University of Warsaw
Faculty of Economic Sciences

Working Papers
No. 19/2020 (325)

ARTIFICIAL NEURAL NETWORKS PERFORMANCE IN WIG20 INDEX OPTIONS PRICING

Maciej Wysocki
Robert Ślepaczuk

Warsaw 2020

WORKING PAPERS 19/2020 (325)

Artificial Neural Networks Performance in WIG20 Index Options Pricing

Maciej Wysocki[a], Robert Ślepaczuk[b]*

[a] *Faculty of Economic Sciences, Quantitative Finance Research Group, University of Warsaw*
[b] *Faculty of Economic Sciences, Quantitative Finance Research Group, Department of Quantitative Finance, University of Warsaw*
** Corresponding author: rslepaczuk@wne.uw.edu.pl*

Abstract: In this paper the performance of artificial neural networks in option pricing is analyzed and compared with the results obtained from the Black – Scholes – Merton model based on the historical volatility. The results are compared based on various error metrics calculated separately between three moneyness ratios. The market data-driven approach is taken in order to train and test the neural network on the real-world data from the Warsaw Stock Exchange. The artificial neural network does not provide more accurate option prices. The Black – Scholes – Merton model turned out to be more precise and robust to various market conditions. In addition, the bias of the forecasts obtained from the neural network differs significantly between moneyness states.

Keywords: option pricing, machine learning, artificial neural networks, implied volatility, supervised learning, index options, Black – Scholes – Merton model

JEL codes: C4, C14, C45, C53, C58, G13

Note: The views presented in this text are those of the authors and do not necessarily represent those of Labyrinth HF project.
This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors

1. Introduction

The history of neural networks starts in the early 1940s, when McCulloch and Pitts (1943) proposed the first computational model for neural networks. Throughout the years, many upgrades and improvements were proposed. The popularity of neural networks started to grow in 1974, when Werbos published his work about the backpropagation algorithm that actually enabled operational training of models. Within growth in computational power and usage of GPUs (graphical processing units) as compute engines, the era of deep learning came. Neural networks are nowadays used for image and 3D objects recognition, as well as in pattern recognition, in which these algorithms achieve better scores than humans do. In fact, neural networks became so popular, that they are used in nearly all fields of science, such as biology, chemistry, medicine and finance.

In finance, neural networks are broadly used in many different applications. At the very beginning, the main aim of machine learning methods in finance was forecasting stock movements and pricing derivatives. Nowadays these methods are applied for a variety of different tasks, such as preventing credit frauds, modeling volatility, building automatic transactional systems and algorithmic trading. Moreover nets are often used as tools for speeding up processes of calibrating parametric models as well as solving partial differential equations (e.g. the Black – Scholes equation).

Although the performance of neural networks has already been described in different papers, most of them focus on simulated markets or data from the New York Stock Exchange with the approach of boosting the performance of Black – Scholes model. The main aims of this paper are exploration of deep learning possibilities in option pricing and analysis of the market data-driven approach for neural networks training. None of the previous works covered the topic of machine learning approach to pricing derivatives on emerging markets with relatively low liquidity and high volatility. Throughout the years many different models have been proposed for the purposes of pricing options and modeling their movements, however none of them was ever proven to be the best one. Therefore the chase for the unbiased and reliable approach to options pricing continues. Taking into account previous results of neural networks, these models might turn out to be a solution for problems occurring in pricing contracts quoted on emerging markets. While many previous papers present simulated (or randomly generated) data for training purposes, this approach will most likely not work for conditions of the Polish derivatives market. In order to cover as many

market behaviors as possible, the market data seems to be a much better solution for model fitting purposes. The goal of this research is to design the proper architecture of the neural network for data-driven approach and test its performance comparing to the traditional Black – Scholes – Merton model.

Testing different approaches and methods is not only important for scientific purposes, but also from investors' point of view. Options are one of the most popular financial derivatives, which are traded on worldwide stock exchanges as well as over-the-counter instrument. These contracts are in portfolios of many institutions such as banks, hedge funds or pension funds. In order to efficiently deal with the financial management of portfolios constructed of various instruments, including derivatives, one must first possess appropriate tools for pricing and valuation of traded assets. Unbiased and accurate pricing might turn out to be a substantial advantage over other participants of the market. Therefore taking up the topic of effective option pricing with new approaches is not only interesting, but also a very important topic that many institutions and financial theorists constantly research.

The first research hypothesis is that neural networks trained on real-world market data are able to perform better than the Black – Scholes model in terms of pricing errors. Similarly to the standard machine learning development process, common evaluation metrics are compared for both neural network and Black – Scholes model price pricing in order to decide which of these two gives better, more accurate pricing. When it comes to financial applications, not only accurate results are important, but also stability and robustness of the method is crucial. In order to properly compare approaches and therefore verify the hypothesis, the hyperparameters tuning for the neural network is conducted.

The assumption that neural networks might, in fact, perform at least as good as the traditional Black – Scholes model is based on many previous kind of research, such as Jang and Lee (2019) or Yang *et. al* (2017). As these articles state, properly designed neural networks trained on market data are able to significantly outperform other models, including Black – Scholes. Different types of network architecture are used for a variety of different market conditions (e.g. various volatility structures), but for the very first research on machine learning methods on Polish derivatives more general type of neural network is used.

The second hypothesis is that one can observe a difference in pricing errors of the neural network taking into account the moneyness ratio. For different moneyness states, there

could be various error distributions. The distinction between moneyness ratios is important not only for the purposes of the analysis, but also for investors who may use different pricing methods. The revealing pattern in error distribution might provide an explanation of its magnitude. Moreover for purposes of future neural network architectures, such analysis might be helpful when deciding whether options with particular characteristics should be excluded from the training process.

Taking into account the properties of options and differences between moneyness states, it seems to be a reasonable assumption that the pricing error will differ for every possibility. Kokoszczyński *et al.* (2010a) provided evidence from the Polish market, that indeed different moneyness ratios and time to maturities reveal patterns of the pricing error distribution. It turns out, that the worst results are for low both time to maturity and moneyness ratio and the best results for high both time to maturity and moneyness ratio. The same has been done for the Japanese option market, regardless of the chosen model and its characteristics (Kokoszczyński *et al.*, 2017).

This paper includes a comparison of artificial neural networks designed for pricing options, trained on market data from Polish stock exchange with the Black – Scholes – Merton model with historical volatility. The results for ANN (artificial neural network) were obtained using *R* in version 3.6.1 along with *Python* in version 3.7.4. Deep learning libraries used for design, training and testing the network are Keras (version 2.3) and TensorFlow (version 2.0). The rest of the calculations, as well as graphs and tables were done using only *R* language with the *RStudio* development environment.

The remaining parts of the paper are organized as follows. Firstly literature review is provided, followed by a chapter consisting of option pricing models included in the paper and methodology description with an introduction to neural networks. The third chapter is devoted to particular parts of ANNs (e.g. nodes, layers or activation functions), their meaning and usage, as well as a description proper network architecture development. Subsequently, data description and its preprocessing for purposes of ANN fitting are provided. The next chapter presents empirical results, model performance and comparison. Verification of main hypotheses, summary and proposals for further research are in the conclusion of this paper.

2. Literature Review

The rapid growth of interest in the options started in the 1970s, when the Chicago Board of Options Exchange (CBOE) was founded, as well as the first comprehensive model for pricing the European-style options was proposed. The paper "The Pricing of Options and Corporate Liabilities" published by F. Black and M. Scholes (1973) together with the paper "Theory of Rational Option Pricing" written by R. Merton (1973), introduced a new era of pricing to the options market. The Black – Scholes – Merton (BSM) model seemed simple and easy to implement in the practical issues, giving reliable results however after not a very long time researchers published results based on the real-world data suggesting that some of the BSM model assumptions are not realistic.

Article "Empirical option pricing: a retrospection" (Bates 2003) provides a very exhaustive discussion of previous empirical research concerning option pricing, especially with the BSM model. The author points out that many tests of the BSM model were conducted in an inappropriate way that is testing one of the assumptions and ignoring all of the others. Moreover it is stated, that the three most unrealistic assumptions are that the stock price follows the Geometric Brownian Motion with constant volatility and constant risk-free interest rate. The author suggests that the BSM model is not the right model however the results obtained while testing should be considered a crucial indication where to look for further development. At that time researchers focused on the so-called volatility smile and negatively skewed risk-neutral distributions, typical for returns series. Difficulties with finding a suitable estimator that would properly forecast underlying asset volatility are one of the main reasons for the imprecise prices obtained from the BSM model. The work of Bakshi *et. al* (1997) also revealed inconsistency of the BSM model between different moneyness and maturities. This paper proves that there is a significant pricing bias between different states. Moreover it was shown that introducing stochastic volatility (Heston 1993; Hull and White 1987) and stochastic volatility jumps (Bates, 1996) does indeed improve the performance of option pricing models.

When it comes to modeling the volatility, the BSM model is used in order to calculate the implied volatility that is believed to comprehend current uncertainty on the markets. Implied volatility was said to have huge predictive power in forecasting the future volatility on the markets in order to address issues with biased volatility indicators. However further researches conducted to investigate the implied volatility abilities to forecast future volatility

showed very poor performance of such approach. Work of Canina and Figlewski (1993) clearly states that the implied volatility calculated with the BSM model has practically no correlation with future volatility, as well as it does not follow information contained in recently observed volatility. Similarly work of Fleming (1998) shows that implied volatility is a biased estimator however it does contain valuable information about the future realized volatility. Moreover implied volatility has a better predictive accuracy than historical volatility. Although it's proven to be biased, implied volatility has the advantage over other estimators when used in the BSM model, as shown in many previous works (Raj and Thurston 1998; Ferreira *et. al* 2005).

A large variety of machine learning methods was used in financial applications due to their flexibility and reliability. Park *et. al* (2014) provided a comparison of a few machine learning algorithms – support vector regression (SVR), Gaussian process regression (GPR) and artificial neural network (ANN) with parametric methods – Black-Scholes model, Heston model and Merton model. It turned out that non-parametric machine learning methods significantly outperformed the BS model, as well as performed comparably well to other parametric models, depending on moneyness and maturity. Machine learning algorithms did not show any evidence of overfitting, as well as provided reliable out-of-sample forecasts. SVR had the best performance in-sample, while out-of-sample pricing error was the lowest for one of the following: SVR, NN or Merton jump-diffusion model, depending on characteristics of the option. Similar conclusions were stated by Wang (2011) in an article that summarizes the performance of SVR in currency options pricing. He focuses on the selection of initial input variables of SVR, as well as introducing stochastic volatilities with jumps to the model. As a result a great improvement in forecast accuracy is obtained. Results obtained on Hong Kong derivatives Market (Liang *et.al* 2009) also confirm that the use of SVR and NN does improve pricing accuracy and provide better, more reliable results. The best out-of-sample results were obtained using SVR and slightly worse using multi-layer perceptron (MLP). Clearly the worst results among non-parametric methods were obtained using a linear neural network (LNN). Nevertheless, all of these methods turned out to be more accurate in predicting the options prices than the conventional, parametric methods that are binomial trees, finite differences method and Monte Carlo simulations.

The very first papers concerning the use of artificial neural networks for purposes of derivatives pricing are the articles of Malliaris and Salchenberger (1993) and Hutchinson *et. al* (1994). The results from both papers were promising, as in the first work the ANNs

managed to outperform the BSM model for pricing in- and out-of-the-money options, as well as gave similar results for at-the-money options. The second article investigated not only predictive power and ability to directly forecast the price, but also the hedging performance of the neural networks. It turned out that the MLP performs significantly better in delta hedging, providing more accurate and stable predictions, while the BS model exhibited greater bias variability for different moneyness and maturities.

Throughout the years many papers were published providing evidence for advantages of a non-parametric approach to the problem of pricing derivatives and other assets, as well as new methods were developed to address issues of certain market conditions. In addition the methodology of modeling and initial parameter choice differs between authors. The simplest approach is feeding the network with the price of underlying asset and strike price without any processing along with the other parameters used in the BSM model. This can be found in articles of Herrmann and Narrr (1997), Palmer and Gorse (2017) or Mitra (2012). Nevertheless this approach is far less popular than the use of transformed spot price and strike price. The most common transformation is a division of the underlying asset price by the strike price. Such data preprocessing can be found in articles published by Andreou *et.al* (2006, 2008, 2010), Amilon (2003), Gencay and Salih (2003) and Hahn (2013).

The most recent papers concerning the evaluation of the neural network's performance in pricing liabilities focus mainly on testing different types and architectures of the neural networks, such as the recurrent neural networks (RNNs) or the convolution neural networks (CNNs). Yang *et. al* (2017) proposed the use of the gated neural networks that not only provide reliable prices of the options, but also contain a guarantee of economically reasonable and rational results. The model is able to outperform other network-based models and some of the econometric methods described in the paper. When it comes to the portfolio hedging, the long short-term memory (LSTM) RNN is said to be outperforming conventional methods (Huang and Zhang 2019).

3. Methodology and Option Pricing Models

The terminology used in the paper refers to the typical nomenclature used in the literature concerning options pricing and machine learning. In order to compare the models and verify hypotheses error metrics were introduced along with a description of methodology and neural networks. The architecture of a neural network developed

for purposes of the options pricing is discussed together with the conducted hyperparameters tuning.

3.1. Terminology and Metrics

An option is a contract giving the holder a right to purchase (a call option) or to sell (a put option) a fixed amount of underlying asset at a specific date that is the expiration date. Typically two most popular types of options are distinguished, namely European options and American options. European style options are the ones that cannot be exercised before the day of expiration, whereas owners of American style options can use their right at any time before maturity. Due to the fact that options are traded on options exchange organized markets as well as over-the-counter, there are many different types of options, called the exotics. The pricing of the exotics is rather different from the traditional styles and is not the scope of this study.

At the beginning, the moneyness should be introduced. It describes an actual profit for the owner of an option from exercising the option immediately at that time. Three different moneyness states are distinguished: out-of-the-money (OTM), at-the-money (ATM) and in-the-money (ITM). In-the-money options are the ones that would have positive intrinsic value if they were exercised today. Similarly, out-of-the-money options are the ones that would have negative intrinsic value if they were exercised today. At-the-money options fill in the last possibility which is the current price and strike price are the same, so the theoretical profit is equal to zero. In order to properly classify options within their current moneyness, the moneyness ratio (MR) is used, expressed as (Kokoszczyński 2010a):

$$MR = \frac{S}{K \cdot e^{-r \cdot \tau}} \tag{1}$$

where S is the spot price of the underlying, K is the strike price, r is the risk-free rate and τ is the time to maturity. For call options, moneyness ratio in the range $[0; 0.95)$ refer to out-of-the-money options, the moneyness ratio in the range $[0.95;\ 1.05)$ means the option is at-the-money, and options with MR higher than 1.05 are in-the-money. Put options are classified using the same ranges, but in reverse order, so that the first range is for ITM options and the last one is for OTM options.

Secondly, the term hedging is to be described. Hedging is an investment strategy that assumes resigning from potential profits in order to neutralize the risk influence. Hedging

the portfolio means making it as much resistant to the market risk as possible. In order to properly carry out this procedure, after taking a position in one investment one must take another one in an opposing investment. For hedging purposes many different types of financial instruments are used e.g. stocks, forward contracts, swaps or derivatives, such as options. Many types of strategies designed for neutralizing the market risk are available for the investors and choosing which should be used depends on the portfolio that has to be hedged. The BSM model described below is based on the assumption that a risk-neutral portfolio can be created taking a short position in an option and a long position in the stock. This means, that the option's value is not dependent on the stock's price, therefore, the risk of an unfavorable price change is neutralized.

When it comes to model evaluation and comparison between different methods, error metrics have to be introduced. Statistics used in this paper are typical for any work concerning regression problems (James *et. al* 2013):

– mean absolute error (MAE)

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|Y_i - \hat{Y}_i\right| \tag{2}$$

– mean square error (MSE)

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2 \tag{3}$$

– root mean square error (RMSE)

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2} \tag{4}$$

– mean absolute percentage error (MAPE)

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{Y_i - \hat{Y}_i}{Y_i}\right| \tag{5}$$

For each case Y_i is the real value and $\hat{Y}_i$ is the value calculated by the model.

3.2. Black – Scholes – Merton Model

The very first comprehensive option pricing model, that is the Black – Scholes – Merton (BSM) model, assumes the stock price follows a geometric Brownian motion with constant drift and volatility. The further assumptions (Black and Scholes, 1973) are:

– there is a constant, risk-free rate of return

– the underlying does not pay a dividend

– there are no arbitrage opportunities on the market

– the trade on the market is continuous

– any amount of stocks can be bought or sold on the market

– any amount of money can be borrowed at the risk-free rate

– there are neither transaction costs, nor taxes.

These assumptions are said to be unrealistic, as described in the literature review part, therefore, many of them have been removed or relaxed by further works (e.g. Merton's correction for dividend-paying assets). The model used in this paper is the framework BSM model with the continuous dividend paid by the underlying asset. The model is based on the partial differential equation, namely the Black – Scholes (Black and Scholes 1973) equation:

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0 \tag{6}$$

where V is the price of the option at time t, S is the price of the underlying at time t, r is the risk-free interest rate and σ is the standard deviation of the underlying asset's returns. The main idea standing behind this equation is that market participants are able to hedge positions in options by buying or selling the underlying asset, borrowing the money at the riskless rate. The solution that is the price of either put or call price given by the model is ought to satisfy the equation 6. The prices of European options can be obtained using the following formulas (Black and Scholes 1973):

$$P_c(S_t, \tau) = S_t \cdot e^{-q\cdot\tau} \cdot N(d_1) - K \cdot e^{-r\cdot\tau} \cdot N(d_2) \tag{7}$$

$$P_P(S_t, \tau) = K \cdot e^{-r\cdot\tau} \cdot N(-d_2) - S_t \cdot e^{-q\cdot\tau} \cdot N(-d_1) \tag{8}$$

$$d_1 = \frac{\ln\left(\frac{S_t}{K}\right) + (r - q)\cdot\tau}{\sigma\sqrt{\tau}} + \frac{\sigma\sqrt{\tau}}{2} \tag{9}$$

$$d_2 = \frac{\ln\left(\frac{S_t}{K}\right) + (r - q)\cdot\tau}{\sigma\sqrt{\tau}} - \frac{\sigma\sqrt{\tau}}{2} = d_1 - \sigma\sqrt{\tau} \tag{10}$$

where P_C is the price of the call option and P_P is the price of the put option. In these equations, additional symbol q that denotes the dividend rate, K stand for the strike price and $N(\cdot)$ is the cumulative distribution function of the standard normal distribution.

The one parameter that cannot be directly observed on the market is the volatility of the underlying asset, which is also one of the most important issues when it comes to options pricing. Accurate modeling of the volatility process is crucial for precise options pricing, therefore many approaches have been proposed for its forecasting and measuring. The volatility estimator used in this paper is the historical volatility which is also very popular among market practitioners (Kokoszczyński *et. al* 2010b). It is based on the historical returns from the underlying and calculated using the formula:

$$HV_n = \sqrt{T}\sqrt{\frac{1}{n-1}\sum_{t=0}^{n}(u_t - \overline{u})^2} \tag{11}$$

$$u_t = \ln\frac{S_t}{S_{t-1}} \tag{12}$$

where u_t is a logarithmic rate of return from the underlying asset, $\overline{u}$ is the mean of all log returns in the sample, T is the number of trading days in a year and n is the size of the sample. Typically T is equal to 252 trading days in the year. The size of the sample, which is a number of days taken into account when calculating volatility estimator, is chosen to be 60 in this paper, which is based on the assumptions that too few days would not allow to catch the long term trend and too many days would result in a heavily biased and smoothed estimator.

It is worth mentioning that there are many other approaches to modeling volatility that could be used to obtained estimators introduced to the BSM formulas, such as realized volatility (Black 1976), stochastic volatility (Hull and White 1987; Heston 1993) or implied volatility calculated from observed market options prices. Executive review of various volatility estimators can be found in Ślepaczuk and Zakrzewski (2013).

3.3. Artificial Neural Network

Artificial Neural Networks are statistical tools inspired by the architecture of the human brain, designed to mimic the way humans process the information. This approach allowed creating flexible and accurate statistical models used in nearly any field of modern science.

3.3.1. Architecture of Artificial Neural Networks

Artificial Neural Networks are the black-box models that process the input data through the nodes and the layers to generate the output data without any prior knowledge of the closed-form functional relationship between them. This is called a non-parametric approach, as one does not need to assume any type of parametric function that links the input with the output. This is a huge advantage over the parametric approach, as the non-parametric models are able to react to any new conditions with changing the black-box functional form which is impossible for any parametric model such as the BSM model. When the neural network consists of more than one layer then it is called a deep learning model. ANNs are also supervised learning models which means that the labeled data has to be used in the learning process in order to properly train the model. The learning process is, in this case, training the algorithm on the example of input and output pairs so that the model knows how to generate the output when fed with new input.

The class of NNs used in this paper is the multilayer perceptron (MLP), also called the feedforward network, which consists of at least three layers of neurons: the input layer, the output layer and at least one hidden layer between them. There are no restrictions for the number of layers or neurons in them, so one has to find a proper architecture depending on the data the NN is fed with. The very first description of MLPs can be found in articles of Ivakhnenko (1973) and Fukushima (1980). The very first layer of MLP is the input layer that consists of data vector and a bias term from which the weighted sum is calculated and then fed forward to the first hidden layer. In each layer of the network there is a chosen activation function that is responsible for activation of the layer if the output from the previous one exceeds the threshold of activation. The output of the activation function from each neuron is then passed further to the next layer of nodes where the weighted sum is calculated again. This process continues until the output layer is introduced where the output vector is calculated.

The basic component of ANNs is a neuron, also called a node. The neurons take the input either from the initial data set or from the previous layer and combine it with an optimal threshold calculated using the activation function. These functions task is introducing non-linearity to the model as well as creating a differentiable transition as the input changes. The neurons are attached to connections that are responsible for assigning weights in the way that more valuable information has a higher weight. The output from the previous neuron is taken as input for the next node after the propagation function has calculated a weighted sum as described above. Each set of neurons create

a single layer in the way that neurons from one layer can be connected only to these from the previous layer and the following layer. Between layers a full connection can exist as well as a dropout could be introduced. Dropout is a technique used for preventing the ANN from overfitting to the training dataset and it consists of dropping out randomly selected nodes from a single layer.

Activation functions play a very important role in the neural networks, therefore a large variety of candidates for these functions has already been proposed. These functions introduce non-linearity to the model so that the NNs can be used as such efficient tools for various problems. The choice of activation functions depends on the type of the problem (regression or classifications) and type or amount of the data that the NN is fed with. Typical activation functions used for regression problems are: rectified linear unit (ReLU), leaky rectified linear unit (Leaky ReLU), logistic (sigmoid), exponential linear unit (ELU) and softmax.

Training a neural network for regression problems rests on training the model on the example of input and output pairs from the training dataset. This process fits the statistical tool for observations so that the out-of-sample data could be used for forecasting the output within learned rules and patterns. The training process is conducted by minimizing the observed errors between real and predicted values in order to maximize the accuracy of the fitted values. The errors are expressed using cost function which form depends on the type of the problem. The cost function is evaluated in every run and then the weights in connections between the neurons are updated in order to optimize the function. Typically the learning process continues as long as the error is reduced, so intentionally the cost function reaches its global extremum. Nevertheless, in the case of an unsatisfactory result, the architecture of the neural network should be redesigned. A part of the training process is also the choice of certain hyperparameters for the NN. Hyperparameters and their optimal selection will be described in further part of this paper.

3.3.2. Backpropagation and optimization

Backpropagation (Werbos 1974) is an algorithm used in training MLPs for supervised learning problems. This method was designed for adjusting the weights in a connection between neurons in the way that the error is minimized. The algorithm calculates the loss functions gradient with respect to each weight during the training process. It is done using the chain rule calculating the gradient of neural network's cost function backwards for one layer at a time. Starting from the output layer, partial derivatives are calculated through every layer

to the input layer and then for each of them the algorithm returns gradient with respect to adequate weights. The main advantage of backpropagation is its efficiency which allows the use of gradient-based optimization techniques. Thanks to the backpropagation algorithm, training the neural network can be conducted as an iterative process of updating weights.

The gradient-based optimization techniques are designed to search for global optima of a function in directions pointed by the gradients at the specific point. The most common technique is the gradient descent algorithm used for finding a local minimum of a differentiable function. Nevertheless there are many alternatives that could be used for optimization purposes. One such alternative is stochastic gradient descent (SGD), which updates the weights each time a partial derivative is calculated. Another one is adaptive moment estimation (ADAM) in which both the gradients and the second moment of the gradients are calculated and used for weights updating (Kingma and Ba 2014). There is no optimizer proven to be the best one and each of them has different convergence characteristics, so once again the choice of the method depends on the nature of the problem as well as the data used in the training process.

3.3.3. Hyperparameters tuning

Having discussed the core parts of neural networks, the hyperparameters will be introduced. The hyperparameters are a type of parameters that are arbitrarily set before the learning process starts and do not change through the training phase. Many different hyperparameters also play different roles, e.g. speed up the computation or influence the accuracy of predictions. In order to properly develop neural network architecture, a variety of such parameters have to be chosen. As the option pricing is a supervised regression problem, the chosen objective function to be minimized is MAE.

The basic hyperparameters are the number of layers in a neural network and a number of neurons in each layer. Both of them have to be defined at the beginning of architecture design. Typically neural networks used for purposes of option pricing do not have too many layers (Liu *et. al*, 2019) and the search for the optimal number should be done rather using the trial and error method (Hamid and Iqbal 2002). Following this technique, the proper number of layers was found to be 6, including the input and the output layer. The search for the optimal number of layers started from one hidden layer and consequently networks up to six hidden layers were tested. The input layer is responsible for taking the initial data for the neural network. The output layer is producing the results for the given data. The most

important part happens in the hidden layers where all the calculations are done. Every hidden layer in the neural network is learning various paths in the data through the minimization of the cost function.

For the first stage of neural network architecture design, the number of nodes was chosen along with the activation functions, a batch size, a dropout rate and an optimizer. At that point, the task was to find an initial set of parameters that were performing well and stable in order to use them as a starting point in the tuning phase. The final values of parameters were chosen in the process of hyperparameters tuning conducted for all of the following hyperparameters: batch size, dropout rate, optimizer, initializer, learning rate, $\beta 1$, $\beta 2$.

The batch size is the parameter defining how many examples are used in one backward pass of the learning process. It defines how many samples from the training data are used in one complete training pass through the network. Small batch size requires less memory however it might happen, that the gradient estimates will be inaccurate for too small batches. The dropout rate refers to ignoring randomly selected nodes during one pass of the training phase in a certain layer. The dropouts are introduced for chosen layers in order to prevent the algorithm from overfitting to the training data. When the fully developed neural network is trained without ignoring random nodes, it happens that there are codependences between the layers. Therefore the dropout is introduced for the purpose of learning more robust features. The optimizers, as described in the previous section, are algorithms responsible for updating the attributes of NNs such as the weights and the learning rate in order to reduce the error and minimize the objective function. Therefore the optimizers assign the impact of the gradient on the parameters. The initializers are functions defining the way to set the initial random weights for connections in each dense layer. Choosing a proper initialization technique is crucial, as only proper weights allow optimizing the function in a rational amount of time. When the initial weights are set incorrectly, the convergence to the global minimum is impossible. For the very first training pass, weights are randomly set sampled from e.g. normal or uniform distribution. The learning rate is said to be one of the most important hyperparameters, as it is responsible for controlling how much the weights are updated in response to the evaluated cost function at every pass in the training process. This tuning parameter influences directly every step of the optimization process by assigning how much the new information replaces the old features. Too high learning rate will cause the

algorithm to pass by the global minimum, while too low learning rate will stuck in a local minimum or saddle point as well as it could fail to converge in a rational amount of time.

Table 1. Possible values of the hyperparameters investigated during the first stage

Parameter	Options or range
Neurons (each layer)	250, 500, 1000, 1500, 2500
Batch Size	250, 500, 1000, 1500, 2000, 2500
Dropout Rate	0, 0.05, 0.1, 0.2
Optimizer	RMSProp, Adam
Activation Function	ELU, ReLU, Softmax, Sigmoid

Note: Different values of hyperparameters checked in the first stage of development of neural network architecture

Table 2. Hyperparameters of the neural network framework

Parameter	Chosen option or value
Neurons (each layer)	1000
Batch Size	1500
Dropout Rate	0.1
Optimizer	Adam
Activation Function	ReLU

Note: The hyperparameters chosen as the optimal values from all of the possibilities in Table 1.

The first stage of the neural network architecture design was conducted as an iterative process of training the neural network with another combination of hyperparameters until all possible sets of hyperparameters were checked. Then the results were summarized and the set that resulted in the lowest value of loss function was chosen. This approach allowed designing a neural network that gave stable and satisfactory results for that moment. Nevertheless a tuning of hyperparameters along with preventing from overfitting had to be done.

The hyperparameters tuning was conducted in the following way. For each of the parameters a set of possible values was chosen and then network with framework architecture (Table 2) was trained using different values of just one parameter with other parameters set to be constant. Such approach allows comparison between different hyperparameter values. Moreover changing just one hyperparameter at a time ensures that the changes in the results are caused by the investigated parameter and not by the others. The number of epochs is set to 5 so that the algorithm responsible for updating weights runs 5 times during a single training process. In this way, the optimal values could be found for each of the hyperparameters. Different models were compared using the values of the loss function and MSE. The Table 3 contains all of the ranges or possible options for different parameters of the model.

Table 3. Values of the hyperparameters investigated during the hyperparameters tuning

Parameter	Options or range
Neurons (each layer)	500, 1000, 1500, 2000
Batch Size	1000, 1500, 2000
Dropout Rate	0.05, 0.1, 0.15, 0.2, 0.25
Optimizer	SGD, Adam, Adamax, Adagrad, Adadelta, Nadam
Initializer	Random Normal, Random Uniform, Glorot Normal, Glorot Uniform, Lecun Normal,
Learning Rate	0.0001, 0.005, 0.001, 0.005, 0.01
β1	0.75, 0.8, 0.9, 0.95, 0.975
β2	0.95, 0.975, 0.999, 0.9999

Note: Hyperparameters values for the tuning phase aiming to improve the performance of the Neural Network.

The number of neurons was chosen to be the same for every layer. Although the initial value of nodes was selected from a similar range, it was decided to check for different values once again on the framework in order to confirm and double-check the results. The table below summarizes the results for each value, so that the statistics chosen are final values obtained after the whole learning process was conducted for a specific value of a parameter.

Table 4. A number of neurons and error metrics evaluated after each training process

Neurons	MAE	MSE
500	0.0232028	0.0025026
1000	0.0232188	0.0025038
1500	0.0232206	0.0025054
2000	0.0232127	0.0025038

Note: Final values of the error metrics calculated for different number of nodes for the hyperparameters tuning. Other hyperparameters are: batch size – 1500, dropout rate – 0.1, optimizer – adam, activation function – ReLU.

As clearly visible, values are very similar in each case. The final number of neurons at each layer was chosen to be 500 as for that number the cost function was monotonically decreasing during the training process in opposite to other possible numbers of neurons for which either MAE or MSE were behaving in a non-monotonic way.

Table 5. The batch size and error metrics evaluated after each training process

Batch size	MAE	MSE
1000	0.0231536	0.0024729
1500	0.0231796	0.0024737
2000	0. 0231300	0. 0024713

Note: Final values of the error metrics calculated for different batch sizes for the hyperparameters tuning. Other hyperparameters are: neurons – 500, dropout rate – 0.1, optimizer – adam, activation function – ReLU.

When it comes to the batch size, the optimal value was different as the one chosen in the first stage. In the final model there are 1000 examples of input and output data during one backward pass in the training process. Although the final values of MAE and MSE

are slightly higher than for the batch size set to 2000, the learning process for the batch size equal to 1000 is more stable, therefore this value was chosen.

Table 6. The dropout rate and error metrics evaluated after each training process

Dropout Rate	MAE	MSE
0.05	0. 0232059	0. 0025036
0.1	0. 0232209	0. 0025042
0.15	0. 0232166	0. 0025025
0.2	0. 0232186	0. 0025025
0.25	0. 0232311	0. 0025060

Note: Final values of the error metrics calculated for different dropout rates for the the hyperparameters tuning. Other hyperparameters are: neurons – 500, batch size – 1000, optimizer – adam, activation function – ReLU.

Once again, the results are very similar for each value of the hyperparameter. The dropout rate was set to 0.2, as for that value the process remained the most stable among the others as well as the error metrics evaluated after the first epoch was the lowest. The stability of the process means that no sudden jumps upward or downward of the error metrics were observed during the training process, whereas error metrics for each epoch reveal how far from the expected minimum is the model at the given stage of optimization.

Table 7. The optimizer and error metrics evaluated after each training process

Optimizer	MAE	MSE
SGD	0. 0235614	0. 0025065
Adam	0. 0232262	0. 0025050
Adamax	0. 0232267	0. 0025074
Adagrad	0. 0232103	0. 0025060
Adadelta	0. 0237524	0. 0025160
Nadam	0. 0232326	0. 0025073

Note: Final values of the error metrics calculated for different optimizing methods for the the hyperparameters tuning. Other hyperparameters are: neurons – 500, batch size – 1000, dropout rate – 0.2, activation function – ReLU.

The results differ a little between optimizers, especially when it comes to comparing errors. The worst results were obtained for Adadelta and SGD, as for both of these MAE exceeded 0.0235. Better results are obtained using Nadam, for which MAE was around 0.02323. The best results were obtained for Adam, Adamax and Adagrad. Since Adam gave reproducible results and it is most commonly used in financial applications, it was chosen as the final optimizer. The Adam optimizer provides results which do not differ between the training runs as well as tend to converge to the desired extremum.

Table 8. The initializer and error metrics evaluated after each training process

Initializer	MAE	MSE
Random Normal	0. 0232714	0. 0025059
Random Uniform	0. 0232260	0. 0025062
Glorot Normal	0. 0232198	0. 0025041
Glorot Uniform	0. 0232212	0. 0025027
Lecun Normal	0. 0232197	0. 0025050

Note: Final values of the error metrics calculated for different initializers for the the hyperparameters tuning. Other hyperparameters are: neurons – 500, batch size – 1000, dropout rate – 0.2, optimizer – Adam, activation function – ReLU.

The highest MAE values were obtained for random normal initializing function. Random uniform performed clearly the worst in terms of mean squared error value. Glorot normal and lecun uniform perform very similarly. The final method of random weights assignment was chosen to be lecun normal due to its monotonically decreasing error metrics.

Table 9. The learning rate, β1, β2 and error metrics evaluated after each training process

Learning Rate	β1	β2	MAE	MSE
0.001	0.9	0.9999	0.0232107	0.0025051
0.005	0.8	0.9999	0.0232112	0.0025075
0.001	0.8	0.9999	0.0232116	0.0025121
0.001	0.8	0.95	0.0232172	0.0025096
0.001	0.9	0.975	0.0232194	0.0025064

Note: Values of error metric for 5 best runs of the neural network with corresponding hyperparameters used in the run. Other hyperparameters are: neurons – 500, batch size – 1000, dropout rate – 0.2, optimizer – Adam, activation function – ReLU, initalizer – lecun normal.

For the analysis of learning rate and β1, β2 hyperparameters another approach was taken. These parameters were investigated together due to their similarity and the roles that they have. All of them are parameters of the optimizer that influence the model flexibility that is how much the model is updated in every pass of the training phase. This approach required running the algorithm 100 times in order to check every possibility. The table above summarizes the best 5 runs. The parameters chosen to be in the final architecture of the model are the learning rate at the level of 0.001, β1 equal to 0.9 and β2 equal to 0.9999.

The whole process of choosing parameters took a long time and effort however a properly developed neural network needs both of these. A little change in the parameters can result in huge mispricing in the final stage which is out-of-sample testing. In order to prevent the model from overfitting to the in-sample data, the dropout was introduced before the second, third and fourth hidden layers. Introducing the dropout for the first hidden layer

did not result in any improvement of the neural network performance. The cross-validation was also run in order to check for possible overfitting. The results of the cross-validation, as well as out-of-sample results are left for the chapter concerning the empirical results.

4. Data description

Dataset used in the research was gathered using web scraping methods and its core parts are daily quotes of WIG20 index European options and WIG20 index from the Warsaw Stock Exchange.

4.1. Data distribution

The quotes cover the period from the beginning of January, 2009 to the end of November, 2019. Such a wide time frame of the data allows covering many different conditions on the market. 3 months Warsaw Interbank Offer Rate (WIBOR3M) was used as an estimator of risk-free interest rate, similarly to Kokoszczyński *et. al* (2010a). The options included in the dataset are all of these quoted on the Warsaw Stock Exchange (WSE), so the research covers many different strike prices and maturities. For the modeling purposes, the time to maturity was calculated in years, where one year was 252 trading days. As a dividend rate estimator a continuous dividend yield[1] from the WIG20 index was used. The historical volatility estimator was calculated using the method described in one of the previous chapters.

Due to very limited access to the derivatives data, only quotes from the opening and closing of the trading day along with the minimum and the maximum price that day were obtained. It is an objective limitation of the research that could not be resolved in any way. Nevertheless, it is assumed that such a long time frame of the data and such a great amount of data gathered are sufficient for the aim of this research and modeling based on this data can be used for verification of the stated hypotheses. For modeling purposes the close price was assumed to be the proper option's price and is then used as a true value.

A number of the records in the collected dataset is equal to 139371, where 68285 observations concern the call options and the remaining 71086 observations concern the put options. Dataset is then well balanced and a large amount of data for both put and call options is included. The Table 10 presents a summary of the descriptive statistics for the whole dataset

[1]Data obtained from: https://stooq.com/q/?s=wig_dy

Table 10. Summary statistics for chosen variables

	Mean	Standard Deviation	Minimum	Median	Maximum
Option Price	59.820	101.651	0.01	21.4	1580
WIG20 Index	2265.15	266.968	1327.64	2308.44	2932.62
Strike Price	2252.22	384.034	900	2275	3400
Interest Rate	0.027	0.013	0.016	0.017	0.058
Dividend Rate	0.029	0.011	0.011	0.030	0.060
Time to Maturity	0.352	0.330	0.000	0.250	1.460
Historical Volatility	0.1847	0.073	0.0675	0.1656	0.5087

Note: Statistics calculated for the Polish market in years 2009 – 2019. All options quoted on the WSE during that period of time are summarised.

The options prices are distributed in a very wide range between 0 and 1600. The mean price is near 60 and the median price is 21, so the distribution is uneven and probably outliers are introduced to the dataset. The WIG20 index is distributed between 1327 and 2935, which is not a wide range for a stock market index. Both dividend and interest rates do not seem to deviate a lot and remain stable. As seen in the graphs presented below, the intuition concerning the outliers in the dataset is confirmed. Prices close to 0 dominate the dataset however there are some observations with prices higher than 100. There are 1354 records with a price higher than 500 and 102 observations with a price above 1000. Nevertheless these are correct observations, so they have to remain in the dataset.

Figure 1. Histogram of the options prices and comparison of market and strike prices

Note: Plots of the options price distribution from the 10 years period on WSE. The histogram of option prices is strongly influenced by the accumulation of the observations near zero. The typical strike prices are between 1800 and 2500.

The data was also divided with respect to the moneyness states as described in chapter 2 of this paper. The Table 11 summarizes descriptive statistics for put and call options

distinguished between three moneyness states. As clearly visible, most of the observations are OTM options. There are only around 7100 ITM calls and the same amount of ITM puts. For both types, there are 25000 observations ATM both calls and puts. Moreover, the prices are the highest for ITM options, while both OTM and ITM are cheaper as all of the statistics are smaller for them.

Table 11. Summary statistics for options concerning their moneyness

Type	Moneyness	Number of options	Mean Price	Minimum Price	Maximum Price
Call	OTM	35888	14.83	0.01	256.60
	ATM	25259	59.40	0.01	400.00
	ITM	7138	265.30	30.00	1429.00
Put	OTM	40771	17.11	0.01	289.00
	ATM	23219	64.58	0.01	468.00
	ITM	7096	312.00	22.00	1580.1

Note: Summary statistics for all the options quoted on the WSE in years 2009 – 2019 with distincstion between types and moneynes.

4.2. Data preprocessing for neural network

For purposes of modeling with a neural network the dataset had to be firstly divided into two subsets. The first, larger one is called the training or in-sample set and it is used for purposes of fitting and learning the model. The second, smaller one is called the testing or out-of-sample set and it is used for validation purposes. A test sample is used in order to verify whether introducing the model to the new data will result in at least comparable results as well as check if there is overfitting to the training sample. The performance of the statistical model has to be checked on out-of-sample data to verify its correctness and goodness of fit. In this paper 80% of the initial data was used as the training set and the remaining 20% was used as a testing data on which the model was validated. There is no golden rule specifying how to divide the data and the discussion on this topic continues (Ruf and Wang, 2019). Some authors say that the time series characteristic of the data should not be interfered with, while others claim that the ability to catch different market conditions is more important. In this paper the second approach was taken and the data was split with respect to varying price distribution. This means that the intent was to feature the training data with as many different market conditions as possible. The network could then be trained on the data coming from low volatility as well as high volatility periods or low liquidity as well as high liquidity periods on the market.

Data preprocessing means preparing it for statistical modeling. There is a large variety of transformations that could be used in order to properly modify the data. The main aim of preprocessing is changing the distribution and range of the data. Applying statistical models such as neural networks typically require such preparations with respect to the characteristics and the abilities of the models. Neural networks in supervised learning applications learn in the process described above using example pairs from labeled data. When it comes to the application of neural networks in option pricing the typical transformation is scaling the data to a mean of 0 and a variance of 1 (Anders *et. al*, 1998) in the following way:

$$X_{preProcessed} = \frac{X - \overline{X}}{\sqrt{Var(X)}}$$

where X denotes a vector of values for a single variable in the dataset. The same transformation has been used in this research in order to obtain reliable and unbiased results by methods widely described in other papers.

The last issue when it comes to modeling with the use of neural networks is choosing what input should the model be fed with. In the literature many different suggestions can be found. One way is feeding the network with just spot price, strike price and time to maturity (Phani *et. al* 2011; Barunikova and Barunik, 2011). A more common approach is using option price divided by strike price and time to maturity (Zheng *et. al*, 2019; Park *et. al* 2017). The most common approach found in the literature is using spot price divided by strike price along with time to maturity, interest rate and volatility (Liu and Zhang, 2011; Hahn, 2013; Palmer and Gorse, 2017). As this paper's aims to compare the performance of a BSM model and NNs in pricing options, the machine learning model was decided to be introduced to the same data as used in the BSM model. In other words, the input variables are spot price, strike price, interest rate, continuous dividend rate, time to maturity and volatility. Moreover, the spot price was divided by strike price. Similarly the output of the neural network was chosen to be the option's price divided by its strike price. This transformation was done in order to cumulate the prices in a narrow range, because as shown above the prices of options on WSE are cumulated near 0, but many outliers are ranging even up to 1500.

5. Empirical results

5.1. Cross-Validation Results

The *k*-fold cross-validation was conducted in order to check if the overfitting was introduced to the model as well as verify its abilities without using the out-of-sample dataset. This method consists of randomly dividing the data set into *k* folds of equal size that are then used to train and test the model (James *et. al*, 2013). In typical machine learning applications, *k* is usually chosen to be between 3 and 10, depending on the size of the available dataset. The first group is left as an out-of-sample data and the remaining *k*-1 groups are used to train the model that is then validated on the data left and the error metric is calculated. This procedure is then repeated *k* times and as a result there are *k* estimates of test error that are then averaged and treated as an out-of-sample error that could be compared to in-sample errors. The main goal of cross-validation was checking whether the parameters chosen in the way described in the previous chapter are not fitted only to the in-sample data.

In this paper the number of folds that the in-sample data is divided into, was chosen to be 5. The neural network was trained 5 times on slightly different datasets and each time estimated on the remaining part of data. The results below summarize the errors calculated after validation on a single fold.

Table 12. Cross-validation training and testing errors

	Fold	1	2	3	4	5
Train						
	MAE	0.023221	0.023141	0.023278	0.023200	0.023331
	MSE	0.002539	0.002448	0.002521	0.002524	0.002496
Test						
	MAE	0.023335	0.023624	0.023052	0.023298	0.022861
	MSE	0.002418	0.002795	0.002436	0.002441	0.002501

Note: Values of the error metrics calculated after training of the neural network on a partial dataset and testing on the data left from the partition. The hyperparameters are: neurons – 500, batch size – 1000, dropout rate – 0.2, optimizer – Adam, activation function – ReLU, learning rate -0.001, β1 – 0.9, β2 – 0.9999.

The main conclusion is that no overfitting was detected, therefore it is confirmed that the model is designed correctly. The performance is very stable and errors remain low. It can be used to obtain the prices without worries about bias resulting from improper training and there is no need to redesign the architecture of the neural network or repeat the process of the hyperparameters tuning. The model evaluated in such a way is prepared to be used on out-of-sample data as a pricing tool with the given hyperparameters: neurons –

500, batch size – 1000, dropout rate – 0.2, optimizer – Adam, activation function – ReLU, learning rate -0.001, β1 – 0.9, β2 – 0.9999.

5.2. In-sample results

The in-sample results are the results obtained on the dataset that was used to train the neural network. This subgroup is 80% of the actual dataset that was used in this research. The Black – Scholes – Merton model is the benchmark model so its results will be summarized in the first order. It's worth noticing that there is no need for splitting the dataset when using the BSM model as it does not require any training. Nevertheless, the results obtained with the BSM model are used as the first benchmark of the goodness of fit for the neural network as well as an indicator of the possible error range. The Table 13 summarizes the error metrics for prices obtained using the Black – Scholes – Merton model.

Table 13. Error metrics for the BSM model prices

Type	Moneyness	MAE	MSE	RMSE	MAPE
Call	OTM	**11.293**	436.624	20.896	1.2445
	ATM	13.470	422.868	20.564	0.3764
	ITM	19.709	766.129	27.679	**0.0913**
Put	OTM	**7.529**	184.029	13.566	0.6277
	ATM	12.779	409.507	20.236	0.2733
	ITM	24.304	1197.690	34.608	**0.0904**

Note: The values of the error metrics for prices obtained using the BSM model divided between types and moneyness of the options in in-sample period

The first conclusion is that the quality of pricing with the use of the BSM model differs a lot between the types of options as well as their moneyness. For the purpose of comparison of the pricing accuracy of the model, the mean average percentage error (MAPE) will be used. The reason for that is the difference between prices. In-the-money options are much more expensive comparing to out-of-the-money options, which often have a price close to 0. MAPE allows comparing the errors with the difference in price already taken into consideration. Both call and put OTM options are priced with the highest percentage bias. For call option, the MAPE exceeds 1.22 when it comes to OTM options, while for put options it is close to 0.62. At-the-money options are priced more accurate with MAPE close to 0.37 for call options and 0.27 for put options. The ITM options are priced with the lowest percentage bias, close to 0.09 for both calls and puts.

The neural network was trained using 80% of the dataset that is 111498 observations. It is the same dataset as the one used for the results obtained in the Table 13 with the BSM

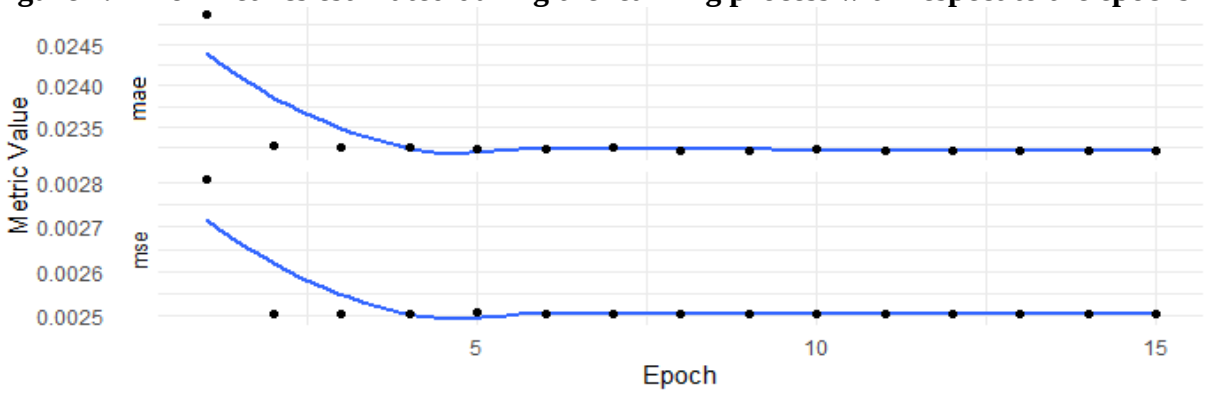
model. The number of epochs was chosen to be 15, so the learning process took 15 repeats of passing the entire set backward and forward through the algorithm. Different values of the epoch were checked in order to find the number of epochs that allows the network to converge. Too small number of epochs would not allow the network to properly train as there would not be enough data passes through the network resulting in underfitted algorithm. However a large number of epochs could cause a possibility of overfitting the network to the training data and learning the features characteristic for that particular dataset, but not for the option prices in general. The aim is to obtain a stable process of learning without visible overfitting.

Table 14. Error metrics estimated during the learning process

Epoch	MAE	MSE
1	0.02502907	0.002833899
2	0.02327030	0.002507878
3	0.02324068	0.002505204
4	0.02322462	0.002504939
5	0.02323742	0.002506087
6	0.02322323	0.002505938
7	0.02323078	0.002503588
8	0.02321084	0.002503795
9	0.02320583	0.002502452
10	0.02321385	0.002505376
11	0.02320986	0. 002503139
12	0.02321202	0.002502502
13	0.02321516	0.002503510
14	0.02321169	0.002503338
15	0.02320890	0.002503376

Note: The values of the error calculated after every epoch during the final training of the neural network with the following hyperparameters: neurons – 500, batch size – 1000, dropout rate – 0.2, optimizer – Adam, activation function – ReLU, learning rate -0.001, β1 – 0.9, β2 – 0.9999.

Figure 2. Error metrics estimated during the learning process with respect to the epochs



Note: Values of the MAE and the MSE metrics calculated after every epoch of training the neural network with the following hyperparameters: neurons – 500, batch size – 1000, dropout rate – 0.2, optimizer – Adam, activation function – ReLU, learning rate -0.001, β1 – 0.9, β2 – 0.9999.

As the Table 14 and the Figure 2 both show, the learning process remained stable without any unexpected jumps or random disruptions. The loss function stabilized with the final value around 0.02321 and the MSE metric stabilized near 0.0025. The neural network learned fast as the process started to remain stable at the 5th echo and then the loss function remained at similar values.

Nevertheless, the errors shown in the Table 14 and the Figure 2 are values just for the output of the neural network which was chosen to be option price by its strike: $\frac{C}{K}$. In order to obtain results, the transformation was turned back and error metrics for real and fitted values were calculated.

Table 15. Error metrics for the neural network model in-sample pricing

Type	Moneyness	MAE	MSE	RMSE	MAPE
Call	OTM	**22.763**	758.021	27.532	16.758
	ATM	39.516	2967.675	54.476	2.3205
	ITM	242.46	78512.48	280.201	**0.9020**
Put	OTM	**14.626**	374.482	19.351	10.508
	ATM	44.773	3683.993	60.696	1.9122
	ITM	282.897	113397	336.745	**0.8903**

Note: The values of the error metrics divided between types and moneyness of the options priced using the neural network with the following hyperparameters: neurons – 500, batch size – 1000, dropout rate – 0.2, optimizer – Adam, activation function – ReLU, learning rate -0.001, β1 – 0.9, β2 – 0.9999.

The obvious conclusion is that the accuracy of pricing is not stable between different moneyness states. The most reliable prices are obtained for the ITM options. In-the-many call options are priced with MAPE around 0.9, while the same metric for the OTM call options is close to 16.7. The OTM put options are priced with mean average percentage error close to 10.5, while for the ITM options this metric is near 0.89. Once again, at-the-money options are priced more accurate than the OTM options, but less accurate than the ITM options. At-the-money call options are priced with the MAPE around 2.3, while this metric for the put options was closer to 1.91.

When it comes to comparison of the neural network and the Black – Scholes – Merton model performance, the mean average error was used to compare accuracy of pricing for the same types of options in the same moneyness. The Tables 13 and 15 reveal that the BSM model provides much more reliable pricing than the neural network. The OTM calls are priced by the deep learning model with MAE close to 22.8 and for puts this metric is around 14.6. The same options are priced by the BSM model with MAE adequately 11.3 and 7.5.

Similarly for the at-the-money call options, the neural network pricing resulted in MAE around 39.5 and for the put options around 44.8. The BSM model priced these options with the error metric equal adequately 13.5 and 12.8. When it comes to ITM options the parametric model priced the calls with the error metric nearly 19.7 and the puts with the error metric close to 24.3. The neural network priced the ITM call options with MAE around 242.5 and the put options with MAE around 282.9. That is a huge difference between the methods and the neural network performed much worse in pricing the ITM options. The in-sample results show that the non-parametric approach is much more instable as well as its model prices are far more biased than these obtained using the BSM model.

Although the training process remained stable and no overfitting was detected in the model, the resulting in-sample pricing performance is not satisfactory. So far, the prices provided by the neural network are less reliable than these from the BSM model, as the resulting errors are higher for the deep learning model. Nevertheless, in order to verify the machine learning model the out-of-sample results had to be compared.

5.3. Out-of-sample results

The final verification of machine learning methods such as neural networks is based on the out-of-sample results. In this paper, the dataset was split in proportions of 80% for the training purposes and 20% for the testing purposes. For out-of-sample results, the remaining smaller part of the data was used. This data was not used before so that the neural network could be fed with the new input that consists of 27873 observations.

Firstly, in order to benchmark the non-parametric approach, the pricing errors obtained using the BSM model are summarized. The same dataset, namely the testing data was used to price options with the parametric approach.

Table 16. Error metrics for the BSM model prices

Type	Moneyness	MAE	MSE	RMSE	MAPE
Call	OTM	**11.009**	404.216	20.105	1.234
	ATM	13.835	445.259	21.101	0.3634
	ITM	20.173	839.290	28.970	**0.0915**
Put	OTM	**7.950**	201.196	14.184	0.6317
	ATM	13.047	409.144	20.227	0.2927
	ITM	23.863	1156.281	34.004	**0.0939**

Note: The values of the error metrics for prices obtained using the BSM model divided between types and moneyness of the options in out-of-sample dataset

In the Table 16 it is visible, that the error values vary depending on moneyness. Moreover, the similar conclusions could be drawn as for in-sample period. The ITM options are priced with the lowest MAPE while the OTM options are priced with the highest error. The OTM call options are priced with MAPE near to 1.23, while the MAPE for the OTM put options is slightly above 0.6. The ATM options are priced with a similar MAPE which is close to 0.36 for the call options and 0.3 for the put options. The MAPE metric is slightly above 0.09 for both in-the-money calls and in-the-money puts what means that options in this moneyness state are priced the most accurately.

The neural network trained on the market data was used to price options from the testing dataset. The same approach as in the training phase was used, so the output of the neural network is the options price divided by its strike price. Then the transformation was inverted to analyze the error metrics for prices. This action had to be taken up to provide comparable results as the main aim is to weigh up the pricing accuracy of the Black – Scholes – Merton model and the neural network. The market-driven approach consisted of learning the model on the example pairs from the real-world market in order to provide reliable model prices and detect as many details and patterns as possible.

Data or information leakage means that observations from outside the learning set were used in the process of model development. This often leads to bullish results and not reliable models. Data leakage is a complex problem when it comes to time series datasets and creating their partitions. The easiest way of detecting data leakage is the verification of the results. If the results are way too accurate and optimistic, probably the leakage happened and the development process should be double-checked. In this paper, the neural network was trained only on the training data that differs from the testing sample. What is more, the data transformations and cross-validation were performed to prevent the data leakage problem and its fallouts as well as overfitting the model. Both information leakage and overfitting lead to over-optimistic outcomes and useless models. It could result in failing to provide any reliable results after using these models on a completely new dataset. As in previous parts of this paper, the resulting error metrics are summarized in the table with respect to the option's type and its moneyness. The out-of-sample verification allowed checking for the occurrence of the problems described above.

Table 17. Error metrics for the neural network out-of-sample prices

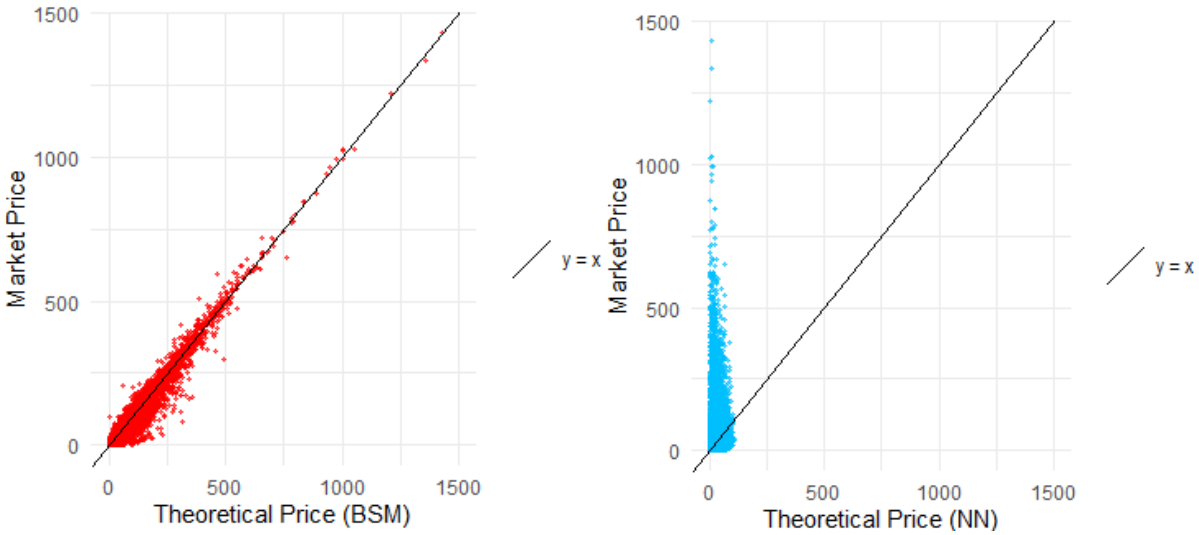
Type	Moneyness	MAE	MSE	RMSE	MAPE
Call	OTM	**22.993**	767.184	27.698	17.124
	ATM	39.781	2937.782	54.201	2.135
	ITM	246.013	80779.38	284.217	**0.9041**
Put	OTM	**14.742**	381.378	19.529	10.505
	ATM	44.745	3670.09	60.581	1.974
	ITM	277.419	107795.5	328.322	**0.8897**

Note: The values of the error metrics divided between types and moneyness of the options prices obtained using the neural network with the following hyperparameters: neurons – 500, batch size – 1000, dropout rate – 0.2, optimizer – Adam, activation function – ReLU, learning rate -0.001, β1 – 0.9, β2 – 0.9999.

Obtained results suggest that no overfitting was introduced to the neural network as obtained out-of-sample metrics are comparable to in-sample ones. Moreover the results are similarly biased and not any optimistic, so it is concluded that the data leakage is not a problem in this paper. The least reliable results in terms of MAPE are obtained for the OTM options, both the calls and the puts. For the OTM call options the MAPE metric is above 17, while for the OTM put options this error metric is equal to 10.5. The MAPE for the ATM options is equal 2.13 for the calls and 1.97 for the puts. For both in-the-money calls and puts the MAPE is close to 0.9.

Clearly the obtained results are much worse than those from the BSM model. Once again there are huge differences between the moneyness states. In addition the pattern is the same that ITM options are priced the most precisely in terms of the MAPE metric, while the OTM options are priced the worst. Comparison of the results from the Table 16 and the Table 17 shows that the BSM model priced the option more accurately. The OTM call options are priced by the BSM model with MAE close to 11 and the put options with MAE close to 8. When it comes to neural network, the error metric was adequately 23 and 14.7. The BSM model priced the ATM calls with mean average error near to 13.8 and puts with MAE close to 13. The same options were priced by the neural network with MAE adequately close to 40 and 44.75. The highest MAE was obtained for the ITM options, as the neural network priced the calls with mean average error nearly 246 and puts nearly 277.4. The same options were priced using the BSM model and the errors are close to 20 for the calls and nearly 23.9 for the puts.

Figure 3. The BSM model prices and market prices along with NN model prices and market prices with curve $y = x$



Note: The model prices from the BSM model from the out-of-sample data and the model prices from the Neural Network from the out-of-sample data with the corresponding market prices revealing the bias of the models.

The figures compare the market price with the corresponding model price obtained from the BSM model (Figure 3) and the neural network (Figure 4). The line $y = x$ is a curve determining the perfect pricing where the model price and the market price are exactly the same. For the BSM model, there are more dots below the curve what means the model prices are positively biased. Therefore the model tends to overprice options which means that the prices obtained by the parametric model are higher than the corresponding real values. Nevertheless the dots more or less follow the straight line. This conclusion, however, could not be stated when it comes to Figure 4 and the neural network pricing. The non-parametric model tends to strongly under-price the options what results in prices cumulated in the range between 0 and 75. The neural network prices are often close to 10 while the real price of the option exceeded 1000. The out-of-the-money options that are priced with the lowest errors are the cheapest options. Even though the non-parametric model was trained using the market data, it was not able to catch the similarities in order to properly price options with varying moneyness and types.

Both the tables and the analysis of the figures leads to the conclusion that the Black – Scholes –Merton model performed much more stable and provided less biased results. Firstly, the error metrics are similar between the options types and moneyness, even though they vary. Secondly, the BSM model tends to only slightly overprice the options which can be observed in Figure 3. The neural network does not provide any reliable results for pricing options and tends to under-price the options.

5.4 Discussion of the results

There could be a few reasons for such unsatisfactory results when it comes to modeling options prices with neural networks. Firstly, the data used for the training purposes that was taken from the real-world market consisted mostly of the out-of-the-money options. Above 61000 observations from the training sample were the OTM options, while only 11401 observations were in-the-money options. This might have led to fitting the model to the low prices that are typical for the OTM options, while the ITM options were not properly recognized. Some authors suggest that filtering methods should be applied to the data in order to prevent such problems (Andreou *et. al* 2006; Barunikova and Barunik 2011; Yao *et. al* 2000). There are different methods that could be applied, such as selecting only in-the-money options or selecting only options that satisfy various maturity constraints. Nevertheless such filtering the input data means narrowing the possibilities for use of the neural network in pricing of options. The main goal of developing pricing models is preparing not only a stable, but also a robust method that could be used in various market conditions. Filtering the data with many constraints for maturity, moneyness or other characteristics actually leads to mining the data as long as the results satisfy the expectations. In this paper the intention was to compare the pricing of options with the use of the parametric and non-parametric methods, therefore no filtering was introduced. The Black – Scholes – Merton model is designed to provide prices that are more or less accurate and the obtained results suggest that although the errors vary for different characteristics, they are still comparable. When it comes to neural networks, the resulting errors are way different for various characteristics and they are not comparable.

Secondly the polish derivatives market is an emerging market with typical problems for such markets that is low liquidity or non-synchronous trading (Kokoszczyński *et. al* 2010a). This leads to various price distributions on a wide range. As described in the previous section, the prices in the dataset vary between 0.01 and 1500, but are cumulated near 0. The prices above 1000 are so untypical that they could be treated as outliers, while the strike prices exceed 1000. The neural network turned out not to be robust for such a wide range of data. The main barrier is the lack of in-the-money options on the emerging markets with little participants. This is also a reason that filtering methods could not be applied as the obtained data would not be reliable for the characteristics of the market.

The research hypothesis concerning the neural network model and the Black – Scholes – Merton model turned out to be rejected. The machine learning method does not perform any better than the parametric model. The out-of-sample pricing errors summarized in Tables 5 and 6 state that for both call and put options the BSM model provides more accurate prices. The second hypothesis concerning the differences in errors between moneyness states was also rejected. The conducted modeling results reveal that the neural network tends to fit the most numerous options type and somehow ignore other observations. This leads to great differences in pricing errors for different moneyness and types. As the dataset consisted mostly of cheap, out-of-the-money options, the neural network fitted to these observations and provided low prices.

6. Conclusions

The efficiency and accuracy of the parametric Black – Scholes – Merton model and the non-parametric neural network in options pricing were verified using the data from Warsaw Stock Market. The neural network was developed in a data-driven approach which means it was designed and trained using real-world market data. The BSM model was used with the 60 days historical volatility estimator. In order to check the research hypotheses the 10 years data was split into the training sample used for modeling purposes and the testing sample used for validation of the model. The input to the neural network consisted of the same variables as theses used in the BSM model, however the value of the WIG20 index as an underlying asset was divided by the strike price of an option. The output of the neural network was the option price divided by its strike price. The output was then transformed in order to analyze and compare error metrics for prices. The results were presented for call and put options divided between three different moneyness states.

The results obtained from empirical analysis do not confirm the research hypothesis that the neural networks trained on the market data are able to outperform the BSM model. In fact, the neural network used for pricing options does not perform significantly better than the BSM model. The prices obtained using the data-driven NN are far more biased than these from the parametric model. Even though the neural network was developed with the use of the market data as well as no overfitting was introduced to the model, the prices provided by the NN are not satisfactory. The neural network is not robust to the varying market conditions typical for emerging markets such as WSE.

We cannot reject the second hypothesis that the prices provided with the neural network are not robust to the varying moneyness states. In terms of MAPE, the most accurate model prices are obtained for the ITM options and the worst prices are obtained for the OTM options. The pattern in pricing accuracy is visible in the in-sample as well as out-of-sample results.

To sum up, the conclusion is that using neural networks to price options for all maturities and moneyness states does not lead to significant improvement in pricing accuracy. The BSM model is more robust to various market conditions and provides much more stable and reliable prices. One of the possibilities to deal with the difficulties met by the neural network is applying filtering methods to the data. These are filters with constraints that are designed to create dataset with more stable and narrow data that could be used for training and testing the model. Nevertheless, this approach leads to limiting the use of the non-parametric method only to the situations when certain conditions are met. Another possibility is to develop different models for various characteristics, e.g. three NNs for three moneyness states. The results obtained by developing the neural network on the wide dataset consisting of options with different characteristics are neither satisfactory nor reliable.

Further research for machine learning methods applications in pricing options on such markets as WSE could focus on dealing with the varying accuracy between moneyness states. As suggested above either filtering data or developing the models for different conditions could be tested. Secondly, more effort could be put to the detection of the outliers and effective dealing with such observations. Moreover, in order to provide a more diversified sample, the high-frequency data could be used. Lack of in-the-money options was the main reason for such biased prices provided by the neural network.

References

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

Anders U., Korn O., Schmitt Ch., 1998, Improving the pricing of options: a neural network approach. Journal of Forecasting, 17(5-6), pp. 369-388.

Andreou P. C., Charalambous C., Martzoukos S., 2006, Robust artificial neural networks for pricing of European options. Computational Economics, 27, pp. 329 – 351.

Andreou P. C., Charalambous C., Martzoukos S. H., 2008, Pricing and trading European options by combining artificial neural networks and parametric models with implied parameters. European Journal of Operational Research, 185(3), pp. 1415 – 1433.

Andreou P. C., Charalambous C., Martzoukos S. H., 2010, Generalized parameter functions for option pricing. Journal of Banking & Finance, 34(3), pp. 633 – 646.

Bakshi G., Cao C., Chen Z., 2012, Empirical Performance of Alternative Option Pricing Models. The Journal of Finance, 52(5), pp. 2003 – 2049.

Barunikova M., Barunik J., 2011, Neural networks as semiparametric option pricing tool. Bulletin of the Czech Econometric Society, 18(28).

Bates D., 1996, Jumps and Stochastic Volatility: Exchange Rate Processes Implicit in Deutsche Mark Options. Reveiew of Financial Studies, 9(1), pp. 69 – 107.

Bates D., 2003, Empirical option pricing: a retrospection. Journal of Econometrics, 116(1-2), pp. 387 – 404.

Black F., Scholes M., 1973, Pricing of Options and Corporate Liabilites. Journal of Political Economy, 81(3), pp. 637 – 654.

Black F., 1976, The Pricing of Commodity Contracts. Journal of Financial Economics, 3(1-2), pp. 167 – 179.

Canina L., Figlewski S., 1993, The Informational Content of Implied Volatility. The Review of Financial Studies, 6(3), pp. 659 – 681.

Ferreira E., Gago M., Leon A., Rubio G., 2005, An Empirical Comparison of the Performace of Alternative Option Pricing Model. Investigaciones Economicas, 29(3), pp. 483 – 523.

Fleming J., 1998, The quality of market volatility forecasts implied by S&P 100 index option prices. Journal of Empirical Finance, 5(4), pp. 317 – 345.

Fukushima N., 1980, A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological Cybernetics, 36, pp. 193 – 202.

Gencay R., Salih A., 2003, Degree of mispricing with the Black–Scholes model and nonparametric cures. Annals of Economics and Finance, 4(1), pp. 73 – 101.

Hahn J. T., 2013, Option Pricing Using Artificial Neural Networks: An Australian Perspective. PhD thesis, Bond University.

Haug E., 1997, Complete Guide to Options Pricing Formulas. McGraw-Hill, New York.

Heston, S., 1993, A Closed-Form Solution for Options with Stochastic Volatility with Applications to Bond and Currency Options. Review of Financial Studies, 6(2), pp. 327 – 343.

McCulloch W.S., Pitts W., 1943, A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics, 5, pp. 115 – 133.

Hamid S., Habib A., 2005, Can neural networks learn the Black–Scholes Model? A simplified approach. Southern New Hampshire University, Working Paper No. 2005-01.

Herrmann R., Narr A., 1997, Neural networks and the evaluation of derivatives: some insights into the implied pricing mechanism of german stock index options, Retrieved on April 23, 2020 from http://finance.fbv.kit.edu/download/dp202.pdf

Huang W., Zhang J., 2019, Option Hedging Using LSTM-RNN: An Empirical Analysis, Beihang University.

Hull, J., A. White, 1987, The Pricing of Options with Stochastic Volatilities. Journal of Finance, 42(2), pp. 281 – 300.

Hull J., 2012, Options, Futures and Other Derivatives, 9th Edition. Pearson, Boston.

Hutchinson J., Lo A., Poggio T., 1994, A Nonparametric Approach to Pricing and Hedging Derivative Securities Via Learning Networks. The Journal of Finance, 49(3), pp. 851 – 889.

Ivakhnenko A. G., 1973, Cybernetic Predicting Devices. CCM Information Corporation, New York

James G., Hastie T., Witten D., Tibshirani R., 2013, An Introduction to Statistical Learning: With Applications in R, Springer, New York.

Jang H., Lee J., 2019, Generative Bayesian neural network model for risk-neutral pricing of American index options. Quantitative Finance, 19(4), pp. 587 – 603.

Kingma, D., Ba J., 2014, Adam: A Method for Stochastic Optimization. International Conference on Learning Representations, San Diego.

Kokoszczyński, R., Nehrebecka N., Sakowski P., Strawiński P., Ślepaczuk R., 2010a, Option Pricing Models with HF Data – a Comparative Study. The Properties of the Black Model with Different Volatility Measures. University of Warsaw, Faculty of Economic Sciences, Working Papers 3/2010.

Kokoszczyński, R., Sakowski P., Ślepaczuk R., 2010b, Midquotes or Transactional Data? The Comparison of Black Model on HF Data. University of Warsaw, Faculty of Economic Sciences, Working Papers 15/2010.

Kokoszczyński R., Sakowski P., Ślepaczuk R., 2017, Which Option Pricing Model Is the Best? HF Data for Nikkei 225 Index Options. Central European Economic Journal, 4(51), PP. 18 – 39.

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

Liu S., Oosterlee C., Bohte S., 2019, Pricing options and computing implied volatilities using neural networks. Risks, 7(16), pp. 1 – 22.

Liu D., Zhang L., 2011, Pricing Chinese warrants using artificial neural networks coupled with Markov regime switching model. International Journal of Financial Markets and Derivatives, 2(4), pp. 314 – 330.

Malliaris M., Salchenberger L., 1993, A neural network model for estimating option prices. Applied Intelligence, 3(3), pp. 193 – 206.

Merton R., 1973, Theory of Rational Option Pricing. The Bell Journal of Economics and Management Science, 4(1), pp. 141 – 183.

Mitra S., 2012, An option pricing model that combines neural network approach and Black Scholes formula. Global Journal of Computer Science and Technology.

Palmer S., Gorse D., 2017, Pseudo-analytical solutions for stochastic options pricing using Monte Carlo simulation and breeding PSO-trained neural networks. European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, Bruges.

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), pp. 5227 – 5237.

Phani B., Chandra B., Raghav V., 2011, Quest for efficient option pricing prediction model using machine learning techniques. The 2011 International Joint Conference on Neural Networks, San Jose.

Raj M., Thurston D., 1998, Transactions Data Examination of the Effectiveness of the Black Model for Pricing Options on Nikkei Index Futures. Journal of Financial and Strategic Decisions, 11, pp. 37 – 45.

RStudio Team, 2015, RStudio: Integrated Development for R. RStudio Inc., Boston.

Ruf J., Wang W., 2019, Neural Networks for Option Pricing and Hedging: A Literature Review. SSRN Electronic Journal.

Ślepaczuk R., Zakrzewski G., 2013, High-Frequency and Model-Free Volatility Estimators. Lap Lambert Academic Publishing.

Van Rossum G., Drake, F.L., 2009, Python 3 Reference Manual. Scotts Valley, CA: CreateSpace.

Wang P., 2011, Pricing currency options with support vector regression and stochastic volatility model with jumps. Expert Systems with Applications, 38(1), pp. 1 – 7.

Werbos P., John P., 1974, Beyond regression: new tools for prediction and analysis in the behavioral sciences. Ph.D. Thesis, Harvard University.

Yang Y., Zheng Y., Hospedales T., 2017, Gated neural networks for option pricing: rationality by design. Association for the Advancement of Artificial Intelligence, pp. 52 – 58.

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

Zheng Yu, Yang Y., Chen B., 2019, Gated neural networks for implied volatility surfaces. Retrieved on June 17, 2020 from https://arxiv.org/abs/1904.12834

University of Warsaw
Faculty of Economic Sciences
44/50 Długa St.
00-241 Warsaw
www.wne.uw.edu.pl