Abstract

This paper proposes a data-driven approach to price European call and put options by using a wide and deep neural network (WDNN). The model was trained with options data on the S&P 500 index ranging from January 2016 to August 2020, validated on data from September to October 2020 and finally tested on data from November to December 2020, while the Black-Scholes model served as a benchmark to compare the pricing accuracy. The pricing accuracy of the WDNN model was far superior to that of Black-Scholes looking at multiple error measures for both call and put options, across a big majority of all groups of maturity and moneyness. In addition to being more accurate, the Machine Learning approach also has the advantage of evading any of the potentially false assumptions about financial mechanics of the Black-Scholes model. The WDNN furthermore has the edge of being much faster in deployment in practice for the pricing of derivatives, when compared to other computationally more demanding models needed to achieve pricing accuracy superior to that of the Black-Scholes model. The superior accuracy of the WDNN in pricing options can also be useful for the development of better hedging strategies, by calculating the greeks based on the partial derivatives of the WDNN instead of those of the Black-Scholes model.

Contents

List of Figures

List of Tables

Acronyms

AE Absolute Error

AI Artificial Intelligence

ANN Artificial Neural Network

ATM at-the-money

BS Black-Scholes

CS Coradu su

DITM deep-in-the-money

DNN Deep Neural Networks

DOTM deep-out-of-the-money

GA Genetic algorithms

GPU central processing unit

GPU graphics processing unit

ITM in-the-money

leaky ReLu leaky rectified linear unit

LGBM Light Gradient boosting Machine

MAE Mean Absolute Error

MAPE Symmetric Mean Absolute Percentage Error

MAPE Mean Absolute Percentage Error

MASE Mean Absolute Scaled Error

MDA Median Absolute Deviation

ML Machine Learning

MSE Mean Square Error

NN Neural Network

NTM near-the-money

OTM out-of-the-money

ReLu rectified linear unit

RF Random Forest

RMSE Root Mean Square Error

SELU scaled exponential linear unit

SVR Support Vector Machine

WDNN Wide and Deep Neural Network

XGB XGBoost

1 Introduction

"*All models have faults - that doesn't mean you can't use them as tools for making decisions*" (Scholes, 1975).

Options are financial derivatives that give the owner the right, but not the obligation, to buy (call) or sell (put) a particular security on (European type) or before (American type) an expiration date. In 1973 Fischer Black and Myron Scholes remarkably managed to find a closed-form solution to price European call options. Although the model has some limitations, thanks to it's relatively simple arithmetic and the limited amount of input features needed, the Black-Scholes model went on to have a significant impact on the development of quantitative finance and remains to this day, one of the most frequently used option pricing models (Mezofi & Szabo, 2018). The focus of this paper lies on the pricing of call and put options on the S&P 500 index, which have a European style exercise feature (no early exercise permitted), and can be priced according to the Black-Scholes Model, with the addition of a dividend component:

$$\mathrm{C} = \mathrm{S}e^{-qT}N(\mathrm{d}_1) - \mathrm{K}\mathrm{e}^{-rT}\mathrm{N}(\mathrm{d}_2) \tag{1}$$

$$\mathrm{P} = \mathrm{K}e^{-rT}N(-d_2) - \mathrm{S}\mathrm{e}^{-qT}\mathrm{N}(-d_1) \tag{2}$$

where

$$\mathrm{d}_1 = \frac{1}{\sigma\sqrt{\mathrm{T}}}\left[\ln\left(\frac{S}{K}\right) + T\left(r - q + \frac{\sigma^2}{2}\right)\right] \tag{3}$$

$$\mathrm{d}_2 = \mathrm{d}_1 - \sigma\sqrt{\mathrm{T}} \tag{4}$$

and

$$N(x) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{x} \mathrm{e}^{-\frac{1}{2}z^2}dz \tag{5}$$

Notation

C = Price of a call option

P = Price of a put option

S = Current underlying security price

K = Strike price of the option

r = Risk-free interest rate

σ = Volatility of the underlying security

T = Time to option maturity (in years)

q = Continuously compounded annualized dividend yield

N = Normal cumulative distribution function

The Black-Scholes model is a theory-driven model and relies on the assumptions that stock prices are continuous and follow a geometric Brownian motion, that the risk free rate and volatility are known and remain constant over the lifetime of an option and that the market is frictionless, meaning that there are no transactions costs, short squeezes (as seen in early 2021 with GME), or taxes and that markets are efficient, meaning that market movements cannot be predicted. Additional market assumptions include the no arbitrage opportunity (no way to make a riskless profit), the ability of all market participants to buy and sell any amount of securities and their ability to borrow and lend any amount at the risk-free rate (Hull, 2018, p. 324-369). Obviously, some of these assumptions do not always hold true in reality (Natenberg, 2014, p. 464-485), but even when controlled for assumptions, the Black-Scholes model is not able to reproduce some empirical findings and fails to explain the so-called implied volatility surface (Hull, 2018, p. 453-471). As a consequence of this, mathematically more demanding methods have been developed over time such as the Heston model, jump diffusion processes or Monte Carlo simulations, which are also computationally more expensive. The significant improvements in terms of computational power, data sets, and algorithmic advances throughout the 2000s have also led to a strong rise in the importance and potential usage of Machine Learning models in the last decade. Even though it can be computationally expensive and take a lot of time to train a Machine Learning (ML) model, the deployment to price derivatives is very fast once the model is trained and the weights of the neurons are saved.

Thanks to these advances, we hope to build a data-driven Machine Learning model trained with historical options data, not only evading some of the false assumptions about financial mechanics in the Black-Scholes model, but also able to outperform the Black-Scholes model in terms of pricing accuracy.

This paper therefore empirically investigates whether Machine Learning models can price European options more accurately than the Black-Scholes (BS) model. The data used to train, test and validate the models ranges from 2016 to 2020, consisting of over 13 million observations. Two separate models will be trained for the pricing of European call and put options respectively and the Black-Scholes model with a dividend component will serve as a benchmark. Furthermore it will be investigated, what influence the moneyness of options and time to maturity has on the pricing accuracy of the WDNN and the Black-Scholes model. To train the model the same inputs as for the Black-Scholes model will be used as features and the focus lies on the development of a ML model based on a rather new neural network architecture introduced in 2016 called ”Wide

and Deep Neural Network" (WDNN) using the keras and TensorFlow interface. Additionally, some of the results of this paper have also been implemented in a practical way in form of *this web applications*, which can be used to price European options and simulate different scenarios and will be further developed and maintained over time.

The paper is structured in the following way: The second chapter provides a short literature review. The third chapter gives information about the data used to train the Wide and Deep Neural Network (WDNN) models and price the options with the Black-Scholes model. The fourth chapter addresses the methodology used in training and optimizing the model and gives a summary of some of the most important developments in the field of Deep Learning. The fifth chapter contains the analysis and comparison of the pricing accuracy of the models. The final chapter concludes the most important results of this paper and gives a brief outlook.

2 Literature review

2.1 Early research on option pricing with Neural Networks

Malliaris and Salchenberger (1993) were some of the first to use a neural network to estimate market prices of options on the S&P 100 index, using data from 1 January 1990 to 30 June 1990 published in the Wall Street Journal. In addition to the 5 input variables required for the Black-Scholes model, they added yesterday's market price of the underlying security and yesterday's market price of the options as explaining variables to train their neural network and split their data into in-the-money (ITM) and out-of-the-money (OTM) groups to train two models. They used a feedforward neural network with 4 hidden layers and 1 output node. The sigmoidal function was used as activation function and the learning rate and momentum were set initially at 0.9 and 0.6, while the learning rate was adjusted downwards and the momentum upwards during training to improve performance. Their neural networks had a lower MAPE for 4 out of the 5 two-weeks periods used to test their model for out-of-the money options, but a higher MAPE for 4 out of the 5 two-weeks periods for in-the-money options.

Anders, Korn, and Schmitt (1998) applied statistical inference techniques to build a neural network to price call options on the German stock index DAX. They used a single hidden-layer feedforward neural net with a linear output and the tangent hyperbolic function as activation function. They found S/K, r, σ, and T to have a statistically significant influence on the option price and their model showed an overall higher accuracy than the Black-Scholes model in terms of R^2, RMSE, MAE and MAPE.

2.2 NN to price american options and inclusion of dividend component

Yao, Li, and Tan (2000) used a neural network to forecast prices of American style call options on Nikkei 225 index futures. The data used to train the model ranged from 4 January 1995 to 29 December 1995, containing 17'790 observations and they partitioned their data according to moneyness (at-the money, in-the-money, out-of-the-money, deep-in-the-money and deep-out-of-the-money) to train different models. Additionally they trained different subsets of models where the training data was kept in the original sequence by date or randomly sampled. The hyperbolic tangent function is used as activation function. They only used three input features (S, K and T), 2 to 3 hidden layers, and one output (C) in the output layer. The testing error of the neural network in terms of MSE was lower than that of Black-Scholes for in-the-money and

out-of-the-money options, but higher for at-the-money options.

Bennell and Sutcliffe (2004) were the first to study the performance of artificial neural networks in pricing UK options on European-style FTSE 100 call options ranging from 1 January 1998 to 31 March 1999 and also the first to compare it to the adapted Black-Scholes model including dividends. They split their data into in-the money and out-of-the money groups to train two distinct models and different subsets of the following input variables S, K, S/K, T, Q, r and σ to train multiple models with either C or C/K as output. Using C as an output their ANN was inferior for pricing both out-of-the money options and in-the-money options. Using C/K as an output some models were superior to the Black-Scholes model for out-of-the money options, but inferior for in-the-money options. When deep-in-the-money (DITM) and options with long maturities (3.4% of total volume) were excluded, the performance of the ANN became comparable to that of Black–Scholes for in-the-money options.

2.3 Other Machine Learning models

Park, Kim, and Lee (2014) compared the performance of 6 different models for the pricing of options on the KOSPI 200 index from January 2001 to December 2010. Overall on out-of-sample predictions the mean-squared error was the lowest for the Gaussian process model, followed by a support vector machine and neural network model with very similar performances, followed by the Merton model, then the Heston model and finally the Black-Scholes model with the highest MSE of all 6 models. For options with a moneyness ratio (S/K) bigger than 0.97 and a short-term and medium-term maturity the Merton model had the best performance according to MSE and with long-term maturity the Neural Network (NN) performed best. On options with a moneyness level between 0.97 and 1.13 and a short and medium-term maturity the performance was best with the SVR model and long-term with the NN model again. For short-term options with a moneyness ratio above 1.13, the performance was best with the NN model and medium and long-term with the Merton model.

Ivașcu (2021) compared the option pricing accuracy of some of the most popular Machine Learning algorithms with classical methods such as Black-Scholes and Corrado-Su on European call options on WTI crude oil future contracts. He trained 5 different models using different subsets of S, K, S/K, T, r and σ as features and if we take the best performing submodel of all models the highest accuracy using RMSE as a metric was achieved with Light Gradient boosting Machine (LGBM) and XGBoost (XGB) with a RMSE of 0.31, followed by Random Forest (RF)

with a RMSE of 0.39, NN with a RMSE of 0.54, Support Vector Machine (SVR) with a RMSE of 0.59, Genetic algorithms (GA) with a RMSE of 0.77, Coradu su (CS) with a RMSE of 0.91 and BS with a RMSE of 1.4.

Table 1 shows a more extensive overview of previous research about option pricing with neural networks and some of the main parameters.

Author(s)	Security	Data	Input	Output	Model	Activation	Measure
Malliaris and Salchenberger (1993)	S&P 100	January 1990-June 1990	S, K, r	C/P	NN with 4 hidden layer	sigmoid	MAD, MAPE, MSE
Hutchinson, Lo, and Poggio (1994)	S&P 500	1987-1991	S/K, T	C/K	NN with 4 hidden layer	sigmoid	R^2
Lajbcygier, Boek, Palaniswami, and Flitman (1996)	SPI	1992-1994	S/K, T, r, σ	C/K	NN with 4, 10 and 20 hidden layer	sigmoid	R^2, NRMSE, MAPE
Qi and Maddala (1996)	S&P 500	1994-1995	S, K, T, r, σ	C	NN with 5 hidden layer	sigmoid	R^2, ASE, MAE
Anders et al. (1998)	DAX	1992-1994	S/K, r, σ, T	C	NN with 1 hidden layer	tanh	R^2 RMSE, MAE, MAP
Yao et al. (2000)	Nikkei 225	1995	S, K, T	C	NN with 2 and 3 hidden layer	tanh	NMSE
Amilon (2003)	Swedish Stock Index	1997-1999	S/K, T, r	C/K	NN with 10, 12, 14 hidden layers	tanh and logistic	RMSE
Bennell and Sutcliffe (2004)	FTSE 100	1998-1999	S, K, S/K, T, Q, r, σ	C, C/K	NN with 10, 12, 14 hidden layer	tanh and logistic	PEs, MD, MAD, MPD, R^2, MSD
Han and Lee (2008)	KOSPI 200	2006	S/K, T	C/K	NN with 1 hidden layer and GPR	NA	MSPE
Park et al. (2014)	KOSPI 200	2001-2010	S/K, T	C/K	NN with 3 hidden layer	tanh	RMSE, MSPE
Liu, Oosterlee, and Bohte (2019)	European call options	Generated data	S/K, T, r, σ	C/K	NN with 4 hidden layer	ReLu	MSE, RMSE, MAE, MAPE
Pagnottoni (2019)	Bitcoin	2015-2018	S, K, T, r, σ	C, P	NN with 1 hidden layer	sigmoid	MAE, MAPE, MSE
Ivascu (2021)	WTI oil futures contracts	2017-2018	S, K, T, r, σ	C, C/K	Various models	NA	RMSE
Current work	S&P 500	2016-2020	S, K, T, r, σ	C, P	WDNN with 3 hidden layer	ReLu	MAE, MAPE, MDA, PEs, MSE

Table 1: Summary of main parameters in previous research

3 Data

3.1 Data source and preparation

The option prices and security prices were obtained from OptionMetrics provided by the Wharton Research Data Services (2021). Option metrics records the daily outcomes of all listed option contracts and corresponding security prices among other financial data. We obtained data from the best bid and best ask price of all options on the S&P 500 index since 1 January 1996 with the respective strike price and exercise date of each option. Furthermore we obtained the implied dividend yield and closing price of every trading day of the S&P 500 index since 1 January 1996. The US treasury yield rate used as risk-free rates was obtained from the US Department of the Treasury (2021).

3.2 Overview of data used for training, validation and testing of the models

The different data sets were prepared, merged and evaluated using R (R Core Team, 2021) and RStudio (RStudio Team, 2021) and the data.table (Dowle & Srinivasan, 2021), tidyverse (Wickham et al., 2019), and plotly (Sievert, 2020) packages. Although data since 1996 is available, the data used to train the model only ranges from 1 January 2016 to 31 August 2020, the validation set ranges from 1 September 2020 to 31 October 2020 and the test set ranges from 1 November 2020 to 31 December 2020, as the model accuracy did not benefit from training with "older" data. As we have a very high amount of observations, this split ($\approx$ 90% train, $\approx$ 5% validation and $\approx$ 5% test) allows us to train our model with more recent data from which the model pricing accuracy benefits, while still leaving enough observations for validation and testing of the model without any forward-looking bias. The overview of the train, validation and test split for both the WDNN model to price call options and the WDNN model to price put options is visible in Table 2 and a summary statistics of the whole data set used to train, validate and test the model is visible in Table 3. The option prices according to Black-Scholes have been computed with the same data according to the Black-Scholes formula including a dividend component visible in equation (1) and (2) of the Introduction.

Model	Train data range	Observations	%	Validation data range	Observations	%	Test data range	Observations	%
WDNN - CALLS	January 2016 - August 2020	6'923'085	≈ 90%	September - October 2020	367'948	≈ 5%	November - December 2020	355'159	≈ 5%
WDNN - PUTS	January 2016 - August 2020	6'922'960	≈ 90%	September - October 2020	367'948	≈ 5%	November - December 2020	355'159	≈ 5%

Table 2: Overview of train, validation and test split of data set

Variable	N	Mean	St. Dev.	Min	25%	75%	Max
S	13,846,045	2,714	356	1,829	2,468	2,940	3,508
K	13,846,045	2,405	651	100	2,025	2,840	5,200
T	13,846,045	0.284	0.442	0.003	0.052	0.296	3.000
σ	13,846,045	0.160	0.156	0.034	0.075	0.197	0.976
r	13,846,045	0.013	0.008	0.000	0.005	0.021	0.030
q	13,846,045	0.017	0.005	0.000	0.016	0.019	0.023
Best bid	13,846,045	248	389	0.000	3.4	328.6	3,403
Best offer	13,846,045	252	392	0.05	3.8	335.9	3,413
Bid-ask mean	13,846,045	250	390	0.025	3.6	332.2	3,405

Table 3: Summary statistics of the data used to train, validate and test the models

"S" represents the underlying security price, in our case the S&P 500 index. "K" represents the strike price of the option. "T" represents the time to maturity in years (maturity in days divided by 365). "r" represents the riskfree rate and was assigned by taking the US treasury yield which matches the maturity of the option on the respective trading day (for example the 3 month treasury yield of the respective day as risk-free rate for options with a maturity of less than 92 days, the 6 month treasury yield for options with a maturity of less than 183 days etc.). "σ" represents the realized historical volatility over the last 21 trading days (approximately one calendar month) and was computed by using a simple standard deviation calculation on the logarithm of the close-to-close daily total returns of the S&P 500 index (underlying security), assuming there are 253 trading days per year. "q" represents the continuous implied dividend yield of the S&P 500 index. It is assumed that dividends get payed continuously, according to continuously-compounded dividend yield. Furthermore a put-call parity is assumed by the data

provider to calculate the implied index dividend yield and the dividend rate "q" was lagged by one day to train the models as it only gets available on OptionMetrics at 9am of the next trading day in practice. The best bid variable contains the best, or highest, closing bid price across all exchanges on which the option trades. The best offer variables contains the best, or lowest, closing ask price across all exchanges on which the option trades. The "bid-ask mean" variable represents the arithmetic mean of the best bid and best offer price. S, K, T, σ, r, q will serve as features to train the WDNN model while the Bid-ask mean price is the target. The input features for the WDNN model were scaled by removing the mean and scaling to unit variance, while the target variable was not scaled. More information about the data source is available in the IvyDB "File and Data Reference Manual" from Option Metrics. The R Code used to prepare and merge the data into one tidy data set is available under *this Github link*.

4 Methodology

4.1 Rise of Neural Networks

Even though McCulloch and Pitts already developed the first Neural Network in 1943 (based on simple linear models motivated from a neuroscientific perspective), deep learning and other more advanced Machine Learning only started to really take off in recent years. The three main drivers behind this rapid development of ML in recent years are improvements in hardware, as central processing unit (GPU) and graphics processing unit (GPU) became significantly faster, data sets (quantity, availability and quality of data) and algorithmic advances. According to Chollet (2017) Machine Learning is not pure mathematics or physics where majors breakthroughs can be achieved with just a pen and piece of paper, but rather an engineering science also guided by experimental finding. Algorithmic innovations are therefore only possible when both the hardware and data is available to try new ideas or scale up old ideas. From the 1990s until the 2000s the main limitations for algorithmic advances were hardware and data, but luckily the internet took off during these years and graphic chips were developed for the needs of the gaming industry (Chollet, 2017, p. 20).

Hardware

In terms of hardware, companies like NVIDIA and AMD have been investing billions of dollar to develop faster GPUs (mainly with the gaming market in mind) and therefore indirectly subsidizing super-computing for the next generation of Artificial Intelligence (AI), as Deep Neural Networks (DNN), mainly consisting of many small matrix multiplications, are also highly parallelizable and therefore benefit incredibly from the development of these faster massively parallel chips (GPUs). In recent years, companies like NVIDIA have also started to invest and develop GPUs specifically for ML and in 2016 Google announced their tensor processing unit (TPU) project, developing a new chip design to run deep neural networks, allowing for 15–30X more throughput than contemporary CPUs and GPUs while being far more energy efficient than state-of-the-art GPUs. (Chollet, 2017, p. 21) Paid cloud platforms such as Google's GCP, Amazon's AWS, and Microsoft's Azure and even free ones such as Google Colab have further democratized AI, making large computing power available to almost everybody and the next big advancement in terms of hardware might come soon with quantum computing.

Data

In terms of data, equally enormous advances have been made in the last 20 years. Besides the exponential progress in storage capacity over the last few decades following Moore's law (1965), the rise of the internet was the real game-changer, making it possible to collect and distribute very large data sets, which can now be used to train Machine Learning models. Platforms like Kaggle have made more data sets available to the general public and the various public competitions set a great incentive for researchers, engineers as well as students to get into the field of Data Science and push for innovation and new best-practices. In addition to that, a deep wave of investments both from venture capital and even more internal investments from big tech companies such as Google, Amazon, Facebook, Microsoft, Apple, or Baidu has facilitated the development and innovation in the fields of AI, which has become central to the product strategy of most of these big tech companies (Chollet, 2017, p. 22). New digital devices, better cameras and sensors and the Internet of Things further increase both the availability and quality of data.

Algorithmic advances

In term of algorithms, the discovery of better activation functions for neural layers, better weight-initialization schemes, and better optimization schemes (such as RMSProp and Adam) allowed to reduce the problem of the feedback signal used to train NN fading away as the number of hidden layers increased. Over time even more advanced techniques like batch normalization, residual connections, depthwise separable convolutions were developed to help gradient propagation.

Hornik showed already in 1991 that any function can be approximated by a neural network with as few as a single hidden layer. It should therefore be possible to price our options with a neural network viewing the option price as a function of S, K, T, Q, r and σ. In contrast to the Black-Scholes model, we can avoid any potentially false assumptions about financial mechanics with a wide and deep neural network that learns from historical data instead. Two WDNN models will be trained, one for the pricing of call options and another one for the pricing of put options.

4.2 Wide and deep neural networks

The wide and deep neural network architecture was first introduced in a paper in 2016 by Google for recommender systems productionized and evaluated on their Google Play Store (Cheng et al.). Although used for recommender systems in the first introduction, the wide and deep

neural network architecture can also be used for regression tasks and Google open-sourced their implementation in TensorFlow, which we will use to train our model. This kind of network architecture allows our model to learn both deep pattern (through the hidden layers) and simple rules (through the short path) and has therefore some interesting properties for the pricing of derivatives. For a more visual representation of our network architecture please see Figure 1. It is also be possible to just send a subset of the features through the wide path and a different subset (overlapping possible) through the deep path, although this did not increase the accuracy of the models and was therefore not applied in the final configuration.

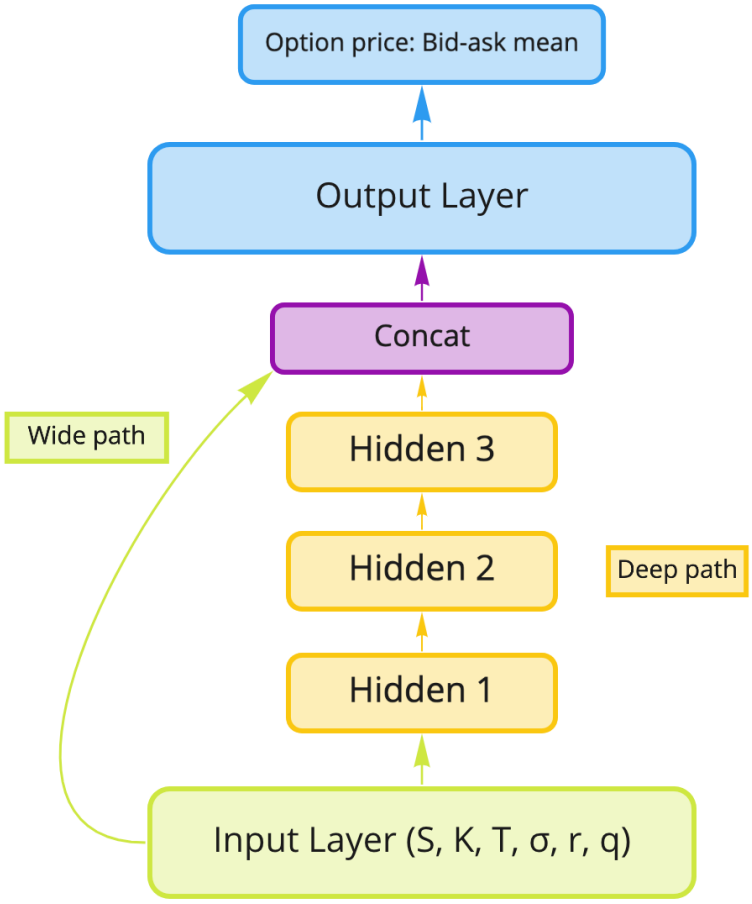


Figure 1: Architecture of our wide and deep neural network

4.3 Network architecture and hyperparameter tuning

Finding the right network architecture and hyperparameter tuning can sometimes be more of an art than a science in itself. We arrived at the final configuration of our WDNN through extensive

manual experimenting, prototyping and cross-validated randomized hyperparameter searches using RandomizedSearchCV with scikit-learn (Pedregosa et al., 2011). Overall we experimented with the number of hidden layers (1-10), number of neurons per hidden layer (100-1000), learning rates using different learning rate schedulers (piecewise constant scheduling, performance scheduling, and exponential scheduling), different optimizer (SGD, Adagrad, RMSprop, Adam and Nadam), different activation functions (ReLu, leaky ReLu, SELU, and ELU) and different initialization techniques (Glorot, He, and LeCun initialization).

Liu et al. (2019) found that drop-out and batch normalization did not improve accuracy when pricing options with Artificial Neural Network (ANN) and we can confirm this as drop-out and batch normalization did not improve the accuracy of our models either. They believe that this could be because option prices are sensitive to all inputs compared to sparse features in image classification for example, where these techniques usually perform well.

The application of Max-Norm Regularization in each hidden layer on the other hand did slightly improve the accuracy of our model. Max-norm regularization enforces an absolute upper bound on the weight of each neuron such that $\| w \|_2 \leq$ x, x being the max-norm hyperparameter and $\| \cdot \|_2$ the l_2 norm. Instead of adding a regularization loss term to the overall loss function, $\| w \|_2$ is computed after each training and only rescaled if necessary. The lower the "x", the higher the amount of regularization and the lower the risk of overfitting. Max-norm regularization can also reduce the unstable gradients problems when not using Batch normalization. (Géron, 2019, p. 370-371)

Besides Max-norm Regularization, the main technique we used to avoid overfitting is an early stopping callback. Early stopping is one of the most frequently used regularization techniques used in deep learning due to its effectiveness and simplicity (Bengio, Goodfellow, & Courville, 2017, p. 246-247). Early stopping interrupts training when no improvement is measured on the validation set error anymore for a predefined amount of epochs and restores the weights of the model with the lowest validation set error at the end of training (weights only get saved every time the validation set error improves). This method enables us to set a higher number of epochs for training without wasting too much time and computational power for training, while also reducing overfitting, as training will automatically stop after some time when there is no more progress on the validation set and only the "best" weights get saved.

In the final configuration we maintained a wide and deep neural network using three hidden layers with 400 neurons at each layer, a learning rate of 0.001, Adam as optimizer, a rectified

linear unit (ReLu) as activation function and Glorot uniform for the initialization of the weights. S, K, T, Q, r and σ served as input for the model and we used one output node predicting the equilibrium of the bid and ask price of the option as output using yet again ReLu (as option premiums are never negative) as activation function. The input variables were scaled by removing the mean and scaling to unit variance, meaning we removed the mean (of the train data set) and divided by the standard deviation (of the train data set) for all input features for each observation. The target was not scaled. The output node uses a concatenation of the initial input and the output of the third hidden layer as input for the pricing of the premiums of the options on the S&P 500 index.

The use of Adam has especially improved the accuracy and training speed of our models. The Adam optimizer is a first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments and was first introduced in 2014 (Kingma and Ba). The used Adam optimizer is straightforward to implement, computationally efficient, has little memory requirements, is invariant to diagonal rescaling of the gradients and works very well on problems that are large in data (like in our case) and/or parameters. Adam can be seen as a combination of RMSprop and momentum with some distinctions. First the momentum is incorporated directly as an estimate of the first order moment with exponential weighting of the gradient and secondly Adam includes bias corrections to the estimates of both the first-order moments and the second-order moments to account for their initialization at the origin. (Bengio et al., 2017, p. 308-309) As already mentioned we used ReLu as activation function, which is the default recommendation for modern neural networks defined by the activation function $\sigma(x) = \max(0, x)$ and visible in figure 2.

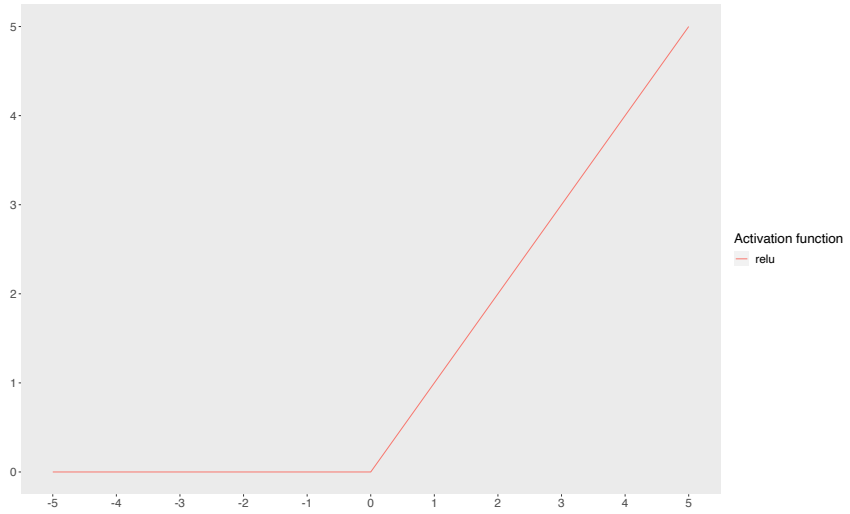


Figure 2: Visualisation of ReLu function

When we apply the ReLu function on the output of a linear transformation, we reach a nonlinear transformation, although it remains very close to a linear transformation, being a piecewise linear function with two linear pieces (Bengio et al., 2017, p. 174-175). The finding of Glorot and Bengio (2010) that the the previously frequently used logistic sigmoid activation function is not suited for deep learning with random initialization because of it's mean value, which has a tendency to drive the top hidden layer into saturation, has played a big role to overcome the unstable gradients problem for deeper networks, partly due to a poor choice of activation function in combination with random initialization.

The non-linearity of the ReLu function in combination with glorot uniform initialization is therefore beneficial for our model in comparison to older research were mostly sigmoid and tanh were used as activation function. The ReLu function preserves many of the properties that make linear models easy to optimize while at the same generalizing well (Bengio et al., 2017, p. 175).

Even though scaled exponential linear unit (SELU) and leaky rectified linear unit (leaky ReLu) are said to often outperform ReLu (Géron, 2019, p. 733), as ReLu can suffer from the dying neuron problem, the use of leaky ReLu with a similar configuration or the use of SELU with LeCun initialization did not improve the accuracy of the models. An overview of the range of hyperparameters tested and the configuration of the final model is visible in Table 4.

Parameters	Range	Final Configuration
Activation	ReLu, leaky ReLu, SELU, ELU	ReLU
Hidden Layers	1-10	3
Neurons	100-1000	400
Initialization	Glorot uniform, He uniform, LeCun normal	Glorot uniform
Batch normalization	Yes, No	No
Dropout rate	0.0-0.5	0.0
Max-norm regularization	0.5-3	1
Optimizer	SGD, RMSprop, Adagrad, Adam, Nadam	Adam
Learning schedulers	None, Exponential, Piecewise constant, Performance	None
Learning rate	0.1-0.00001	0.01
Batch size	500-20'000	10'000

Table 4: Range of hyper-parameters tested for optimization and final configuration

We arrived at these hyperparameters and this model configuration through extensive prototyping on the call options and the same model configuration and hyperparameters were used for both the pricing of call and put options and only the respective target variable and train, validation and test data changed for the training of each model. The Python code to train the models using the keras (Chollet et al., 2021) interface and TensorFlow (Abadi et al., 2021) is available under *this Github link*.

4.4 Trial and error

While the section above explains the final models used, this section addresses some of the things that have been tried out, but unfortunately did not work out as well as expected. We also trained models with three output nodes, trying to predict the "Best bid", "Best ask", and "Bid-ask mean" price of each option at the same time, instead of just predicting the equilibrium price. While the performance of these models was still good, it was slightly less accurate than the model with just one output node and training took much longer.

Furthermore we tried to increase the model's accuracy by adding the skew of the the S&P 500 daily returns (N=21) as an additional feature, although this unfortunately did not improve the models accuracy either. In the early stages of prototyping we also tried out other Machine Learning algorithms such as Regression Decision Trees, random Forest, XGBoost and Support

Vector Machines, but as the accuracy of these models did not come close to that of the WDNN model, the focus was set on the further development of the WDNN model. From all the other ML models tested XGBoost seemed as the most promising after the WDNN.

We also tried to predict the simple pricing error of the Black-Scholes model with a WDNN, DNN and XGBoost and adding/subtracting it to the price given by the Black-Scholes model instead of directly predicting the option premium, but it turned out that this approach led to a lower overall accuracy than directly pricing the option premium. Moreover we tried to improve the models accuracy with some feature engineering and by taking different subsets of the Black-Scholes inputs as features or adding the moneyness of the option (S/K ratio) or the Black-Scholes price as an additional feature, but none of these steps improved the accuracy of the model. To further improve the WDNN architecture we also tried passing different subsets of features through the wide and deep path, although we were not able to find a configuration that systematically has a better accuracy than the finally used model configuration visible in Figure 2 and Table 4.

5 Results

5.1 Overall accuaracy

To check whether the pricing accuracy of the WDNN is superior to that of the Black-Scholes model, we will compare the accuracy through various metrics such as the Root Mean Square Error (RMSE), Mean Square Error (MSE), Median Absolute Deviation (MDA), Mean Absolute Error (MAE), and the Absolute Error (AE). Because Mean Absolute Percentage Error (MAPE) calculation is impossible for option prices at 0 and unstable for option prices close to 0 and outliers, we will remove all options with a premium of less than 1$ from our data for evaluation and use the Symmetric Mean Absolute Percentage Error (MAPE) and the Mean Absolute Scaled Error (MASE), which overcome some of the shortcomings of the MAPE, as additional metrics. Another useful measures are the PEX% intervals, which give the percentage of price predictions within ±X% of the actually observed equilibrium market price for each model.

For robustness the accuracy comparison will solely focus on the results of the pricings on the out-of-sample test data set ranging from November to December 2020, which the model has never seen before in order to avoid any forward-looking bias. All of the plots and accuracy measures in the following sections therefore always relate to the data and accuracy on the test set. The test data contains 329'108 observations for the call options and 302'570 observations for the put options (after removing the options with a premium of less than 1$).

As it is visible in Table 5, the accuracy of the WDNN models is significantly better than that of the Black-Scholes model across all measures. The R-Code for the evaluation of the results is available *on Github again*. Furthermore all the visualizations are available in better quality and size *under this link*, as the size upload limit makes it impossible to include all of them in full resolution inside this paper.

Model	RMSE	MSE	MDA	MAE	MASE	BIAS	MAPE	SMAPE	PE1	PE5	PE10	PE20
BS - CALL	41.85	1751.18	13.14	24.95	46.33	21.19	20.31%	20.95%	28.34%	52.70%	65.87%	75.31%
WDNN - CALL	9.00	80.95	5.62	6.97	12.95	1.81	9.20%	7.12%	41.42%	78.05%	86.33%	92.06%
BS - PUT	39.49	1559.35	13.05	24.06	1.17	17.94	56.86%	103.1%	9.97%	19.50%	25.31%	33.37%
WDNN - PUT	9.27	85.99	3.65	6.21	0.30	4.47	21.05%	31.06%	13.86%	36.71%	52.67%	70.36%

Table 5: Overall accuracy comparison of Black-Scholes model vs WDNN model

As we can see in Table 5, the mean or median pricing error is significantly lower for the WDNN model for both call and put options, across all measures, both in relative and absolute terms. Some of the error measures might seem a little bit high for both models, but this is due to the fact that we have some big outliers in our data sets with options with maturities up to 3 years that are very deep-in-the-money or very deep-out-of-the-money, which were not removed from the data set and therefore negatively affect the accuracy measures of both the WDNN model and the Black-Scholes model.

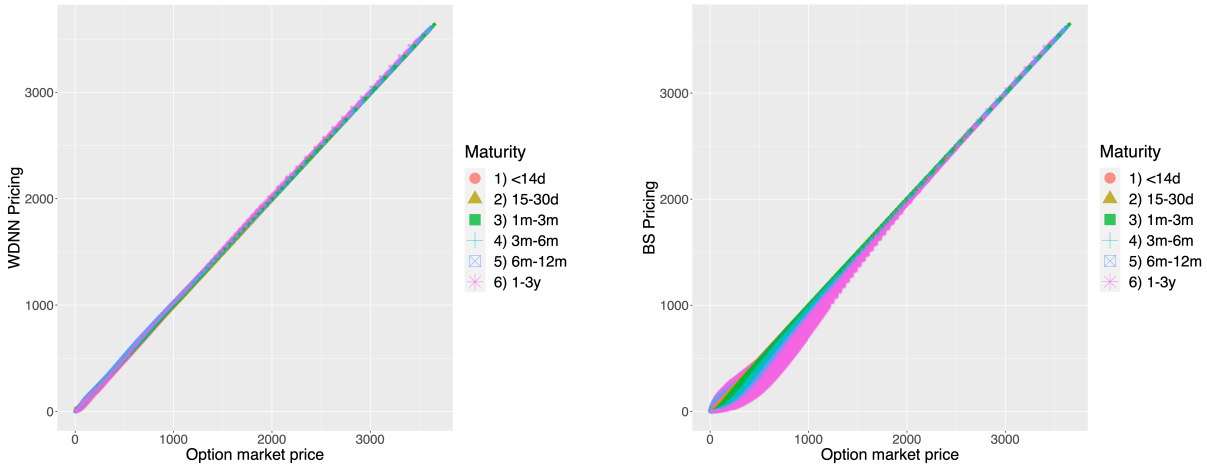
Despite these outliers and a naive historical estimation of volatility, the WDNN models was able to predict 41.24% of the observations in the test set for call options within ±1% of the actually observed market equilibrium price vs 28.34% for the Black-Scholes model. For the put options the accuracy was significantly worse with only 13.86% of the pricings within ±1% of the actual equilibrium market price for the WDNN model vs 9.97% for the Black-Scholes model. If we increase the $\pm X\%$ interval, the outperformance of the WDNN model becomes more significant. From the bias variable we can see that the Black-Scholes model has a strong bias of underestimating the real value of options for both calls and puts and even though this bias gets corrected considerably by the WDNN models, some bias of underestimating the price remains also with the WDNN models.

Model Variable	N	Mean	St. Dev.	Min	25%	75%	Max
CALLS - BS ERROR	329,108	−21.186	36.088	−357.117	−31.294	−1.796	88.151
CALLS - WDNN ERROR	329,108	−1.814	8.812	−45.097	−7.318	3.599	47.903
CALLS - BS APE	329,108	0.203	0.446	0.000	0.008	0.196	12.534
CALLS - WDNN APE	329,108	0.092	0.393	0.00000	0.005	0.041	22.341
CALLS - BS SE	329,108	1,751.181	6,139.364	0.000	18.535	1,066.278	127,532.600
CALLS - WDNN SE	329,108	80.946	135.867	0.000	7.119	97.586	2,294.707
PUTS - BS ERROR	302,570	−17.939	35.179	−321.485	−28.544	−1.175	119.549
PUTS - WDNN ERROR	302,570	−4.474	8.122	−70.105	−7.490	−0.324	24.065
PUTS - BS APE	302,570	0.569	0.422	0	0.1	1.0	10
PUTS - WDNN APE	302,570	0.211	0.306	0	0.03	0.2	20
PUTS - BS SE	302,570	1,559.349	5,217.113	0.000	20.207	992.250	103,352.500
PUTS - WDNN SE	302,570	85.986	212.521	0.000	2.744	70.084	4,914.697

Table 6: Summary statistics of the the simple error, APE, and squared error

In Table 6 we can see the summary statistics of the simple error, the absolute percentage error (APE) and the squared error (SE) for the Black-Scholes and the WDNN models for both call and put options. We can see that not only the mean values of the WDNN are much closer to 0 than those of the Black-Scholes model across all measures, but also that the interquartile ranges are much smaller and closer to 0 for the WDNN models. Only when looking at the Max APE, we can see that some outlier pricing errors are bigger for the WDNN than for Black-Scholes, although this is relativized when looking at the upper quartile of the APE, which is around 4% for the WDNN vs 20% for BS for the calls and around 20% for puts for the WDNN vs 100% for BS.

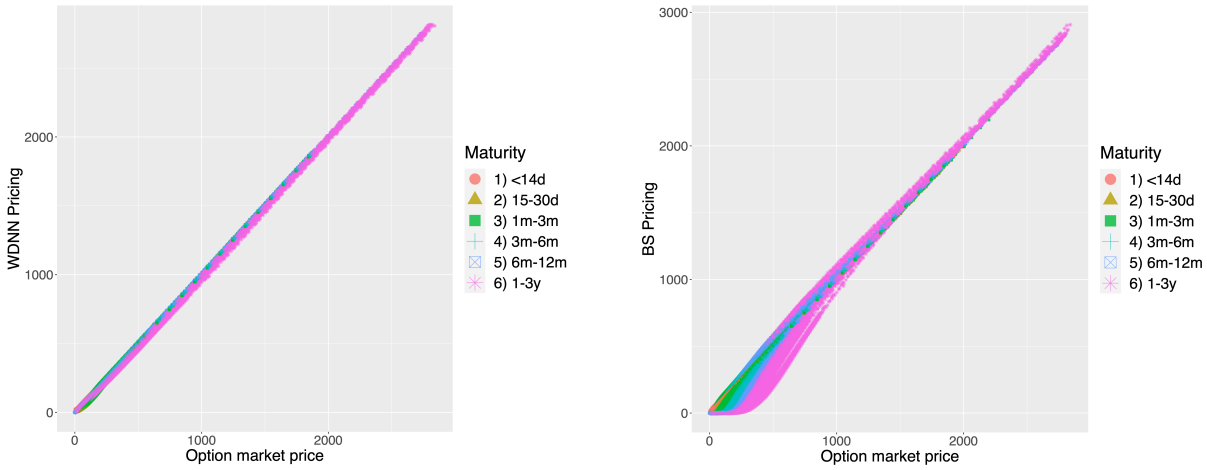
The superior accuracy of the WDNN model in both pricing call and put options is also visible in the scatter plots in Figure 3 and 4, where we have the historically observed option market prices of the test set on the x-axis and the the pricings of the WDNN model and Black-Scholes model on the y-axis. As can be seen, the WDNN models fits the actual market prices much better than the Black-Scholes model and even though we have some overplotting, it remains visible, that the Black-Scholes model pricing error has a tendency to increase for options with longer maturities. The next section will address this in more detail.



(a) WDNN model - Calls

(b) Black-Scholes model - Calls

Figure 3: Calls - Scatter plot fit with historical market prices and pricings of models



(a) WDNN model - Puts

(b) Black-Scholes model - Puts

Figure 4: Puts - Scatter plot fit with historical market prices and pricings of models

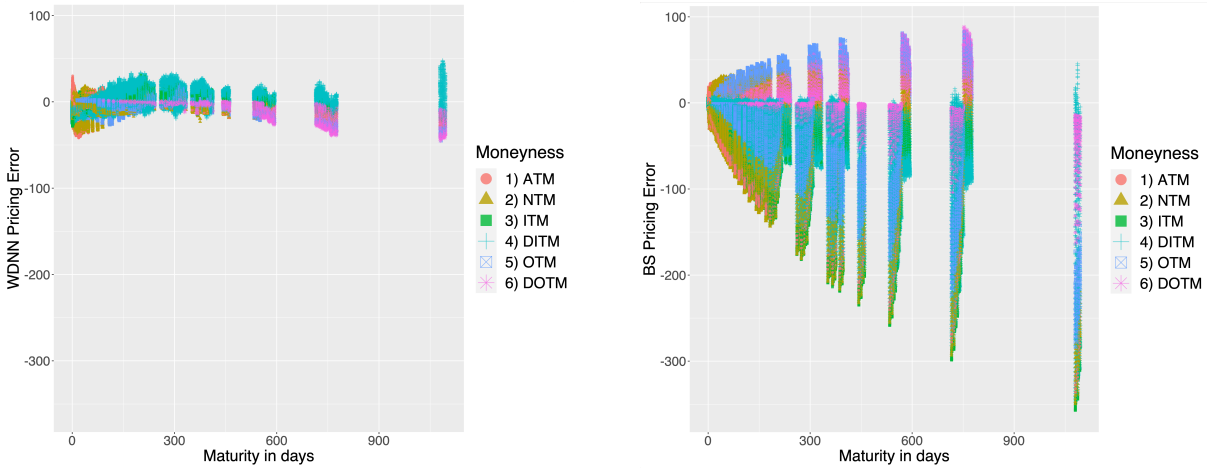
5.2 Accuracy across different levels of moneyness or maturity

To analyze the accuracy across different maturities we have labeled the maturity of the options in the test set into the following six groups:

1) <14 days
2) 15-30 days
3) 1-3 months
4) 3-6 months
5) 0.5-1 year
6) 1-3 years

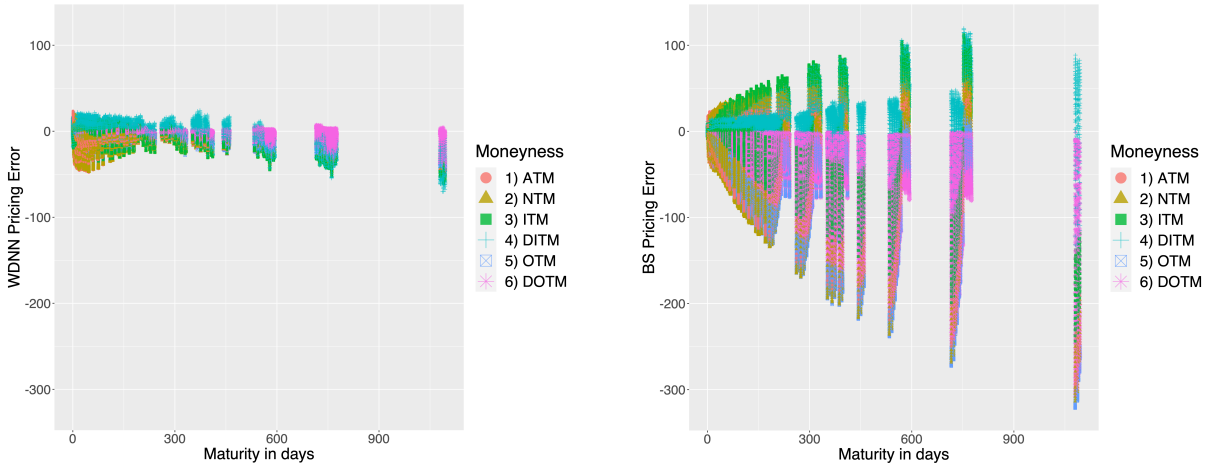
In order to also compare the accuracy through different levels of moneyness we will compare the accuracy for at-the-money (ATM), near-the-money (NTM), in-the-money (ITM), deep-in-the-money (DITM), out-of-the-money (OTM), and deep-out-of-the-money (DOTM) options. For the moneyness of the options we have calculated the ratio S/K for the call options and K/S for the put options and labeled them according to the following criteria (numbers relate to the ratio S/K for call and K/S for put options):

1) ATM = 0.99-1.01
2) NTM = 0.95-0.99 & 1.01-1.05
3) ITM = 1.05-1.15
4) DITM > 1.15
5) OTM = 0.85-0.95
6) DOTM < 0.85



(a) WDNN model - Calls (b) Black-Scholes model - Calls

Figure 5: Calls - Pricing error over options with different maturities



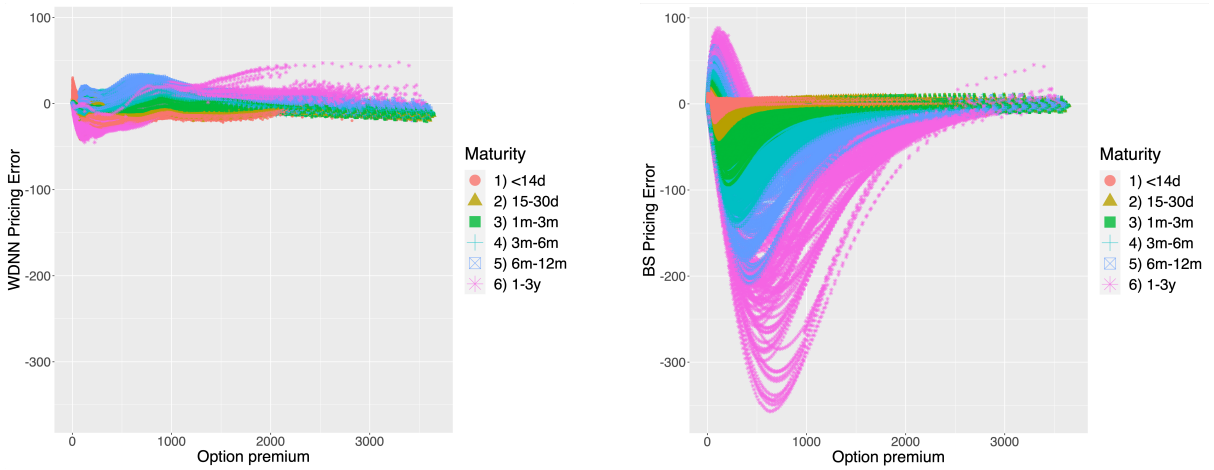
(a) WDNN model - Puts (b) Black-Scholes model - Puts

Figure 6: Puts - Pricing error over options with different maturities

If we look at the scatter plot of the pricing errors across different maturities for the call options in Figure 5, we can reconfirm that the Black-Scholes model has a tendency to underestimate the option premium and this tendency increases as the maturity increases. The pricing errors of the Black-Scholes model range from -357 to +88, while they only ranges from from -48 to +45 for the WDNN model. It can also been in Figure 5, that for call options that are deep-in-the-money, the WDNN model tends to overestimate the option premium, especially for options with longer maturities, while it tends to underestimate them for DOTM and OTM options. The

Black-Scholes model most underestimates the value of NTM, ITM, and OTM options and for short maturities also ATM options.

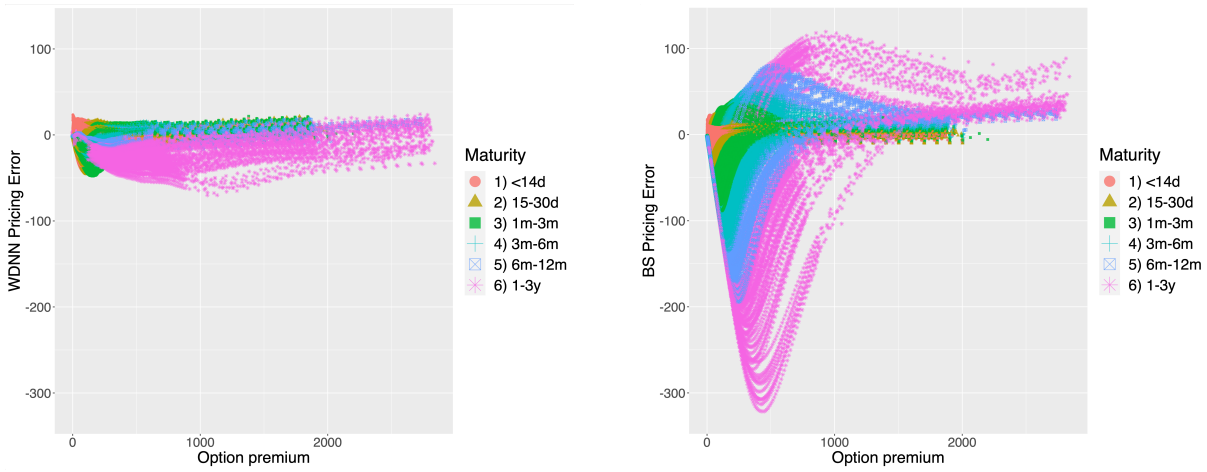
If we look at the same plot for the put options in Figure 6, we can see that the WDNN model also start to have the problem of generally underestimating the value of put options with longer maturities, although to a much lower extent than Black-Scholes with pricings errors down to -70 vs down to -321 for the Black-Scholes model. We can see that for DITM put options with a maturity of about 3 years, Black-Scholes overestimates the option premium, while the WDNN underestimates them. While the Black-Scholes model generally overestimates the value of DITM put options, the WDNN model only does this for option with a maturity of less than 1.8 years and starts to underestimate the premiums for options with a longer maturity than that. For a maturity of less than 6 months most of the big pricing errors of the WDNN model happen for ATM, NTM, and ITM options.



(a) WDNN model - Calls

(b) Black-Scholes model - Calls

Figure 7: Calls - Pricing error over options with different market prices



(a) WDNN model - Puts

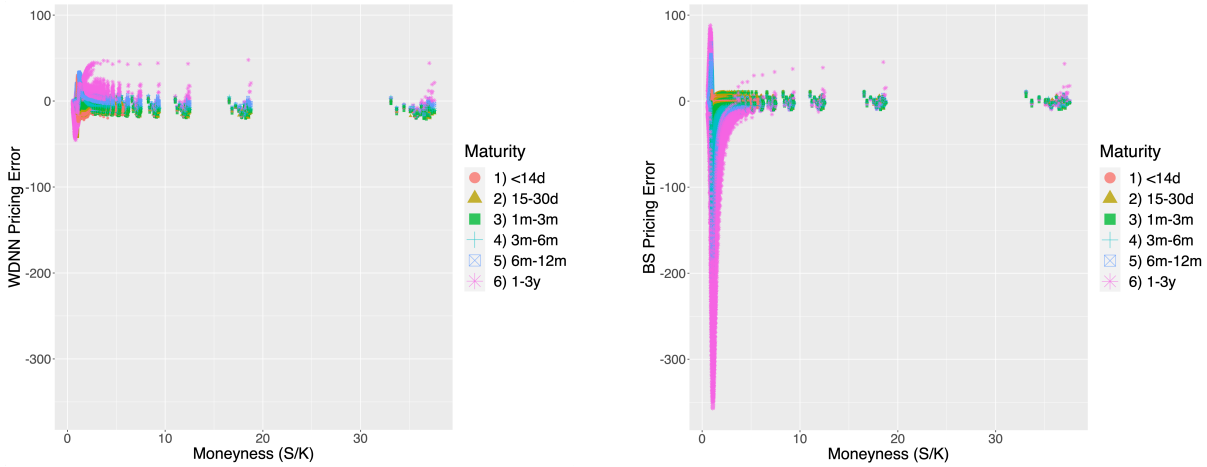
(b) Black-Scholes model - Puts

Figure 8: Puts - Pricing error over options with different market prices

When looking at the plot for call options across different premiums visible in Figure 7, an interesting pattern is observable for the pricing errors of the Black-Scholes model. Up to a certain option premium the pricing errors starts increasing for more expensive options and only after a certain thresholds the pricing error starts decreasing again. The longer the maturity of the option, the higher the market price thresholds the option needs to reach until the pricing error starts decreasing again. For the WDNN model we can see that it has the problem of heavily overestimating the option premium for some of the options which have a maturity of less than 14 days and an actually observed very low market value, although the Black-Scholes model also has a lot of trouble pricing this group correctly. Furthermore the WDNN has tendency of underestimating the premiums for options with maturities between 1 and 3 years and a market value below 1'000$, while overestimating the premiums of the same maturity group with a higher market price.

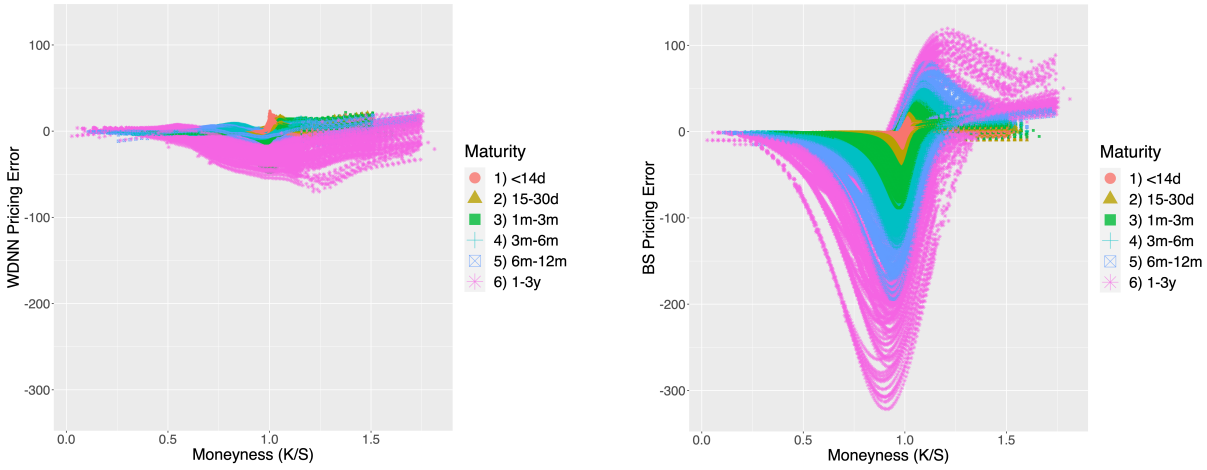
Looking at the same plot for the put options in Figure 8, we can see the same pattern as already seen in Figure 7 for the Black-Scholes model. The WDNN model has also yet again the problem of overestimating the premiums of ATM options with a maturity of less than 14 days and an actually observed very low market price. As option market premiums increase, the WDNN model tends to underestimate the market prices of put options with maturities between 1 and 3 years. At a market price of around 220$, a lot of options with a maturity between 1 and 3 months get heavily underestimated by both models, although the underestimation of the

Black-Scholes model is much larger.



(a) WDNN model - Calls (b) Black-Scholes model - Calls

Figure 9: Calls - Pricing error over options with different levels of moneyness



(a) WDNN model - Puts (b) Black-Scholes model - Puts

Figure 10: Puts - Pricing error over options with different levels of moneyness

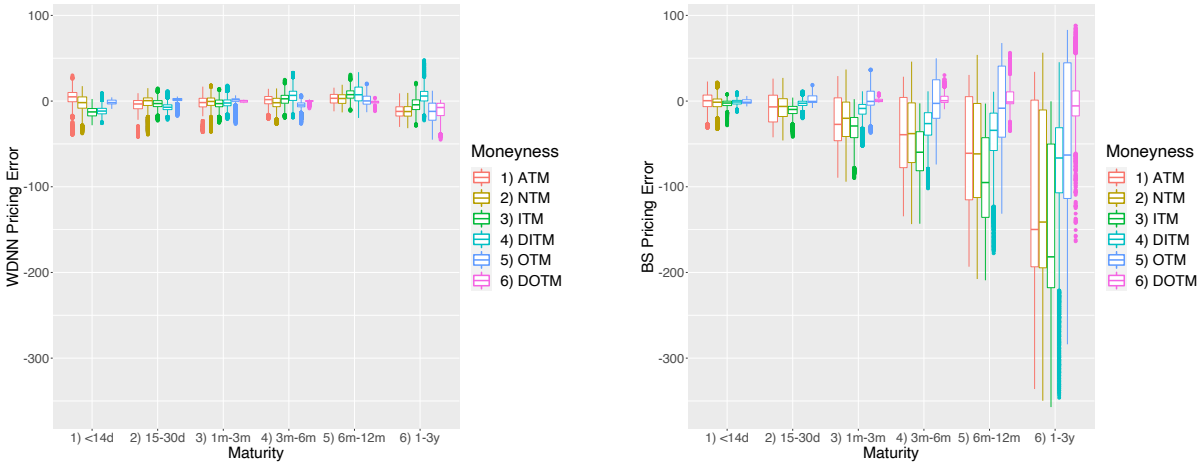
Looking at Figure 9, we can see that some of the biggest pricing errors for both the Black-Scholes and the WDNN model for call options happen at a maturity level (S/K) closer to 1 and this pricing error yet again becomes bigger as the maturity increases, although it is difficult to see more details as the range of S/K is very large on the x-axis.

For the put options we can see in Figure 10 that the peak of underestimating the option premium is reached for Black-Scholes as the options start to move from ATM to OTM and

the pricing error decreases again as the options get very-deep-out-of-the money. For another group of put options the peak of overestimating the price is reached for the Black-Scholes model for ITM options and the pricing error starts decreasing again for very DITM options. For the WDNN model some of the biggest pricing errors happen for ATM options with a maturity of less than 14 days and for ITM and DITM options with longer maturities.

5.3 Calls - Accuracy across moneyness for grouped maturities

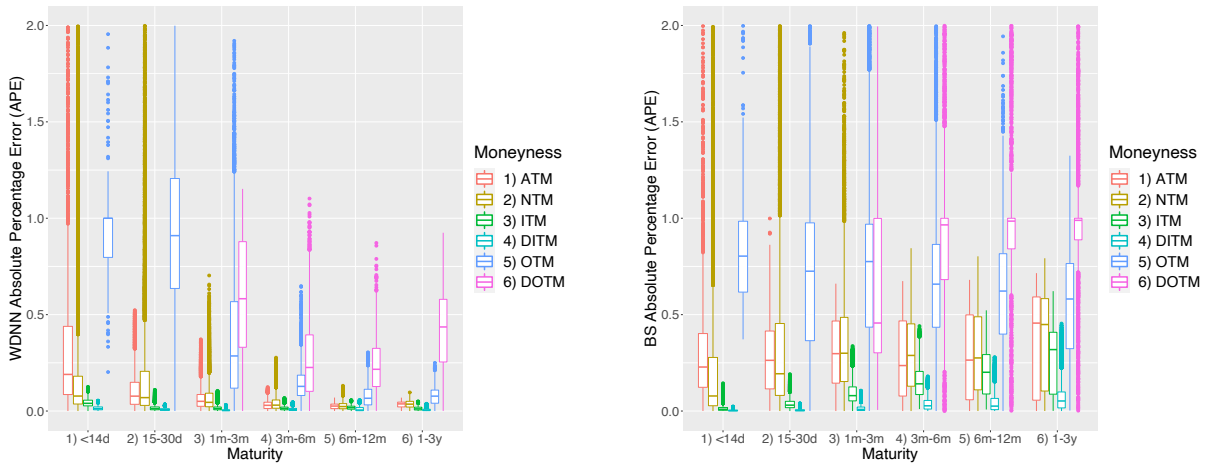
The boxplots of the simple error and APE distribution visible in Figure 11 and Figure 12 give a very good overview of the absolute and relative pricing accuracy while considering the moneyness and maturity of the option at the same time, having grouped the options in a total of 36 groups, composed of a combination of 6 different maturity groups times 6 different levels of moneyness. The next subsections will always refer to Figure 11 when talking about the pricing error in absolute terms (simple error boxplot) and to Figure 12 when talking about the relative error (APE boxplot). As the APE is very unstable for outliers we cut off the y-axis at 2.0 (200%) for the call options for better visibility, despite the fact that the APE of some outliers go up to over 1500% for both models.



(a) WDNN model - Calls

(b) Black-Scholes model - Calls

Figure 11: Calls - Simple error distribution across different moneyness and maturity groups



(a) WDNN model - Calls

(b) Black-Scholes model - Calls

Figure 12: Calls - APE distribution across different moneyness and maturity groups

5.3.1 Maturity less than 14 days

For a maturity of less than 14 days we can see that the median error is actually closer to 0 for the BS model than the WDNN model for call options which are ATM, ITM, and DITM looking at Figure 11. When we look at the same boxplot for the absolute percentage error, we can see that the biggest outliers are for ATM and NTM options, with some of the APE pricing errors going up to over 1500% for both models. For the OTM options both the BS and WDNN model seem to have a lot of trouble pricing theses options correctly when we look at the median APE

5.3.2 Maturity between 15 and 30 days

For options with a maturity between 15 and 30 days the simple pricing error of the WDNN model compared to the BS model starts to become much better. For ATM, and NTM call options the median error and median APE is much closer to 0 for the WDNN model than for the BS model, while also having a much smaller interquartile range. While the median error for ITM options is closer to 0 for the WDNN model, it is closer to 0 for the BS model for DITM options. When looking at the OTM options we can yet again see that both models have a lot of trouble pricing this group of options accurately.

5.3.3 Maturity between 1 and 3 months

When looking at call options with a maturity between 1 and 3 months the WDNN model really starts to shine compared to the BS model looking at the simple error distribution. The median error of the BS model for ATM, NTM, ITM, and DITM is much further away from 0 for the BS model than for the WDNN model and has a much larger interquartile range. The median APE is very close to 0 for the WDNN model for these options, while being still rather high for the BS model. For OTM both models still have trouble pricing them correctly, although the WDNN model has a much lower median APE and less high APE outliers. For DOTM call options the median APE is closer to 0 for the Black-Scholes model.

5.3.4 Maturity between 3 and 6 months

When looking at the simple error distribution for call options with a maturity between 3 and 6 months, the pattern observed for the options with a maturity between 1 and 3 months repeats and intensifies itself. If we look at the APE distribution we can see that the median APE is now significantly lower for the WDNN across all levels of moneyness, while also having a much smaller interquartile range.

5.3.5 Maturity between 6 and 12 months

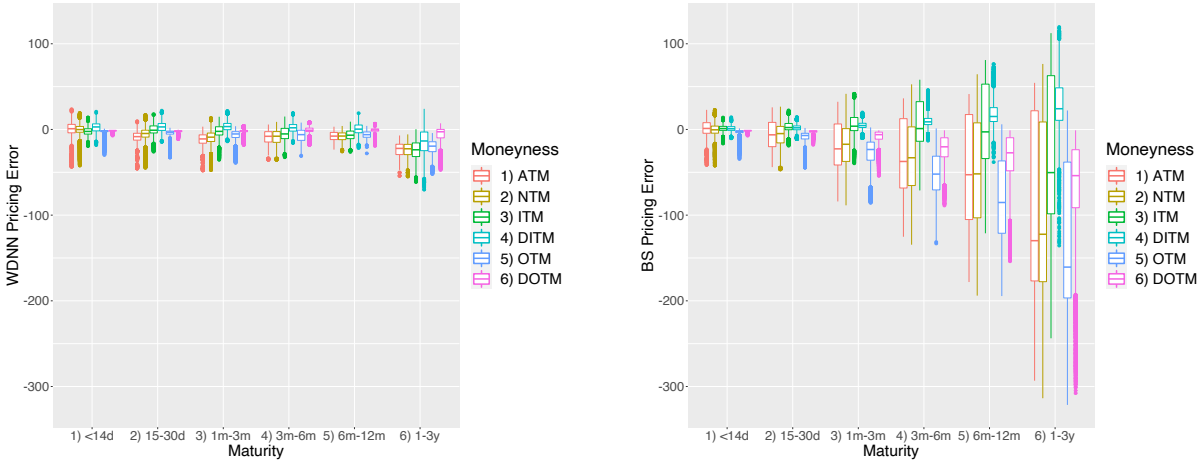
With call options of a maturity between 6 and 12 months we can yet again see a repetition and intensification of the pattern observed with the previous maturity group. The median pricing errors for BS diverge farther from 0 for the BS model while the interquartile ranges increases. Looking at the APE distribution we can see that the WDNN is very accurate for ATM, NTM, ITM, and DITM options. While the accuracy on the OTM options has further improved in terms of APE, the WDNN still has significant trouble pricing the DOTM options, although it does a lot better than the BS model.

5.3.6 Maturity between 1 and 3 years

For call options with a very long maturity between 1 and 3 years, the pricing error for BS is the highest in both absolute and relative terms for ATM, NTM, ITM & DITM options. For OTM, and DOTM the pricing errors also start to become very large in absolute terms and similar to the previous group in relative terms. It is also interesting to see that the WDNN starts

to underestimate the option premiums in this group for ATM, NTM, ITM, OTM, and OTM options, while overestimating them for DITM options. The BS model generally underestimates the option premiums, with the underestimation being the largest for ATM, NTM, ITM, DITM and OTM options.

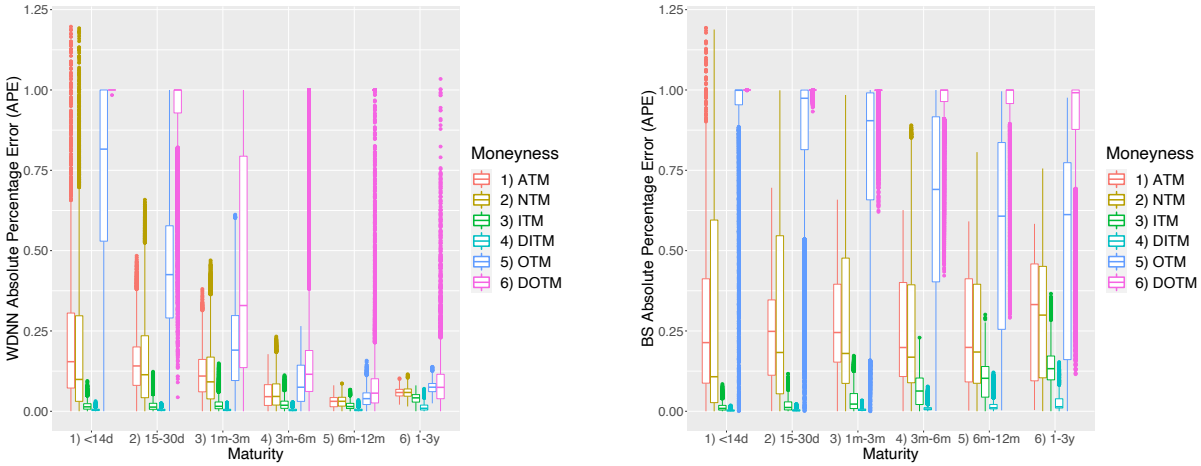
5.4 Puts - Accuracy across moneyness for grouped maturities



(a) WDNN model - Puts

(b) Black-Scholes model - Puts

Figure 13: Puts - Simple error distribution across different moneyness and maturity groups



(a) WDNN model - Puts

(b) Black-Scholes model - Puts

Figure 14: Puts - APE distribution across different moneyness and maturity groups

5.4.1 Maturity less than 14 days

When looking at the error distribution for put options with a maturity less than 14 days, we can see that the distribution is rather similar for both the BS and WDNN model. Like with the call options we have a lot of APE outliers again for ATM and NTM options. The interquartile APE range is narrower for ATM and NTM options with the WDNN model. With both the WDNN and BS model the biggest median absolute percentage pricing errors happen for OTM and DOTM options, while the median APE is lower for the WDNN model for OTM options and about the same as BS for DOTM options.

5.4.2 Maturity between 15 and 30 days

If we look at the maturity group between 15 and 30 days, we can see that the interquartile error range gets much bigger for ATM and NTM options for the BS model, while being narrower for the WDNN model, although this comes at the cost of more outliers. The median APE of the WDNN is much lower for ATM, NTM, and OTM options, while being about the same for ITM, DITM, and DITM put options. Both models struggle the most with the accurate pricing of OTM and DOTM options, although the WDNN does a lot better for OTM options in terms of APE.

5.4.3 Maturity between 1 and 3 months

With the group of 1 to 3 months the tendency of the BS model to underestimate the value of ATM, NTM, OTM, and DOTM put options further increases. Now the median APE is significantly lower with the WDNN model for ATM, NTM, OTM, and DOTM options, while it is unfortunately not really visible for ITM, and DITM options in this series due to the strong concentration around 0% for both the WDNN and the BS model.

5.4.4 Maturity between 3 and 6 months

For put options with a maturity between 3 and 6 months the median APE is extremely close to 0 for the WDNN models for ATM, NTM, ITM, DITM options and also closer to 0 compared to the previous maturities for OTM and DOTM options. If we compare it to the APE's of the BS model, we can see that the WDNN is especially more accurate for ATM, NTM, OTM, DOTM options.

5.4.5 Maturity between 6 and 12 months

For the put options with a maturity between 6 and 12 months for ATM, NTM, OTM, and DOTM options both models have a tendency to underestimate the price, although the BS model does this to a much greater extent. The median error for ITM options is pretty close to 0 for the WDNN, while the BS model has a tendency to overestimate the value of these options. Looking at the median APE we can see that the WDNN model is much more accurate, especially for OTM and DOTM options where the BS model has a median APE of about 68% and 100%, while only being around 3% and 6% for the WDNN model.

5.4.6 Maturity between 1 and 3 years

For the group of options with a maturity between 1 and 3 years, the median APE of the BS model further increases for ATM, NTM, ITM, and DITM options. For ATM, and NTM options the median APE of the BS model is around 37% and 33% compared to a median APE around 6% for the WDNN model. While it is the norm to heavily underestimate the DOTM options for the BS model, this only happens to such an extent in some outlier cases for the WDNN model.

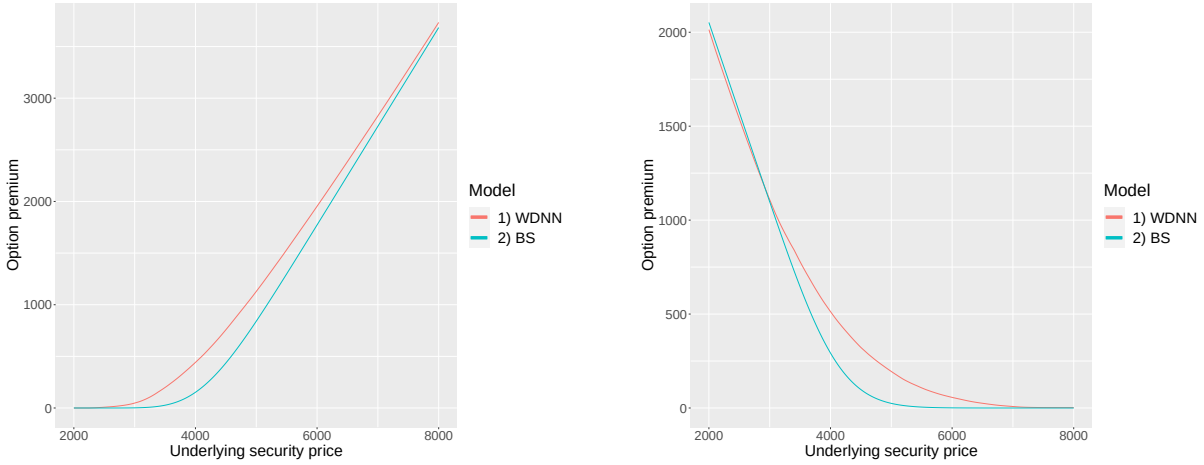
5.5 Option premium as a function of the underyling security price

In the previous section we have seen at which levels of moneyness and for which maturities the BS model most heavily over- or underestimates the option price. We have also seen that the WDNN model is able to correct this pricing error considerably in many cases. When we fix all of the Black-Scholes input variables except ”S” (security price) and generate a sequence of ”S” in a synthetic data frame we use the Black-Scholes model and the WDNN model to plot the premium of the option as a function of the price of the underlying security and see how the pricing function of the WDNN model diverges from that of Black-Scholes.

As we have seen in Figure 11 and 12, some of the biggest price underestimations for the Black-Scholes model happen for options with a maturity between 1 and 3 years. For this maturity group the BS model strongly underestimates the price of ITM call options and OTM put options, especially during times of low volatility, although this is partially also due to the fact that we fixed N at 21 for the volatility calculation. In these cases, the BS model pricing errors goes down to around -350 for the BS model in multiple cases, while only being around -20 for the WDNN model for the calls and down to around -320 for the puts with the BS model while only being

around -40 for the WDNN model.

If we fix K (strike price) at 4'000, σ (volatility) at 8%, the risk-free rate at 0.3%, q (dividend yield) at 1.5% and T (maturity in years) at 3.0, we can see in Figure 15 that the pricing function of the WDNN heavily deviates from that of the Black-Scholes model. In some of these cases with similar variables the Black-Scholes model consistently underestimated the option market price by a large margin. The divergence of the pricing functions in this case is clearly visible in Figure 15 and we can see that the WDNN model is able to correct the underestimation of the option premium to a great extent in such a scenario.

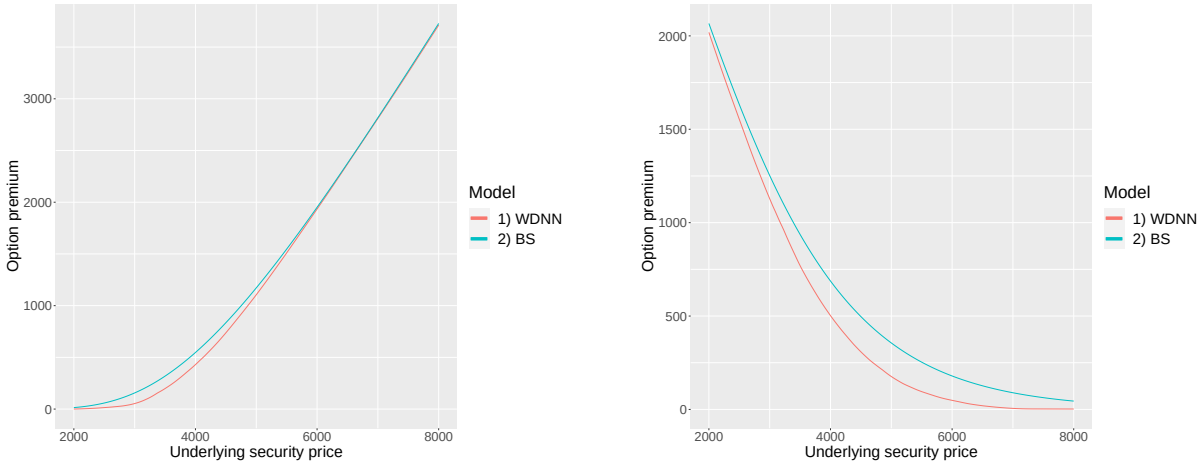


(a) Calls: K=4'000, T=3.0, σ=8%, r=0.3%, q=1.5%

(b) Puts: K=4'000, T=3.0, σ=8%, r=0.3%, q=1.5%

Figure 15: Pricing function for long maturity during times of low volatility

If we take another scenario, where we fix K at 4'000, T at 3.0, σ at 23%, r at 0.3% and q at 1.5%, we can see the opposite. In this scenario, with a long maturity during high market volatility, the Black-Scholes model tends to heavily overestimate the option price, up to +88 for calls and up to +120 for puts vs up to only +48 (calls) and +28 (puts) for the WDNN model. The divergence of the pricing function for this case is visible in Figure 16.

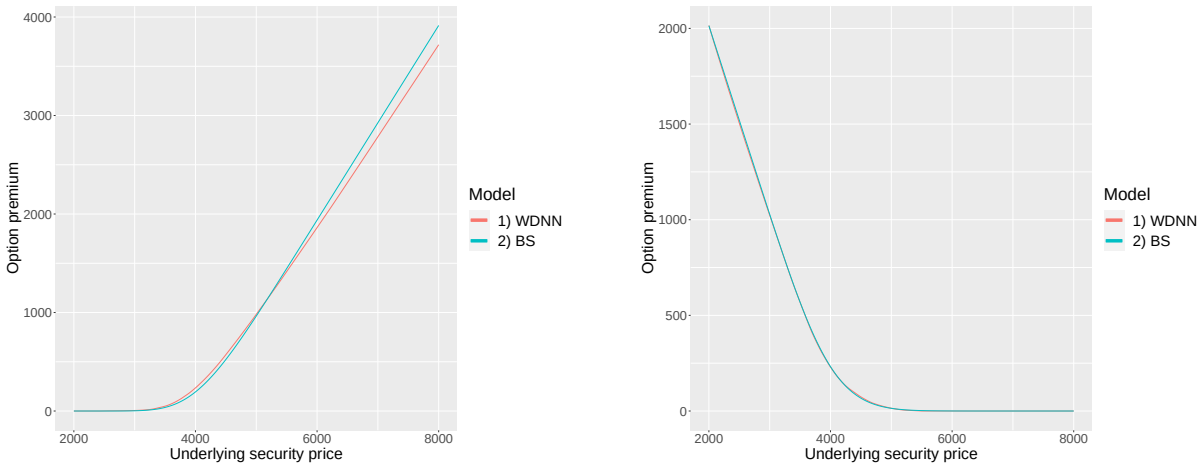


(a) Calls: K=4'000, T=3.0, σ=23%, r=0.3%, q=1.5%

(b) Puts: K=4'000, T=3.0, σ=23%, r=0.3%, q=1.5%

Figure 16: Pricing function for long maturity during times of high volatility

If we take the same fixed variables as before, but fix σ at a more average level of 15% and decrease T to 0.8 (years), we can see in Figure 17 that the pricing function of the WDNN and BS model become almost identical to each other for the put options and much closer to each other than in the previous scenarios for the call options. In this case the WDNN values ATM, NTM, and ITM options higher than the Black-Scholes model, while valuing options that are very DITM slightly lower than the Black-Scholes model. Depending on the selection of the fixed variables there are many other scenarios where the pricing function of the WDNN and BS model can be very close or very different to each other. If someone is interested to visualize or analyze other scenarios, the code to do this and the saved models are available *on Github again*.



(a) Calls: K=4'000, T=0.8, σ=15%, r=0.3%, q=1.5%

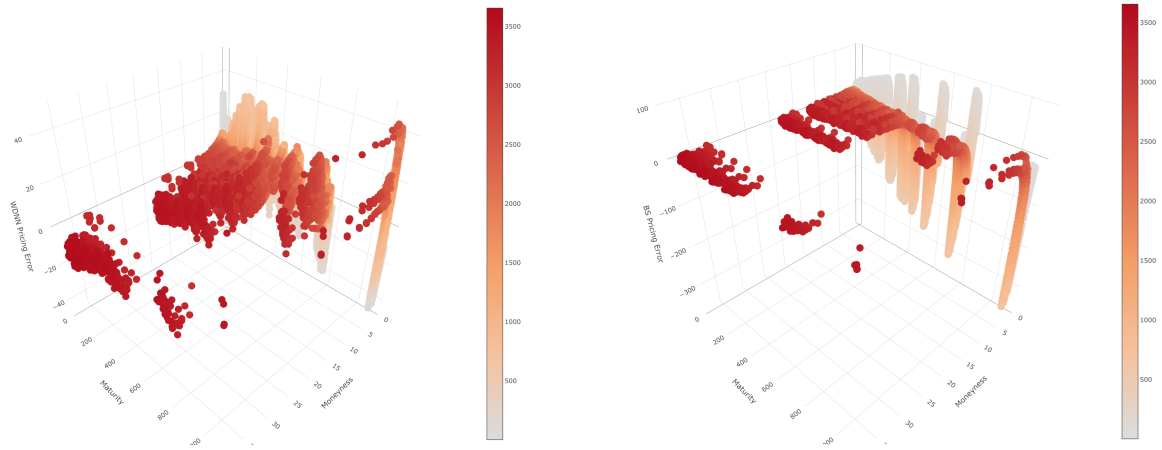
(b) Puts: K=4'000, T=0.8, σ=15%, r=0.3%, q=1.5%

Figure 17: Pricing function for medium maturity during times of average volatility

5.6 Interaction between premium, moneyness, maturity and pricing error

The 3D plot visible in Figure 18 for the call options and Figure 19 for the put options, shows the pricing error on the z-axis in interaction with the maturity (y-axis), level of moneyness (x-axis) and level of the options market price (color scaling of the legend). As the interpretation of a 3D plot printed on 2D becomes limited, an interactive version of these 3D plots for the pricing errors of the models is available on the following websites:

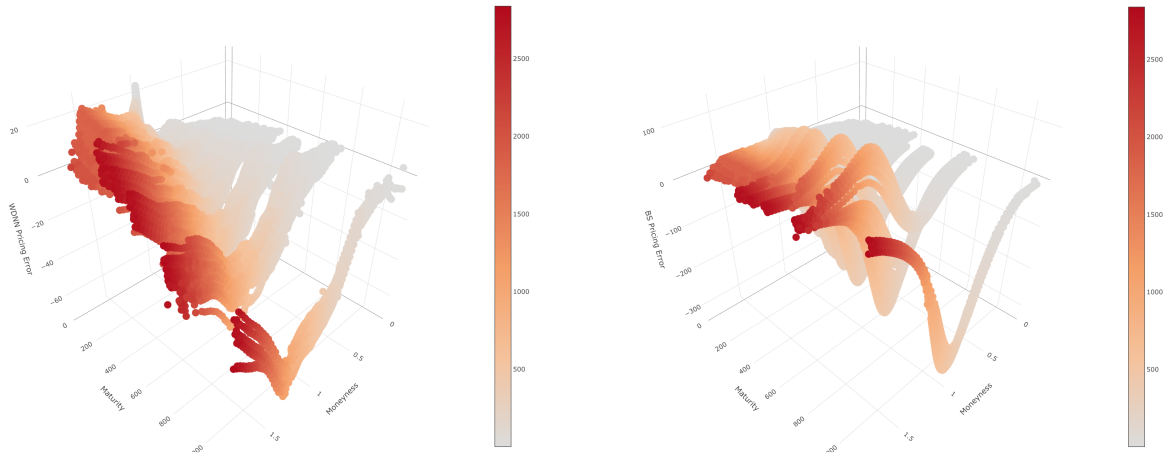
1. *WDNN CALLS - 3D Plot Pricing Errors*
2. *BS CALLS - 3D Plot Pricing Errors*
3. *WDNN PUTS - 3D Plot Pricing Errors*
4. *BS PUTS - 3D Plot Pricing Errors*



(a) WDNN Calls Pricing Error - 3D Plot

(b) BS Calls Pricing Error - 3D Plot

Figure 18: Calls - 3D Plot showing interaction of pricing error across variables



(a) WDNN Puts Pricing Error - 3D Plot

(b) BS Puts Pricing Error - 3D Plot

Figure 19: Puts - 3D Plot showing interaction of pricing error across variables

When visualizing the plots, for comparison it is very important to note here that the z-axis with the pricing errors has not the same range at all on the plots for the Black-Scholes model and the plots for the WDNN model (Calls: -40 to +40 for the WDNN & -300 to +100 for BS; Puts: -60 to +20 for the WDNN & -300 to +100 for BS).

For the call models we can see that for both the WDNN model and the BS model some of the biggest pricing errors happen for options with a very low market value and a level of moneyness close to 1 (S/K). In the short term the WDNN tends to overestimate these options

while it tends to underestimate them in the long run. The BS model generally overestimates the value of NTM options and an actually observed very low market value for maturities up to two years, while underestimating them for maturities around 3 years.

For the put options, we can observe that for the BS model the pricing errors start increasing as the moneyness (K/S) increases to reach a peak for options that are OTM at a level around 0.9 (K/S) and start decreasing again as the options become DOTM. For the WDNN model we can observe that the pricing errors are getting bigger as the moneyness (K/S) increases, to reach a peak for DITM options and decreases again for options that are very DITM. Some of the biggest pricing errors are for options with an observed market price between 400$ and 1'200$ and the pricing errors are usually less severe for options with a very high market price (above 2'000$).

6 Conclusion

Can Machine Learning models challenge the Black-Scholes model? In this paper we have proposed a wide and deep neural network (WDNN) to price European call and put options. The results have shown that the WDNN can compute option prices more accurately than the Black-Scholes model in a significant way across a big majority of the 36 maturity and moneyness groups, using the same input features as the Black-Scholes model with a naïve historical estimation of volatility.

Through the use of a WDNN model we can avoid any potentially false assumptions about financial mechanics and view the option prices as a function of the input features that is well approximated by the WDNN. While the pricing errors significantly increase for options with longer maturities with the Black-Scholes model and the Black-Scholes model has a strong tendency to underestimate the premium, the WDNN model does not suffer from this flaw. Especially when one is unsure about the volatility estimation, the WDNN model can be very useful, as it has shown to be much more accurate than the Black-Scholes model, during times of very low or high volatility for options with longer maturities. In general we can also say that the pricing error is biggest in both absolute an relative terms for OTM and DOTM options for both the Black-Scholes model and the WDNN model. Compared to other computationally more demanding methods to price options, the Machine Learning approach has the advantage of heavily reducing the computing time to price the options, as once the models are trained and saved, the on-line prediction will be very fast.

The superiority of the Wide and Deep Neural Network (WDNN) looks promising, given that European-style options are the home ground of the Black-Scholes model. Even though the focus of this paper lied on the pricing of European options on the S&P 500 index, it should be possible to successfully use the same approach without loss of generality for American, Bermudan, or exotic options and other types of options for which the pricing can be trickier and no reliable closed-form solution exists, as long as enough data is available to train, validate and test the model. With further advances in data, hardware and algorithms over time, Machine Learning models will become even more accurate and efficient in the future. The options greeks, representing different risk measures, play an important role in practice for hedging purposes and can also be calculated based on the WDNN model and could therefore be useful for the development of better hedging strategies. The WDNN architecture has interesting properties for the

pricing of derivatives and further research could focus on the pricing of other types of options in combination with a more sophisticated estimation of volatility. Another interesting approach could be to estimate the implied volatility using a neural network with news sentiment data as an additional feature among other financial data and then feed this estimation back to another WDNN to finally price the option.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... others (2021). Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)* (pp. 265–283).

Amilon, H. (2003). A neural network versus black–scholes: a comparison of pricing and hedging performances. *Journal of Forecasting*, *22*(4), 317–335.

Anders, U., Korn, O., & Schmitt, C. (1998). Improving the pricing of options: A neural network approach. *Journal of forecasting*, *17*(5-6), 369–388.

Bengio, Y., Goodfellow, I., & Courville, A. (2017). *Deep learning* (Vol. 1). MIT press Massachusetts, USA:.

Bennell, J., & Sutcliffe, C. (2004). Black–scholes versus artificial neural networks in pricing ftse 100 options. *Intelligent Systems in Accounting, Finance & Management: International Journal*, *12*(4), 243–260.

Cheng, H.-T., Koc, L., Harmsen, J., Shaked, T., Chandra, T., Aradhye, H., ... others (2016). Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems* (pp. 7–10).

Chollet, F. (2017). *Deep learning with python*. Manning.

Chollet, F., et al. (2021). *Keras*. GitHub. Retrieved from `https://github.com/fchollet/keras`

Dowle, M., & Srinivasan, A. (2021). data.table: Extension of 'data.frame' [Computer software manual]. Retrieved from `https://CRAN.R-project.org/package=data.table` (R package version 1.14.0)

Géron, A. (2019). *Hands-on machine learning with scikit-learn, keras, and tensorflow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media.

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (pp. 249–256).

Han, G.-S., & Lee, J. (2008). Prediction of pricing and hedging errors for equity linked warrants with gaussian process models. *Expert Systems with Applications*, *35*(1-2), 515–523.

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural networks*, *4*(2), 251–257.

Hull, J. C. (2018). *Options, futures and other derivatives*. Pearson.

Hutchinson, J. M., Lo, A. W., & Poggio, T. (1994). A nonparametric approach to pricing and hedging derivative securities via learning networks. *The Journal of Finance*, *49*(3), 851–889.

Ivaşcu, C.-F. (2021). Option pricing using machine learning. *Expert Systems with Applications*, *163*, 113799.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Lajbcygier, P., Boek, C., Palaniswami, M., & Flitman, A. (1996). Neural network pricing of all ordinaries spi options on futures. In *Neural networks in financial engineering: Proceedings of the third international conference on neural networks in the capital markets* (pp. 64–77).

Liu, S., Oosterlee, C. W., & Bohte, S. M. (2019). Pricing options and computing implied volatilities using neural networks. *Risks*, *7*(1), 16.

Malliaris, M., & Salchenberger, L. (1993). Beating the best: A neural network challenges the black-scholes formula. In *Proceedings of 9th ieee conference on artificial intelligence for applications* (pp. 445–449).

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, *5*(4), 115–133.

Mezofi, B., & Szabo, K. (2018). Black-scholes: A new option for options pricing.

Moore, G. (1965). Moore's law. *Electronics Magazine*, *38*(8), 114.

Natenberg, S. (2014). *Option volatility and pricing: Advanced trading strategies and techniques, 2nd edition*. McGraw-Hill Education.

Pagnottoni, P. (2019). Neural network models for bitcoin option pricing. *Frontiers in Artificial Intelligence*, *2*, 5.

Park, H., Kim, N., & Lee, J. (2014). Parametric models and non-parametric machine learning models for predicting option prices: Empirical comparison study over kospi 200 index options. *Expert Systems with Applications*, *41*(11), 5227–5237.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Édouard Duchesnay (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, *12*(85), 2825-2830. Retrieved from `http://jmlr.org/papers/v12/pedregosa11a.html`

Qi, M., & Maddala, G. (1996). Option pricing using artificial neural networks: the case of s&p

500 index call options. In *Neural networks in financial engineering: proceedings of the third international conference on neural networks in the capital markets* (pp. 78–91).

R Core Team. (2021). R: A language and environment for statistical computing [Computer software manual]. Vienna, Austria. Retrieved from `https://www.R-project.org/`

RStudio Team. (2021). Rstudio: Integrated development environment for r [Computer software manual]. Boston, MA. Retrieved from `http://www.rstudio.com/`

Sievert, C. (2020). *Interactive web-based data visualization with r, plotly, and shiny.* Chapman and Hall/CRC. Retrieved from `https://plotly-r.com`

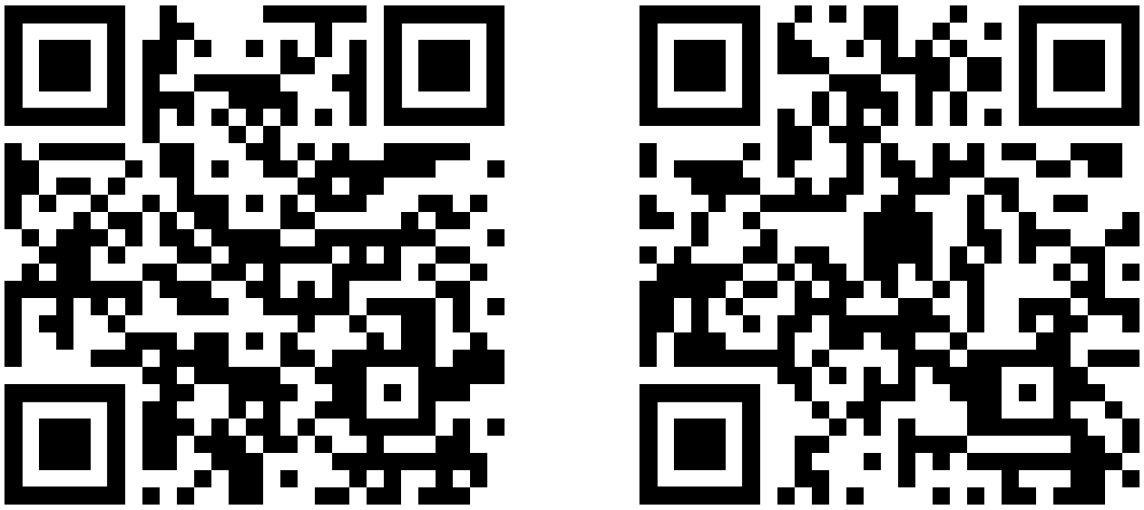
US Department of the Treasury. (2021). *Daily treasury yield curve rates.* Retrieved 2021-05-09, from `https://www.treasury.gov/resource-center/data-chart-center/interest-rates/pages/TextView.aspx?data=yieldAll`

Wharton Research Data Services. (2021). *Optionmetrics.* Retrieved 2021-05-09, from `https://wrds-www.wharton.upenn.edu/pages/get-data/optionmetrics/ivy-db-us/`

Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L. D., François, R., ... Yutani, H. (2019). Welcome to the tidyverse. *Journal of Open Source Software*, *4*(43), 1686. doi: 10.21105/joss.01686

Yao, J., Li, Y., & Tan, C. L. (2000). Option price forecasting using neural networks. *Omega*, *28*(4), 455–466.

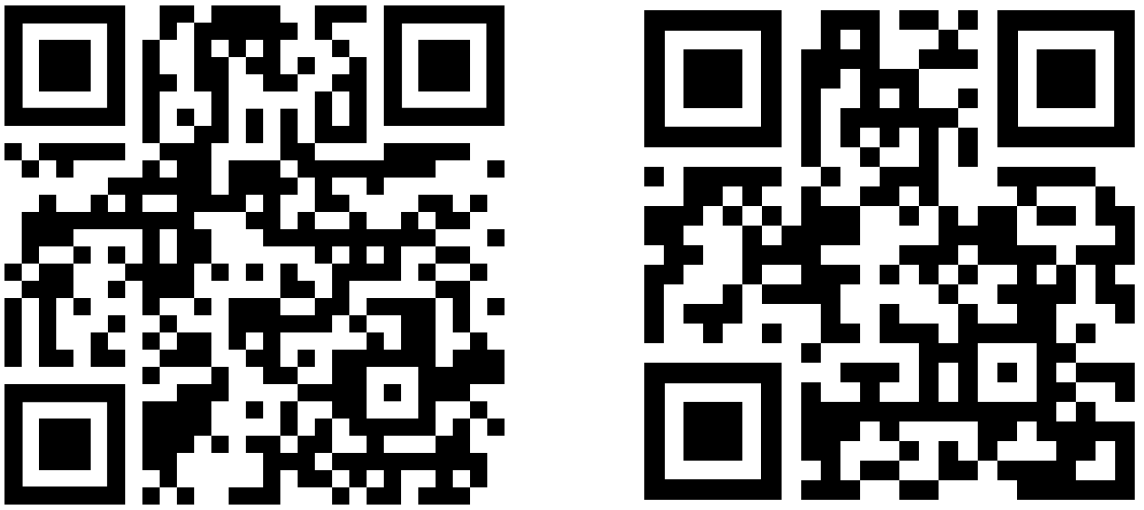
A Appendix



(a) Code on Github

(b) Option Pricing Tool

Figure 20: Code on Github & Option Pricing Tool



(a) Visualizations

(b) 3D-Plots

Figure 21: Visualizations & 3D-Plots