Decisions in Economics and Finance
https://doi.org/10.1007/s10203-025-00518-9

Option pricing with deep learning: a long short-term memory approach

Rita Pimentel[1] · **Morten Risstad**[1] · **Sondre Rogde**[1] · **Erlend S. Rygg**[1] · **Jacob Vinje**[1] · **Sjur Westgaard**[1] · **Cassandra Wu**[1]

Received: 30 April 2024 / Accepted: 11 March 2025

Abstract
This paper presents a deep learning approach for option pricing using a long short-term memory (LSTM) neural network applied to European call options on the S&P 500 index. We utilize a rolling window approach that trains 12 instances of the LSTM model, one for each month of 2021. To gain further insight into the model performance, we use explainable artificial intelligence (XAI) through SHapley Additive Explanations (SHAP). We find that the LSTM model outperforms the Black–Scholes and the Heston models and a multilayer perceptron (MLP) neural network regarding overall pricing accuracy. Most notably, the time-sequencing nature of LSTM enables the proposed model to capture sufficient short-term volatility from recently traded options. This result is still robust when controlling for time-varying volatility dynamics. Thus, the model is less prone to measurement errors in volatility.

Keywords Option pricing · LSTM · XAI

JEL Classification G13 · C63 · C45

1 Introduction

This paper studies a deep learning method for pricing options using a long short-term memory (LSTM) neural network trained and tested on European call options on the S&P 500 index. LSTM networks have been explored for multiple applications in finance. Several of these studies highlight the advantages of LSTM networks in analyzing relationships among time-series data through its memory function. Sezer et al. (2020) find LSTM networks to be the most prevalent type of deep learning method for financial time-series forecasting. Among the most popular applications is to study individual stock markets (Chen et al. 2015; Jin et al. 2019) and equity indices

✉ Rita Pimentel
rita.pimentel@ntnu.no

[1] Department of Industrial Economics and Technology Management, Norwegian University of Science and Technology, Alfred Getz v. 3, 7491 Trondheim, Norway

Published online: 12 April 2025

(Li et al. 2017; Yan and Ouyang 2018; Qiu et al. 2020; Gao et al. 2021; Bhandari et al. 2022). In spite of the ability of machine learning methods to learn flexible, functional forms and thus handle the complex non-linear relationship between underlying assets and option values, the number of studies on the use of LSTM networks for option pricing is limited. Compared to previous research on using an LSTM model[1] for option pricing, a key contribution of this study is to train and test the model using rolling windows to imitate how real market participants could deploy the model and regularly retrain it. The LSTM model is further analyzed using SHapley Additive exPlanations (SHAP) to identify how different input features contribute to model predictions. To further investigate the effect of these features on the predicted price, we analyze the performance of the LSTM model for different levels of maturity and moneyness[2]. Our research contributes to the limited number of studies demonstrating the feasibility of applying deep learning methods using LSTM to price options.

Specifically, our results show that the LSTM model generally outperforms the Black–Scholes (BS), the Heston, and the multilayer perceptron (MLP) neural network benchmark models for options with longer time to maturity (greater than 3 months) in each of the defined moneyness intervals. However, for options with a shorter time to maturity (less than 3 months), the BS and Heston models perform better than both deep learning methods. Most notably, the time-sequencing nature of LSTM enables the model to capture sufficient short-term volatility from traded options over the past five trading days. We find support for this conjecture from SHAP analysis, showing that the volatility estimate in general does not impact predictions to any material extent within the LSTM framework. We do, however, find that the volatility estimate is relatively more important in the LSTM framework when volatility is low compared to when volatility is high. This reinforces Bollerslev et al. (2016), that report larger measurement errors in S&P 500 data when volatility is high. The true volatility is a latent variable and must be estimated with error. Such measurement errors translate to noise in training data. Hence, a higher signal-to-noise ratio naturally extends to relatively higher importance of the volatility proxy at low volatility levels within the LSTM framework. To further verify the robustness of this finding, we estimate a GJR-GARCH (Glosten et al. 1993) model, in short GG, and use the resulting conditional volatility estimates as an alternative volatility proxy in the BS, MLP, and LSTM. For the LSTM model, including conditional volatility estimates as predictors does not improve forecasts compared to our baseline LSTM specification, on the contrary - the LSTM-GG tends to perform worse across most combinations of moneyness and time to maturity. We interpret this as a confirmation of our hypothesis that the LSTM model is capable of learning the pricing dynamics from history without relying on explicit volatility estimates. As such, the proposed LSTM model is less exposed to

[1] In this paper, "model" broadly refers to any framework for estimating option prices. Traditional stochastic models, like Black–Scholes, rely on probabilistic assumptions, while data-driven approaches, such as LSTMs, learn pricing dynamics from historical data. Both are termed "models" as they share the goal of predicting option prices.

[2] Moneyness is defined as the price of the underlying divided by the strike price of the option. At-the-money options have a strike equal to or close to the current price of the underlying. For call options, out-of-the-money options have a strike higher than the current price of the underlying, and in-the-money options have a strike lower than the current price of the underlying.

measurement errors in the volatility input compared to benchmark models. This is particularly important under rapidly changing market conditions.

Our research demonstrates that an LSTM model trained and tested using rolling windows can improve pricing performance. Comparable studies employ different approaches. Liang and Cai (2022) use a training-validation-test split that is rather theoretical, as it samples from the same time period. This potentially gives their LSTM model an advantage as the training and test sets could have more similar characteristics and feature values compared to sampling from different time periods. Liu and Wei (2022) do an overall split of their data set, of which 70% is training data, and 30% is testing data. Compared to previous studies, our training-validation-test split using rolling windows more realistically imitates how a real market participant could deploy and retrain the model.

Considering previous studies on deep learning methods for option pricing, we observe that few researchers report on the interpretability of their proposed method. This is somewhat surprising as interpretability is often a requirement for the real-life deployment of financial models. In our research, we analyze model interpretability using SHAP, from which we draw insights to guide further research on the development of LSTM for option pricing.

2 Literature review

Many extensions and alternative pricing models have emerged since the breakthrough derivation of the analytical option pricing formula by Black and Scholes (1973) and Merton (1973). The most prevalent criticism of the BS model is the assumption about constant volatility over the life of the option, which is inconsistent with the volatility smile observed in real option markets and empirical observations of the stochastic behavior of volatility in asset prices and their derivatives (Yang et al. 2017; Ball and Roma 1994). Several alternative option pricing models that incorporate stochastic volatility have been proposed since, including (1987; 1988), Johnson and Shanno (1987); Scott (1987); Wiggins (1987); Heston (1993) and Hagan et al. (2002). Other extensions of the BS model are adjusted for skewness and kurtosis different from the lognormal distribution, such as models developed by Jarrow and Rudd (1982), Corrado and Su (1996) and Jurczenko et al. (2004). Another approach has been to allow for discontinuous asset prices to take into account the influence of rare events on the underlying price, like jump-diffusion models proposed by Merton (1976) and Bates (1991). When there are no known analytical solutions, numerical methods are useful to derive an approximate solution. This class of models includes tree-based models (Cox et al. 1979; Rendleman and Bartter 1979) and finite difference methods (Brennan and Schwartz 1978). When the dimensions grow, Monte Carlo simulations (Boyle 1977) provide another approach to option pricing.

In contrast to the option pricing models presented above, machine learning models do not pre-specify functional relations. Within machine learning, several types of neural networks have been studied by financial researchers as methods for option pricing. Ruf and Wang (2020) provide an extensive literature review of studies using artificial neural networks (ANNs) for option pricing. They find that more than half of

the paper abstracts explicitly emphasize the positive performance of ANNs in option pricing and hedging. Hutchinson et al. (1994) is one of the first papers to use ANNs to price options. They find that the ANN outperforms BS on average when applied to daily call options on S&P 500 futures. Using intraday data for call options on the German DAX index, Anders et al. (1998) also find that ANNs show a higher pricing accuracy than BS. Garcia and Gencay (2000) use ANNs to price European call options on the S&P 500 index and present one of the first studies using moneyness as input rather than the strike and the underlying price separately. Gencay and Qi (2001) and Gradojevic et al. (2009) provide further evidence that ANNs perform better than BS for call options on the S&P 500 index. Gradojevic et al. (2009) more specifically study modular ANNs based on moneyness and time to maturity and demonstrate the ability of such ANNs to specialize in pricing certain types of options. Amilon (2003) apply ANNs to call options on the Swedish OMX index and demonstrate better pricing performance when benchmarked against BS. Bennell and Sutcliffe (2004) apply ANNs to European call options on the FTSE 100 index. The ANN outperforms BS for out-of-the-money options, whereas BS outperforms the ANN for in-the-money options.

More recent studies show some examples of increasingly complex ANN architectures. Yang et al. (2017) use a gated ANN to price European call options on the S&P 500 index. Ferguson and Green (2018) demonstrate that deep ANNs can achieve high pricing accuracy for derivatives valuations using a basket option as an example. Liu et al. (2019) successfully price European call options using deep ANNs that consist of 4 layers with 400 nodes each. Wei et al. (2021) propose an ensemble learning mechanism for option pricing that aggregates the estimations of three parametric models (stochastic volatility, jump-diffusion, and ad hoc BS[3]) and a convolutional neural network (CNN). The framework is applied to call options on the Shanghai Stock Exchange (SSE) 50 ETF and shows that the ensemble model outperforms the parametric models. Wei et al differentiates their research by considering the co-movements of options trading at the same time when estimating option prices and implied volatility. Du Plooy and Venter (2021) find that ANNs are able to price European call options on the Johannesburg Stock Exchange All Share Index with a high degree of accuracy when given an implied volatility surface constructed using option price data from the South African market. Jang et al. (2021) demonstrate improved pricing performance by deep ANNs compared to parametric methods and other machine learning methods when applied to call options on the S&P 500 index, call options on the EuroStoxx50 index, and put options on the Hang Seng Index. Qian et al. (2022) show that ANNs based on the genetics algorithm achieve better pricing accuracy than standard ANNs and the BS model when applied to call options on the Shanghai Shenzhen CSI 300 index. Liang and Cai (2022) propose an LSTM and a 1D-CNN to price call options on the SSE 50 ETF and call options on the S&P 500 index. Both models are trained using daily option quotes, from 2017 to 2019 for SSE 50 ETF, and 2019 for S&P 500. In both option markets, the proposed LSTM model shows significantly improved pricing accuracy and robustness compared to benchmark models, including BS. Liu and Wei (2022) develop two LSTM models: an unregulated version and a regulated version

[3] Ad hoc BS is a modified version of BS that estimates implied volatilities using the determined volatility function.

with added technical indicators. Similar to Liang and Cai (2022), Liu and Wei train their model using daily data for options on the SSE 50 ETF. However, while Liang and Cai focus on call options, Liu and Wei study both call and put options, and find that both LSTM models demonstrate better pricing accuracy than the BS model. The best pricing performance is achieved by the regulated LSTM model. In a recent review of the literature investigating UK options data, Zouaoui and Naas (2023) find that deep neural networks are capable of outperforming BS. However, Zouaoui and Naas (2023) report mixed results in terms of the preferred deep learning architecture. Investigating the relevance of exogenous variables, Andreou et al. (2008) report that lags of implied volatility increase the pricing performance of ANNs. Liu and Zhang (2023) find that ETF50 option pricing accuracy can be improved when realized skewness serves as an input feature in LSTM.

Machine learning models, such as neural networks, are often seen as black-box models that are difficult to interpret. While it is well-understood how neural networks can approximate any continuous function (Hornik et al. 1989), this does not necessarily provide sufficient insight into the characteristics of the approximated function. These limitations have driven the field of explainable AI (XAI), which is a set of methods to help us understand and interpret predictions made by machine learning models. The XAI methods that have been proposed can be categorized into local and global explainability methods (Molnar 2022). Local explainability evaluates how individual components contribute to each particular prediction made by a model. In contrast, global explainability shows how individual components impact model predictions on average. SHapley Additive exPlanantions (SHAP) is a local explainability method proposed by Lundberg and Lee (2017). Although these values are calculated locally, the local results can be aggregated to derive global explanations. Accumulated Local Effects (ALE) is a global explainability method proposed by Apley and Zhu (2020).

3 Data

This section presents the data. First, we present the data preparation process and descriptive statistics and then describe the time-sequencing of the data necessary for using the LSTM model.

3.1 Data preparation

The data set consists of the daily closing price for SPX European call options on the S&P 500 index from April 1st, 2020 to December 31st, 2021, provided by optionsDX[4]. The data set is made up of 3,371,420 option quotes. Data filtering has been kept at a minimum to maximize the generalization of the proposed LSTM model. The data filtered out primarily consists of missing values, and options with more than three years to maturity as these contracts were introduced during the testing period for the LSTM model. About 3.2% of the data is discarded with these filtering criteria. Furthermore, in creating the time-sequenced data structure as described in subsection 3.2, data for the first four trading days are used to create lags and hence discarded. Data for the

[4] https://www.optionsdx.com.

Table 1 Summary statistics of processed data grouped by moneyness

Feature	Moneyness Count	<0.97 755203	0.97−1.03 475368	>1.03 2033753
Option price	mean	28.20	111.79	963.88
	std	53.15	90.00	753.26
	min	0.02	0.02	64.00
	25%	0.20	43.75	415.55
	50%	3.35	95.05	724.20
	75%	31.30	154.60	1264.94
	max	582.30	747.15	4641.80
S&P500	mean	3862.98	3936.56	3909.54
	std	605.90	558.37	564.42
	min	2470.28	2470.28	2470.28
	25%	3357.43	3446.95	3418.66
	50%	3925.24	4019.95	3943.64
	75%	4395.08	4405.87	4399.53
	max	4792.94	4792.94	4792.94
Strike	mean	4489.94	3938.90	2970.11
	std	878.47	563.32	886.76
	min	2550.00	2400.00	100.00
	25%	3800.00	3460.00	2450.00
	50%	4500.00	4030.00	3075.00
	75%	4975.00	4415.00	3650.00
	max	9200.00	4940.00	4650.00
Time to maturity (Days)	mean	132.25	78.21	121.78
	std	180.51	124.13	166.99
	min	1.00	1.00	1.00
	25%	19.00	14.00	20.00
	50%	58.00	30.00	57.00
	75%	168.00	94.00	154.00
	max	1095.00	1095.00	1095.00

first four days of the training, validation, and test set are also discarded to avoid data leakage due to overlapping time periods between the sets. This reduces the size of the data set by an additional 15%. Table 1 summarizes the statistics for relevant features in the processed data set before creating lags, grouped by level of moneyness.

The option price is calculated as the mid-price (bid-ask average).

Risk-free rates are interpolated from U.S. Treasuries[5]. Finally, min-max scaling is used to scale each input value between 0 and 1, as given by Eq. 1. To prevent data leakage, the min-max scaling is calibrated with the minimum and maximum values of

[5] Publicly available at https://home.treasury.gov.

Table 1 continued

Feature	Moneyness Count	<0.97 755203	0.97−1.03 475368	>1.03 2033753
Volatility	mean	0.22	0.20	0.21
	std	0.15	0.13	0.14
	min	0.10	0.10	0.10
	25%	0.12	0.12	0.12
	50%	0.15	0.15	0.15
	75%	0.21	0.20	0.21
	max	0.54	0.54	0.54
Risk-free rate	mean	0.08	0.07	0.08
	std	0.06	0.05	0.06
	min	0.00	0.00	0.00
	25%	0.05	0.04	0.04
	50%	0.07	0.06	0.07
	75%	0.10	0.09	0.10
	max	0.99	0.99	0.99
Moneyness	mean	0.87	1.00	1.59
	std	0.09	0.02	1.88
	min	0.50	0.97	1.03
	25%	0.81	0.98	1.11
	50%	0.91	1.00	1.22
	75%	0.94	1.01	1.48
	max	0.97	1.03	47.93

the training set and not the entire data set. The same scaling ranges from the training set are then applied to the validation and the test set.

$$X_{scaled} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{1}$$

3.2 Data time-sequencing

Data must be in a time-sequenced format for the LSTM model to utilize time-series information. A sequence length of five days is used, following Liang and Cai (2022). This means that data from the last four trading days and the current day is used to price options for the current day. The time-sequencing method is illustrated in Figure Fig. 1 for an option with a time to maturity of 30 days. As a result of creating time sequences for all options in the data set, we get a three-dimensional matrix with axes for options, time steps, and features as visualized in Fig. 2. The LSTM model takes this three-dimensional matrix as input.

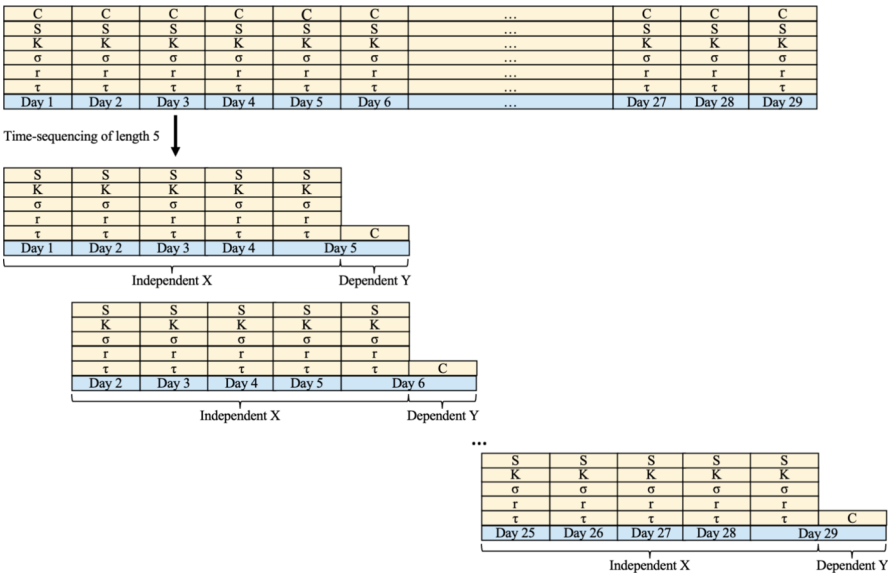

Fig. 1 Time-sequencing of a 30-day maturity option using a sequence length of 5 days

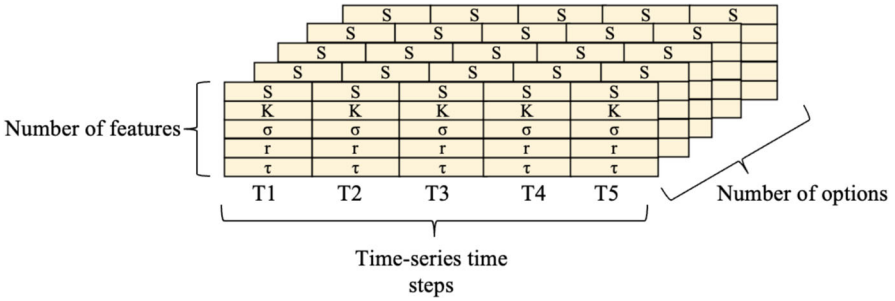

Fig. 2 3D format of the LSTM model input structured by options, time-steps, and features

4 Methods

This section starts by describing the volatility estimators considered in the study, next introduces the proposed LSTM model, describes the specific model architecture implemented, how the model is trained, the hyperparameter search, and finally presents the methods used to benchmark model performance[6].

[6] The Python code with the implementations of all models is available upon request.

4.1 Volatility estimation

Following Liang and Cai (2022), our baseline specification considers the 90-day moving average volatility of the underlying,

$$\sigma = \sqrt{252 \frac{\sum_{i=1}^{n}(R_i - \bar{R})^2}{n-1}}, \tag{2}$$

where $R_n = log(S_n/S_{n-1})$ and S_n is the closing price for day n.

Although this is a simple estimate of the true volatility, we hypothesize that it is sufficient, given the recurrent nature of the LSTM and its inherent capability to learn the option pricing dynamics. Arguably, a time-series model for conditional volatility could potentially serve the same purpose. Hence, to control for this, we estimate a GJR-GARCH (1, 1) with t-distributed residuals (Glosten et al. 1993). The dynamic equation for the conditional variance σ_t^2 of the GG(1, 1) is

$$\sigma_t^2 = \omega + \alpha u_{t-1}^2 + \beta \sigma_{t-1}^2 + \gamma u_{t-1}^2 I\left(r_{t-1} < 0\right), \tag{3}$$

where u is the return, σ is the volatility, α, β, γ are parameters, and I is an indicator function which equals 1 if $r_{t-1} < 0$ and zero otherwise. The last element captures the well-known leverage effect in equities, causing asymmetrical effects on volatility from positive and negative returns.

4.2 The LSTM model

LSTM neural networks, introduced by Hochreiter and Schmidhuber (1997), are modified versions of RNNs. The structure of RNNs allows previous hidden states to feed into and thereby influence the outputs of subsequent steps, which makes this class of neural networks suitable for sequential or time series data. However, a key challenge for RNNs is vanishing gradients that hamper their ability to learn long-term dependencies. This means that RNNs may not be able to connect information from prior states to the current state to make accurate predictions, especially if prior states are not sufficiently recent. The LSTM network provides a solution to the challenge of long-term dependencies[7].

4.2.1 Model implementation

The proposed LSTM model uses the same inputs as BS, meaning strike price, the underlying price, time to maturity, the risk-free rate, and volatility. The structure of the LSTM model consists of five LSTM layers, each followed by a batch normalization (BN) layer. The overall structure of the LSTM model is visualized in Fig. 3.

A common issue with deep neural networks is gradient disappearance or gradient explosion. To mitigate this, BN layers are implemented to normalize the activations

[7] See Appendix subsection A.1 for a general description of LSTM architecture.

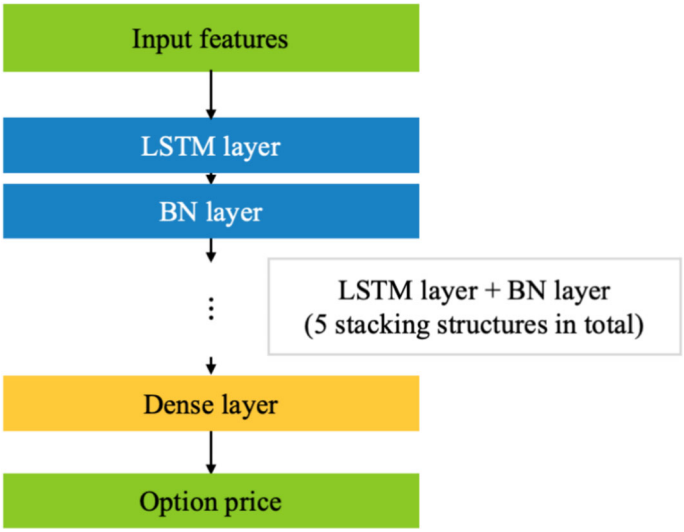

Fig. 3 Structure of the LSTM model

of the preceding layer in each batch. Implementing BN layers also helps prevent overfitting and speeds up the training process by requiring fewer iterations to converge. The BN normalization algorithm is presented below. For further details, see Ioffe and Szegedy (2015).

Input: Values of x over a mini-batch B = $\{x_1, ..., x_m\}$
Parameters to be learned: γ and β
Output: $y_i = BN_{\gamma,\beta}(x_i)$

$$\mu_\beta \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{Mini-batch mean}$$

$$\sigma_\beta^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_\beta)^2 \qquad \text{Mini-batch variance}$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu_\beta}{\sqrt{\sigma_\beta^2 + \epsilon}} \qquad \text{Normalize}$$

$$y_i \leftarrow \gamma\hat{x}_i + \beta \equiv BN_{\gamma,\beta}(x_i) \qquad \text{Scale and shift}$$

Where ϵ is a small constant for numerical stability.

ReLU is used as an activation function in each neuron, for which the function is defined as

$$h(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

The initial weights in the model are randomly generated. The weights are then repeatedly updated to reduce the mean squared error (MSE) with the AdamW optimizer, which extends the Adam optimizer (Kingma 2014) utilizing weight decay to penalize large weights and avoid exploding gradients, see Loshchilov and Hutter (2017). Furthermore, we utilize early stopping once validation loss has not improved for 20 consecutive epochs during hyperparameter search and final model training. For each time validation loss is improved, checkpointing is used to save the model, which will be used to make predictions.

4.2.2 Model training

The proposed LSTM model employs a chronological split where the training data comes before the validation and test data in time. Compared to random partitioning, this prevents breaking the time-series structure and creating information leakage between the test and training sets (Ruf and Wang 2020). Furthermore, Yao et al. (2000) show that the test error obtained with random partitioning underestimates the generalization error.

We implement rolling windows over 10-month periods using an 8-1-1 training-validation-test split. This means that to price options in January 2021, the model is trained using option prices from April 2020 to November 2020, and with a validation set consisting of option prices for December 2020. Similarly, to price options in February 2021, a new instance of the model is trained and validated using data from time periods shifted a month forward compared to when pricing options in January. Over the entire training process for pricing options in each month of 2021, 12 instances of the LSTM model are created.

By utilizing rolling windows, the training data is arguably more up-to-date, representative, and relevant to the out-of-sample options to be priced. Rolling windows are also better equipped to handle the existence of time-inhomogeneity in financial data, where relationships change over time (Ruf and Wang 2020). In this paper, we conduct monthly retraining, for which Fig. 4 illustrates the corresponding rolling windows set-up. When deployed and under ideal conditions, a market participant can retrain the model at arbitrary intervals as needed, such as on a weekly or daily basis.

4.2.3 Hyperparameter search

We conduct an extensive automated hyperparameter search based on a random search algorithm that obtains 100 different combinations of hyperparameters, within a predefined range, that the model is configured and tested with[8]. For each combination of hyperparameters, we train a model on the training set of the first model period and test on the corresponding validation set. The first model period was used to avoid data leakage if the test data for any of the model periods were included in the training or validation set of the hyperparameter search. We use the same hyperparameter configuration for all model instances.

[8] The hyperparameter search is conducted using *wandb* (https://wandb.ai/). *wandb* is an experiment-tracking tool for machine learning which allows running multiple hyperparameter searches in parallel.

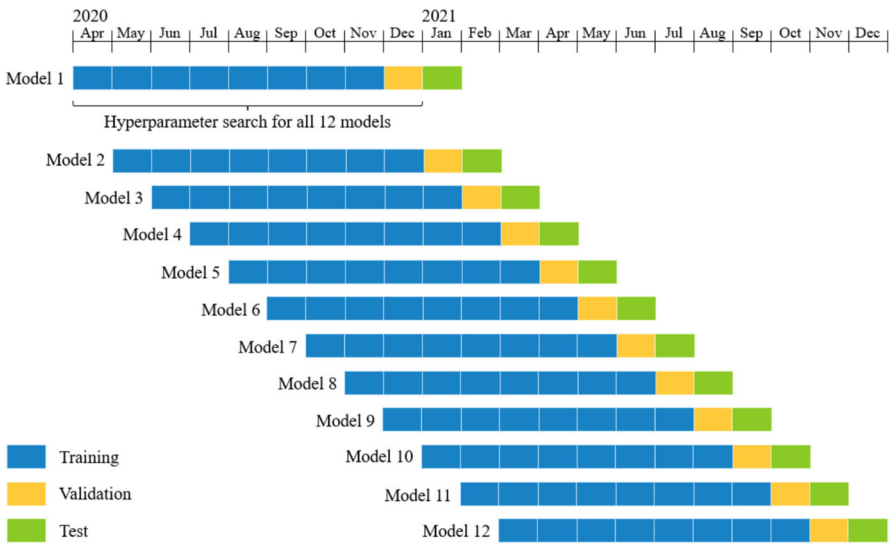

Fig. 4 Rolling windows using 12 separate model instances with 1-month test sets for each month in 2021

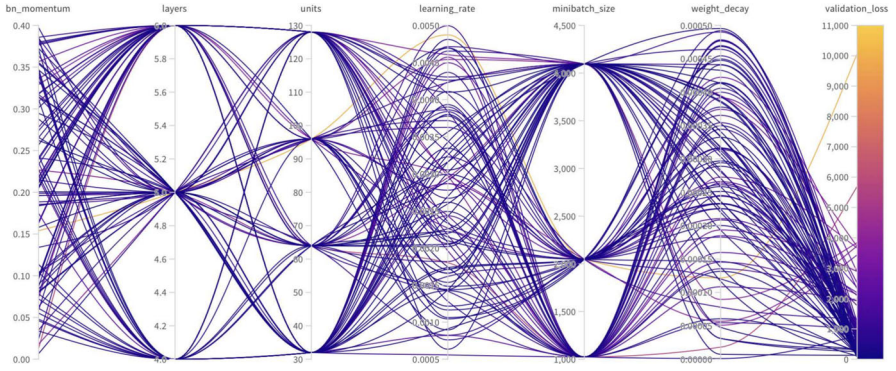

Fig. 5 Validation loss from 100 model initializations with different hyperparameters for the LSTM pricing model

Figure 5 shows all the runs from the hyperparameter search for the LSTM model. It shows the validation loss for the 100 different model configurations, each with a distinct set of hyperparameters. The figure helps identify the configurations that minimize the validation loss, guiding the selection of the best-performing model. The variation in validation loss across the different configurations indicates how sensitive model performance is to changes in hyperparameter values.

Figure 6 shows a feature importance metric for each hyperparameter and the linear correlation between each hyperparameter and the validation loss. Since we want to minimize the validation loss, a positive correlation implies that we want a lower parameter value. The feature importance metric is calculated by training a random forest with the hyperparameters as input and the validation loss as the target output. The

Fig. 6 Estimated feature importance metric for each hyperparameter, and linear correlation between each hyperparameter and the validation loss. A green correlation bar indicates a positive correlation and a red correlation bar indicates a negative correlation

Table 2 Hyperparameter configuration for the LSTM and LSTM-GG models

	LSTM	LSTM-GG
Layers	5	4
Units per layer	64	32
Learning rate	0.002595	0.001293
Weight decay	0.000333	0.000052
BN momentum	0.262121	0.001874
Minibatch size	4096	4096
Epochs	Early stopping	Early stopping
Optimizer	AdamW	AdamW

metrics are interpreted as how much each hyperparameter contributes to model performance regarding validation loss. We take into account feature importance metrics in our analysis to mitigate some limitations of linear correlation, such as not capturing second-order interactions between inputs. The hyperparameter values presented in Table 2 are based on analysis of Figs. 5 and 6, and the inspection of top performing hyperparameter combinations during the search.

4.3 Benchmark models

To evaluate the performance of the proposed LSTM network, we compare it with several benchmark models. First, we consider two stochastic models. The Black–Scholes model is included as it remains a widely used benchmark (Hutchinson et al. 1994). As noted in the comprehensive review by Ruf and Wang (2020), other commonly used parametric benchmarks are stochastic volatility pricing models. Therefore, we also incorporate the Heston model as a benchmark. Finally, we include a data-driven approach using a simpler neural network that does not account for time sequencing, namely a multilayer perceptron.

4.3.1 Black–Scholes model

The Black–Scholes model (Black and Scholes 1973; Merton 1973) assumes the underlying asset follows a geometric Brownian motion under the risk-neutral measure,

$$dS_t = rS_t dt + \sigma S_t dW_t, \tag{4}$$

where S_t is the underlying asset price at time t, σ is the volatility of the underlying asset, r is the risk-free interest rate, and W is a Wiener process. Under this model, the price for a European call option C for a non-dividend paying asset is given as

$$C(S_0, T) = S_0 N(d_1) - Ke^{-rT} N(d_2) \tag{5}$$

$$d_1 = \frac{log(S_0/K) + (r + \sigma^2/2)/T}{\sigma\sqrt{T}} \tag{6}$$

$$d_2 = \frac{log(S_0/K) + (r - \sigma^2/2)/T}{\sigma\sqrt{T}} = d_1 - \sigma\sqrt{T} \tag{7}$$

where S_0 is the current price of the underlying, K is the exercise price of the option, T is the time to maturity, and $N(\cdot)$ is the cumulative normal distribution. As for the LSTM model, the volatility in the Black–Scholes is the 90-day moving average volatility of the underlying asset and the GG.

4.3.2 Heston model

The Heston model is a stochastic volatility model used to price European call options, proposed by Heston (1993). Borrowing the notation already defined to the Black–Scholes model, the Heston model assumes that the underlying asset price, S_t, and its variance, V_t, follow the stochastic differential equations

$$dS_t = rS_t dt + \sqrt{V_t} S_t dW_t^S, \tag{8}$$

$$dV_t = \kappa(\theta - V_t)dt + \eta\sqrt{V_t} dW_t^V, \tag{9}$$

where V_t is the variance of the asset returns, κ is the rate of mean reversion of the variance, θ is the long-term variance mean, η is the volatility of volatility, and W^S and W^V are two Wiener processes with correlation ρ. The price of a European call option for a non-dividend paying asset under the Heston model is given by

$$C(S_0, T) = S_0 P_1 - e^{-rT} K P_2, \tag{10}$$

where P_1 and P_2 are risk-neutral probabilities defined as

$$P_j = \frac{1}{2} + \frac{1}{\pi} \int_0^\infty \mathrm{Re}\left[\frac{e^{-iu \ln K} f_j(u)}{iu} \right] du, \quad j = 1, 2, \tag{11}$$

Table 3 Hyperparameter configuration for the MLP and MLP-GG models

	MLP	MLP-GG
Layers	4	6
Units per layer	32	96
Learning rate	0.004469	0.004102
Weight decay	0.000425	0.000202
BN momentum	0.300571	0.327534
Minibatch size	4096	2048
Epochs	Early stopping	Early stopping
Activation function	ReLU	ReLU
Optimizer	AdamW	AdamW

where $\mathrm{Re}(x)$ represents the real part of the number $x \in \mathbb{C}$. The characteristic functions $f_1(u)$ and $f_2(u)$ are given by

$$f_j(u) = \exp\left(iu \ln S_0 + C_j(T,u) + D_j(T,u)V_0\right), \tag{12}$$

where V_0 is the current volatility of the underlying asset price and

$$C_j(T,u) = ruTi + \frac{\kappa\theta}{\eta^2}\left[(b_j - \rho\eta ui + d_j)T - 2\ln\left(\frac{1-g_j e^{d_j T}}{1-g_j}\right)\right], \tag{13}$$

$$D_j(T,u) = \frac{b_j - \rho\eta ui + d_j}{\eta^2}\frac{1-e^{d_j T}}{1-g_j e^{d_j T}}, \tag{14}$$

with

$$b_j = \kappa + \lambda + (j-2)\rho\eta \tag{15}$$

$$d_j = \sqrt{(b_j - \rho\eta ui)^2 + \eta^2(u^2 + (-1)^j ui)}, \tag{16}$$

$$g_j = \frac{b_j - \rho\eta ui + d_j}{b_j - \rho\eta ui - d_j}. \tag{17}$$

where λ is the linear constant of the function representing the price of volatility risk.

4.3.3 The multilayer perceptron model

A multilayer perceptron (MLP) neural network is a classical feedforward ANN with multiple layers. The MLP network used to benchmark utilizes the same input as the BS, but in contrast to LSTM only uses the current time values as input. It follows that it is not able to capture information embedded in the time-series structure as the LSTM model. The implementation and training of the MLP follows the same procedure as the LSTM. We also conduct a hyperparameter search for the MLP to derive the hyperparameter values shown in Table 3.

4.4 Interpretability analysis using SHAP

SHapley Additive exPlanations (SHAP) is a unified framework for interpreting model predictions. It was introduced by Lundberg and Lee (2017) to address the issue that there is a growing number of interpretability methods, yet a lack of understanding of how these methods relate and when one method is preferable over another. Instead of relying on a single method, the proposed SHAP value unifies six methods to provide a unified measure of feature importance that can be assigned to each feature. In selecting these six methods, Lundberg and Lee introduce the perspective of viewing any explanation of predictions by a certain model as a model in itself, termed the explanation model. The six methods that SHAP values are based on all use the same explanation model. Mathematically, let f be the original prediction model to be explained and g be the explanation model. Explanation models often map the original inputs x through a mapping function $x = h_x(x')$ to get simplified inputs x'. Then the explanation model g can be defined as:

$$g(z') = \phi_0 + \sum_{i=1}^{M} \phi_i z_i' \tag{18}$$

where $z' \in \{0, 1\}^M$, M is the number of simplified input features, and $\phi_i \in R$ is the effect attributed to each feature by methods with explanation models matching Eq. 18. For more detailed explanations, see Lundberg and Lee (2017).

Since SHAP is a local model-agnostic method that calculates the impact of each single data point, a key challenge is that it is computationally expensive. Therefore, we adopt SHAP using K-means with $k = 20$ to extract representative option quotes from the training set to be used by the explainer. From the test set, we sample 10,000 options for which SHAP values are calculated. The analysis of the model pricing options is done in January 2021[9].

5 Results and discussion

This section presents the pricing performance of the LSTM model and the MLP model, hereafter collectively referred to as the deep learning methods, as well as the performance of BS and Heston, which together we call parametric models. We consider two different ways of estimating the volatility, namely, a 90-day moving average and a GJR-GARCH. For simplicity, we refer to the models with the GJR-GARCH estimates as BS-GG, MLP-GG, and LSTM-GG, respectively.

Results presented for the deep learning methods combine the results of the 12 model instances tested in each month of 2021 using the rolling window approach. Our main finding is that the LSTM models achieve higher pricing accuracy than benchmark models at an aggregated level, considering all options in the data set and for most options with longer time to maturity in each of the defined moneyness intervals.

[9] SHAP plots for all 12 model instances, using a smaller sample size, are available upon request. All plots show similar trends.

Table 4 Pricing performance metrics for all models for the full-year 2021

	BS	BS-GG	Heston	MLP	MLP-GG	LSTM	LSTM-GG
RMSE	37.82	39.68	35.90	30.59	35.08	27.94	**26.86**
MAE	21.09	22.23	20.00	20.93	25.78	19.94	**18.82**

Bold values indicate the best model in each category

In the following discussion, we first analyze the overall pricing performance of all models. Then, we conduct an interpretability analysis of the LSTM model. We further analyze the pricing performance of the models with regard to time to maturity and moneyness.

5.1 Overall pricing performance

To evaluate the forecasting accuracy of the different models, root mean squared error (RMSE), mean absolute error (MAE), and the Theil U_1 coefficient are employed. Additionally, bias, variance and covariance proportion are also calculated to assess estimation performance and types of errors. The bias proportion indicates how far the forecast's mean is from the actual series's mean. Variance proportion indicates how far the forecast's variance is from the actual series's variance. The covariance proportion measures the remaining unsystematic forecasting errors. These three measures should all sum to one. Low bias and variance proportions are preferred. See Appendix subsection A.2 for further details on performance metrics.

Table 4 shows the pricing error aggregated across all option quotes for the full year of 2021. The table shows that generally both deep learning methods perform better than BS and Heston, and that LSTM outperforms the MLP model, both in terms of RMSE and MAE. We also note that the LSTM and the LSTM-GG present a similar pricing accuracy. The overperformance of the LSTM model could be attributed to its ability to capture more complicated non-linear and time-dependent patterns, allowing it to capture more dynamic market conditions better. The greater flexibility of the deep learning methods could reduce the bias. The outperformance of the deep learning methods is consistent with results by Liang and Cai (2022) and Liu and Wei (2022). For comparison, it should be noted that the deep learning methods Liang and Cai (2022) refer to include an MLP, a 1D-convolutional neural network, and an LSTM model. The models are applied to options on both the S&P 500 and SSE 50 ETF. Liu and Wei (2022) use LSTM to price options on the SSE 50 ETF.

Table 5 shows the bias, variance, and covariance proportion for all models. In terms of bias, the Heston model has the lowest bias proportion, followed closely by the LSTM models, which still maintain a significantly lower bias compared to MLP and BS. It can be seen that the difference in bias proportion between MLP and BS is less than that of LSTM and BS. Both deep learning methods exhibit a lower variance proportion than BS, though the Heston model achieves the lowest variance overall. However, Heston's advantage in variance comes at the cost of a higher covariance proportion, suggesting potential over-smoothing. In contrast, LSTM models strike a

Table 5 Bias, variance and covariance proportion

Model	Bias	Variance	Covariance
BS	13.71%	8.94%	77.34%
BS-GG	7.06%	6.21%	86.74%
Heston	0.07%	3.25%	96.68%
MLP	19.51%	5.95%	74.54%
MLP-GG	18.99%	0.30%	80.71%
LSTM	1.29%	4.47%	94.24%
LSTM-GG	1.08%	2.94%	95.98%

better balance, maintaining low bias and moderate variance while keeping covariance slightly lower than Heston, which contributes to their superior overall performance. The DM statistic (Diebold and Mariano 1995) in Table 6 shows that the differences in pricing by the models are statistically significant. Table 7 shows Theil U_1 statistics that support the finding that LSTM is the superior model, followed by MLP, Heston and finally BS.

The aggregated impact of each feature across all lags is shown in Fig. 7. The x-axis measures the SHAP value, where a positive value indicates an increased option price and a negative value represents a reduced option price. A red color indicates a high feature value, and a blue color indicates a low feature value. The SHAP values estimate the impact of each feature value in terms of explaining deviations in output values from a baseline of expected option value if no feature values were given. The interpretations of feature effects are thus relative to the other options in the data set.

The SHAP values shown in Fig. 7 indicate that the largest deviation from the baseline option price is caused by the strike price, followed by the value of the underlying S&P 500 index, time to maturity, volatility, and lastly, the risk-free rate. Having the intrinsic value of the option has the more relevant characteristic is expected. These findings are consistent with the interpretability analysis by Liang and Cai (2022) using ALE. The most important features are also those that show the broadest range of feature values in the data set as shown in Table 1. From Fig. 7, we clearly see that higher values for the S&P 500 index and time to maturity contribute to increasing the estimated option price, while higher values for the strike contribute to reducing the estimated option price, and vice-versa. This confirms known relationships for call options.

Contrary to the common understanding of option pricing, the SHAP values indicate volatility is not so relevant to estimating the option price within the LSTM framework. A reasonable explanation is that the time-sequencing nature of LSTM enables the model to infer sufficient short-term volatility from the past five trading days of S&P 500 prices in its input. This is also supported by the fact that the performances of the LSTM and LSTM-GG do not differ significantly. Ruf and Wang (2020) points out that the rolling window is used by some practitioners when volatility is not used as an input feature. The rolling window approach thus allows each instance of the LSTM model for each window period to learn the implied volatility during the training period, thereby reducing the need for an explicit volatility input feature. This could imply that a model with a rolling window but without a volatility measure or that pays little

Table 6 DM statistics

	BS	BS-GG	Heston	MLP	MLP-GG	LSTM	LSTM-GG
BS		−56.61***	49.07***	113.87***	58.41***	194.77***	209.95***
BS-GG	56.61***		84.64***	135.44***	88.71***	208.19***	226.05***
Heston	−49.07***	−84.64***		77.37***	16.12***	144.78***	160.36***
MLP	−113.87***	−135.44***	−77.37***		−83.07***	47.58***	65.69***
MLP-GG	−58.41***	−88.71***	−16.12***	83.07***		248.87***	231.87***
LSTM	−194.77***	−208.19***	−144.78***	−47.58***	−248.87***		45.44***
LSTM-GG	−209.95***	−226.05***	−160.36***	−65.69***	−231.87***		−45.44***

$p<0.01$

Table 7 Theil U_1 statistics

Model	Theil U
BS	0.000031
BS-GG	0.000032
Heston	0.000029
MLP	0.000025
MLP-GG	0.000029
LSTM	0.000023
LSTM-GG	0.000022

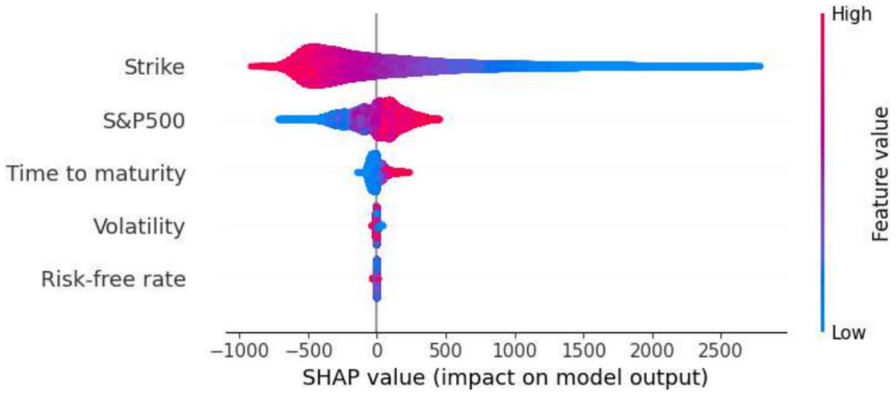

Fig. 7 SHAP values aggregated for each feature across all lags for the LSTM model using a sample of 10,000 option quotes

attention to the volatility measure could be vulnerable to large changes in volatility between the training and test period. It could also imply that the volatility measure would be more important with a longer training period. The same argument could be given for the risk-free rate, as it also does not change that much during the training period.

To explore that further, Fig. 8 depicts how the volatility explainability changes depending on the volatility and S&P 500 prices.[10] For high values of volatility, the volatility itself does not contribute that much to the option price prediction. However, when the volatility is low, it contributes more significantly to decreasing the estimated option price. The observation that volatility is relatively more important at low levels coincide well with Bollerslev et al. (2016), who report that low levels of volatility are associated with less measurement errors.

5.2 Impact of time to maturity and moneyness on pricing performance

The interpretability analysis finds strike price, underlying price, and time to maturity as the most important features determining model estimates. To examine how pricing

[10] Note all features are standardized.

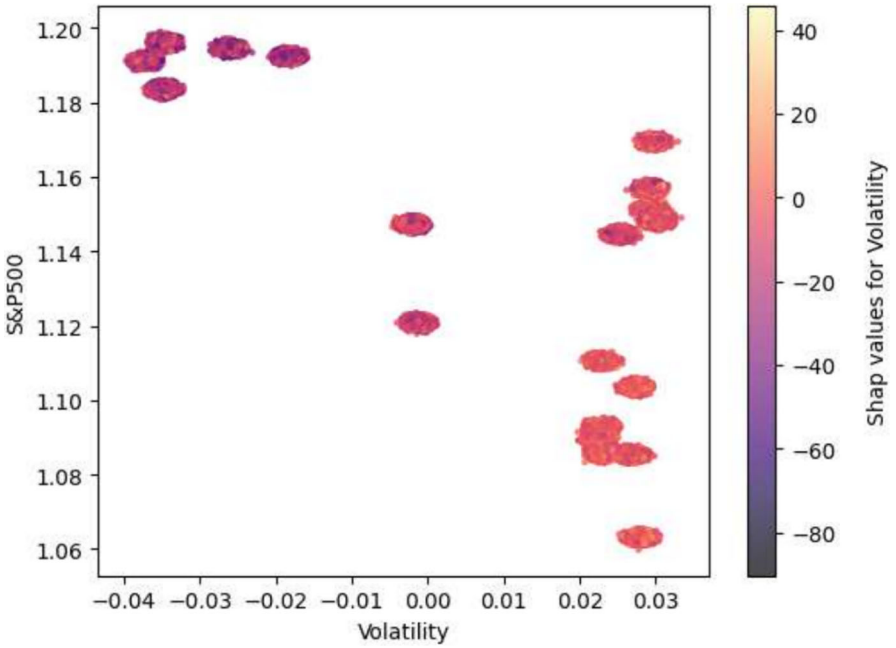

Fig. 8 SHAP value for volatility aggregated across all lags for the LSTM model using a sample of 10,000 option quotes

performance changes with regard to different option characteristics, we therefore further analyze the results by moneyness and time to maturity since these are the most important input features. Along these dimensions, performance is shown in Table 8 as RMSE and in Table 9 as MAE. These tables show that the deep learning methods outperform BS and Heston for options with longer time to maturity across all levels of moneyness, whereas BS and Heston outperform for options with shorter time to maturity. The LSTM model outperforms the MLP model for almost all maturity and moneyness combinations.

Tables 8 and 9 also reveal interesting implications with respect to the relevance of conditional volatility estimates, as represented by GG. Notably, the marginal effects of including GG estimates differ for MLP and LSTM. The overall tendency is that MLP benefits from including an explicit conditional volatility estimate. This is not surprising, since the MLP is not a recurrent neural network and hence does not inherently have memory. The LSTM architecture, on the other hand, is well suited to learn predictive signals from time-series dynamics. In most instances, LSTM outperforms LSTM-GG[11]. A likely explanation is that GG volatility estimates contain a too low signal-to-noise ratio, conditional on the information already contained in the history of option prices and learnt by the LSTM model. It is well established that true volatility is a latent variable and must be estimated with error, irrespective of the choice

[11] These conclusions are further supported by the pairwise Diebold-Mariano model comparisons in Table 6.

Table 8 RMSE of model predictions for full-year 2021 grouped by moneyness and maturity

Moneyness	Maturity	BS	BS-GG	Heston	MLP	MLP-GG	LSTM	LSTM-GG
<0.97	1-90	**4.20**	11.62	16.44	7.05	7.25	6.92	8.03
	91-180	**13.67**	29.22	36.68	18.53	18.75	16.24	19.28
	181-270	20.24	34.98	40.29	20.41	19.29	**15.81**	20.39
	271-450	29.47	46.45	48.80	23.85	22.74	**19.00**	26.02
	451-630	47.10	61.27	62.82	29.75	29.02	**22.30**	28.33
	631-1085	56.50	81.13	80.52	30.71	35.03	**25.66**	34.19
0.97−1.03	1-90	**16.22**	18.11	24.36	21.33	30.19	20.90	20.61
	91-180	44.85	50.85	47.62	33.48	36.09	**27.76**	29.48
	181-270	69.02	74.25	64.47	40.78	39.39	**29.02**	33.32
	271-450	86.40	93.92	80.36	42.63	40.96	**31.72**	40.07
	451-630	113.32	118.86	106.43	49.22	43.88	**33.22**	39.75
	631-1085	117.73	134.60	125.74	44.69	47.74	**32.36**	46.30
>1.03	1-90	13.19	11.75	**10.32**	28.19	37.22	28.52	23.06
	91-180	38.98	36.69	28.87	32.07	36.47	28.08	**27.18**
	181-270	49.46	46.71	37.15	37.84	39.50	31.89	**31.24**
	271-450	67.91	65.32	53.49	42.59	44.46	**39.01**	39.33
	451-630	84.95	83.37	71.25	55.41	52.93	**43.39**	44.51
	631-1085	100.20	102.62	92.54	**55.88**	64.97	55.90	63.39

Bold values indicate the best model in each category

Table 9 MAE of model predictions for full-year 2021 grouped by moneyness and maturity

Moneyness	Maturity	BS	BS-GG	Heston	MLP	MLP-GG	LSTM	LSTM-GG
<0.97	1-90	**1.79**	5.02	7.65	3.13	3.06	2.87	3.52
	91-180	**8.16**	19.27	24.51	13.22	12.90	10.77	13.31
	181-270	11.16	21.79	25.41	13.21	12.75	**10.00**	12.77
	271-450	17.27	30.60	32.11	15.90	15.59	**12.44**	16.84
	451-630	31.50	43.23	44.04	20.33	21.06	**15.17**	19.29
	631-1085	39.81	62.05	58.64	22.92	28.05	**19.47**	25.06
0.97−1.03	1-90	**10.57**	12.50	17.59	15.91	22.65	15.83	16.32
	91-180	38.32	43.55	38.98	28.70	28.86	**21.65**	23.61
	181-270	61.72	65.15	52.58	35.17	32.86	**23.89**	25.40
	271-450	78.82	82.90	65.45	37.32	34.80	**26.78**	31.17
	451-630	106.60	108.13	87.68	44.32	39.07	**28.20**	32.65
	631-1085	109.30	122.32	103.82	41.09	44.89	**27.58**	36.90
>1.03	1-90	7.63	6.94	**5.99**	20.37	30.60	22.76	18.42
	91-180	30.93	28.48	**21.36**	25.16	29.28	22.57	21.38
	181-270	38.54	35.85	28.08	30.42	31.41	25.60	**24.29**
	271-450	53.22	49.44	38.81	31.10	32.69	27.95	**27.03**
	451-630	67.78	64.24	53.03	41.60	35.58	**26.29**	29.26
	631-1085	78.95	78.31	67.66	30.84	38.86	**27.84**	40.80

Bold values indicate the best model in each category

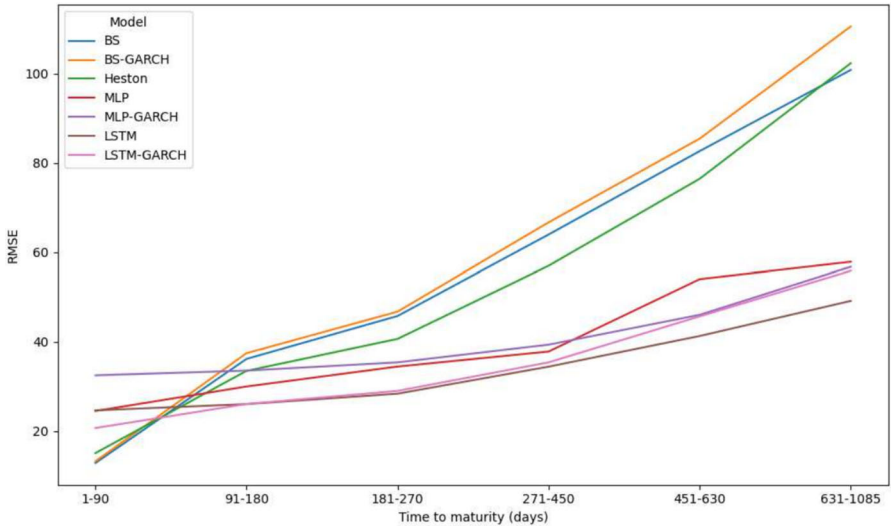

Fig. 9 RMSE of model predictions sorted by time to maturity

of volatility model. Hence, measurement errors, in the form of bias and variance in volatility estimates, are likely to be less distorting for MLP compared to LSTM.

Figure 9 shows pricing performance changes with respect to time to maturity. As RMSE is an absolute measure, it is natural that the RMSE increases for all models as maturity increases because this should increase the value of the options, all else being equal. Nonetheless, we see a steeper increase in RMSE for the BS and Heston models than for the deep learning methods. This might be caused by the BS assumptions, such as constant volatility and lognormal distribution, being less likely to hold over longer time periods. Similarly, while the Heston model accounts for stochastic volatility through the Cox-Ingersoll-Ross (CIR) process, its increasing RMSE suggests that its volatility dynamics may not fully capture long-term price behavior. Market participants could use a different volatility specification for longer maturity options, which could further impact the pricing accuracy of these models. The improved performance by the deep learning networks compared to BS and Heston for options with longer time to maturity is consistent with results by Liang and Cai (2022). Stark (2017) also show that neural networks perform better than BS for options with longer time to maturity when applying an MLP to price European call options on the DAX 30 index. The finding that BS overall performs better in options with a shorter time to maturity contradicts the results of Liang and Cai (2022). However, they perform less well relative to BS on short-maturity options and for short-maturity at-the-money options they indeed outperform the deep learning methods. Liang and Cai (2022) also find that the Heston model's RMSE is lower than the deep learning models for short-maturity options. Fig. 9 assures that the LSTM model outperforms the other models for maturities longer than 90 days. For shorter maturities, the LSTM-GG is the deep learning method with the best performance. However, BS, BS-GG, and Heston present the lowest RMSE in these scenarios.

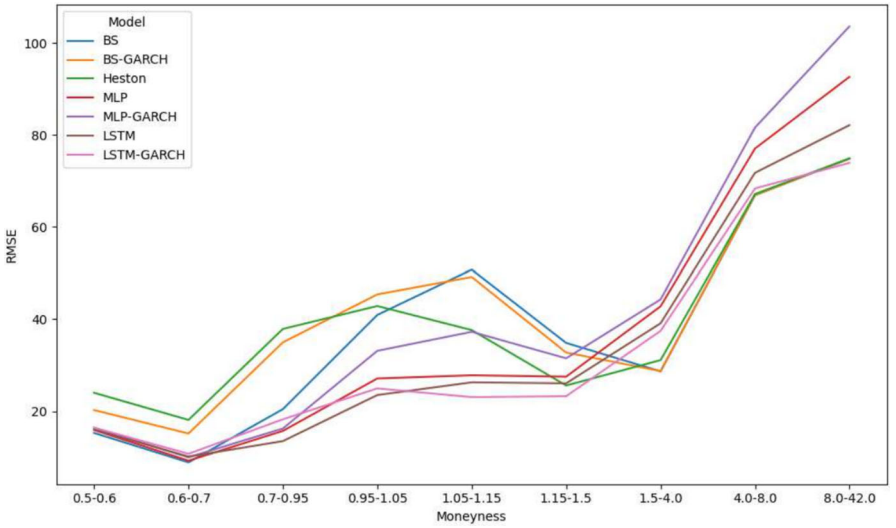

Fig. 10 RMSE of model predictions sorted by moneyness

Figure 10 shows how pricing performance varies according to moneyness. As with time to maturity, RMSE would generally be expected to increase for increasing moneyness as this makes the options more valuable. However, a more non-linear pattern is observed with a curved local peak around at-the-money options, especially for the parametric models. This could be attributed to the volatility smile, indicating that the volatility used is perhaps closer to the implied volatility we would see for out-of-the-money and in-the-money options than at-the-money options. The RMSE nonetheless continues to increase as the options become deeper in-the-money. The lower pricing errors on absolute measures for out-of-the-money options are in line with other literature (Liang and Cai 2022; Bennell and Sutcliffe 2004; Gencay and Salih 2003). For comparison, Bennell and Sutcliffe (2004) use an MLP to price European call options on the FTSE 100 index, and Gencay and Salih (2003) use an MLP to price European call options on the S&P 500 index.

The deepest out-of-the-money options also have a high RMSE compared to the less out-of-the-money options. All else equal, these options should be worth less, thus having a higher relative error. Both parametric and deep learning methods have trouble pricing the deep-in-the-money options, being the latter clearly worst. This could be due to these options being less traded, leading to more noise and mispricings, thus making it difficult for the models to price these options correctly.

An attempt to address the issue of varying performance across moneyness levels is proposed by Gradojevic et al. (2009) by training separate neural networks for each class of options categorized by moneyness and time to maturity. Yang et al. (2017) argues that these manual heuristics for dividing the options is suboptimal as it does not adapt to changing market conditions over time and proposes a dynamic divide-and-conquer method to divide the option space. Another approach to increase the performance is simply to filter out deep-in-the-money options from the data set to

increase the performance of our deep learning methods. Yang et al. (2017) discard in-the-money options and justifies this by stating that trading is very inactive for those options, and thus their prices are not reliable. Bennell and Sutcliffe (2004) show that their neural networks performs better than BS for out-of-the-money options, but worse for in-the-money options. However, when they restrict the sample space by excluding options with moneyness greater than 1.15 and maturity longer than 200 days, the performance of the neural networks becomes comparable to that of BS, even for the in-the-money options.

6 Conclusion

This paper presents an LSTM model to price European call options on the S&P 500 index. To simulate the real-world scenario of model deployment, we utilize a rolling window approach, resulting in the training of 12 instances of the LSTM model, one for each month of 2021. The LSTM model is benchmarked against the BS, the Heston, and an MLP network. The results show that the LSTM model outperforms benchmark models for options with longer time to maturity in each of the defined moneyness intervals. However, for options with a shorter time to maturity, the BS and Heston models perform better than both deep learning methods. Our interpretability analysis using SHAP shows that the strike price, the price of the underlying, and the time to maturity are most important for model predictions. Most notably, the time-sequencing nature of LSTM enables the model to capture sufficient short-term volatility from traded options over the past five trading days. As such, the proposed LSTM model is less exposed to measurement errors in the volatility input. This is in contrast to parametric option pricing models, which rely heavily on implied volatilities inferred from option prices - inevitably involving model risk. Hence, the proposed LSTM model sidesteps the crucial applied problem of explicitly estimating an appropriate volatility parameter specification in parametric models.

An interesting area for further research is to use implied volatility instead of historical volatility, which increased the performance of the ANN proposed by Andreou et al. (2008), or using forecasting methods for volatility. To further utilize the time-forecasting ability of LSTM models, a separate LSTM model could be trained to forecast volatility using a time series of the S&P 500 as input. Exploring other volatility measures could also be beneficial for the BS model. Other benchmarking models like the parametric model including jumps from Merton (1976) or other neural networks like CNNs could also be explored.

Another limitation is only using the same input variables as the parametric models. These variables likely do not capture all the information a market participant would use when trading options, making the model less flexible in terms of pricing options during changing financial conditions. Further research could therefore explore the use of different input features for our proposed LSTM model. Examples of other input features include prices or other data for concurrently traded options. It seems intuitive to assume that such information contains useful insight into what investors see as the likely risk-neutral probability distribution at different times in the future. To make

this feasible in terms of input dimensionality, an initial CNN layer could be used for feature extraction.

Finally, our research could be extended to other use cases for options like hedging, risk management, or trading that could benefit from the ability of LSTM models to capture complicated non-linear and time-dependent patterns while learning and adapting to dynamic market conditions.

A Appendices

A.1 Core concepts of LSTM networks

The LSTM network provides a solution to the challenge of long-term dependencies with a structure that consists of three gates: an input gate i_t, a forget gate f_t and an output gate o_t. The subscript t represents a single time step. These gates control the flow of information used to calculate the output at each state. Figure 11 illustrates the LSTM unit structure, referred to as a memory cell, at a single time step.

The forget gate decides what information to throw away from the cell state c_t using sigmoid as the activation function, as given by Eq. 19. It takes the output from the previous hidden state h_{t-1} and the new input x_t as the total set of inputs at this time step. It outputs a number between 0 and 1 for each number in the previous cell state

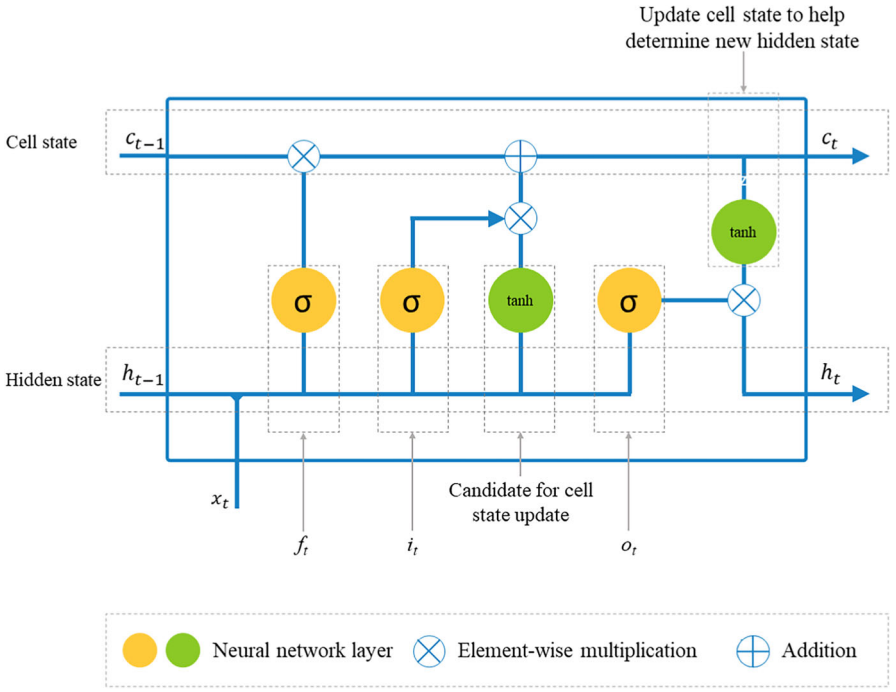

Fig. 11 A single LSTM memory cell showing the forget gate, the input gate and the output gate

c_{t-1}. An output of 0 signals to completely get rid of the information, and an output of 1 signals to keep all the information.

$$f_t = \sigma(W_{f,x}x_t + W_{f,h}h_{t-1} + b_f) \tag{19}$$

where σ denotes the sigmoid activation function, $W_{f,x}$ and $W_{f,h}$ denote weight matrices, and b_f denotes the bias term.

The process of deciding what information to store in the cell at this time step can be viewed as two steps. In step one, the input gate decides which values to update using sigmoid activation, as given by Eq. 20. Then, we generate a vector of new candidate values $\tilde{c}_t$ using tanh, as given by Eq. 21. To create an update to the state, we combine the outputs from these two steps.

$$i_t = \sigma(W_{i,x}x_t + W_{i,h}h_{t-1} + b_i) \tag{20}$$

$$\tilde{c}_t = tanh(W_{\tilde{c},x}x_t + W_{\tilde{c},h}h_{t-1} + b_{\tilde{c}}) \tag{21}$$

where $W_{i,x}$, $W_{i,h}$, $W_{\tilde{c},x}$ and $W_{\tilde{c},h}$ denote weight matrices, and b_i and $b_{\tilde{c}}$ denote bias terms.

The update from the old cell state c_{t-1} to the new cell state c_t is given by Eq. 22.

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{22}$$

where $\odot$ represents element-wise multiplication.

The final output is a filtered version of the cell state c_t. First, a sigmoid layer decides which parts of the cell state to output, as given by Eq. 23. Then, tanh is applied to the cell state to transform its values between -1 and 1. These two outputs are multiplied so that only the desired parts are given as output, according to Eq. 24.

$$o_t = \sigma(W_{o,x}x_t + W_{o,h}h_{t-1} + b_o) \tag{23}$$

$$h_t = o_t \odot tanh(c_t) \tag{24}$$

where $W_{o,x}$ and $W_{o,h}$ denote weight matrices, and b_o denote the bias term.

A.2 Performance metrics

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2} \tag{25}$$

$$MAE = \frac{1}{m}\sum_{i=1}^{m}|y_i - \hat{y}_i| \tag{26}$$

$$\text{Theil } U_1 = \frac{\sqrt{\frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)}}{\frac{1}{m}\sum_{i=1}^{m} y_i^2} \tag{27}$$

$$\text{Bias proportion} = \frac{(\frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i))^2}{\frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2} \tag{28}$$

$$\text{Variance proportion} = \frac{(\sigma_{\hat{y}} - \sigma_y)^2}{\frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2} \tag{29}$$

$$\text{Covariance proportion} = \frac{2(1-\rho)\sigma_{\hat{y}}\sigma_y}{\frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2} \tag{30}$$

Where m is the number of options, y_i is the observed price and $\hat{y}_i$ is the price given by the model, ρ is the correlation between $\hat{y}$ and y, and $\sigma_{\hat{y}}$ and σ_y is the standard deviation of $\hat{y}$ and y respectively.

Diebold Mariano (DM) test (Diebold and Mariano 1995) is used to perform a 2-by-2 comparison of the models using the MSE loss function to evaluate the equality of the model predictions. Given a loss function $l(\cdot)$, the series of difference in loss between the two prediction series is calculated as:

$$d_i = l(\hat{y}_{i,1} - y_{t,1}) - l(\hat{y}_{i,2} - y_{i,2}) \tag{31}$$

Given the null hypothesis of:

$$H_0 : E[d_i] = 0 \tag{32}$$

The DM test statistic is:

$$DM = \frac{\frac{1}{m}\sum_{i=1}^{m} d_i}{\sqrt{2\pi \hat{f}_d(0)/m}} \tag{33}$$

Where $\hat{f}_d(\cdot)$ is the spectral density of $\{d_i\}$.

Acknowledgements This research was partially funded by The Research Council of Norway throughout the project COMPAMA (https://www.ntnu.edu/compama/), with grant number 314609.

Author Contributions Rygg, Vinje, and Wu: Conceptualization, data curation, methodology, formal analysis, implementation, interpretation. Rodge: Implementation, interpretation, and reviewing. Risstad: Supervision, conceptualization, methodology, validation, interpretation, writing—improving original draft, editing, and reviewing. Westgaard: Supervision, conceptualization, validation. Pimentel: Funding, interpretation, writing - improving original draft, editing, and reviewing.

Funding Open access funding provided by NTNU Norwegian University of Science and Technology (incl St. Olavs Hospital - Trondheim University Hospital)

Declarations

Conflict of interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Amilon, H.: A neural network versus Black-Scholes: a comparison of pricing and hedging performances. J. Forecast. **22**, 317–335 (2003)

Anders, U., Korn, O., Schmitt, C.: Improving the pricing of options: a neural network approach. J. Forecast. **17**(5–6), 369–388 (1998)

Andreou, P.C., Charalambous, C., Martzoukos, S.H.: Pricing and trading European options by combining artificial neural networks and parametric models with implied parameters. Eur. J. Oper. Res. **185**(3), 1415–1433 (2008)

Apley, D.W., Zhu, J.: Visualizing the effects of predictor variables in black box supervised learning models. J. R. Stat. Soc. Ser. B: Stat. Methodol. **82**(4), 1059–1086 (2020)

Ball, C., Roma, A.: Stochastic volatility option pricing. J. Financ. Quant. Anal. **29**(4), 589–607 (1994)

Bates, D.S.: The crash of 87: was it expected? The evidence from options markets. J. Financ. **46**(3), 1009–1044 (1991)

Bennell, J., Sutcliffe, C.: Black-Scholes versus artificial neural networks in pricing FTSE 100 options. Intell. Syst. Account. Financ. Manag. **12**(4), 243–260 (2004)

Bhandari, H.N., Rimal, B., Pokhrel, N.R., et al.: Predicting stock market index using lstm. Mach. Learn. Appl. **9**, 100320 (2022)

Black, F., Scholes, M.: The pricing of options and corporate liabilities. J. Political Econ. **81**(3), 637–654 (1973)

Bollerslev, T., Patton, A.J., Quaedvlieg, R.: Exploiting the errors: a simple approach for improved volatility forecasting. J. Econom. **192**(1), 1–18 (2016)

Boyle, P.P.: Options: a Monte Carlo approach. J. Financ. Econ. **4**(13), 323–338 (1977)

Brennan, M.J., Schwartz, E.S.: Finite difference methods and jump processes arising in the pricing of contingent claims: a synthesis. J. Financ. Quant. Anal. **13**, 461–474 (1978)

Chen, K., Zhou, Y., Dai, F.: A LSTM-based method for stock returns prediction: A case study of China stock market. 2015 IEEE International Conference on Big Data (Big Data) (2015)

Corrado, C., Su, T.: Skewness and kurtosis in S&P 500 index returns implied by option prices. J. Financ. Res. **19**, 175–192 (1996)

Cox, J., Ross, S., Rubinstein, M.: Option pricing: a simplified approach. J. Financ. Econ. **7**(3), 229–263 (1979)

Diebold, F.X., Mariano, R.S.: Comparing predictive accuracy. J. Bus. Econ. Stat. **13**(3), 253–263 (1995)

Du Plooy, R., Venter, P.J.: Pricing vanilla options using artificial neural networks: Application to the South African market. Cogent Econ. Financ. **9**(1), 1914285 (2021)

Ferguson, R., Green, A.: Deeply learning derivatives (2018)

Gao, Y., Wang, R., Zhou, E.: Stock prediction based on optimized lstm and gru models. Sci. Program. **1**, 4055281 (2021)

Garcia, R., Gencay, R.: Pricing and hedging derivative securities with neural networks and a homogeneity hint. J. Econom. **94**(1–2), 93–115 (2000)

Gencay, Q., Qi, M.: Pricing and hedging derivative securities with neural networks: Bayesian regularization, early stopping, and bagging. IEEE Transact. Neural Netw. **12**(4), 726–734 (2001)

Gencay, R., Salih, A.: Degree of mispricing with the Black-Scholes model and nonparametric cures. Ann. Econ. Financ. **4**(1), 73–101 (2003)

Glosten, L.R., Jagannathan, R., Runkle, D.E.: On the relation between the expected value and the volatility of the nominal excess return on stocks. J. Financ. **48**(5), 1779–1801 (1993)

Gradojevic, N., Gencay, R., Kukolj, D.: Option pricing with modular neural networks. IEEE Transact. Neural Netw. **20**(4), 626–637 (2009)
Hagan, P.S., Kumar, D., Lesniewski, A.S., et al.: Managing smile risk. The Best of Wilmott **1**, 249–296 (2002)
Heston, S.: A closed-form solution for options with stochastic volatility with applications to bond and currency options. Rev. Financ. Stud. **6**(2), 327–343 (1993)
Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
Hornik, K., Stinchcombe, M., White, H.: Multilayer feedforward networks are universal approximators. Neural Netw. **2**(5), 359–366 (1989)
Hull, J.: An analysis of the bias in option pricing caused by a stochastic volatility. Adv. Futures Options Res. **3**, 29–61 (1988)
Hull, J., White, A.: The pricing of options on assets with stochastic volatilities. J. Financ. **42**(2), 281–300 (1987)
Hutchinson, J.M., Lo, A.W., Poggio, T.: A nonparametric approach to pricing and hedging derivative securities via learning networks. J. Financ. **49**(3), 851–889 (1994)
Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Bach F, Blei D (eds) Proceedings of the 32nd International Conference on Machine Learning, Proceedings of Machine Learning Research, vol 37. PMLR, Lille, France, pp 448–456 (2015)
Jang, J., Yoon, J., Kim, J., et al.: DeepOption: a novel option pricing framework based on deep learning with fused distilled data from multiple parametric methods. Inf. Fusion **70**, 43–59 (2021)
Jarrow, R., Rudd, A.: Approximate option valuation for arbitrary stochastic processes. J. Financ. Econ. **10**(3), 347–369 (1982)
Jin, Z., Yang, Y., Liu, Y.: Stock closing price prediction based on sentiment analysis and LSTM. Neural Comput. Appl. **32**, 9713–9729 (2019)
Johnson, H., Shanno, D.: Option pricing when the variance is changing. J. Financ. Quant. Anal. **22**(2), 143–151 (1987)
Jurczenko, E., Maillet, B., Negrea, B.: Stock closing price prediction based on sentiment analysis and LSTM. Quant. Financ. **4**(5), 479–488 (2004)
Kingma, DP.: Adam: A method for stochastic optimization (2014)
Li, J., Bu, H., Wu, J.: Sentiment-aware stock market prediction: A deep learning method. 2017 International Conference on Service Systems and Service Management pp 1–6 (2017)
Liang, L., Cai, X.: Time-sequencing European options and pricing with deep learning-analyzing based on interpretable ale method. Expert Syst. Appl. **187**, 115951 (2022)
Liu, D., Wei, A.: Regulated LSTM artificial neural networks for option risks. FinTech **1**(2), 180–190 (2022)
Liu, S., Oosterlee, C., Bohte, S.: Pricing options and computing implied volatilities using neural networks. Risks **7**(1), 16 (2019)
Liu, Y., Zhang, X.: Option pricing using lstm: a perspective of realized skewness. Mathematics **11**(2), 314 (2023)
Loshchilov, I., Hutter, F.: Decoupled weight decay regularization (2017)
Lundberg, S., Lee, SI.: A unified approach to interpreting model predictions. Adv. Neural Inf. Process. Syst. 30 (NIPS 2017) (2017)
Merton, R.: Theory of rational option pricing. Bell J. Econ. Manag. Sci. **4**(1), 141–183 (1973)
Merton, R.C.: Option pricing when underlying stock returns are discontinuous. J. Financ. Econ. **3**(1–2), 125–144 (1976)
Molnar, C.: Interpretable Machine Learning: A Guide for Making Black Box Models Explainable, 2nd edn. Lulu. com (2022)
Qian, L., Zhao, J., Ma, Y.: Option pricing based on GA-BP neural network. Proc. Comput. Sci. **199**, 1340–1354 (2022)
Qiu, J., Wang, B., Zhou, C.: Forecasting stock prices with long-short term memory neural network based on attention mechanism. PloS one **15**(1), e0227222 (2020)
Rendleman, R., Bartter, B.: Two-state option pricing. J. Financ. **34**, 1093–1110 (1979)
Ruf, J., Wang, W.: Neural networks for option pricing and hedging: a literature review. J. Comput. Financ. **24**(1), 1–46 (2020)
Scott, L.: Option pricing when the variance changes randomly: theory, estimation, and an application. J. Financ. Quant. Anal. **22**(4), 419–438 (1987)

Sezer, O.B., Gudelek, M.U., Ozbayoglu, A.M.: Financial time series forecasting with deep learning: a systematic literature review: 2005–2019. Appl. Soft Comput. **90**, 106181 (2020)
Stark, L.: Machine learning and options pricing: A comparison of black-scholes and a deep neural network in pricing and hedging dax 30 index options. Master's thesis, Aalto University (2017)
Wei, X., Xie, Z., Cheng, R., et al.: An intelligent learning and ensembling framework for predicting option prices. Emerg. Markets Financ. Trade **57**(15), 4237–4260 (2021)
Wiggins, J.: Option values under stochastic volatility: theory and empirical estimates. J. Financ. Econ. **19**(2), 351–372 (1987)
Yan, H., Ouyang, H.: Financial time series prediction based on deep learning. Wirel. Pers. Commun. **102**(2), 683–700 (2018)
Yang, Y., Zheng, Y., Hospedales, T.: Gated neural networks for option pricing: Rationality by design. In: Proceedings of the AAAI conference on artificial intelligence (2017)
Yao, J., Li, Y., Tan, C.: Option price forecasting using neural networks. Omega **28**(4), 455–466 (2000)
Zouaoui, H., Naas, M.N.: Option pricing using deep learning approach based on lstm-gru neural networks: case of London stock exchange. Data Sci. Financ. Econ. **3**(3), 267–284 (2023)