Artificial Neural Networks Performance in WIG20 Index Option Pricing

Maciej Wysocki and Robert Ślepaczuk
QFRG and DSLab Monthly Meetings

Quantitative Finance Research Group
Faculty of Economic Sciences at University of Warsaw

14-12-2020

Agenda

- Motivation, hypotheses and research questions
- Literature review
- Methodology
  - Terminology and Metrics
  - Black – Scholes – Merton Model
  - Artificial Neural Network
    - Architecture of Artificial Neural Networks
    - Backpropagation and optimization
    - Hyperparameters tuning
- Data description
  - Data distribution and data preprocessing for neural network
- Empirical results
  - Cross-Validation
  - In-sample and out-of-sample results
- Conclusions and research extensions

Motivation I

The main aims:

- the exploration of deep learning possibilities in option pricing and
- the analysis of the market data-driven approach for NNs training.
- to design the proper architecture of the NN for data-driven approach and test its performance comparing to the traditional BSM model.

because

none of the previous works covered the topic of **ML approach to pricing derivatives on emerging markets with relatively low liquidity and high volatility.**

and . . .

Motivation II

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.**

Hypotheses and research questions

First Hypothesis:

The neural networks trained on real-world market data are able to perform better than the Black – Scholes - Merton model in terms of pricing errors.

Second Hypothesis:

One can observe a difference in pricing errors of the neural network taking into account the moneyness ratio.

Software, Libraries, Hardware

- 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.
- Computer specification: Intel Core i7-6700 2.60GHz, 16GB RAM, NVIDIA GeForce GTX950M.
- One full training (15 epochs) lasted around 2 minutes.

Literature review

- **Black and Scholes (1973), Merton (1973) -> BSM model**
- Heston (1993), Hull and White (1987), Bates (1996) -> introduce stochastic volatility
- Bakshi et al. (1997), Bates (2003) -> drawbacks of BSM model
- Kokoszczyński et al. (2010a) & Kokoszczyński et al. (2017) -> Polish & Japanese options market, different moneyness ratios (MR), various TTM
- **Malliaris and Salchenberger (1993), Hutchinson et al. (1994) -> the first attempt to use of ANNs for the purpose of derivatives pricing**
- Herrmann and Narrr (1997), Mitra (2012), Palmer and Gorse (2017) -> 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
- Amilon (2003), Gencay and Salih (2003), Andreou et al. (2006; 2008; 2010), Hahn (2013) -> used transformed spot price and strike price
- Park et al. (2014) -> non-parametric machine learning methods significantly outperformed the BS model
- **Jang and Lee (2019) or Yang et al. (2017) -> NNs perform at least as good as BSM model**
- Huang and Zhang (2019) -> recent papers concerning the evaluation of NNs

Terminology and Metrics

Methodology. Terminology and Metrics

- **European** style put and call options
- Moneyness ratio in order to define: OTM, ATM and ITM states, calculated according to the formula:

$$MR = \frac{S}{K * e^{-r*t}} \tag{1}$$

- MR for call: OTM$[0; 0.95)$, ATM$[0.95; 1.05)$, ITM$[1.05; +\infty]$
- MR for PUT in reverse order
- Error metrics for model evaluation: **MAE, MSE**, RMSE, MAPE

Black – Scholes – Merton Model

Methodology. Black – Scholes – Merton Model I

- Black – Scholes – Merton (BSM) model assumes **the stock price follows a Geometric Brownian Motion with constant drift and volatility**
- Most of the **inital assumptions were found to be unrealistic** and they were removed or relaxed by further works
- The model is based on **the partial differential equation**, namely the Black – Scholes (Black and Scholes, 1973) equation (2):

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

Methodology. Black – Scholes – Merton Model II

- The **prices of European options** can be obtained using the following formulas (Black and Scholes, 1973):

$$P_c(S_t, \tau) = S_t * e^{-q*\tau} * N(d_1) - K * e^{-r*\tau} * N(d_2) \tag{3}$$

$$P_p(S_t, \tau) = K * e^{-r*\tau} * N(-d_2) - S_t * e^{-q*\tau} * N(-d_1) \tag{4}$$

$$d_1 = \frac{\ln(\frac{S_t}{K}) + (r - q) * \tau}{\sigma\sqrt{\tau}} + \frac{\sigma\sqrt{\tau}}{2} \tag{5}$$

$$d_2 = \frac{\ln(\frac{S_t}{K}) + (r - q) * \tau}{\sigma\sqrt{\tau}} - \frac{\sigma\sqrt{\tau}}{2} = d_1 - \sigma\sqrt{\tau} \tag{6}$$

Methodology. Black – Scholes – Merton Model III

- **Volatility is not directly observed** on the market
- We use **HV estimator**:

$$HV_n = \sqrt{T}\sqrt{\frac{1}{n-1}\sum_{t=0}^{n}(u_t - \bar{u})^2}; u_t = \ln\frac{S_t}{S_{t-1}} \tag{7}$$

- T equals 252 trading days per year
- **n - memory of the process equals 60 trading days**

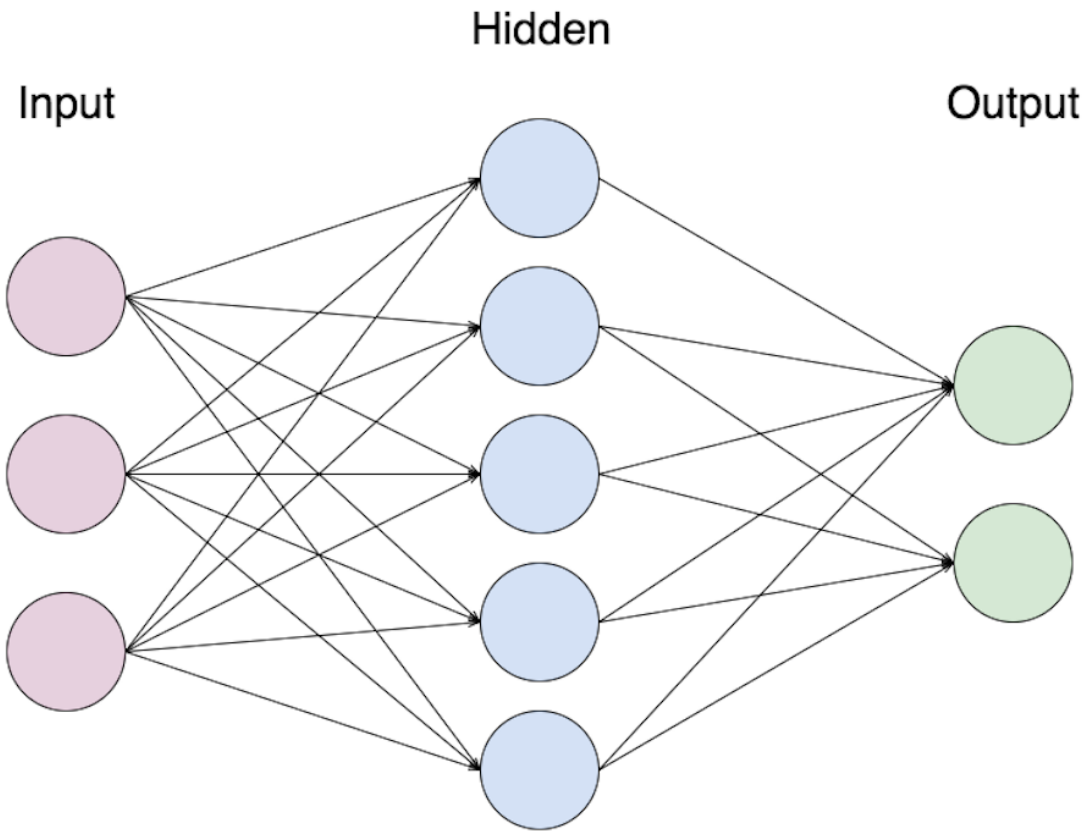
Architecture of Artificial Neural Networks

Methodology. Artificial Neural Network. Architecture of Artificial Neural Networks I

- **ANNs are a non-parametric approach** that links the input with the output.
- The non-parametric models **react to any new conditions with changing the black-box functional form**.
- **ANNs - supervised learning - labeled data**
- The learning process, **is 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 - **multilayer perceptron (MLP) - feedforward network**, which consists of **at least 3 layers** of neurons: the input layer, the output layer and at least one hidden layer between them.

Methodology. Artificial Neural Network. Architecture of Artificial Neural Networks II

Figure 1. Architecture of Artificial Neural Network



Source: https://medium.com/@jamesdacombe/an-introduction-to-artificial-neural-networks-with-example-ad459bb6941b

Methodology. Artificial Neural Network. Architecture of Artificial Neural Networks III

- The basic component of ANNs is a **neuron**. 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.
- 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.
- 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.**
- **The output of the activation function from each neuron is then passed further** to the next layer of neurons.

Methodology. Artificial Neural Network. Architecture of Artificial Neural Networks IV

- **The activation function's task is introducing non-linearity to the model**.
- 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 loss function which 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 loss 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.**

Methodology. Artificial Neural Network. Backpropagation and optimization

- **Backpropagation (Werbos, 1974) is an algorithm used in training MLPs** for supervised learning problems.
- The **algorithm calculates the loss function's gradient with respect to each weight** during the training process.
- **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.**

Hyperparameters tuning

Methodology. Artificial Neural Network. Hyperparameters tuning I

- 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 NN architecture, a variety of such parameters have to be chosen. As the option pricing is a supervised regression problem, **the chosen loss function to be minimized is MAE.**
- The basic hyperparameters are:
  - the **number of layers** in a neural network
  - a **number of neurons** in each layer.
- Both of them have to be defined at the beginning of architecture design.

Methodology. Artificial Neural Network. Hyperparameters tuning II

- 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 of layers 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, so we had 4 hidden layers.
- The search for the optimal number of layers **started from one hidden layer** and consequently networks up to six hidden layers were tested.
- **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 loss function.

Methodology. Artificial Neural Network. Hyperparameters tuning III

- **For the first stage of neural network architecture design,** the number of neurons was chosen along with the activation functions, a batch size, a dropout rate, the number of epochs 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: number of neurons, batch size, dropout rate, optimizer, initializer, learning rate, β_1, β_2.
- **The stability of the process means that no sudden jumps upward or downward of the error metrics were observed during the training process.**

Methodology. Artificial Neural Network. Hyperparameters tuning IV

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 indicated in Table 1 were checked** (5 * 6 * 4 * 2 * 4 = 960).

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

Parameter	Options or Range
Neurons (each layer)	250, 500, 1000, 1500, 2000
Batch Size	250, 500, 1000, 1500, 2000, 2500
Epochs	15
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

Methodology. Artificial Neural Network. Hyperparameters tuning V

Then the results were summarized and **the set that resulted in the lowest value of the loss function was chosen**. This approach allowed designing an **initial framework of NN that gave stable and satisfactory results** for that moment. Nevertheless, a tuning of hyperparameters along with preventing from overfitting had to be done.

Table 2. Hyperparameters of the NN framework before tuning phase

Parameter	Options or Range
Neurons (each Layer)	1000
Batch Size	1500
Epochs	15
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.

Methodology. Artificial Neural Network. Hyperparameters tuning VI

- 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 was set to 5** so that the algorithm responsible for updating weights runs 5 times during a single training process. **The change in the number of epochs from 15 to 5 was done after careful analysis of initial learning process and its error estimates**. Moreover, reducing the number of epochs allowed to substantially speed up the tuning of parameters. In this way, the optimal values could be found for each of the hyperparameters.
- Different models were compared using **the loss function (MAE) and MSE.**

Methodology. Artificial Neural Network. Hyperparameters tuning VII

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
Activation Function	ReLU
Epochs	5
Initializer	Random Normal, Random Uniform, Glorot Normal, Glorot Uniform, Lecun Normal
Learning Rate	0.0001, 0.005, 0.001, 0.005, 0.01
$\beta 1$	0.75, 0.8, 0.9, 0.95, 0.975
$\beta 2$	0.95, 0.975, 0.999, 0.9999

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

Methodology. Artificial Neural Network. Hyperparameters tuning VIII. Neurons

The number of neurons was chosen to be the same for every layer. Although the initial value of neurons was selected from a similar range, it was decided to check for different values once again in order to confirm and double-check the results.

Table 4. The number of neurons and values for error metrics.

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 neurons 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 loss 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.

Methodology. Artificial Neural Network. Hyperparameters tuning IX. Batch size

When it comes to the batch size, the optimal value was different as the one chosen in the first stage.

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.

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.

Methodology. Artificial Neural Network. Hyperparameters tuning X. Dropout Rate

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.

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.

Methodology. Artificial Neural Network. Hyperparameters tuning XI. Optimizer

The results differ a little between optimizers, especially when it comes to comparing errors. The worst results were obtained for Adadelta and SGD, better results were obtained using Nadam, while the best results were obtained for Adam, Adamax and Adagrad.

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.

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.

Methodology. Artificial Neural Network. Hyperparameters tuning XII. Initializer

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.

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 final method of random weights assignment was chosen to be **lecun normal due to its monotonically decreasing error metrics.**

Methodology. Artificial Neural Network. Hyperparameters tuning XIII. Learning Rate

For the analysis of learning rate and $\beta 1$, $\beta 2$ 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, $\beta 1$ equal to 0.9 and $\beta 2$ equal to 0.9999.**

Table 9. The learning rate, $\beta 1$, $\beta 2$ and error metrics evaluated after each training process

Learning Rate	$\beta 1$	$\beta 2$	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
0.0001	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.

Methodology. Artificial Neural Network. Hyperparameters tuning XIV

- **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.

Data description

Data description. Data distribution I

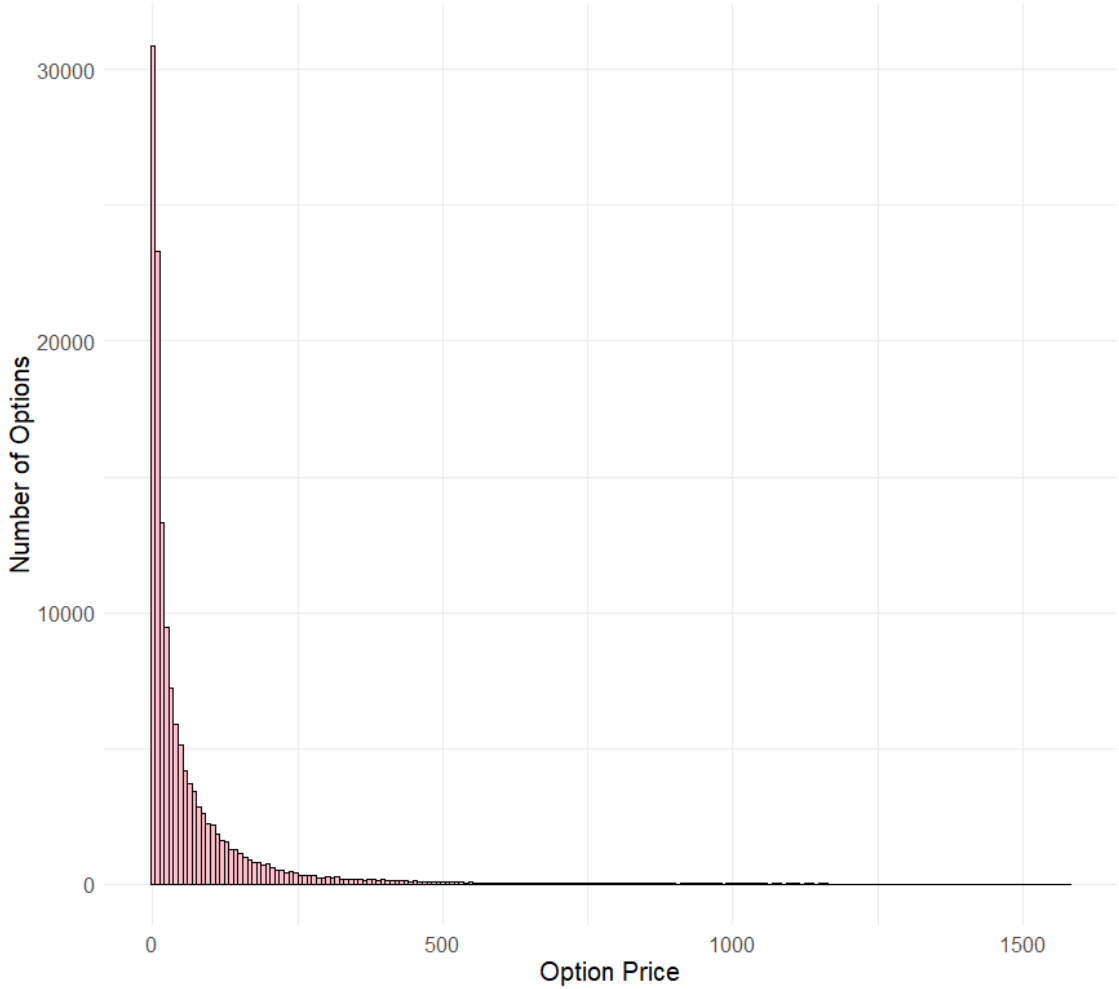
- Data sources: stooq and Warsaw Stock Exchange
- Data period: 01-01-2009 to 30-11-2019
- Put and call WIG20 index european options with all available strikes and maturities
- Daily close price options
- Risk-free interest rate: WIBOR3M
- Dividend rate: WIG20 dividend yield
- Historical volatility estimator
- Database consists of 139371 option prices, with 68285 call options and 71086 put options
- Training data: 80% of the dataset, split with respect to varying price distribution
- Testing data: 20% of the dataset, split with respect to varying price distribution

Data description. Data distribution II

- **The options prices are distributed in a very wide range between 0 and 1600.**
- **Prices close to 0 dominate the dataset however there are some observations with prices higher than 100.**
- 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.
- There are 1354 records with a price higher than 500 and 102 observations with a price above 1000.

Data description. Data distribution III

Figure 2. Histogram of the options prices



Note: Plot 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.

Data description. Data distribution IV

Figure 3. Comparison of market and strike prices

Note: Plot of the options price distribution from the 10 years period on WSE. The typical strike prices are between 1800 and 2500.

Data description. Data distribution V

Table 11. Summary statistics for options concerning their moneynes

Type	Moneyness	No. of Options	Mean Price	Min Price	Max Price
Call	OTM	35888	14.83	0.01	256.6
	ATM	25259	59.40	0.01	400.0
	ITM	7138	265.30	30.00	1429.0
Put	OTM	40771	17.11	0.01	289.0
	ATM	23219	64.58	0.01	468.0
	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.

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 ATM and OTM are cheaper as all of the statistics are smaller for them.

Data description. Data preprocessing for neural network

- Data was split into training and testing samples based on options price distribution
- The intent was to feature the training data with as many different market conditions as possible
- Scaling the data to a mean of 0 and variance of 1:

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

- Time-to-maturity calculated in years consisting of 252 trading days
- **Spot price was divided by the strike price as the input price to ANN**
- Inputs to ANN: spot price, strike price, interest rate, dividend rate, time to maturity, volatility
- **The output of the neural network is the option's price divided by its strike price**

Black - Scholes - Merton Model

Empirical Results. In-sample BSM Model I

Table 12. Error metrics for the BSM model prices on in-sample data

Type	Moneyness	MAE	MSE	RMSE	MAPE
Call	OTM	**11.293**	436.624	20.896	1.2445
	ATM	13.47	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.

Empirical Results. In-sample BSM Model II

BSM Model Results

- The quality of pricing with the use of the BSM model differs a lot between the types of options.
- The accuracy of pricing is not stable between different moneyness states.
- Both call and put OTM options are priced with the highest percentage bias.
- In-the-money options are priced with the lowest percentage bias, close to 0.09 for both calls and puts.

Artificial Neural Network

Empirical results. 5-Fold Cross-Validation Results

Cross Validation Summary

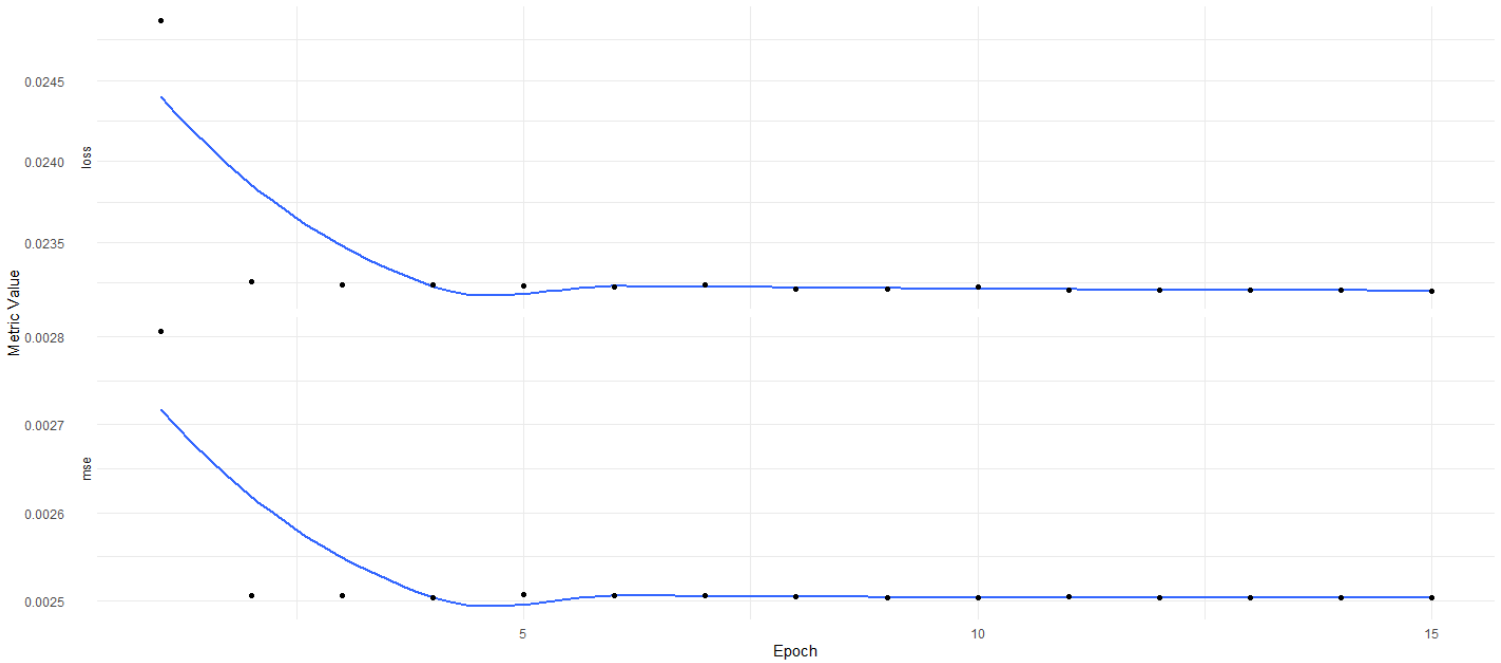
- No overfitting was introducted to the model.
- The model structure was desgined correctly.
- The performance of the model is stable on slightly different datasets.

Final set of hyperparameters

- Neurons: 500
- Batch size: 1000
- Dropout rate: 0.2
- Optimizer: ADAM
- Activation function: ReLU
- Epochs $= 5$
- Learning rate: 0.001
- $\beta 1$: 0.9
- $\beta 2$: 0.9999

Empirical results. Learning Error Metrics

Figure 4. Error metrics estimated during the learning process with respect to the epochs



Note: Values of MAE and MSE 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, $\beta 1$ – 0.9, $\beta 2$ – 0.9999.

Empirical results. In-sample ANN I

Table 15. Error metrics for the ANN model prices on in-sample data

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	78512.480	280.201	**0.902**
Put	OTM	**14.626**	374.482	19.351	10.508
	ATM	44.773	3683.993	60.696	1.9122
	ITM	282.897	113397.000	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, epochs = 5, learning rate -0.001, $\beta 1$ – 0.9, $\beta 2$ – 0.9999.

Empirical results. In-sample ANN II

ANN Model In-Sample Results Summary

- The quality of pricing with the use of the ANN model differs a lot between the types of options.
- The accuracy of pricing is not stable between different moneyness states.
- The most reliable prices are obtained for the ITM options.
- ATM options are priced more accurate than the OTM options, but less accurate than the ITM options.

ANN and BSM Models In-Sample Comparison

OTM:

- Calls are priced by ANN 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

ATM:

- The ANN model pricing resulted in MAE around 39.5 and for the put options around 44.8
- BSM model priced options with MAE equal adequately 13.5 and 12.8

ITM

- ANN priced the ITM call options with MAE around 242.5 and the put options with MAE around 282.9
- BSM model priced the calls with the MAE nearly 19.7 and the puts with MAE close to 24.3

Black - Scholes - Merton Model

Empirical results. Out-of-sample BSM Model I

Table 16. Error metrics for the BSM model prices on out-of-sample data

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	**7.95**	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.

Empirical results. Out-of-sample BSM Model II

BSM Model Results Summary

- Error metrics values vary depending on type of an option.
- The accuracy of pricing is not stable between different moneyness states.
- ITM options are priced with the lowest MAPE.
- OTM options are priced with the highest percentage bias.

Artificial Neural Network

Empirical results. Out-of-sample ANN I

Table 17. Error metrics for the ANN model prices on out-of-sample data

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	80779.380	284.217	**0.9041**
Put	OTM	**14.742**	381.378	19.529	10.505
	ATM	44.745	3670.090	60.581	1.974
	ITM	277.419	107795.500	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, epochs = 5, learning rate -0.001, $\beta 1$ – 0.9, $\beta 2$ – 0.9999.

Empirical results. Out-of-sample ANN II

ANN Model Results Summary

- No overfitting was introduced to ANN as obtained out-of-sample metrics are comparable to in-sample ones.
- The accuracy of pricing is not stable between different moneyness states and option types.
- ITM options are priced with the lowest MAPE.
- OTM options are priced with the highest percentage bias.

ANN and BSM Models Out-of-Sample Comparison

OTM:

- Calls are priced by ANN model with MAE close to 23 and for puts this metric is around 14.7
- Call options are priced by the BSM model with MAE around 11 and the puts with MAE around 8

ATM:

- The ANN model pricing resulted in MAE around 40 and for the put options around 44.75
- BSM model priced options with MAE equal adequately 13.8 and 13

ITM

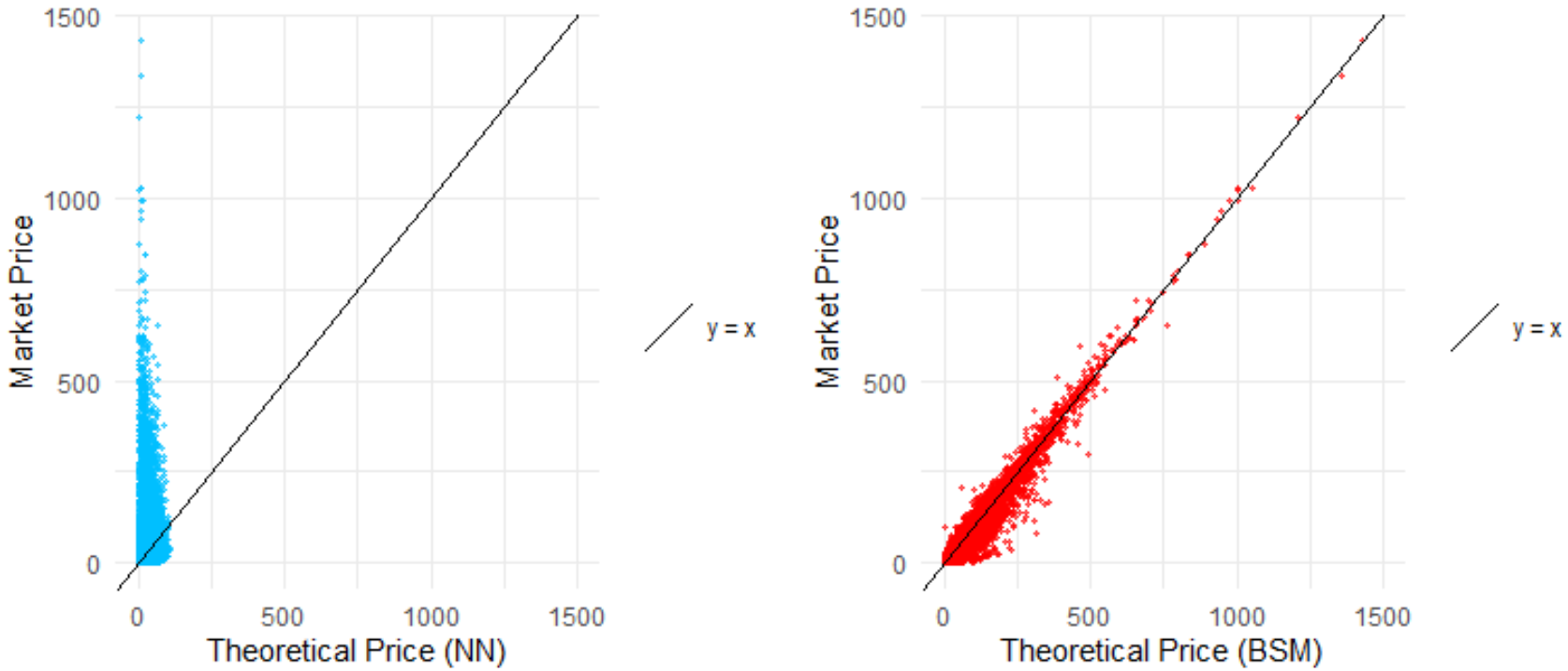
- ANN priced the calls with mean average error nearly 246 and puts nearly 277.4
- BSM model priced the calls with the MAE around 20 and the puts with MAE around 23.9

Empirical results. Discussion I

- The BSM model performed more stable and provided much more reliable option prices than the Neural Network.
- The in-sample and out-of-sample performance of ANN is not satisfactory, moreover there are differences between the moneyness states.
- Above 61000 observations from the training sample were the OTM options, while only 11401 observations were ITM options.
- Options prices above 1000 are so untypical that they could be treated as outliers, while the strike prices exceed 1000.
- Neural network turned out not to be robust for such data and conditions.

Empirical results. Discussion II

Figure 5. 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.

Research Hypotheses Verification

- The research hypothesis concerning the ANN model and the Black – Scholes – Merton model turned out to be rejected. The ML method does not perform any better than the parametric model. The out-of-sample pricing errors 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.

Conclusions

- We compared the accuracy of Black – Scholes – Merton model and ANN in options pricing using the data from Warsaw Stock Market.
- ANN was developed in a data-driven approach which means it was designed and trained using real-world market data which was split into training and validation samples.
- Prices obtained using the ANN model are far more biased than these from the BSM model.
- ANN is not robust to the varying market conditions and moneyness states.
- Using neural networks to price options for all maturities and moneyness states does not lead to significant improvement in pricing accuracy.

Research extensions

- Deal with unbalanced dataset, by applying filtering methods.
- Some authors suggest selecting only ITM options oroptions that satisfy various maturity constraints (Andreou et. al 2006; Barunikova and Barunik 2011; Yao et. al 2000).
- Develop models for each moneyness state and/or option type.
- Provide more diversified sample with high-frequency data that could be used with filtering techniques.
- Outliers detection and dropping.

References I

- Amilon H., 2003, A neural network versus Black–Scholes: a comparison of pricing and hedging performances. Journal of Forecasting, 22(4), pp. 317–335.
- 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.
- 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.
- Herrmann R., Narr A., 1997, Neural networks and the evaluation of derivatives: some insights into the implied pricing mechanism of german stock index options.
- 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.
- 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.

References II

- 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.
- 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.
- 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.
- 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., 2017, Which Option Pricing Model Is the Best? HF Data for Nikkei 225 Index Options. Central European Economic Journal, 4(51), PP. 18–39.
- Liu S., Oosterlee C., Bohte S., 2019, Pricing options and computing implied volatilities using neural networks. Risks, 7(16), pp. 1–22.
- Malliaris M., Salchenberger L., 1993, A neural network model for estimating option prices. Applied Intelligence, 3(3), pp. 193–206.
- 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.
- 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.

Thank you for your attention!

Maciej Wysocki, m.wysocki9@student.uw.edu.pl

Quantitative Finance Research Group

and

Robert Ślepaczuk, rslepaczuk@wne.uw.edu.pl,

Faculty of Economic Sciences,

University of Warsaw,

Department of Quantitative Finance

Quantitative Finance Research Group