methods. Springer Science & Business Media; 2012.

[38] Glorot X, Bordes A, Bengio Y. Deep sparse rectifier neural networks. In: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics; 2011. p. 315–323.

[39] LeCun Y, Bengio Y, Hinton G. Deep learning. nature. 2015;521(7553):436.

[40] Leshno M, Lin VY, Pinkus A, Schocken S. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. Neural networks. 1993;6(6):861–867.

[41] Naftaly U, Intrator N, Horn D. Optimal ensemble averaging of neural networks. Network: Computation in Neural Systems. 1997;8(3):283–296.

[42] Pearlmutter B, Rosenfeld R. Chaitin-Kolmogorov complexity and generalization in neural networks. In: Advances in neural information processing systems; 1991. p. 925–931.

[43] Hashem S. Optimal linear combinations of neural networks. Neural networks. 1997;10(4):599–614.

[44] Haykin S. Neural networks, a comprehensive foundation. Macmilan; 1994.

TRITA -SCI-GRU 2018:162