

# APPLICATION OF ARTIFICIAL NEURAL NETWORKS IN OPTION PRICING AND HEDGING

JERIEL WONG KWAN KIT

QU WENQIN

QF643 | Final Presentation



# AGENDA

- Abstract
- Literature Review
- Method
- Results
- Conclusion and Future Work
- Reference



# ABSTRACT

In recent studies, the non-parametric models such as artificial neural networks (ANN) achieves better pricing performances than the parametric models such as Black-Scholes (BS) pricing formula. This report compares the ANN and the BS pricing formula, first regarding pricing and second regarding delta hedging strategy performance, and aims to explore if the positive results of the previous literature can be replicated using the recent real market data, specifically the European VIX call options from January 2021 to December 2021.

The result shows that the performance of the ANN is close to that of the BS formula in option pricing and delta hedging performance when using the recent real-world European call options data.



# LITERATURE REVIEW

# LITERATURE REVIEW - 1

Ruf and Wang (2020) provide a comprehensive review on the academic literature of the ANN based option pricing and hedging with a statistical perspective.

Components	Key Observations
Input Features	<ul style="list-style-type: none"><li>• Using the underlying price, <math>S_t</math> and the option strike price, <math>X</math> separately</li><li>• Only using their ratio, <math>S_t/X</math> (moneyness)<ul style="list-style-type: none"><li>- Reduces the number of inputs and thus makes the training of the ANN easier</li><li>- Helps generalization and reduces overfitting</li></ul></li><li>• Time to maturity, volatility (historical, implied, GARCH generated, etc.), interest rate</li></ul>
Outputs of ANN	<ul style="list-style-type: none"><li>• The most common output is the option price<ul style="list-style-type: none"><li>- Depending on whether underlying price and strike price are used separately or moneyness is used, the output can be the option price, or the option price divided by the strike price</li><li>- Hybrid ANN; validate to hedging errors; regulation techniques</li></ul></li><li>• Implied volatility - converted to option prices by the Black-Scholes formula</li><li>• Sensitivity / Hedging ratio</li></ul>
Benchmark Models	<ul style="list-style-type: none"><li>• Single period: MAE, MAPE, MSE</li><li>• Multiple period: mean absolute tracking error (MATE) and prediction error (PE)</li><li>• Compare to a benchmark: parametric pricing model<ul style="list-style-type: none"><li>- The most widely used benchmark is the BS pricing formula</li><li>- Stochastic volatility pricing models</li></ul></li></ul>
Data Partition	<ul style="list-style-type: none"><li>• Chronologically - the early data constitutes the training set, and the late data constitutes the test set</li><li>• Randomly - breaks the time structure and introduces information leakage between the training set and the test set</li></ul>
Underlying Assets	<ul style="list-style-type: none"><li>• Simulation data - free of noise and sometimes a close-to-optimal solution is available as a benchmark</li><li>• Most other papers use either both simulation and real data or only real data (options on S&amp;P500, FTSE100 and S&amp;P100)</li></ul>
Regularisation Techniques	<ul style="list-style-type: none"><li>• As the advance of hardware allows for bigger ANNs to be built, regularization techniques have become more important as part of the ANN training. (e.g., L2, dropout, early stopping, etc.)</li></ul>

# MOTIVATION

<p>Why are we trying to fight against a traditional Black-Scholes Model</p>	<ul style="list-style-type: none"> <li>• The dependencies of the Black-Scholes Model are typical examples of traditional parametric models. For example, the volatility has an essential role in the Black-Scholes Model and therefore, it is important to compare different volatility estimation models to guarantee the lowest mispricing in the Black-Scholes pricing prediction.</li> <li>• An inaccurately specified formula could lead to pricing errors. Since changes in economic and market conditions may influence the option price, introducing a data-driven and non-parametric model such as Neural Networks could reduce the pricing error by learning the dynamics behind the option price and minimizing the uncertainty in parametric assumptions.</li> </ul>
<p>Why are we considering delta hedging</p>	<ul style="list-style-type: none"> <li>• A good pricing prediction is not sufficient to support the practical relevance of the non-parametric model. Since the Black-Scholes is a no-arbitrage-based pricing formula, it is important to prove the ability of the model in replicating the option through a dynamic hedging strategy, for instance, delta hedging.</li> </ul>
<p>Pros and Cons of Neural Networks and Black-Scholes pricing formula</p>	<ul style="list-style-type: none"> <li>• <b>Neural network</b></li> <li>• Pros <ul style="list-style-type: none"> <li>- Do not assume log-normality or sample path continuity</li> <li>- Robust to any specification errors applied</li> <li>- Adaptive to structural changes/behaviour from the market</li> </ul> </li> <li>• Cons <ul style="list-style-type: none"> <li>- Data intensive</li> </ul> </li> <li>• <b>Black-Scholes</b></li> <li>• Pros <ul style="list-style-type: none"> <li>- Quick speed in pricing as it uses a mathematical formula to calculate the price of an option</li> </ul> </li> <li>• Cons <ul style="list-style-type: none"> <li>- Neglects important factors, e.g., changes in volatility, external shocks, market regulator actions</li> <li>- Based on many assumptions</li> </ul> </li> </ul>

# LITERATURE REVIEW - 2

Hutchinson et al. (1994) is one of the first papers that proposes to use the ANN method to pricing options in a non-parametric way and to challenge the use of the parametric BS formula and introduce a methodology to evaluate the hedging performance over multiple periods. A remarkable result was shown in this paper.

Components	Key Observations
Objectives	<ul style="list-style-type: none"> <li>Can a neural network learn BS pricing model just through data points?</li> <li>Using inputs to neural network to produce an option price, continuous variable output.</li> <li>To attain a delta-hedge value, to evaluate if the option price produced is a better price than the traditional BS price, as BS aims is to achieve a closed-form solution for option price.</li> </ul>
Assumptions made for testing	<ul style="list-style-type: none"> <li>Interest rate and vol are fixed – this does not mean <math>r</math> and <math>v</math> cannot vary, even if it varies, the neural network will adapt.</li> <li>The statistical distribution of the stock return is independent to the stock price.</li> </ul>
Solution to Problem Statement - Performance Metrics	<ul style="list-style-type: none"> <li>The paper uses 3 metrics to determine if neural network can match BS price: <math>R^2</math>, residual risk and prediction error.</li> <li>By reinforcing the theory of a replicating portfolio, i.e., must hold, where <math>V(P) = V(C) + V(B) + V(S)</math>, and the output price of the neural network is assumed to be a delta-hedge option price, the residual risk would then be the expected value of <math>V(P)</math>. As such with the stated output/option price, it is assumed to be the delta-hedge option price, where the stock and bond position will be rebalanced to satisfy the replicating portfolio.</li> <li>Prediction error = <math display="block">\eta \equiv e^{-rT} \sqrt{E^2[V(T)] + \text{Var}[V(T)]},</math></li> </ul>
Input Features	<ul style="list-style-type: none"> <li>Stock Price / Strike (<math>St/X</math>)</li> <li>Time left to maturity (<math>T-t</math>)</li> </ul>
Simulation and Model	<p>Part 1 = Train-Test Split 80-20 : Actual Black-Scholes versus Predict Neural Network – Option Price</p> <ul style="list-style-type: none"> <li>Train-test split the data in chronological order, in 80% train and 20% test, comparing the neural network's price at different classes, ITM/OTM, different strike levels to evaluate the neural network's option price output</li> </ul> <p>Part 2 = Delta-Hedge comparison between Black-Scholes and NN using price paths</p> <ul style="list-style-type: none"> <li>Monte-Carlo simulation would mean multiple simulated prices (aka price paths) where a stock price can take given that the stock price can be modelled by a stochastic differential equation ( Geometric Brownian Motion )</li> <li>The generalized purpose is to determine the delta-hedge results comparing between Black-Scholes and the neural network. As the neural network coefficients would represent the coefficients of stock and bond in the portfolio, hence able to determine the quantity of stock and bond</li> </ul>



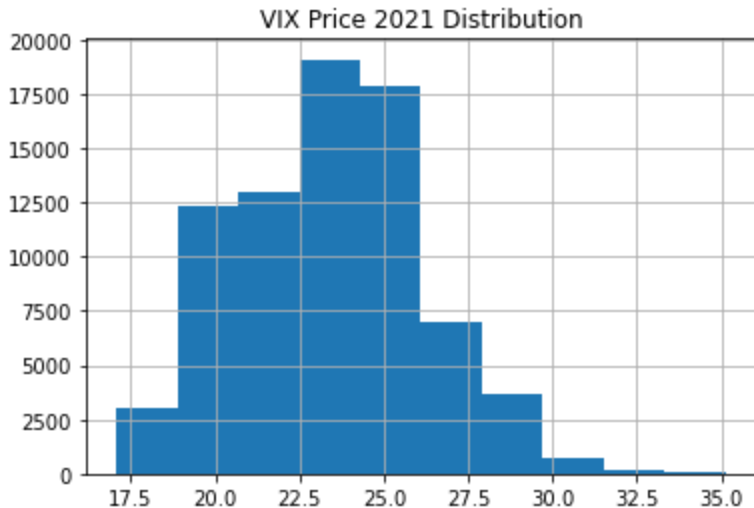
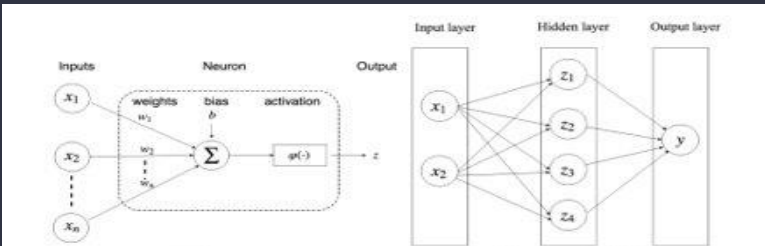


# PART I – OPTION PRICING - METHOD



# BUILD-UP

## Call Options

Interest Rate	Max VIX Price for 2021 = 35.00	Input Features + Data Limitations	ANN – Model (MLP)																								
3-Month Treasury Bill [ Monthly Average ]	Impact = Limiting Strikes to 50 Market Option Prices for OTM Calls are < \$0.10, Inclusion would affect Predicted Option Price outcome	Features 1) Stock / Strike 2) T-t Data Limitations Best Bid and Best Offer x5	2 Inputs 1 Layer – 4 Nodes 1 Output																								
Use to reverse calculate Stock Price  Monthly Average		<table><thead><tr><th>T-t</th><th>S/X</th></tr></thead><tbody><tr><td>0.063492</td><td>2.627350</td></tr><tr><td>0.063492</td><td>2.502238</td></tr><tr><td>0.063492</td><td>2.388500</td></tr><tr><td>0.063492</td><td>2.284652</td></tr><tr><td>0.063492</td><td>2.189458</td></tr><tr><td>...</td><td>...</td></tr><tr><td>0.130952</td><td>0.539298</td></tr><tr><td>0.130952</td><td>0.505592</td></tr><tr><td>0.130952</td><td>0.475851</td></tr><tr><td>0.130952</td><td>0.449415</td></tr><tr><td>0.130952</td><td>0.425762</td></tr></tbody></table>	T-t	S/X	0.063492	2.627350	0.063492	2.502238	0.063492	2.388500	0.063492	2.284652	0.063492	2.189458	...	...	0.130952	0.539298	0.130952	0.505592	0.130952	0.475851	0.130952	0.449415	0.130952	0.425762	<pre>def build(X):     nodes = 4     model = Sequential()      model.add(tf.keras.layers.Dense(nodes, input_dim=X_train.shape[1], activation="sigmoid"))      model.add(tf.keras.layers.Dense(1, activation="sigmoid"))      model.compile(Loss='mse', optimizer='rmsprop') #    model.compile(optimizer="adam", Loss="binary_crossentropy", metrics=['accuracy'])      model.summary()     return model</pre> 
	T-t	S/X																									
0.063492	2.627350																										
0.063492	2.502238																										
0.063492	2.388500																										
0.063492	2.284652																										
0.063492	2.189458																										
...	...																										
0.130952	0.539298																										
0.130952	0.505592																										
0.130952	0.475851																										
0.130952	0.449415																										
0.130952	0.425762																										

# CURRENT MARKET OBSERVATIONS - 1

## Call Option Moneyness Against Stock Price Moneyness

Y-axis :  $C/X$  : Call Option Price per unit of Strike

X-axis :  $S/X$  : Stock (Underlying) per unit of Strike

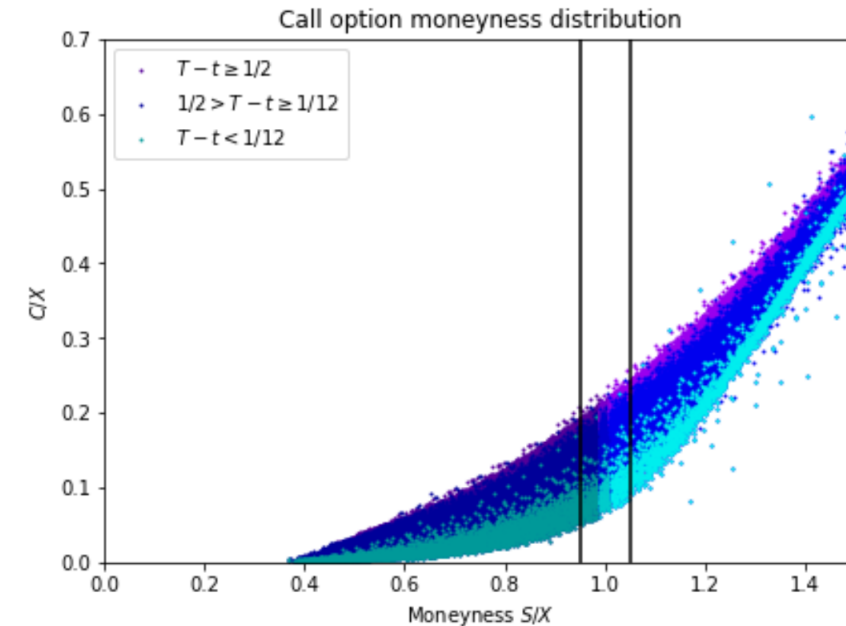
### Takeaway

**Short Maturity** :  $T - t < 30$  : Steepest gradient, highest rate increments

**Medium Maturity** :  $30 < T - t < 180$  : Medium rate of increments

**Long Maturity** :  $T - t > 180$  : Lowest rate of increment

Market call options behaviour are similar to theoretical / textbook concepts

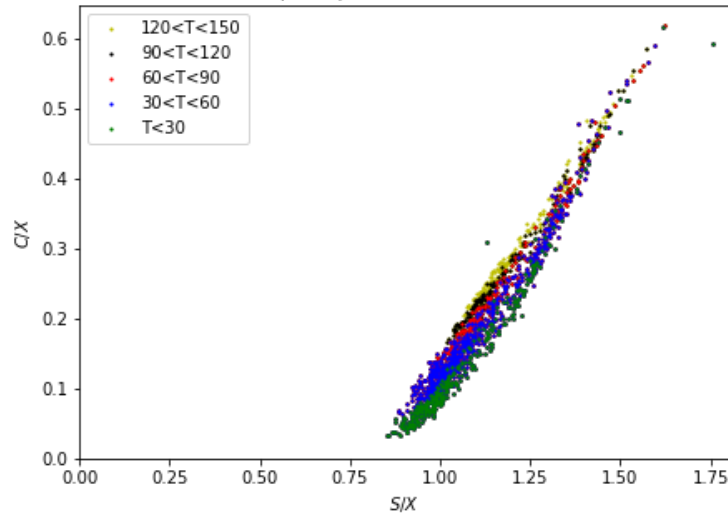


# CURRENT MARKET OBSERVATIONS - 2

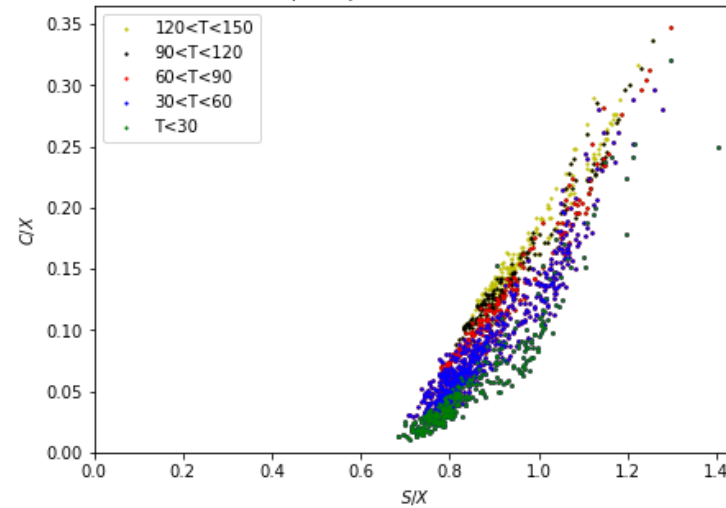
Strikes  
Increases



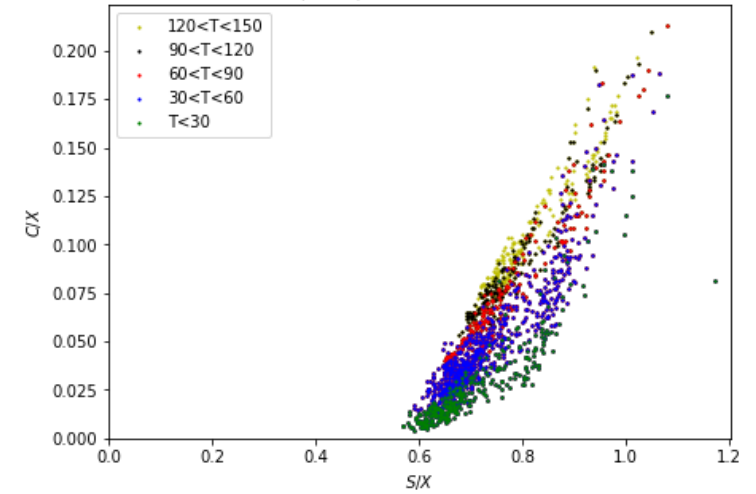
Call option price distribution (X=20)



Call option price distribution (X=25)



Call option price distribution (X=30)



## Takeaway

High strikes with longer maturity tends to be steeper than high strikes with shorter maturity



# PART I – OPTION PRICING - RESULTS

# PART 1 - PRICING RESULTS

Performance comparison results on the test dataset, between the pricing prediction of the BS model ( $y_{BS}/X$ ) vs. the market ( $y_C/X$ ) and the pricing prediction of the NN ( $\hat{y}_{NN}/X$ ) vs. the market ( $y_C/X$ ) cross different moneyness and time-to-maturity, considering MSE, RMSE and MAE.

$$\frac{\hat{C}}{\bar{X}}$$

$$MSE = \frac{1}{N} \sum_i^n (\hat{C}_i - C_i)^2$$

$$RMSE = \sqrt{\frac{1}{N} \sum_i^n (\hat{C}_i - C_i)^2}$$

$$MAE = \frac{1}{N} \sum_i^n |\hat{C}_i - C_i|$$

	test	mse_BS	rmse_BS	mae_BS	mse_NN	rmse_NN	mae_NN	mse_diff (BS-NN)	rmse_diff (BS-NN)	mae_diff (BS-NN)
0	All	0.010340	0.101684	0.074990	0.012491	0.111763	0.049619	-0.002151	-0.010079	0.025371
1	ITM	0.014707	0.121273	0.095716	0.021010	0.144947	0.060215	-0.006303	-0.023674	0.035502
2	OTM	0.004931	0.070220	0.049104	0.000156	0.012482	0.008613	0.004775	0.057738	0.040491
3	ATM	0.013664	0.116894	0.095032	0.000246	0.015677	0.011453	0.013419	0.101217	0.083579
4	Short Maturity	0.000756	0.027494	0.016124	0.005939	0.077065	0.025775	-0.005183	-0.049571	-0.009651
5	Medium Maturity	0.002478	0.049778	0.036729	0.007921	0.088998	0.030840	-0.005443	-0.039220	0.005888
6	Long Maturity	0.023649	0.153781	0.144419	0.017481	0.132215	0.046418	0.006168	0.021566	0.098001

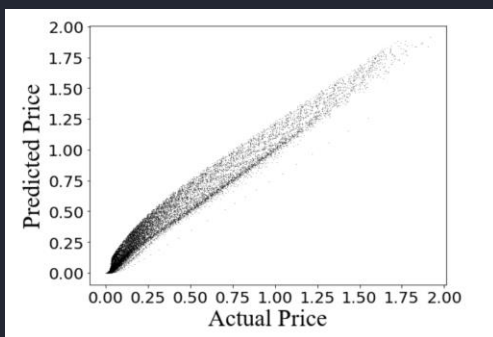
- Similar performance

- ATM, OTM, LM: NN is relatively better

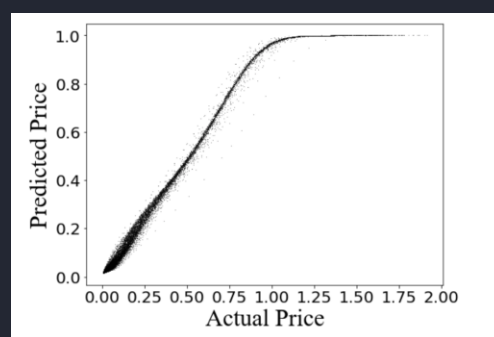
Graphical representation of the entire test dataset results (similar trend when applying to the train datasets) where the predicted price ( $y_{BS}/X$ ,  $\hat{y}_{NN}/X$ ) is plotted as a function of the actual price ( $y_C/X$ ). Technically for the best results is a diagonal linear line, where actual = predicted as unit increases.

Data limitation - Since the ANN is a data driven-model, the model would learn the features associated with the most common type of data in the data set.

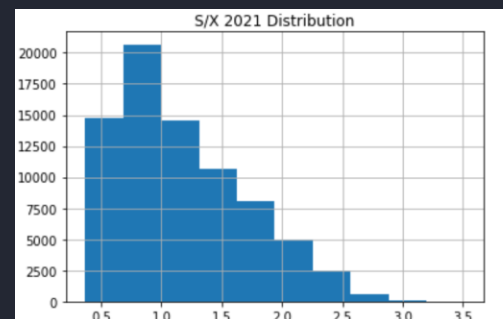
Black-Sholes



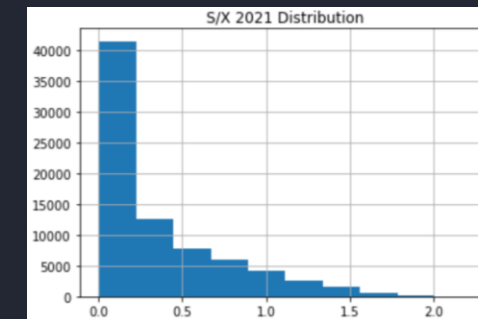
Artificial Neural Network



Density ( $y_{BS}/X$ )

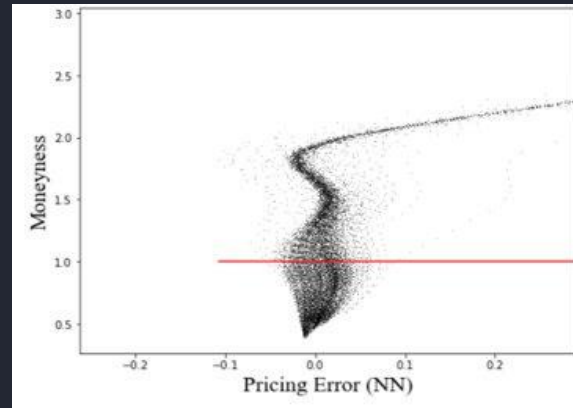
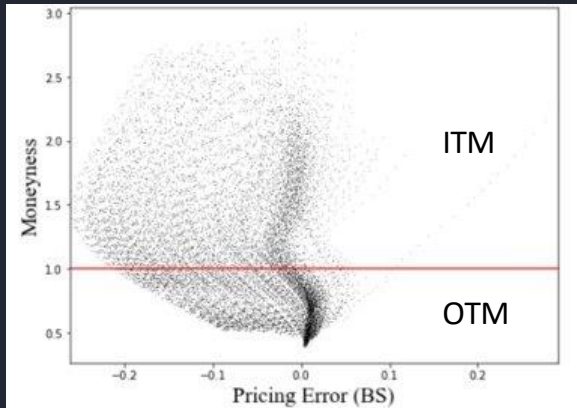


Density ( $C/X$ )



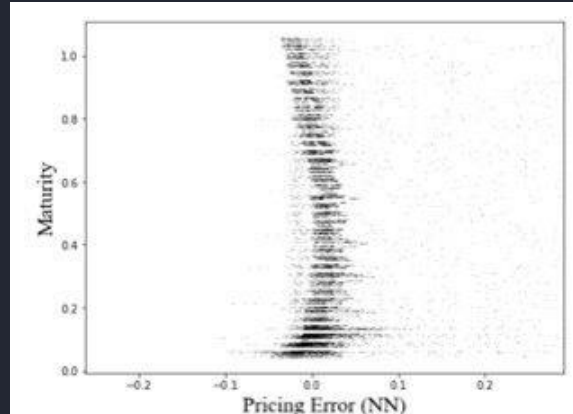
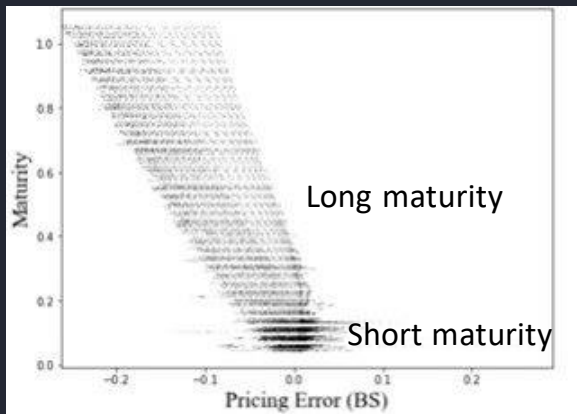
# PART 1 - PRICING RESULTS (CONT'D)

Graphs here represent the pricing error as a function of moneyness. The red horizontal line divides ITM options above from OTM options below.



- pricing error = actual - predicted :
- ve -> the said option is undervalued, +ve -> the said option is overvalued
- (BS) the dispersion on the left side of the mean (0) is bigger, in magnitude.
- Therefore, the options tend to be more underpriced by the BS model than by the ANN

pricing errors of the Artificial Neural Network as a function of time-to-maturity for both models.



- pricing error = actual - predicted : -ve -> overprice, +ve -> underprice
- long maturity =  $T-t > \frac{1}{2}$
- short maturity =  $T-t < \frac{1}{12}$
- medium maturity  $\frac{1}{12} \leq T-t \leq \frac{1}{2}$
- ANN pricing error is centered at 0
- BS pricing error has more variance
- the ANN model outperforms the BS model for Long maturity options
- in the Short maturity case, the mispricing of the two models is similar



# PART II - DELTA HEDGING - METHOD



# PART 2 DELTA-HEDGING FEASIBILITY

## Mathematical Intuition -> Replication

Since the option pricing theory is based on the replication of an option with a portfolio composed of other assets, it is important to prove the ability of the model in replicating the option through a dynamic hedging strategy such as delta hedging. The delta hedging strategy we adopted follows the method used by Hutchinson et al. (1994)

The main idea of the strategy is to set up a replicating portfolio  $V_t$  that offsets the risk of the option position:

- $V_t(S)$  is the value of the underlying asset position
- $V_t(B)$  is the value of a bond position used to finance the position in the underlying asset
- $V_t(C)$  is the value of the option position held in the portfolio at time  $t$

$$V_t = V_t(S) + V_t(B) + V_t(C)$$

$$V_0(C) = -C_{BSM,0}$$

$$V_0(S) = S_0 \Delta_{NN,0}$$

$$V_0(B) = -(V_0(S) + V_0(C))$$

$$\Delta_{NN,0} = \frac{\partial C_{NN,0}}{\partial S_0}$$

The composition of the portfolio at time  $t=0$  is assumed to be:

- $C_{NN,0}$  is the price of the call option predicted by the neural network from which the delta of the call option  $\Delta_{NN,0}$  can be computed.

The strategy consists of

- writing one call option,
- going long for the underlying asset for  $\Delta_{NN,0}$  number of shares at price  $S_0$ ,
- going short for a bond to finance the rest of the long position in the underlying asset that is not financed with the sale of the call option.

The initial value of the replicating portfolio is 0 since the long position is financed entirely with riskless borrowing and the sale of the call option.

$$V_0 = V_0(S) + V_0(C) + V_0(B) = 0$$

Between the initialization of  $V$  at time  $t=0$  and the expiration at  $T$ , at all time  $T-t$  the spot and bond positions are rebalanced daily:

$\tau$  is defined to be 1 day in this paper

$$V_t(S) = S_t \Delta_{NN,t} \quad \text{where} \quad \Delta_{NN,t} = \frac{\partial C_{NN,t}}{\partial S_t}$$

$$V_t(B) = e^{r\tau} V_{t-\tau}(B) - S_t (\Delta_{NN,t} - \Delta_{NN,t-\tau})$$

In this paper, more meaningful performance measures are introduced. the tracking error is defined by the value of the replicating portfolio at expiration date,  $V_T$ :

In this project, we adopt the same performance measures to analyse the hedging strategy performance.

$$V_T = V_T(S) + V_T(C) + V_T(B)$$

- 1) The present value of the expected absolute tracking error:  $\xi = e^{-rT} E[|V(T)|]$

- 2) The prediction error:  $\eta = e^{-rT} \sqrt{E^2[V(T)] + Var[V(T)]}$

- epsilon: information regarding the accuracy of the option pricing formula
- nu: information in the expected tracking error is combined with the information in the variance of the tracking error

$$J_f = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \dots & \frac{\partial y_n}{\partial x_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_1}{\partial x_m} & \dots & \frac{\partial y_n}{\partial x_m} \end{bmatrix}$$

$$J_{ANN} = \begin{bmatrix} \frac{\partial C_t}{\partial X} \\ \frac{\partial S_t}{\partial X} \\ \frac{\partial C_t}{\partial (T-t)} \end{bmatrix}$$

To calculate the delta of the neural network for the call option:

Using ANN, after training, we could derive the optimal weights on the input features and hence in principle are able to derive the partial derivative of the ANN predicted price w.r.t the features.

Using the Jacobian matrix: assuming that  $y$  is the output vector of  $f$ , the main idea is that the Jacobian matrix of  $f$  contains the partial derivatives of each element of the output  $y$  w.r.t each element of the input  $x$ .

# PART 2 DELTA-HEDGING FEASIBILITY

Mathematical Intuition -> Replication

```
# ----- ANN -----
option_data['delta_ANN'] = np.asarray([ getDer(row, sess=sess, grad_func=grad_func, model=model) for index, row in option_data.itertuples()])
option_data['delta_ANN_lag'] = option_data['delta_ANN'].shift(1)

option_data['V_S_ANN'] = option_data['S/X'] * option_data['delta_ANN']
option_data['V_C_ANN'] = -option_data['BS_Opt_Price']
option_data['V_B_ANN'] = getBondValueANN(option_data)

option_data['V_T_ANN'] = option_data['V_S_ANN'] + option_data['V_C_ANN'] + option_data['V_B_ANN']
```

```
def getBondValueBS(option_data):
    tau = 1 # tau is defined to be one day in the paper
    is_first = True
    V_B_BS = list()
    V_B_previous = 0
    for index, row in option_data.itertuples():
        if is_first is True:
            V_B_it = -(row['V_S_BS'] + row['V_C_BS'])
            V_B_BS.append(V_B_it)
            V_B_previous = V_B_it
            is_first = False
            continue

        term_A = np.exp(row['ir'] * tau) * V_B_previous
        term_B = row['S/X'] * (row['delta_BS'] - row['delta_BS_lag'])
        V_B_it = term_A - term_B
        V_B_BS.append(V_B_it)
        V_B_previous = V_B_it
```

```
def getDer(row, sess, grad_func, model):
    x = np.asarray([(0, 0)])

    x[0,0] = row['S/X']
    x[0,1] = row['T-t']

    gradients = sess.run(grad_func, feed_dict={model.input: x.reshape((1, x.size))})
    return np.array(gradients[0][0,:])[0]
```

```
grad_func = tf.gradients(model.output[:, 0], model.input)
```

$$V_S(t) = S(t)\Delta_{RBF}(t), \quad \Delta_{RBF}(t) \equiv \frac{\partial F_{RBF}(t)}{\partial S}$$

$$V_B(t) = e^{r\tau}V_B(t - \tau) - S(t)(\Delta_{RBF}(t) - \Delta_{RBF}(t - \tau))$$

- using Bootstrapping to resample observations from the dataset
- group by option id and date to construct the price path
- within each iteration, for each option id,
  - take the optimal weights trained from the ANN model
  - load it backed onto the boot-strapped data
  - perform the replication method in the previous slide
  - calculate relevant performance metrics
- using the built-in function provided by the TensorFlow package to compute the partial diffraction of the ANN Option Price w.r.t to the feature input



# PART II - DELTA HEDGING - RESULTS

# PART 2 DELTA-HEDGING PERFORMANCE

The hedging performance obtained from the bootstrapping consists of a comparison between the delta-hedge analysis for the Black and Scholes model and the artificial neural network model.

This comparison is developed on the test set considering only the options contracts that have over 10 days of observations in the test set.

num_of_options	epsilon_BS	nu_BS	epsilon_ANN	nu_ANN	tracking_error_ANN	tracking_error_BS	
0	1878	1.578401	1.570971	1.648419	1.623731	2.817918	2.714089

The present value of the expected absolute tracking error  
The prediction error  
The tracking error

$$\xi \equiv e^{-rT} \mathbb{E}[|V(T)|]$$

$$\eta \equiv e^{-rT} \sqrt{\mathbb{E}^2[V(T)] + \text{Var}[V(T)]},$$

$$V(t) = V_S(t) + V_B(t) + V_C(t)$$

- similar performance
- the BS model is slightly better

caution the reader not to give it undue weight. Since we have hedging errors for each option and learning network, we can use a paired  $t$ -test to compare the Black-Scholes model absolute hedging error on each option with the network's absolute hedging error on the same option. The null hypothesis is that the average difference of the two hedging errors is zero, and the (one-sided) alternative hypothesis is that the difference is positive, i.e., the

----- All samples -----				
Tabulate Table:				
Sample	t-statistic	p-value	Observations	% NN < BS
ITM (moneyness > 1.05)	1.59967	0.10985	870	5.86
OTM (moneyness < 0.95)	0.35006	0.72633	881	10.56
ATM (0.95 <= moneyness <= 1.05)	0.30712	0.759	127	8.66
All	1.38461	0.16626	1878	8.25

- to conclude a two-tailed paired  $t$ -test is performed to test for the null hypothesis that two independent samples have identical expected values.
- $H_0$  (difference = 0) is not rejected
- the results are not statistically significant different from each other
- similar hedging performance
- This result is in line with the pricing results presented in the previous section

Table XX

## Paired $t$ -Test Comparing Relative Magnitudes of Absolute Hedging Error, Using Results from all S&P 500 Test Sets from July 1987 to December 1991

The degrees of freedom for each test were 1299, although see comments in the text concerning dependence. RBF indicates radial basis function; PPR indicates projection pursuit regression; MLP indicates multilayer perceptrons; and B-S indicates Black-Scholes formula.

Pair	$t$ -Statistic	$p$ -Value
Linear-1 vs. B-S	-15.1265	1.0000
Linear-2 vs. B-S	-5.7662	1.0000
RBF vs. B-S	2.1098	0.0175
PPR vs. B-S	2.0564	0.02
MLP vs. B-S	3.7818	0.0001



# CONCLUSION AND FUTURE WORK

# CONCLUSION AND FUTURE WORK

## Conclusion

- 1) ANN able to predict an option price that is close to Black-Scholes Theoretical and Market's Option Price -> As seen based on the Performance Metrics Table
  - 1) No statistical distribution required for pseudo-stochastic parameters
  - 2) Clean application of 2 features
- 2) The ANN predicted option price is also able to perform delta-hedging -> As observed from the similar  $V(T)$  values from both Black-Scholes and ANN-Predicted

## Future Pointers

- Exploration of different NN Models such as RNN , LSTM
- Incorporating another element of Black-Scholes parameters, such as  $r$  or  $\sigma$  , i.e. 3 input features
- Predicting different asset classes, such as commodities or currencies options

## Final Thoughts

- The team used bootstrapping, sampling underlying price from historical price points from the dataset, however, could have explore to simulate geometric Brownian motion paths instead, to enforce independent price paths, instead of conditional dependent price point.
- The team's approach of extracting the partial derivative of ANN Option Price w.r.t to Underlying price was to use the tensorflow's keras library method to extract gradient/coefficient from a Jacobian matrix, as the gradient value was passed into the rows and columns of the dataset.
- A different approach could also be taken, where the weights of the 2 inputs could be used, as it would also mean the gradient/coefficient, and then apply directly to new simulated geometric Brownian price paths to calculate  $V(T)$
- The team could have explored a different asset class, as VIX option prices are usually priced with the GARCH model in the industry, instead of Black-Scholes

on: absolute; z-index: 999  
x 5px #ccc}.gbrtl .gbm  
display: block; position  
acity: 1; \*top: -2px; \*le  
/; top: -4px\0/; left: -6  
e-box; display: inline  
isplay: block; list-s  
e-block; line-height  
pointer; display: bl  
tive; z-index: 1000)  
padding-right: 9px  
durl1111

# REFERENCE



# REFERENCE

- P. C. Andreou, C. Charalambous, and S. H. Martzoukos. Generalized parameter functions for option pricing. *Journal of Banking & Finance*, 34(3):633–646, 2010.
- C. Boek, P. Lajbcygier, M. Palaniswami, and A. Flitman. A hybrid neural network approach to the pricing of options. In *Proceedings of ICNN'95—International Conference on Neural Networks*, volume 2, pages 813–817. IEEE, 1995.
- R. Garcia and R. Gencay. Option pricing with neural networks and a homogeneity hint. In *Decision Technologies for Computational Finance*, pages 195–205, 1998.
- R. Garcia and R. Gencay. Pricing and hedging derivative securities with neural networks and a homogeneity hint. *Journal of Econometrics*, 94(1-2):93–115, 2000.
- E. Ghysels, V. Patilea, E. Renault, and O. Torres. Nonparametric methods and option pricing. In D. Hand and S. Jacka, editors, *Statistics in Finance*, chapter 13, pages 261–282. John Wiley & Sons, 1998.
- R. Gencay and R. Gibson. Model risk for European-style stock index options. *IEEE Transactions on Neural Networks*, 18(1):193–202, 2007.
- J. M. Hutchinson, A. W. Lo, and T. Poggio. A nonparametric approach to pricing and hedging derivative securities via learning networks. *The Journal of Finance*, 49(3):851–889, 1994.
- S. Liu, C. W. Oosterlee, and S. M. Bohte. Pricing options and computing implied volatilities using neural networks. *Risks*, 7(1):1–22, 2019b.
- D. Ormoneit. A regularization approach to continuous learning with an application to financial derivatives pricing. *Neural Networks*, 12(10):1405–1412, 1999.
- J. Ruf and W. Wang. Neural networks for option pricing and hedging: a literature review. SSRN 3486363, 2020.
- J. Yao, Y. Li, and C. L. Tan. Option price forecasting using neural networks. *Omega*, 28(4):455–466, 2000.



THANK YOU