Received: 1 August 2021 | Accepted: 11 December 2021

DOI: 10.1002/fut.22302

RESEARCH ARTICLE

THE JOURNAL OF FUTURES MARKETS WILEY

Multistep forecast of the implied volatility surface using deep learning

Nikita Medvedev | **Zhiguang Wang**

Ness School of Management and Economics, South Dakota State University, Brookings, South Dakota, USA

Correspondence
Zhiguang Wang, Ness School of Management and Economics, South Dakota State University, Brookings, SD 57007, USA.
Email: zhiguang.wang@sdstate.edu

Abstract

Modeling implied volatility surface (IVS) is of paramount importance to price and hedge an option. We contribute to the literature by modeling the entire IVS using convolutional long-short-term memory (ConvLSTM) and long-short-term memory (LSTM) neural networks to produce multivariate and multistep forecasts of the S&P 500 IVS. Using daily SPX options data (2002–2019), we find that both LSTM and ConvLSTM fit the training data extremely well with mean absolute percentage error (MAPE) being 3.56% and 3.88%, respectively. The ConvLSTM (8.26% MAPE) model significantly outperforms LSTM and traditional time series models in predicting the IVS out of sample.

KEYWORDS

ConvLSTM, convolutional neural networks, deep learning, implied volatility, LSTM

JEL CLASSIFICATION

G13, G17

1 | INTRODUCTION

Predicting volatility for the S&P 500 Index (SPX) is an important task for traders and dealers of SPX options, because volatility is both a risk measure and an input to options pricing formula. The significance of volatility prediction is reflected in the popularity of SPX options: the Chicago Board of Exchange (CBOE) reported an average daily volume (ADV) of 1.2 million contracts with notional value over $400 billion in 2020. Volatility measure takes two forms realized volatility (RV) and implied volatility (IV). RV is standard deviation of historical returns. IV is embedded in the price of an option, reflecting the investor's view for the underlying asset's future returns. Both measures of volatility are closely connected via the notion of volatility (variance) risk premium, namely their difference. In this study we aim to directly predict IV for the SPX options across the entire surface, that is, IV across strikes and maturities, as opposed to RV commonly seen in the literature (see Samsudin & Mohamad, 2016, for a recent survey).

Predicting the entire IV surface is an enormous challenge given its multidimensional and dynamic nature. For instance, Figure 1 is a snapshot of average IV across 20 moneyness groups and 4 contract months during the period between 2019-08-19 and 2019-12-31. It shows that the IV is higher for OTM puts than OTM calls and that IVS has an upward term structure. The whole implied volatility surface (IVS) also exhibits irregular and changing curvature over time, which further complicates the time series prediction.

The vast majority of volatility forecasting literature is concerned with RV. The literature dates back to the classical autoregressive conditional heteroscedasticity (ARCH) (Engle, 1982) and GARCH (Bollerslev, 1986) models. Hansen and Lunde (2005) applied over 330 GARCH models to volatility forecasting. Corsi's (2008) heterogeneous autoregressive

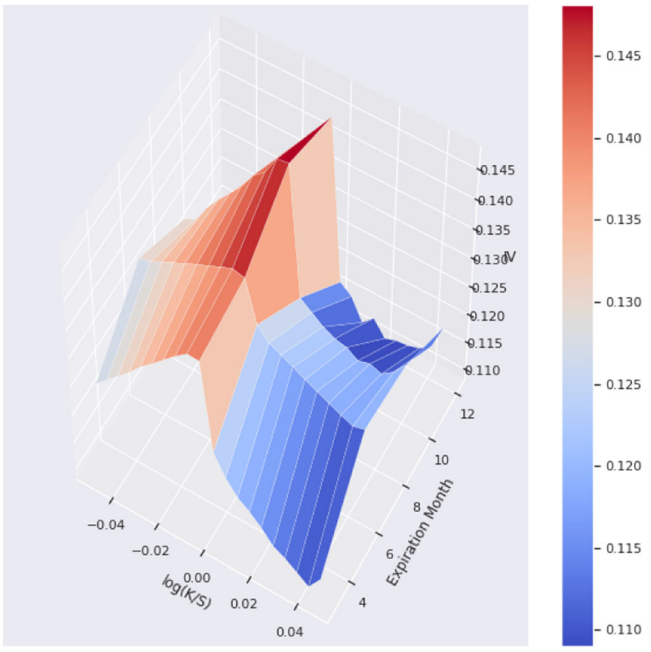

FIGURE 1 Average interpolated implied volatility surface between 2019-08-19 and 2019-12-31

(HAR) model is an improvement over GARCH or AR models in terms of predicting RV. More recently, a variety of machine learning models based on neural network have successfully been applied, such as Xiong et al. (2016) using long-short-term-memory (LSTM) to predict RV of the SPX index, Luong and Dokuchaev (2018) using random forest to predict the direction of RV, Luo et al. (2018) combining recurrent neural networks with GARCH model. Neural networks have shown advantages in estimating complex nonlinear functions, which is often achieved through the depth of the networks, number of neurons, and the activation (transfer) functions between the layers, and given a suitable degree of complexity can be trained to estimate any nonparametric function. This, however, comes at a higher performance cost and a lengthy training process. However, the rise of the new data warehousing techniques, as well as high-performance computing in the 1990s, kick-started a new branch of nonparametric financial derivative research that predominantly focuses on providing efficient solutions to the partial differential equations. Often the use of the nonparametric algorithms and techniques, such as forests, boosting, support vectors, and neural networks for volatility estimation can lead to far more superior results (Park et al., 2014).

There is only scant literature in predicting the IVS, especially using machine learning techniques. One exception is on forecasting the VIX index with neural networks, such as Hosker et al. (2018) using recurrent neural network (RNN) and LSTM. The VIX index is a 30-day IV of the SPX index. It is a univariate prediction, hence a difference exercise from our objective–prediction of the whole IV surface. Another related strand of research is to demonstrate the capability of deep learning architectures as an efficient solver to nonlinear functions, such as the Black and Scholes (1973) formula in Culkin (2017) and the Heston (1993) formula in Liu et al. (2019). Given the functional relationship between IV and Black–Scholes formula, the validity of pricing via deep learning also implies its ability to model IV skew (cross sectional of IV) at a point in time.

To the best of our knowledge, there is no existing research on applying machine learning techniques to predict the entire IVS over time. We contribute to the literature by focusing on IVS forecasting, and by proposing a data-driven machine learning technique for pricing IV and multistep forecast of both IV skew and IV term structure. The goal of this study is to answer the two questions: (1) Can the recurrent neural network architecture significantly outperform traditional time series models in a multistep out-of-sample forecast of the IVS? While multilayer perceptron-based neural network have shown strength in option pricing and volatility estimation, supervised time-series predictions are computationally expensive; (2) Does encoding spatiotemporal dynamics of the IVS significantly improve the IVS forecasts? All traditional time series forecasting models require flattening of the data input vector—this loses the important properties of the IV term structure and the IV skew.

The specific machine learning models we consider are LSTM and Convolutional LSTM (ConvLSTM). We benchmark the two models against classical time series models such as vector autoregressive (VAR) and vector error correction (VEC) models. We apply the models to the training data set of daily SPX index options from 2002-01-02 to

2019-08-16, leaving the last 90 days for out of sample comparisons. We find that both LSTM and ConvLSTM can fit the training (sample) data extremely well with mean absolute percentage error (MAPE) being 3.56% and 3.88%, respectively. For out of sample data, the ConvLSTM (8.26% MAPE) significantly outperforms traditional time series models (13.72% MAPE), as well as the benchmark LSTM model (18.74% MAPE) in predicting the IVS for a 1-, 30-, and 90-day horizon, for out-of-the-money (OTM) and at-the-money (ATM) calls and puts.

The rest of the paper is organized as follows: Section 2 reviews the existing literature and describes how our research fits within the literature; Section 3 describes how we filter and format the data set, and the machine learning models we use. We also detail the algorithms used for training various models and for prediction. Section 4 presents the in-sample fit and out of sample results. Section 5 concludes with the superiority of ConvLSTM in IVS prediction.

2 | LITERATURE REVIEW

In this section, we will review econometric and parametric methods that are relevant for volatility time-series forecasting. We will discuss the more recent methodologies used for option IV forecasting and option pricing, with the help of traditional Artificial Neural Networks (ANN) and more recent deep learning architectures. Lastly we conclude with our intention to fill the gap in the literature by utilizing a new recurrent neural network to predict the IVS.

2.1 | Historical volatility forecasting

In contrary to the theoretical models, time series models seek to explain the movement in volatility using some autoregressive property of the series. Autoregressive conditional heteroskedasticity (ARCH) and general autoregressive conditional heteroskedasticity (GARCH) models for modeling and forecasting volatility are explored in voluminous papers and are among one of the most widely used benchmark models for both realized and IV forecasting. Hansen and Lunde (2005) benchmarked over 339 types of volatility models, against the standard ARCH(1) and GARCH(1,1) for one-day ahead forecast of RV. They find no evidence that any of the tested models can outperform the standard GARCH (1,1) model and find ARCH(1) model to be inferior to other models.

Gospodinov et al. (2006) proposed several parametric and nonparametric methods for estimating RV and IV. They find evidence of volatility clustering, high persistence of the volatility, and volatility to be a long, and slowly mean-reverting process, where the IV has a longer memory than RV. The forecasts were performed one-step ahead for both realized and IV. They find evidence that IV contains valuable information about RV, which, if included, can significantly improve the performance of predicting RV over a long time horizon.

Xiong et al. (2016) proposed a new methodology for predicting future RV of the SPX, using macroeconomic factors obtained from Google Trends. A 25-dimensional vector of 25 trend series representing a selected number of keywords, beginning January 1, 2004, is fed into the LSTM (Hochreiter & Schmidhuber, 1997) neural network (LSTM) with one hidden LSTM cell. One-step forecast of future RV is made, and researchers find that the LSTM outperforms the benchmark GARCH model based on the mean-absolute-percentage error metric and the root mean-squared error.

More recently, Luong and Dokuchaev (2018) used a random forest model to forecast the direction of RV for multistep out of sample forecast on high-frequency data. The proposed model produces accuracy of 80.05%, 72.85%, and 65.22%, for 1-day, 5-day, and 22 forecasts. They find that the long term accuracy of the directional forecast of the random forest model decreases over the 5 to 22-day forecast. However, the random forest model was able to outperform the benchmark HAR model (Corsi, 2008). When compared to GARCH or AR models, the HAR model helps capture the long autoregressive persistence of RV due to the leptokurtosis of the returns that can be observed at different periods. Specifically for predictions, the HAR model uses realized volatilities for the previous day, week, and month interval. The model captures the aggregated high-frequency variance and RV over multiple horizons and has shown to outperform the AR and ARFIMA models for short term volatility forecasts.

Luo et al. (2018) proposed a Neural Stochastic Volatility Model (NSVM), a joint network consisting of a pair of stacked stochastic recurrent neural networks. Internally, the NSVM model closely resembles a special case of GARCH (1,1), often used for volatility forecasting, and a stochastic component described by the GBM with the random disturbance factor. The underlying model belongs to the class of stochastic volatility models, where the generative sequence consists of the joint distribution of the input and stochastic component, which get propagated to the hidden states of the RNN. In comparison to the previous studies, NSVM is composed of two NNs, each with 10 hidden nodes,

which get bundled into a two-layered fully connected neural network. The proposed NSVM model outperforms standard GARCH(1,1), EGARCH(1,1), and GJR-GRACH(1,1,1) models for volatility modeling and forecasting, however, the model takes longer to train when compared to the traditional econometric models.

2.2 | IV forecasting

In terms of traditional time series modeling for IV, Majmudar and Banerjee (2004) explored various GARCH family models for VIX forecasting and concluded, similarly to many other researchers, that EGARCH(1,1) provides the the best overall results. Hosker et al. (2018) compares the performance of six different supervised learning models, including the RNN, and LSTM on the 1-month VIX futures contract and options data. The models are benchmarked against a linear regression, principal component analysis (PCA), and Autoregressive Integrated Moving Average (ARIMA) model over a 3 to 5-day forecast window. They find that RNN and LSTM had overall lower mean absolute errors when compared to other models.

A number of researchers have utilized deep learning architectures for option pricing, or IV estimation. For instance, Liu et al. (2019) proposes a machine learning technique for efficiently computing IV of three types: BS equation, Heston model, and Brent's root finding the calculation of IV. As a result, they present a more efficient solver, which boosts computational performance by a large factor. A similar technique is proposed by Culkin (2017), where a simple NN is trained on a data set of 300,000 simulated options to be able to estimate the BS equation. The model achieves very small out of sample RMSE and MAPE.

In summary, the majority of the literature focuses on future RV forecasting. The domain of nonparametric IV forecasting is still new but started expanding in recent years with the emergence of several new machine learning techniques. Machine learning techniques are more often used in the domain of IV modeling, and neural network architectures are more often used for option pricing. However, utilizing machine learning and neural networks is far less common for forecasting the IV or modeling and forecasting the IVS in general, due to the existing parametric techniques that work reasonably well, and are widely accepted. Neural networks have shown strengths in estimating the BS equation. Recurrent neural network architectures have shown strength in univariate time series forecasting, in many cases outperforming traditional AR, GARCH, and HAR models. The research aims to bridge the gap between forecasting and nonparametric modeling of IV. We apply two recurrent neural network architectures and produce multi-step forecast of the entire IVS.

3 | DATA AND METHODOLOGY

3.1 | Data description

This study focuses on the US stock market index S&P 500 and its most popular financial product offered through the CBOE: the S&P 500 options (SPX) index, SPX option call and put chain. The data for the European call and put options on the SPX are obtained from Delta Neutral for the period between 2002-01-02 and 2019-12-31.[1] The historical open-high-low-close (OHLC) data for IRX—The US treasury bill tracking index and SPX index for the same periods are collected from Yahoo Finance. Delta Neutral data contain the end of day quotes, IV, and sensitivity information for the traded SPX options.

There are a total of 52,914 unique contracts series[2] in the sample. Among them there are numerous contracts with nonstandard expiration dates, such as weekly expiries, and with deep in or out of moneyness. Due to the computational constraints, we begin from down-sampling the contracts to four standard quarterly contracts: March, June, September, December. By down-sampling, the number of unique contracts is reduced to 9195 (Figure 2).

We reconstruct the surface by using a splicing technique where the contracts are stitched “head to toe” on expiration to form continuous series. Continues series are required for constructing balanced panel, because the models used in this study are not aware of the expiring nature of the contracts, and expect a continuous input.

[1]The starting date of 2002-01-02 is chosen due to the data availability.
[2]Series are labeled unique when a series has a unique strike-expiration combination.

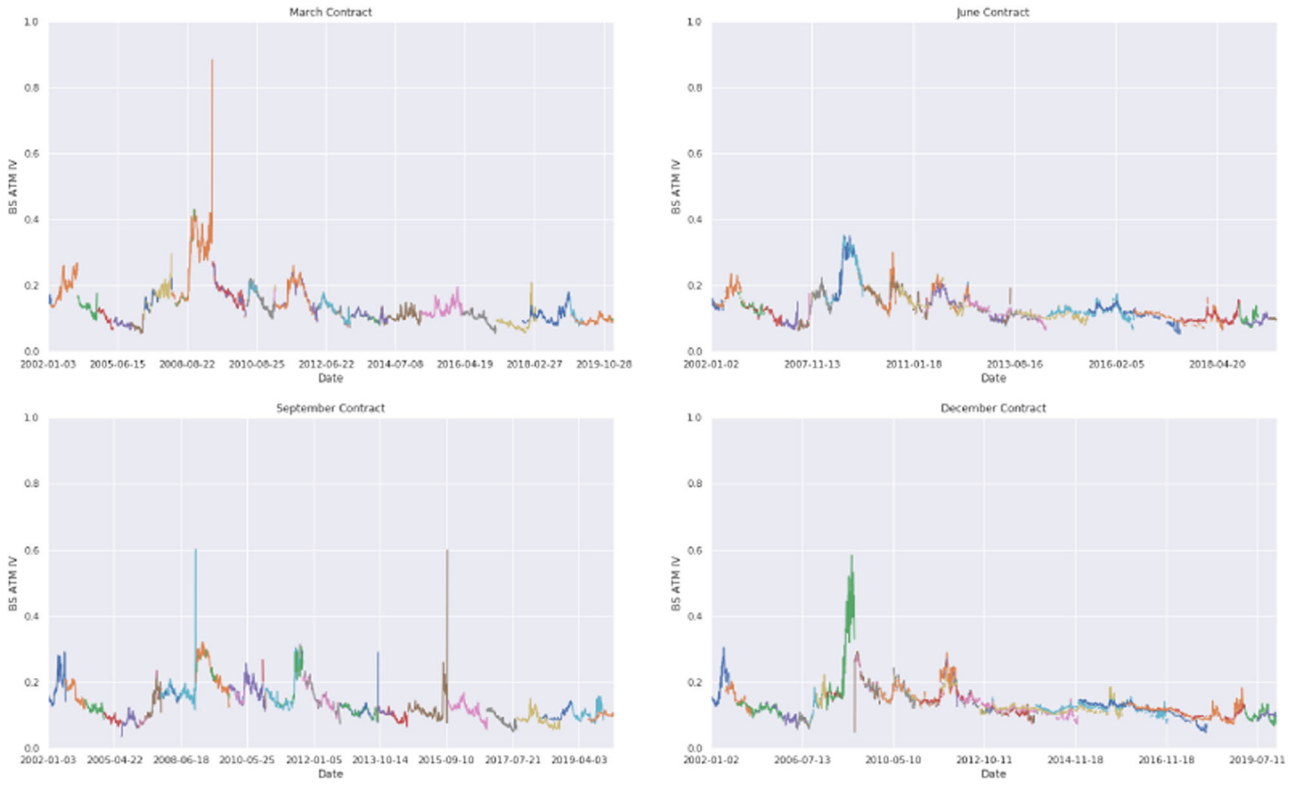

FIGURE 2 At-the-money (ATM) IV for March, June, September and December contracts and expirations. The daily values of average ATM implied volatilities of the S&P 500 index for respective contract months from 2002-01-02 to 2019-12-31 are shown

Because the data set contains an unequal number of time series observations for each contract. To address the unbalanced panel, a pivoting operation where the number of observations is the same for all contract series is required. After pivoting the unbalanced panel, we employed a two-way (forward and backward direction) linear interpolation to fill the missing BS-IV values. We found that in some cases, the IV estimation method produced infinities or zeros, so these values were blanked out, to be interpolated. This interpolation is also needed to produce the target IV bins properly. The interpolation is conducted in three steps: (1) Interpolate based on the IV term structure; (2) Interpolate based on the volatility skew; (3) Backfill based on the contracts log-moneyness group to fill remaining missing values. Descriptive statistics of the transformed IV observations are summarized in Table 1. The step-by-step methodology along with the the IV binning technique discussed in Section 3.3 is displayed in Algorithm 1.

The series follow an identical format where the panel is expressed as either $m \times n$ two-dimensional matrix or three-dimensional matrix, where i is a closing day index, j is contract expiration month index, k is option contract's log-moneyness bin ranging from −0.05 to 0.05, where −0.05 represents OTM put options and 0.05 OTM call options and x is BS-IV:

$$X_{i,j,k} = \begin{bmatrix} x_{0,0,0.10} & x_{0,0,0.15} & x_{0,0,0.20} & \cdots & x_{0,0,k} \\ x_{1,0,0.10} & x_{1,0,0.15} & x_{1,0,0.20} & \cdots & x_{1,0,k} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{i,j,0.10} & x_{i,j,0.15} & x_{i,j,0.20} & \cdots & x_{i,j,k} \end{bmatrix} \tag{1}$$

3.2 | BS solver for IV

Instead of using the IV values from the original data set, we infer the Black–Scholes IV directly from the option market prices by inverting BS formula for IV.

$$\sigma(K, t) = BS^{-1}(C, S, K, t, r) \tag{2}$$

where C = option price, K = strike price, S = SPX index closing price, t = time to expiration in years, and r = interest rate.

TABLE 1 Descriptive statistics of implied volatilities

Expiration month	Type	Train	Count	Mean	*SD*	Min	25%	50%	75%	Max
3	ATM	Test	270.0	0.120142	0.014123	0.090652	0.109070	0.117838	0.130795	0.153857
		Train	13209.0	0.147456	0.056160	0.021373	0.110987	0.131838	0.168764	0.789342
	OTM calls	Test	720.0	0.114413	0.013444	0.087826	0.103942	0.112720	0.124057	0.149029
		Train	35224.0	0.143626	0.055073	0.047769	0.107530	0.127323	0.164976	0.780096
	OTM puts	Test	810.0	0.133558	0.012381	0.103937	0.124324	0.132371	0.141756	0.167763
		Train	39627.0	0.155812	0.056558	0.054969	0.118639	0.141029	0.177716	0.987745
6	ATM	Test	270.0	0.131340	0.008244	0.111260	0.125953	0.131466	0.136507	0.151160
		Train	13209.0	0.148592	0.049762	0.021474	0.115206	0.135462	0.169781	0.626790
	OTM calls	Test	720.0	0.128475	0.007702	0.111815	0.123275	0.128105	0.133247	0.149175
		Train	35224.0	0.144946	0.049716	0.056179	0.111412	0.131362	0.166066	0.890768
	OTM puts	Test	810.0	0.138411	0.006567	0.122188	0.134234	0.137738	0.142059	0.157953
		Train	39627.0	0.156273	0.049911	0.021474	0.122662	0.143082	0.177006	0.883940
9	ATM	Test	270.0	0.126796	0.016199	0.067115	0.119131	0.127817	0.133020	0.275768
		Train	13209.0	0.145946	0.054583	0.036720	0.109509	0.132034	0.167605	0.961242
	OTM calls	Test	720.0	0.122236	0.015108	0.076157	0.115359	0.121831	0.127992	0.299951
		Train	35224.0	0.141935	0.053778	0.047535	0.106202	0.127786	0.163251	0.912145
	OTM puts	Test	810.0	0.140134	0.017403	0.088751	0.131749	0.136936	0.145140	0.295610
		Train	39627.0	0.155250	0.054997	0.048689	0.118381	0.141550	0.176608	0.997274
12	ATM	Test	270.0	0.118084	0.019044	0.046761	0.106774	0.119153	0.130610	0.232869
		Train	13209.0	0.153985	0.064893	0.012742	0.120541	0.138560	0.172216	2.922243
	OTM calls	Test	720.0	0.113049	0.017461	0.062101	0.102109	0.112150	0.126260	0.187349
		Train	35224.0	0.150625	0.063634	0.012742	0.117611	0.135249	0.168916	2.871210
	OTM puts	Test	810.0	0.133669	0.021145	0.080961	0.121350	0.133366	0.143565	0.362683
		Train	39627.0	0.162029	0.064885	0.030445	0.127035	0.146377	0.180116	2.956078

Note: Descriptive statistics of Black–Scholes implied volatilities of the S&P 500 index options for the train and test set, grouped by the contract expiration month, option type and set type. Expiration month 3, 6, 9, and 12 represent March, June, September, and December, respectively. out-of-the-money (OTM) call buckets are $X > 0.001$, OTM puts are $X < -0.004$ and at-the-money (ATM) buckets are $X > -0.004$; $X < 0.001$, where $X = \ln(K/S)$ group defined in Section 3.3. Train set is for the period 2002-01-02 to 2019-08-16 and Test set is for the period 2019-08-19 to 2019-12-31. "*SD*," "min," "max," and "%" denote standard deviation, minimum, maximum, and percentile, respectively.

The required option prices are calculated by taking daily option closing mid-prices ((bid + ask)/2) for each month/strike combination. The volatility is extracted from options prices using a numerical method with the help of Python's pyvollib[3] library, based on LetsBeRational by Peter Jaeckel, to solve for the IV. For the case of option contracts, moneyness can be used to discretize the target IV, to help produce a more generic output. In particular, the information about the underlying, the strike, and the estimated BS-IV is known, so option's $\ln(K/S)$ (log-moneyness) can be used to generate the appropriate BS-IV bins.

3.3 | IV log-moneyness binning

A common problem with the financial time-series data is the stochastic trend and nonstationarity. In particular, option contracts are strike dependent and have a finite time-to-maturity. A number of techniques for parametric time series

[3]https://github.com/vollib/py_vollib.

models can be used to address these common problems. For example, one can integrate the series to address the trend components and apply data transformation techniques when addressing nonstationarity such as heteroscedasticity of the series. The supervised learning methodology used for the neural networks, however, often requires a scaling technique due to the fixed range transfer functions (such as sigmoid) between the layers, which makes the model incapable of accepting new contracts(new strikes) without being refit or retrained. Because of the expiring nature of the contracts encoding, a time-to-maturity of the options series is difficult as no universal convention exists for representing forward moneyness for machine learning models. Two common ways for machine learning models are: (1) Parameterizing moneyness functions such as Equation (2) or $M = (S, K, t, r, \sigma)$ and using parameters as an input to a neural network-based IV/option pricing solver like in Culkin (2017); (2) Standardized forward moneyness: $m = \frac{\ln(S/K)}{\sigma\sqrt{t}}$ and using this as a model feature. When passed to a machine learning model in either format, the important empirically observed properties of the volatility series, such as mean reversion and long-term persistence, are lost.

With a continuous volatility series we can generate a fixed range of buckets for each expiration month of the contract. For each contract month we address this by (1) binning this continuous variable into 20 evenly spaced log-moneys groups, 10 ranging from −0.05 to 0.0 for puts and 10 ranging from 0.0 to 0.05 for call side IV; (2) placing a derived BS-IV into each bin based on each contract's log-moneyness. The algorithm for discretizing the IV term-structure is described in Algorithm 1.

Algorithm 1 Algorithm for discretizing the IVS

1: For each option contract calculate log-moneyness $\ln(K/S)$.
2: Select all observations where log-moneyness $>\ -0.05$ and log-moneyness $<\ 0.05$.
3: Arrange buckets from −0.049 to 0.051 increasing by 0.005.
4: For each observation add a bucket label.
5: Pivot table = index $\longrightarrow$ date, expiration month; columns. $\longrightarrow$ bucket label; values $\longrightarrow \sigma_{BS}$.
6: Linear interpolation of IV skew.
7: Linear interpolation of IV term structure.
8: Select all observations where expiration month is in March, June, September, December.
9: **return** BS-IV (σ_{BS}) for bucket labels and expiration months for each day as Matrix 1.

3.4 | Stationarity and cointegration tests

The purpose of the statistical tests is to build a proper time series model for IV forecasting. The preliminary analysis includes the augmented Dickey–Fuller (ADF), Kwiatkowski et al. (1992) (KPSS), and Maddala and Wu (1999) tests for stationarity, and Johansen (1991) procedure for cointegration analysis to help in the selection of the appropriate lag terms for the AR models. Unit root tests are also needed to determine a degree of differentiation for the AR models.

The ADF tests the null hypothesis of the presence of the unit root against an alternative hypothesis of series being stationary. When the ADF test is applied to the original nondifferenced series, we fail to reject the null-hypothesis, implying that series are nonstationary $I(0)$ in most of the cases and need to be differenced. Nonstationarity is quite common for financial series. The original series are first differenced, and the ADF test is applied to the $I(1)$ series, we reject the null hypothesis for the majority of the series for 2002–2019 samples and conclude that the first differencing is sufficient to make the series stationary. In general, the mean-reverting aspect (no unit root) of the long term IV has been empirically observed due to the long-memory of the volatility series. The unit root cannot however be entirely rejected because, in the short term, the unit root is often observed, which is attributed to the sudden spikes in short term volatility because of the crash-like behavior of the underlying asset or other extreme events. In fact, the impact of heterogeneity of these volatility returns is addressed by Corsi (2008). The findings that nondifferenced IV series follow a nonstationary process are somewhat inconsistent with the earlier research. Although sampling methodology and sampling window or even the IV instrument can also be attributed to differences in ADF results, as previous researchers indicate, these differences are attributed to sampling period or the methodology of constructing the volatility series Gospodinov et al. (2006). As the result of these findings, we continue treating the volatility series as $I(1)$ process.

As a complement to the ADF test, we conduct the KPSS test to examine whether series are level stationery. The null hypothesis for the KPSS test is that the data is level stationary, and the alternative is that the data is not level stationary.

Similar to the ADF test, when applied to the original series, we reject the null hypothesis of stationarity. When they are first differenced, we reject the null hypothesis for the majority of the series.

Although it is confirmed that no unit root is present in the majority of first differenced series, in some cases, the unit root was still present. To test whether the unit root is present cross-sectionally across the panel, we utilize the Maddala–Wu test. Some advantages of the test are especially noteworthy in case of the IV panel data: (1) dimension of N (contract series) can be finite or infinite, (2) each group can have it's own set of stochastic or deterministic components, (3) unbalanced series may be present (4) unit root rule is not strictly enforced and would allow some groups to have unit root while others have no unit root. The null hypothesis matches the ADF null hypothesis, that unit root is present.

Following this test, we find no evidence of the presence of the unit-root and again confirm that that individual intercepts are stationary. Because the test statistic is a χ^2 distribution, we can conclude that although some series tend to exhibit unit-root, these series do not significantly impact the overall stationarity of the panel of differenced series.

Now that we have established that the panel exhibits stationarity when first differenced. We are also interested in exploring a presence of any spurious correlation between groups, that is, if the linear combination of these series is stationary or not. In other words, with the help of the Johansen (1991) procedure, one can observe the existence of any common long-term equilibria between the groups. This step is essential for the model selection, because if the series do not exhibit long-run relationships, then the usage of the long-term error correction models is not appropriate. We utilize the urca[4] package in R to construct a Johansen test. The null hypothesis for the test is $r = 0$ that no cointegration is present and alternative hypothesis is based on the number of cointegration equations. We find strong evidence to reject the null hypothesis of no cointegration relations, and conclude that rank of the matrix is greater than 19, which implies that a combination of all series (moneyness groups) is needed to make the series stationary, so there exists a large number of cointegration relations.

3.5 | Forecasting models

The primary goal is to propose a forecasting methodology that as observed from the statistical tests incorporates: (1) series autocorrelation behavior, (2) is able to nonparametrically estimate the cointegrating behavior of the panel, (3) accounts for long term error correction (or long memory of the series). The model evaluation process is based on $y_{t+1} \cdots y_{t+30}$ forecasting window results where the residual error $e_t = y_t - \hat{y}_t$ is minimized, and residuals of each model are diagnosed and checked for robustness by conducting several statistical tests and side-by-side comparisons of the predicted IVS.

3.5.1 | LSTM model

LSTM neural network (Hochreiter & Schmidhuber, 1997) is a type of RNN that is often used for sequence learning. The network contains recurrent loops that help model retain the past information and carry it forward throughout time. Unlike a simple MLP where the layers and neurons are fully connected to each consecutive layer, and the weight and biases matrices of each layer are propagated unidirectionally forward, the RNN extension reuses the same set of weights that are shared across time. Sharing weights is not only beneficial for retaining the long memory of the input sequence but also an optimization step that makes RNN's use less computational resources than MLP's. Given an input sequence: $x = (x_{t1}, x_{t2}, ..., x_{tn})$ the one layer hidden state at time t, $h_t = f(h_{t-1}, x_t)$ can be written as a function of the previous time step h_{t-1}, which serves as a memory. However, this introduces a problem where the gradients can vanish or explode over time. During the backpropagation, the network accumulates the error gradients recurrently, which could cause large updates to the weights, making a cell output an infinitely large or small gradient. Hochreiter and Schmidhuber (1997) addressed this problem in their paper by expanding an RNN cell to include the forget and output gates, as shown in Figure 3. Forget gate and output gates are the most critical component of the LSTM (Greff et al., 2017). When the sequence passes through an input gate, the hidden state from the previous timesteps and the current timestep pass through an activation function (hyperbolic tangent) and then a forget gate (sigmoid), which dictates

[4]https://cran.r-project.org/web/packages/urca/index.html.

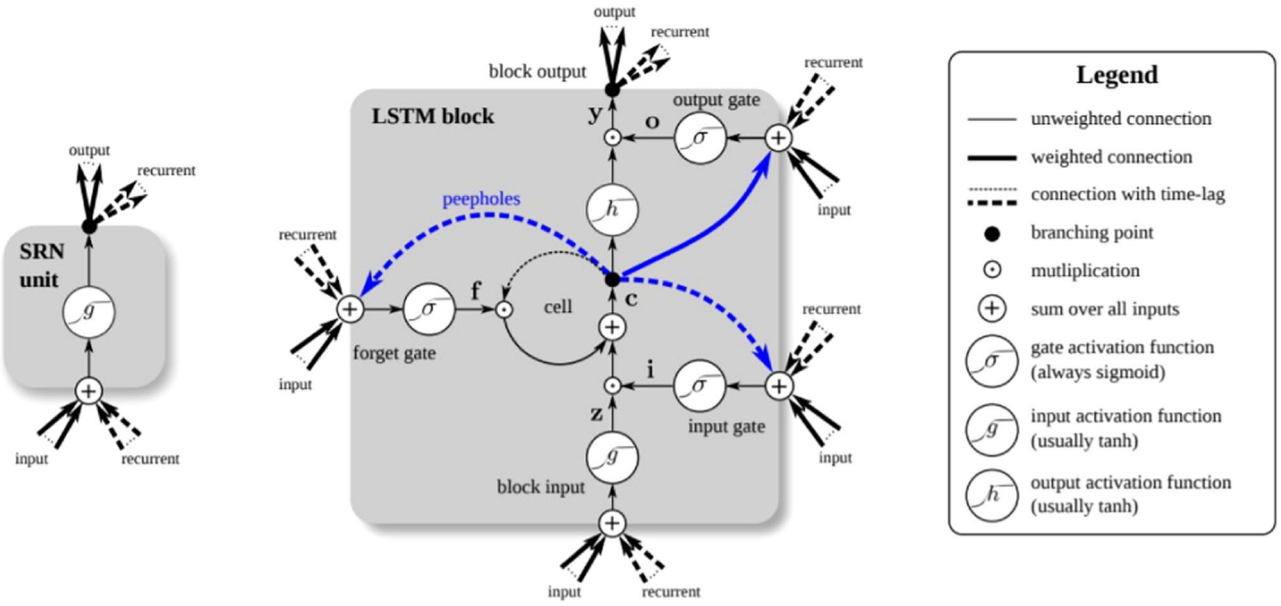

FIGURE 3 Simple recurrent neural network cell and long-short term memory (LSTM) memeory cell. This figure shows the schematic view of a simple recurrent neural network (left) cell and a long short term memory (right) cell by Greff et al. (2017)

whether the new hidden state should be updated and carried forward to the next time step. The simple RNN cell, on the other hand, contains no gates, so the information flow is purely controlled by the activation function.

The model is implemented in Google's deep learning framework Keras[5] and the flow for the models is generated in Tensorboard[6] and visualized in Figure 4. The input passes through a LSTM cell, and the hidden state of the cell at each of 30-time steps passes through a dropout layer, which randomly drops the connections for the output of the hidden state lstm_15 to reduce the possibility of over-fitting. A second LSTM layer is stacked to extract the time distributed hidden state from the first LSTM layer, and finally output a fixed 30-day $(-1, 30, 80)$ vector as an output of the fully connected dense_8 layer.

Regarding the selection of hyperparameters for LSTM, we follow the approach commonly practiced in the computer science literature and by industry practitioners.[7] We choose 128 cell units for the first LSTM layer and 64 cell units for the second LSTM layer. Between layers we allow for a 25% of dropout rate as recommended in Srivastava et al. (2014). We employ a batch size of 32 and a maximum of 1000 epochs to make a trade-off between training time and resource requirements. Finally we use the Adam optimizer as typical for deep learning networks (Kingma & Ba, 2015) and default the learning rate to 0.001.

3.5.2 | Convolutional long-short term memory model

The main disadvantage of the simple fully connected LSTM (FC-LSTM) for IVS forecasting is that the inputs are flattened before being passed to the hidden state, so the essential spatiotemporal relations of the IVS are lost. This is addressed by Shi et al. (2015), who proposed an extension to the FC-LSTM, which adds additional filter dimensions to the cells, hidden states and output gates. The added filters help identify the relationship between implied volatilities of nearby moneyness-maturity option categories as the filters traverse the IV surface. The proposed model determines the future state of the fixed-size cell in the grid by the inputs at current time-step, and the previous states of it's neighboring cells, as visualized in Figure 5.

[5]https://keras.io/.
[6]https://www.tensorflow.org/tensorboard.
[7]More details can be seen in Dr. Jason Brownlee's online tutorial. https://machinelearningmastery.com/how-to-develop-rnn-models-for-human-activity-recognition-time-series-classification/.

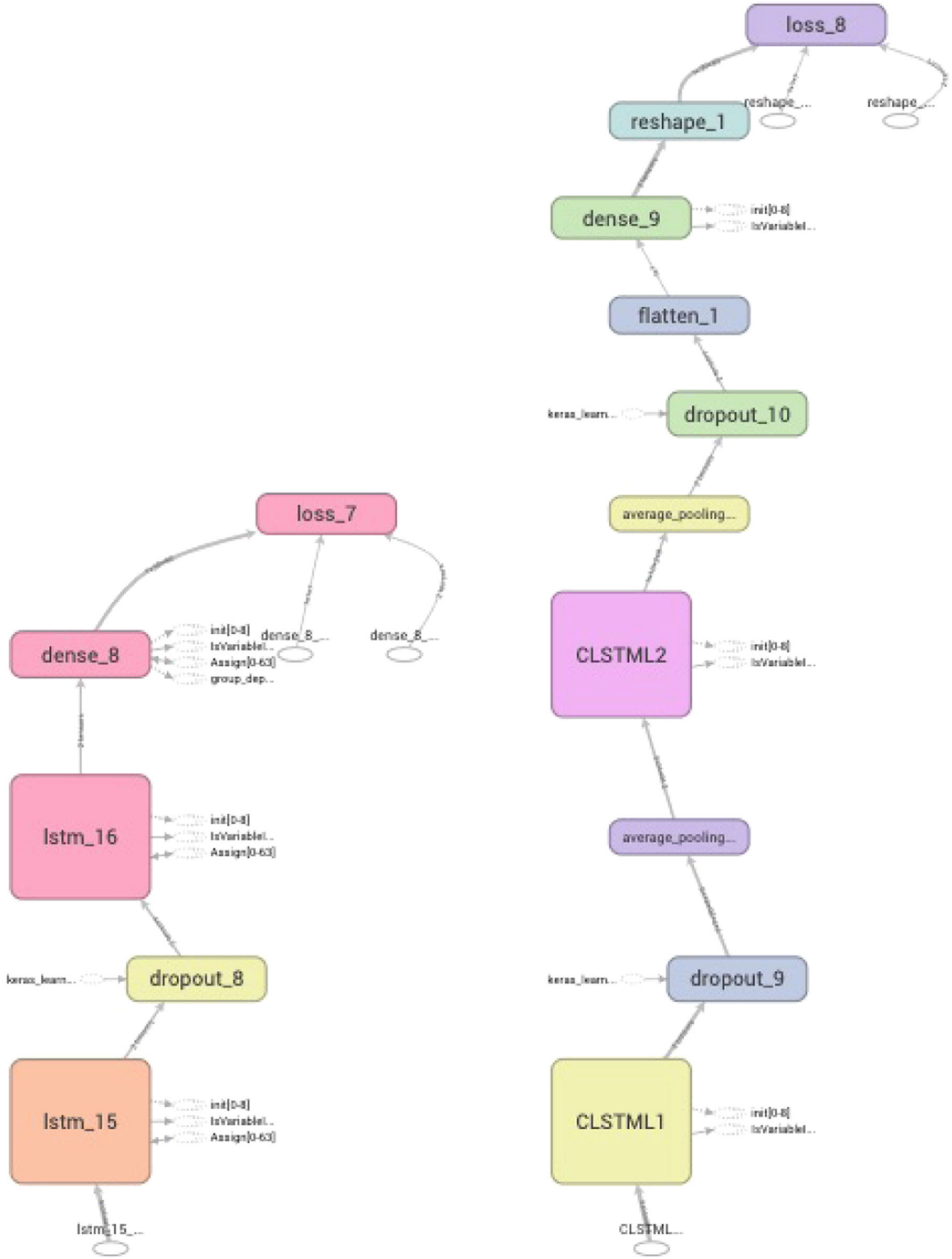

FIGURE 4 LSTM and ConvLSTM model flowcharts. This figure shows the inner working flows from inputs to outputs for the LSTM (left) and ConvLSTM (right) models. ConvLSTM, Convolutional long-short term memory; LSTM, long-short term memory

Zhou (2020) proves theoretically that a deep convolutional neural network is universal and can be used to approximate any continuous function to an arbitrary accuracy when the depth of the neural network is large enough. As shown by Shi et al. (2015), the ConvLSTM is shown to consistently outperform the fully connected LSTM on multiple spatiotemporal data sets, on various configurations. The researchers found that larger kernel size significantly helps the receptive fields in capturing the spatiotemporal correlations and spatiotemporal motion patters, with a smaller number of parameters required. The convolutional filter on its own adds a layer of nonlinearity, which is quite relevant for modeling the IVS. During the convolutional operation, the fixed-sized $N \times M$ grid, known as a filter (kernel), slides over the shape of the IVS and continuously adjusts its weights. Orosi (2012) describes that the majority of the practical methodologies for modeling and interpolating the IVS rely on nonlinear methods such as fitting the quadratic function, cubic spline, or a low-order polynomial, so the mechanics of the ConvLSTM seem to align well with the mechanics of the IVS.

The model is also implemented in Google's deep learning framework Keras, and the flow for the models is generated in Tensorboard and visualized in Figure 4. While a FC-LSTM accepts the 3D vector shaped as (batch, timesteps, features), the ConvLSTM accepts the 5D input vector of (batch, timesteps, rows, columns, and features), where rows and columns represent the grid as shown in Figure 5.

We also add a pooling layer to reduce the dimensionality of the input. An additional benefit of using a pooling layer is addressing the mean-reverting property of the IV. The average pooling layer of the ConvLSTM extracts average past states of the neighboring features, which helps maintain long term dynamics of the IV. Similar to the LSTM architecture, we stack a second ConvLSTM layer, to capture a more detailed representation of the first ConvLSTM layer. Following a second ConvLSTM layer, the outputs are flattened and fully connected to a simple, fully connected (dense) layer, which is shaped as a fixed 30 timestep vector of size (1,30,80), to produce a fixed 30-day forecast for all 20 log-moneyness groups and 4 months (March, June, September, and December).

Finally, we select the hyperparameters for ConvLSTM with the similar approach for LSTM. We also use Shi et al. (2015) as the guidance. Shi et al. (2015) conclude that there is no significant improvement for three-layer relative networks relative to two-layer networks. We choose 16 kernels and 4×4 as the kernel size for the first ConvLSTM layer, and eight kernels and 3×3 as the kernel size for the second ConvLSTM layer. Between layers we allow for a 25% of dropout rate. We employ the Adam optimizer with the default learning rate of 0.001 and 32 as the batch size.

3.5.3 | Vector autoregression (VAR) model

From the stationarity test, we found that series are stationary when first differenced $I(1)$, and due to the multivariate regression requirement we use VAR model:

$$\boldsymbol{y}_t = \text{Const.} + A_1 \boldsymbol{y}_{t-1} + \cdots + A_k \boldsymbol{y}_{t-p} + \boldsymbol{u}_t, \tag{3}$$

where $\boldsymbol{y}_t$ and $\boldsymbol{y}_{t-p}$ are a $K \times 1$ vector of implied volatilities at time t and lagged p periods, respectively, across all strikes and maturities, $A_1, ..., A_k$ are $K \times K$ coefficient matrices and u_t represents k-vector of error terms. The lag term for the model was selected based on the PACF and turned out to be significant only at lag 1. VAR model gives a good baseline model, but because significant cointegrating relationships $r > 0$ are evident between contract log-moneyness groups and expiration months, the error correction term (EC) can be added to the VAR model to account for the long-term relationships (stochastic trend) and still be able to capture the short-term dynamics, to improve the forecast.

3.5.4 | Vector error correction model (VECM)

As observed in Johansen procedure, the volatility series show long-run cointegrating relationship, the VECM is used to model both the large swings in the short-term and the long-term cointegrating relation. So Equation (3) can be rewritten to allow for cointegration:

$$\Delta \boldsymbol{y}_t = Const. + \Pi \boldsymbol{y}_{t-1} + \sum_{i=1}^{p-1} \Gamma_i \Delta \boldsymbol{y}_{t-1} + u_t \tag{4}$$

where Π and Γ_i represent the long-run relationship and the short run relationship, respectively.

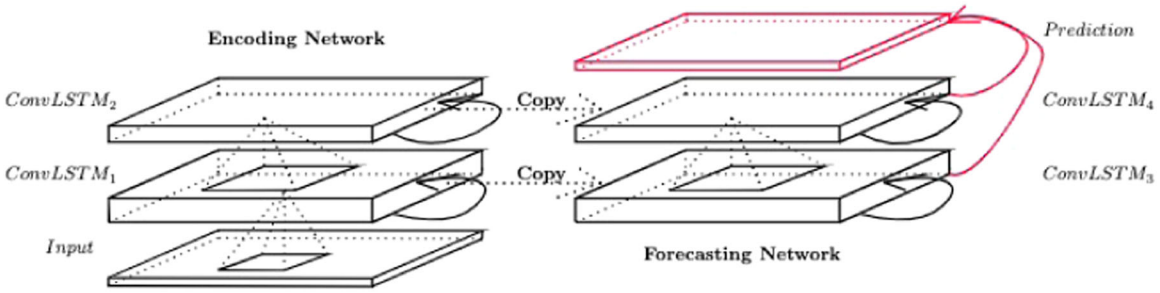

FIGURE 5 Forecasting ConvLSTM. The figure shows how the convolutional neural network LSTM model predicts by combining a encoding network and a forecasting network as presented in Shi et al. (2015). ConvLSTM, Convolutional long-short term memory; LSTM, long-short term memory

3.6 | Algorithms and optimizations

We follow a traditional time series forecasting and backtesting methodology using a train-test split that preserves temporal ordering. This is done for the time-series and the neural networks to have similar and comparable results. Algorithm 2 is modified to interface with R to produce the VAR and VEC models using tsDyn[8] package in R.

Algorithm 2 VAR and VEC training and validation algorithm

train ← Select all from prepared data set where day < 2019-08-16
test ← Select all from prepared data set where day > 2019-08-16
If VAR(1) **then**
  Δ train = $train_t$ - $train_{t-1}$
  model fit X_{t-1}
  model predict X_{t+90}
  predict inverse = Δ train += model predict
  return MAPE, RMSE ← test - predict inverse
end if
if VEC(1,1) **then**
  model fit X_{t-1}
  model predict X_{t+90}
  return MAPE, RMSE ← test - predict
end if

In addition to the algorithm implementation with the deep learning framework Keras, a number of modifications to the LSTM and ConvLSTM Algorithms 3 were done: The data was transformed into a supervised learning problem, where the data is reorganized into two groups: an input group: $X = X_{t-1}, X_{t-2}, ..., X_{t-30}$ and the target group: $Y = X_t, X_{t+1}, ..., X_{t+30}$, to match the fixed 30 day prediction window of the VAR and VEC models. The implemented design utilizes a sliding window approach to help maintain a universal model that can pick up from anywhere in the sequence yet still produce meaningful forecasts. By default, the LSTM in Keras is stateless and treats each batch sequence independently from the previous batch, and resets its internal memory state for each batch; this is done so that the back-propagation algorithm could run between batches and update the weights. Because it is important to carry forward volatility relationships for the lengthy periods of time as discussed in Section 3.4, each batch contains a fixed 30-day history window and produced a fixed output window, which classifies both LSTM and ConvLSTM models, as many-to-many models. This allows us to carry forward the weights from the past time-windows effectively, learning a long-term dependency of the volatility

[8] https://cran.r-project.org/web/packages/tsDyn/index.html.

series. We begin by first scaling the features into a range of −1 to 1 due to the hyperbolic tangent activation function in the LSTM and ConvLSTM hidden layers, before training.

Algorithm 3: LSTM and ConvLSTM training and validation algorithm.

```
train ⟵ Select all from prepared dataset where day < 2019-08-16
test ⟵ Select all from prepared dataset where day > 2019-08-16
Function SUPERVISED(data, history days, prediction days):
    x = data.shift_t − history days; y = data.shift_t + prediction days
    return [x, y]
Function SCALE(data):
    return data.scale(-1,1)
Function INVERSE SCALE(data):
    return prediction inverse scale
if LSTM then
    model ⟵ reshape(n, 30, 80) ⟵ SCALE(.) ⟵ SUPERVISED(train, 30, 30)
    model train for 1000 epochs
end if
if ConvLSTM then
    model ⟵ reshape(n, 30, 20 , 4, 1) ⟵ SCALE(.) ⟵ SUPERVISED(train, 30 , 30)
    model train for 1000 epochs
end if
for i in range 3 do
    model predict X_{t+30}
    INVERSE SCALE(prediction)
end for
return MAPE, RMSE ⟵ test - prediction
```

The model is configured to save weights at each epoch and finally retain the best weights for the models based on the lowest in-sample forecasting mean-squared-error (MSE). For both models, we set the training period for 1000 epochs and add an early stopping callback to stop the training once no improvements in the MSE were made for 50 epochs. The LSTM was trained for 1000 epochs, which took approximately 4 h and 30 min on Google's Research Colab[9] GPU environment.

The ConvLSTM was trained for 361 epochs, triggering an early stop after approximately 2 h. The training process is visualized in Figure 6. A similar reduction in MSE for both LSTM and ConvLSTM is observed; however, the LSTM has a larger training variance when compared to the ConvLSTM. This is an inherent problem when constructing data-driven supervised learning models, often referred to as bias-variance trade-off. High bias in the context of neural network-based supervised learning architectures would mean that the model is underfitting the data, so adding more data, or hyperparameter optimization such as increasing the number of neurons, network depth or learning rate, might assist in capturing more complex feature representation. High variance, on the other hand, means that the model is overfitting the data and captures random noise. For the purpose of constructing the IVS, both low variance and low bias are important, but because of this trade-off, one must tune the hyperparameters to achieve the optimal balance between bias and variance. The LSTM produces overall lower training error, however experiences higher variance. ConvLSTM has a slightly higher (MSE) training error but overall lower variance.

Both models output a 30-day IVS prediction vector. For the LSTM and ConvLSTM, the transformation is a three-step process. Original training values are first scaled to a range of −1 to 1 due to the hyperbolic transfer function in the hidden layer. Appropriate scaling has been shown to significantly improve the out-of-sample neural network performance for IV forecasting Liu et al. (2019); output of the recurrent hidden layer is flattened and connected into a 1D fully connected output layer shaped $1 \times 20 \times 4 \times 30$; an output vector is inverse scaled to the original BS-IV volatility level, as discussed in Section 3.6. The output vector is a 30-day prediction window for the March, June, September, and December month contracts for each of the 20 log-moneyness categories, 10 for the OTM puts, and 10 for the OTM calls.

[9] https://colab.research.google.com/.

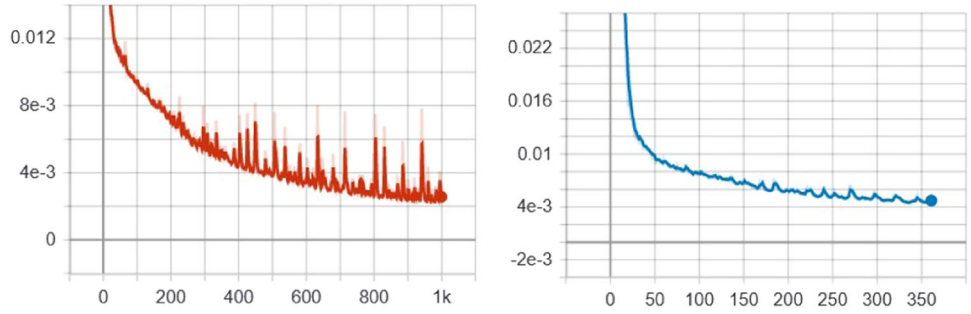

FIGURE 6 LSTM versus ConvLSTM training process. The comparison of loss function values in terms of mean squared error in implied volatility for LSTM (red) in the left panel and ConvLSTM (blue) in the right panel. ConvLSTM, Convolutional long-short term memory; LSTM, long-short term memory

4 | RESULTS AND DISCUSSION

In this section, we discuss the results of the machine learning models and finally compare the forecasting accuracy of these models to the benchmark VAR and VEC model. The comparison is made out of two subtopics to answer two questions: (1) Which of the models performs the best for the 1-, 30-, and 90-day windowed forecast. To support the findings, we conduct the DM test (Diebold & Mariano, 1995) for the equal predictive accuracy of the models; (2) Which of the models can correctly model and predict the IVS dynamics. The four models are compared side-by-side, and the evaluation is based on the 90 trading-day hold out period from 2019-08-16 to 2019-12-31 over a 1-, 30-, and 90-day prediction windows. In this section, we present findings that convolution operation in the LSTM memory cell, significantly improves modeling and forecasting future IV and outperforms traditional time-series approaches for long-horizon predictions.

4.1 | Model evaluation criteria

In Section 3.1, we discussed the interpolation technique, to fill the missing values after pivoting the option chain. Before scoring the models, the predictions are inverted back to the original BS-IV volatility level, and scoring is based on the original BS-IV. For the VAR Model, the data was initially first differenced to achieve stationarity, so the predicted observations are added back to the nondifferenced values, to recreate the original continuous volatility series. For the VEC model, the output is an IV volatility of an appropriate scale, so the inverse difference is not necessary.[10] To fairly compare the models, all 4 models are trained on the same training set of 4373 days. One of the differences in the supervised, sliding window approach for the LSTM and ConvLSTM, when compared to the VAR and VEC models, is that the forecast is a fixed size 30-day vector. The LSTM and ConvLSTM are not retrained for the 90 trading-day out-of-sample predictions. Instead, the 30-day historical window is passed as an input to the model, to get the next fixed 30-day forecast window. This is done because the LSTM learns to generalize the sequences and can predict sequences based on the weights learned from the long series. To make the comparison between the neural networks and time-series models fair, the VAR and VEC models are fit on a train set, and the same set of coefficients is reused for predicting 1-, 30-, and 90-day windows. As the evaluation criteria we choose root mean squared error (RMSE) in the actual BS-IV level:

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\sigma_{i,\text{actual}} - \sigma_{i,\text{forecast}})^2}$$

[10]Differencing in the VEC model is done internally thought an implementation in the tsDyn library in R, through the integration and cointegration factor parameter.

and mean absolute percentage error (MAPE) relative to the actual BS-IV level:

$$\text{MAPE} = \frac{100\%}{N}\sum_{i=1}^{N}\left|\frac{\sigma_{i,\text{actual}} - \sigma_{i,\text{forecast}}}{\sigma_{i,\text{actual}}}\right|$$

where i and N denote the ith observation and the total number of observations, respectively, in the training or test set.

We compare the performance of the models based on the RMSE, MAPE for the entire prediction vector. These loss metrics were chosen to be consistent with previous machine learning research in the area of IV modeling and forecasting, which tends to use a combination of MSE, RMSE, or MAPE (see Culkin, 2017; Gospodinov et al., 2006; Liu et al., 2019; Luong & Dokuchaev, 2018; and Xiong et al., 2016).

4.2 | RESULTS

We summarize the mean in-sample, out-of-sample, and out-of-sample errors with the Savitzky–Golay filter applied in Table 2. Significantly lower in-sample RMSE and MAPE for ConvLSTM and LSTM are observed than their time series counterparts, VAR and VEC. The best out-of-sample fit is achieved by ConvLSTM whereas the best in-sample fit is achieved by LSTM model. The in-sample performance could be attributed to an ability of the neural networks to accurately approximate the BS-IV (Culkin, 2017; Liu et al., 2019). Liu et al. (2019) used neural network for estimating three IV equations (BS, Heston, Brent's) and found that neural network is highly efficient at approximating option prices and IVs, and because of the ability to "batch" process the contracts, it can lead to significant performance gains for on-line predictions. Culkin (2017) found that ANN's can learn to price options with the BS equation and produce very low error rates. The ConvLSTM model produces consistent in-sample and out-of-sample errors, signifying model robustness. However, the discrepancy between the in-sample and out-of-sample performance for LSTM is evident, potentially due to model overfitting.

The out of sample forecasts are divided into multiple time horizons, which are summarized in Table 3. The horizon represents a daily mean IV across all log-moneyness and maturity groups. It is observed that ConvLSTM has the lowest RMSE and MAPE for all prediction horizons among all models considered. VAR and VEC are nearly identical in their performance, both outperforming LSTM. All models tend to produce larger errors for longer prediction horizons. However, what makes ConvLSTM stand out is its slow deterioration of predictive performance. This behavior confirms observations of Shi et al. (2015) that were described for a number of spatiotemporal datasets, that (1) the ConvLSTM is better at handling spatiotemporal relationships than a regular fully connected LSTM and (2) larger kernel sizes allow for the capture of the spatiotemporal motion patterns. By using an additional average pooling layer between two ConvLSTM layers, it also allows us to capture the mean-reverting nature of the IV over a long-term horizon allows and show that historical spatial relationships between the maturities play an important role in the forecasting of the IVS.

Average predictions against true values are visualized in Figure 7. It is observed that all four models seemingly predict the average IV pattern but overestimate the IV jumps and magnitude of these jumps in the longer-term.

TABLE 2 Average in-sample and out-of-sample performances

	In-sample		Out-of-sample		OOS (smoothed)	
Model	**RMSE**	**MAPE**	**RMSE**	**MAPE**	**RMSE**	**MAPE**
VAR	0.022	5.13	0.021	13.72	0.021	13.58
VEC	0.022	5.13	0.021	13.75	0.020	13.50
LSTM	0.014	3.56	0.031	18.74	0.032	18.99
ConvLSTM	0.019	3.88	0.014	8.26	0.015	8.68

Note: This table shows average in-sample, out-of-sample, and out-of-sample (OOS) smoothed errors for the S&P 500 index implied volatility surface. The in-sample period spans from 2002-01-02 to 2019-08-16 while the out-of-sample period spans from 2019-08-19 to 2019-12-31. VAR, VEC, LSTM, ConvLSTM denote the vector autoregression, vector error correction, long short term memory and convolutional neural network long short term memory models, respectively. RMSE and MAPE denote root mean squared error and mean absolute percentage error, respectively. The OOS (smoothed) applies the Savitzky–Golay filter to smooth the prediction performance on the out-of-sample set.

TABLE 3 Average out-of-sample forecast errors

Model	Horizon	RMSE	MAPE
VAR	$h = 1$	0.011	6.68
	$h = 30$	0.016	8.60
	$h = 60$	0.018	10.91
	$h = 90$	0.021	13.50
VEC	$h = 1$	0.011	6.77
	$h = 30$	0.016	8.64
	$h = 60$	0.018	10.92
	$h = 90$	0.021	13.58
LSTM	$h = 1$	0.017	11.71
	$h = 30$	0.019	10.38
	$h = 60$	0.036	19.90
	$h = 90$	0.032	18.88
ConvLSTM	$h = 1$	0.008	4.61
	$h = 30$	0.015	7.61
	$h = 60$	0.014	7.94
	$h = 90$	0.015	8.30

Note: The average out-of-sample forecast errors for the S&P 500 index implied volatility surface across all moneyness and maturity categories over 1-, 30-, 60-, and 90-day horizons are shown. The out-of-sample period spans from 2019-08-19 to 2019-12-31. VAR, VEC, LSTM, ConvLSTM denote the vector autoregression, vector error correction, long short term memory and convolutional neural network long short term memory models, respectively. RMSE and MAPE denote root mean squared error and mean absolute percentage error, respectively.

Time-series models converge to a mean value following 10 timesteps, and the output of the VAR and VEC model is the average IVS of previous timesteps. The VEC model is nearly identical to the VAR model in terms of out-of-sample prediction, indicating a weak cointegrating relationship between moneyness groups and maturities. The VAR and VEC models could hardly generate significant variations in the last quarter of the data. The LSTM model persistently overestimates the true values. The ConvLSTM model produces the closest prediction to the actual IV, which implies that meaningful information is extracted during convolution operation over the volatility skew and the term structure, which is subsequently captured in the average pooling layer. A further look into the end of the training set in August 2019 shows that implied volatilities spiked up from low teens to high teens and sustained at the high level for the month. All four models show varying degrees of short term memory, especially the two time series models, when applied to predict out of sample. The ConvLSTM model on the other hand shows lesser degree of short term memory.

To mitigate the effect of the forecast instability, we apply the Savitzky-Golay smoothing filter based on the 4th degree polynomial for each of the predicted 30-day windows and plot it against the true IV in Figure 8. Similar to that of the unsmoothed forecast, VAR, VEC, and LSTM models overestimate the IV level, but seemingly capture the future dynamics of IVS. For LSTM and ConvLSTM smoothing the forecast with the Savitzky-Golay filter helped reduce some of the output noise, that was observed in the raw forecast. However, applying the Savitzky–Golay smoothing filter does not yield reduction in RMSE and MAPE as seen in Table 2.

We construct a mean predicted IVS for the 90-day holdout period and compare it to the mean actual IVS for the same period to investigate whether the future general dynamics of IVS are captured by the VAR, VECM, LSTM and ConvLSTM models. Figure 9e shows mean actual IVS is based on the same time-frame of 2019-08-19 to 2019-12-31 of the holdout window. The near term contracts observe higher IV levels than longer-term contracts. In addition, the volatility skew is also present in Figure 9e. As discussed in Section 3.3, the IVS is constructed by arranging 20 log-moneyness groups, where values < 0, represent OTM puts, and values > 0 represent OTM calls. So to properly capture the dynamics based on the holdout window, the models need to show overall presence of the IV term structure, which in the case of the holdout window is expressed through a higher IV for the September (Month 9) and December (Month 12) contracts. As a second criterion, the models also need to capture the skew properly. Figure 9 shows the mean

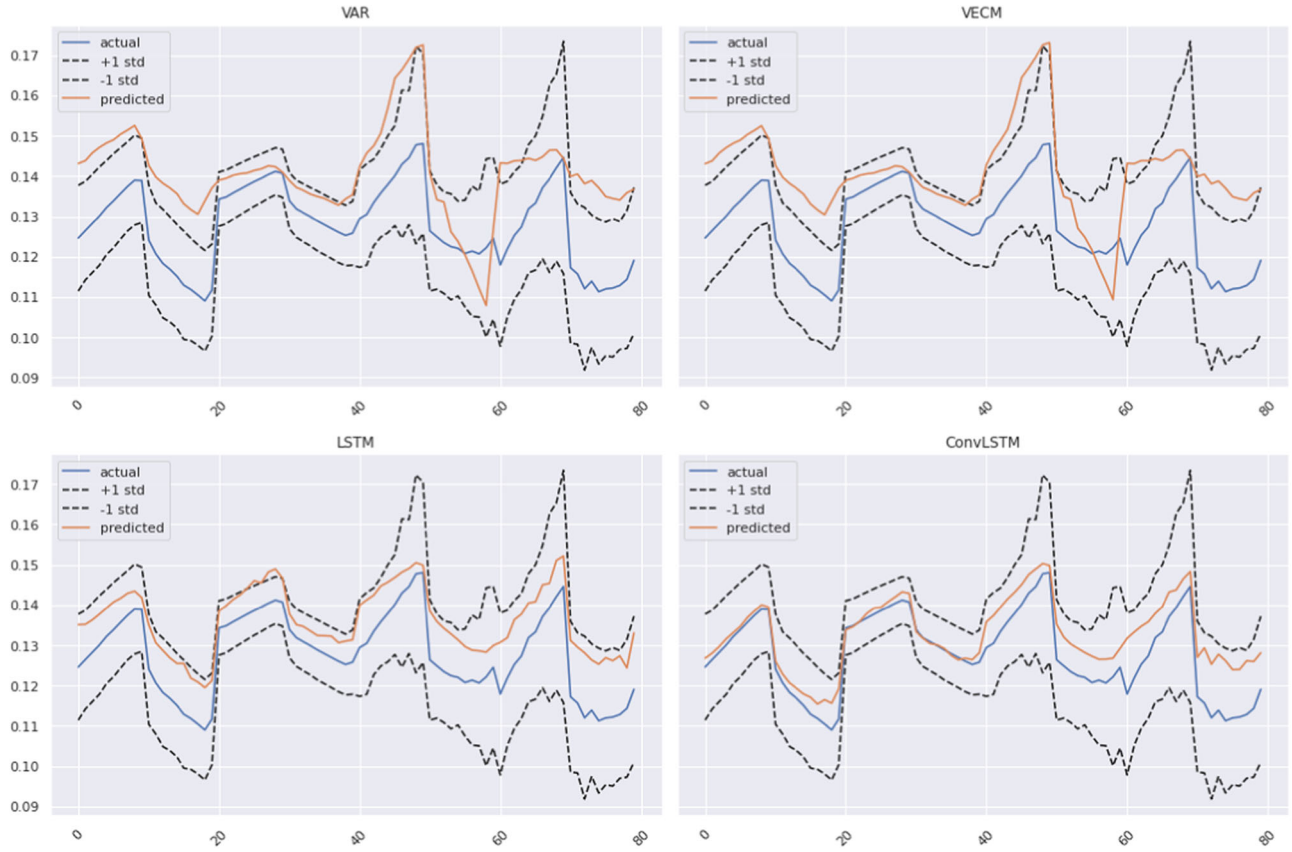

FIGURE 7 Average out-of-sample actual versus predicted IV. The average predicted versus actual implied volatilities (solid lines) during the out-of-sample period are shown. The dashed lines represent the actual implied volatilities plus/minus one standard deviation. The out-of-sample period spans from 2019-08-19 to 2019-12-31

predictions for the entire holdout period for each model. In general, all four models were able to capture the presence of the skew and the term structure component of the IV series.

Table 4 reports RMSE and MAPE for the out-of-sample predictions for each contract month. ConvLSTM produces the lowest overall RMSE and MAPE for all expiration months and option types. The near-term (December and September) contracts observe higher RMSE and MAPE cross-sectionally for all-models, which hints at the difficulty of forecasting the near-term IV. ConvLSTM produces the lowest MSE and RMSE for long-term (March and June) contracts. Another thing to note is that LSTM and ConvLSTM seemingly performs better at capturing the IV skew, since both RMSE and MAPE remain relatively constant between option buckets for all expiration months when compared to VAR and VEC models.

We find that better performance is generally achieved for the March and June contracts than for the September and December contracts, with the June contract being the clear winner. Across moneyness categories, volatilities of near the money, especially slightly OTM put options, are predicted with the lowest errors. Additionally, all models produce lower errors for put options than those for call options. Among the four models, the ConvLSTM model consistently produces low errors across all moneyness groups and expiration months. It is also the only model that can produce a predicted value both higher and lower than the true value across moneyness groups, indicating its flexibility to match the IV skew.

Finally, we employ the Diebold–Mariano (DM) test for the accuracy of the prediction over 1-day forecasting horizon between alternative models, namely AR, VEC, LSTM, and ConvLSTM. We first average the 1-day forecast errors across all moneyness groups and expirations to create a time series of errors. We obtain the DM test statistics and the associated p-values for any two pairs. Note that the test statistics follow a modified DM test proposed by Harvey et al. (1997). We report the pairwise comparison in Table 5, with the model in the row against the model in the column. The null hypothesis for the one-sided test is that two models have the same predictive accuracy, and an alternative hypothesis is that the prediction of the model in the row is significantly more accurate than the predictions of

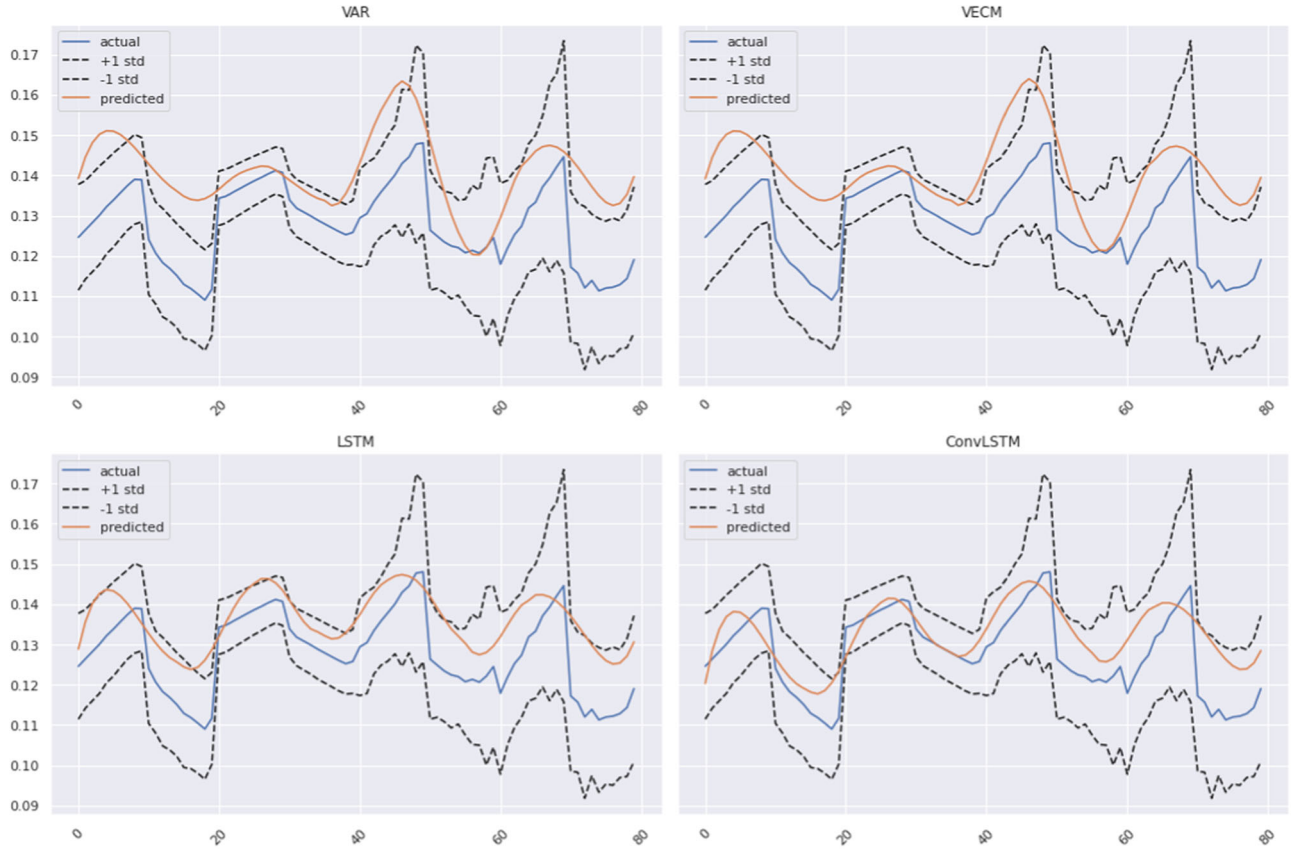

FIGURE 8 Average out-of-sample actual versus predicted Savitzky-Golay smoothed IV. The average predicted versus actual implied volatilities (solid lines) during the out-of-sample period are shown. The dashed lines represent the actual implied volatilities plus/minus one standard deviation. The out-of-sample period spans from 2019-08-19 to 2019-12-31. The predicted values are smoothed using the Savitzky–Golay filter

the model in the column over a 1-day forecast horizon.[11] A positive (negative) DM statistic supports that the model in the row is preferred (inferior) to the model in the column. The first row compares VAR (the row model) to VEC (the first column), LSTM (the second column) and ConvLSTM (the third column), respectively. We find that VAR outperforms VEC and LSTM, but underperforms ConvLSTM. The second row compares VEC to LSTM and ConvLSTM, leading to the conclusion that VEC outperforms LSTM and underperforms ConvLSTM. The third row compares LSTM to ConvLSTM, indicating that LSTM underperforms ConvLSTM. Overall, we confirm that ConvLSTM significantly outperforms the other three models at a significance level of less than 0.001. The performance ranking of the rest models is VAR, VEC, and LSTM, which is consistent with the findings summarized in Table 3.

4.3 | Economic analysis

Based on in-sample fit and out of sample prediction accuracy, the proposed deep learning model, namely ConvLSTM, outperforms the other three alternatives—LSTM, VAR, and VEC models. Option market makers can take advantage of more accurate prediction of IV surface by adjusting their bids and offers accordingly and hedging with better volatility inputs. The practical implication remains a question for a trader: whether the superior statistical performance can translate into economic gains?[12] In the following, we focus on analyzing the profitability of a butterfly strategy that takes advantage of an accurate volatility prediction. Specifically, we examine the profitability of a simple volatility

[11]The DM test will not be valid for 30- and 90-day forecasting horizons due to limited samples.
[12]We thank an anonymous reviewer for suggesting to conduct an economic analysis based on model predictions.

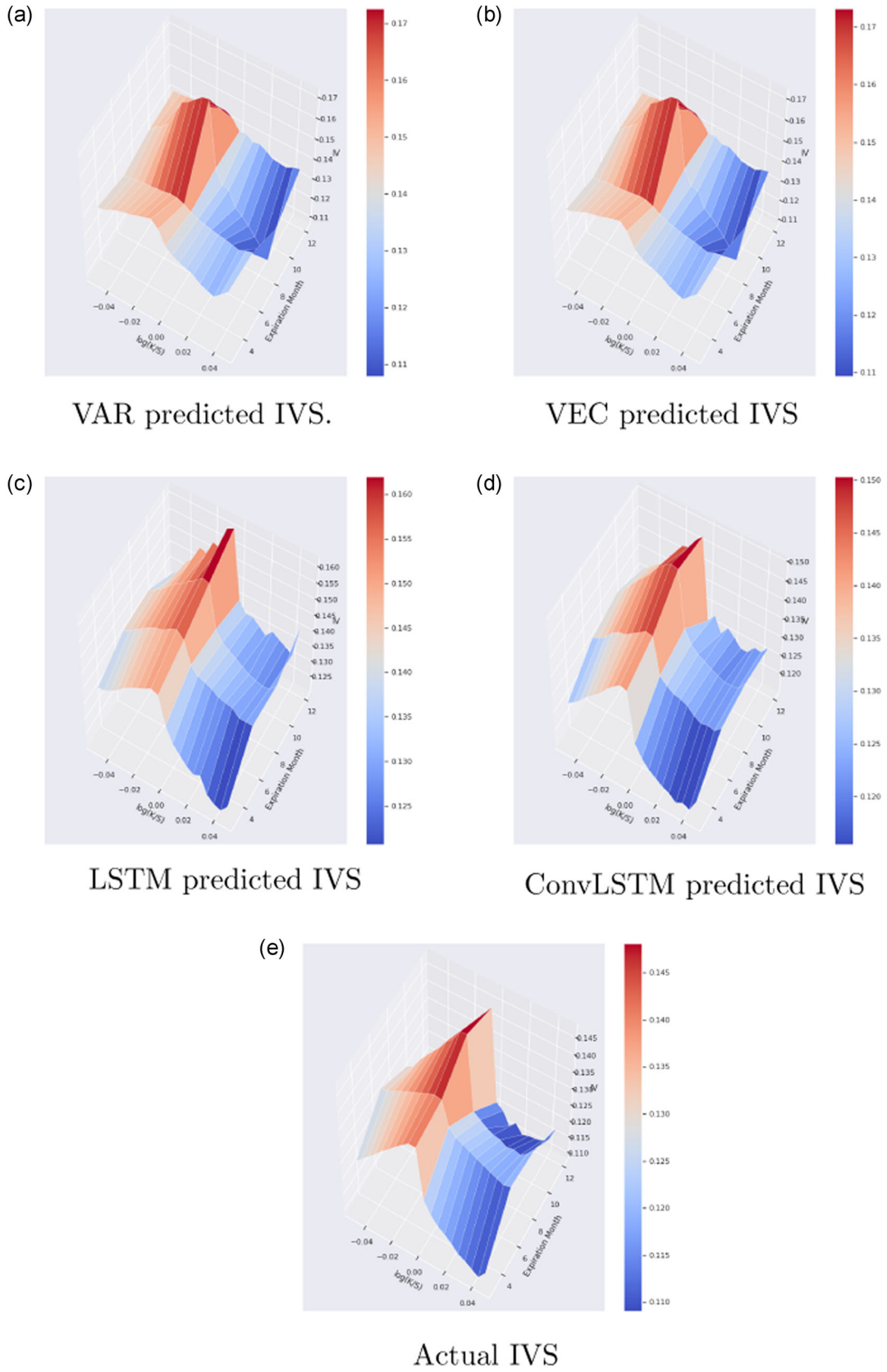

FIGURE 9 Average actual and predicted IVS for the out-of-sample period. The actual versus model-predicted implied volatility surface averaged over the out-of-sample period spanning from 2019-08-16 to 2019-12-31. ConvLSTM, convolutional neural network long short term memory models; LSTM, long short term memory; VAR, vector autoregression; VEC, vector error correction

TABLE 4 Mean out-of-sample RMSE and MAPE by expiration month and moneyness groups

		RMSE				MAPE			
Expiration Month	**Type**	**ConvLSTM**	**LSTM**	**VAR**	**VEC**	**ConvLSTM**	**LSTM**	**VAR**	**VEC**
3	ATM	0.010	0.031	0.025	0.025	7.292	19.343	19.345	19.299
	OTM calls	0.010	0.030	0.024	0.024	7.205	19.789	20.024	19.957
	OTM puts	0.009	0.031	0.019	0.019	5.542	16.905	13.117	13.083
6	ATM	0.006	0.022	0.010	0.010	3.806	12.855	6.502	6.493
	OTM calls	0.006	0.022	0.010	0.010	3.504	13.434	6.687	6.679
	OTM puts	0.006	0.023	0.007	0.007	3.461	12.423	4.155	4.152
9	ATM	0.018	0.032	0.020	0.020	9.717	18.624	11.659	12.084
	OTM calls	0.017	0.028	0.017	0.017	9.334	16.631	8.803	8.752
	OTM puts	0.016	0.036	0.027	0.027	7.477	17.554	17.050	17.453
12	ATM	0.021	0.041	0.029	0.029	16.045	29.083	23.027	22.973
	OTM calls	0.021	0.039	0.029	0.029	17.016	28.928	24.703	24.652
	OTM puts	0.020	0.040	0.024	0.024	11.721	23.041	13.923	13.900

Note: The out-of-sample errors for the S&P 500 index implied volatility surface are shown. The out-of-sample period spans from 2019-08-19 to 2019-12-31. Expiration month 3, 6, 9, and 12 represent March, June, September, and December respectively. VAR, VEC, LSTM, and ConvLSTM denote the vector autoregression, vector error correction, long short term memory and convolutional neural network long short term memory models, respectively. RMSE and MAPE denote root mean squared error and mean absolute percentage error, respectively. OTM call buckets are $X > 0.001$, OTM puts are $X < -0.004$ and ATM buckets are $X > -0.004$; $X < 0.001$, where $X = \ln(K/S)$ group.

TABLE 5 DM test for model comparison

Model	**VEC**	**LSTM**	**ConvLSTM**
VAR	DM = 13.29 (<0.001)	DM = 3.95 (<0.001)	DM = −8.19 (<0.001)
VEC	–	DM = 3.92 (<0.001)	DM = −8.29 (<0.001)
LSTM	–	–	DM = −4.97 (<0.001)

Note: The Diebold–Mariano test statistics and *p*-values (in parenthesis) for pairwise comparisons among VAR, VEC, LSTM, and ConvLSTM modes in terms of 1-day prediction accuracy for the S&P 500 index implied volatilities during the out-of-sample period. The DM test is one-sided with the null (alternative) hypothesis that the model in the row has the same accuracy as (greater accuracy than) the model in the column. A positive (negative) DM statistic supports that the row model is preferred (inferior) to the column model. The out-of-sample period spans from 2019-08-19 to 2019-12-31. VAR, VEC, LSTM, and ConvLSTM denote the vector autoregression, vector error correction, long short term memory, and convolutional neural network long short term memory models, respectively.

strategy that short (long) the butterfly spread if the predicted IV in 5, 10, and 15 days is greater (less) than current IV. A short butterfly spread as a long volatility strategy generally benefits from rising volatility.[13]

We report the strategy's return, standard deviation and Sharpe ratio for the 90-day and 360-day sample periods in Table 6. The statistics are based on 5-, 10-, and 15-day holding periods. We find that time series models produce positive returns for all three holding periods during the 90-day sample period while deep learning models did not. The better results by VAR and VEC models are mainly due to their persistently long volatility strategy because both models consistently overestimate implied volatilities as seen in Figure 7. On the other hand, LSTM and ConvLSTM models recommend more balanced strategies in long versus short volatility in this sample. Several episodes of volatility spikes at the end of 2019 are a favorable condition for the long volatility strategy. In view of the potential bias due to the short sample period, we re-estimate all models with the last 360 days as the holdout period. We find that only ConvLSTM

[13]We note this simple strategy is not a pure volatility strategy because the strategy's exposures to option greeks other than vega might impact its profitability over the holding period, especially as the underlying index moves further away from the initial price level.

TABLE 6 Return statistics for butterfly strategy

	5-Day			10-Day			15-Day		
	Return	**SD**	**SR**	**Return**	**SD**	**SR**	**Return**	**SD**	**SR**
90-day sample									
ConvLSTM	−56.92	51.90	−1.10	−56.66	31.77	−1.78	−4.73	36.29	−0.13
LSTM	−146.34	44.10	−3.32	−70.81	32.75	−2.16	−52.26	35.77	−1.46
VAR	99.59	46.73	2.13	87.17	33.59	2.60	64.47	35.86	1.80
VEC	98.71	46.82	2.11	85.55	33.75	2.53	64.79	35.79	1.81
360-day sample									
ConvLSTM	14.60	46.95	0.31	15.27	36.90	0.41	−14.91	27.86	−0.54
LSTM	−81.41	46.95	−1.76	−45.60	35.31	−1.29	−83.70	28.06	−2.98
VAR	−34.49	47.56	−0.73	−49.88	34.36	−1.45	−65.30	26.95	−2.42
VEC	−34.82	47.55	−0.73	−49.79	34.33	−1.45	−65.30	27.04	−2.42

Note: The annualized return and standard deviation (denoted "sD") in percentage and Sharpe ratio (denoted "SR") for the butterfly strategy that is long (short) volatility when future volatility is predicted to increase (decrease) during the sample period. 5-, 10-, and 15-day denote the respective holding period of the strategy. The 90- and 360-day sample periods start from 2019-08-19 to 2018-7-18, respectively, and end on 2019-12-31. VAR, VEC, LSTM, and ConvLSTM denote the vector autoregression, vector error correction, long short term memory, and convolutional neural network long short term memory models, respectively.

generates positive returns for 5-day and 10-day holding periods during the 360-day sample period. The corresponding Sharpe ratios are 0.31 and 0.41, which are comparable to the historical average of the underlying SPX. Neither VAR nor VEC could produce positive returns during the longer sample period, likely due to their inability to adapt to more changing market conditions. Overall we conclude that the convolution neural network-based deep learning model might produce economic profits over a long horizon, but does not necessarily have an advantage over traditional time series models for shorter time periods despite their statistical edge in predictability.

5 | CONCLUSION

In this paper, we apply and analyze two new deep neural network architectures, namely LSTM and ConvLSTM, for forecasting time-varying multistep evolution of the IVS and benchmark them against traditional time-series VAR and VEC models. The data set contains SPX call and put options traded between 2002-01-02 and 2019-12-31. We evaluate the out-of-sample forecasting performance on a 90 trading day holdout data set from 2019-08-19 to 2019-12-31 using RMSE and MAPE. We conduct the DM test for statistical significance of the forecasts, between time series and machine learning models for multiple time horizons. The forecasts are compared to the actual IV, and machine learning LSTM and ConvLSTM models are benchmarked against the time-series VAR and VEC model. We aim to answer the research question of whether ConvLSTM and LSTM neural networks can significantly outperform traditional time series forecasting methods. Lastly, ConvLSTM was benchmarked against the LSTM to answer the question of whether historical spatial IVS dynamics play a significant role in reducing the forecasting error.

We first determine whether RNN architecture could significantly outperform traditional time series models in the multi-step out-of-sample forecast of the IVS. We find that ConvLSTM produces the lowest overall out-of-sample error RMSE (0.014) and MAPE (8.26%) for the 90 trading day window. Unlike other models, the forecast remains stable and significantly outperforms VAR, VEC, and LSTM models, cross-sectional for almost all moneyness groups and time horizons. We find that ConvLSTM produces cross-sectionally lower RMSE and MAPE for ATM, OTM calls, and OTM puts for all expiration months when compared to other models. We also find that ConvLSTM produces the lowest RMSE and MAPE for longer-term (March and June) contracts, yielding mean RMSE of 0.01 and MAPE of a range between 5.54% and 7.29% for March contract, and mean RMSE of 0.006 and MAPE of a range between 3.46% and 3.81% for June contract. However, similar to previous research, we find that all the other models experience a varying degree of overestimation when predicting IV. We also conduct an economic analysis of a butterfly spread strategy based on model predicted volatility. We find that the convolution neural network-based deep learning model might produce

economic profits over a long horizon, but traditional time series models might perform well for shorter time periods despite their statistical disadvantage out of sample.

We further assess whether incorporating historical spatial dynamics of IV term structure and IV skew, can significantly improve the multistep forecast of the IVS. We find that stacking convolutional and average pooling layers for the ConvLSTM significantly reduces the RMSE when benchmarked against the LSTM, VAR, and VEC models. By introducing the convolutional operation and an average pooling layer, the ConvLSTM handles spatiotemporal patterns better than a regular fully connected LSTM, and larger kernel sizes allow the model to capture the important spatiotemporal properties of the IVS such as mean-reversion and the term structure.

We contribute to the literature on forecasting IV in three aspects. First, we propose a different methodology for encoding IV term-structure as a separate discrete dimension. Previously, the IVS term-structure and IV were encoded as a continuous variable (such as log forward moneyness, forward delta, or forward variance) and could only be used to produce cross-sectional results using basic fully connected neural network architectures. The new encoding method allows for more complex neural network architectures such as LSTM and ConvLSTM to maintain time-varying evolution per each contract group, for a large panel data set. Second, we apply ConvLSTM to the domain of modeling and forecasting IVS. The CNN have shown good results in the domains of computer vision and handwriting recognition, but had not found much application in modeling and forecasting IVS. Lastly, we produce a 30-trading day forecast window for an entire IVS, while previous research focuses on the short term forecast of a few timesteps.

The interpolation technique used to construct an initial training set is an important step in constructing the IVS. In future research, we suggest developing a more advanced technique for interpolating the training IVS. Orosi (2012) references practitioners' way of interpolating the IVS using a quadratic polynomial or a more advanced spline-based nonparametric IVS. Another way could be training a fully connected neural network similar to that of Liu et al. (2019) or Culkin (2017) to interpolate the entire surface and to feed the output to the proposed LSTM and ConvLSTM architectures, given the methodology proposed in this study for encoding IV term-structure as a separate discrete dimension. We suggest using a larger number of contracts and moneyness groups to take advantage of the performance gains from using a larger kernel on the ConvLSTM as referenced in Shi et al. (2015). We understand that empirical results depend on the choice of hyperparameters. Although LSTM is inferior to ConvLSTM in our out-of-sample findings, more hyperparameter tuning may improve the performance of the LSTM model so that it may be comparable to the performance of the ConvLSTM model. One could conduct hyperparameter tuning (such as Grid Search) as discussed by Liu et al. (2019) for the models, which could improve the results.

ACKNOWLEDGMENTS

We would like to thank the Editor, Robert I. Webb, the anonymous reviewer, and Matthew Diersen for their constructive suggestions. We appreciate the helpful comments from participants at SDSU Data Science Symposium 2020, Minnesota Center for Financial and Actuarial Mathematics Seminar, University of Minnesota, and Southwestern Finance Association Annual Meeting 2021. We are grateful for the financial support from Ness School of Management and Economics, South Dakota State University. We would like to acknowledge Li Zhuo's skillful help with data processing.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available from DeltaNeutral. Restrictions apply to the availability of these data, which were used under license for this study. Data are available from the authors with the permission of DeltaNeutral.

ORCID

Zhiguang Wang http://orcid.org/0000-0002-8014-026X

REFERENCES

Black, F., & Scholes, M. (1973). The pricing of options and corporate liabilities. *Journal of Political Economy*, *81*(3), 637–654.

Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, *31*(3), 307–327.

Corsi, F. (2008). A simple approximate long-memory model of realized volatility. *Journal of Financial Econometrics*, *7*(2), 174–196.

Culkin, R., & Das, S. R. (2017). Machine learning in finance: The case of deep learning for option pricing. *Journal of Investment Management*, *15*(4), 15.

Diebold, F., & Mariano, R. (1995). Comparing predictive accuracy. *Journal of Business & Economic Statistics*, *13*(3), 253–63.

Engle, R. F. (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. *Econometrica*, *50*(4), 987–1007.

Gospodinov, N., Gavala, A., & Jiang, D. (2006). Forecasting volatility. *Journal of Forecasting*, *25*(6), 381–400.

Greff, K., Srivastava, R. K., Koutnik, J., Steunebrink, B. R., & Schmidhuber, J. (2017). Lstm: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, *28*(10), 2222–2232.

Hansen, P. R., & Lunde, A. (2005). A forecast comparison of volatility models: does anything beat a garch(1,1)? *Journal of Applied Econometrics*, *20*(7), 873–889.

Harvey, D., Leybourne, S., & Newbold, P. (1997). Testing the equality of prediction mean squared errors. *International Journal of Forecasting*, *13*(2), 281–291.

Heston, S. L. (1993). A closed-form solution for options with stochastic volatility with applications to bond and currency options. *Review of Financial Studies*, *6*, 327–343.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.

Hosker, J., Djurdjevic, S., Nguyen, H., & Slater, R. (2018). Improving vix futures forecasts using machine learning methods. *SMU Data Science Review*, *1*(4), 91.

Interest rate data. Yahoo Finance; 2002-2019; IRX Treasury Yield; https://finance.yahoo.com/

Johansen, S. (1991). Estimation and hypothesis testing of cointegration vectors in gaussian vector autoregressive models. *Econometrica*, *59*(6), 1551.

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In Y. Bengio, & Y. LeCun (Eds.), 3rd *International Conference on Learning Representations*, ICLR 2015, San Diego, CA, May 7–9, 2015, Conference Track Proceedings.

Kwiatkowski, D., Phillips, P. C., Schmidt, P., & Shin, Y. (1992). Testing the null hypothesis of stationarity against the alternative of a unit root. *Journal of Econometrics*, *54*(3), 159–178.

Liu, S., Oosterlee, C. W., & Bohte, S. M. (2019). Pricing options and computing implied volatilities using neural networks. *Risks*, 7(1). doi:10.3390/risks7010016

Luo, R., Zhang, W., Xu, X., & Wang, J. (2018). A neural stochastic volatility model. The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18), 6401-6408.

Luong, C., & Dokuchaev, N. (2018). Forecasting of realised volatility with the random forests algorithm. *Journal of Risk and Financial Management*, *11*(4), 61.

Maddala, G. S., & Wu, S. (1999). A comparative study of unit root tests with panel data and a new simple test. *Oxford Bulletin of Economics and Statistics*, *61*, 631–652.

Majmudar, U., & Banerjee, A. (2004). VIX forecasting. *SSRN Electronic Journal*, 1–23.

Orosi, G. (2012). Empirical performance of a spline-based implied volatility surface. *Journal of Derivatives*, *18*, 16.

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

Samsudin, N. I. M., & Mohamad, A. (2016). Implied volatility forecasting in the options market: A survey. *Sains Humanika*, *8*, 9–18.

Shi, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-k., & Woo, W.-c. (2015). Convolutional lstm network: A machine learning approach for precipitation nowcasting. In T*wenty-Ninth Annual Conference on Neural Information Processing Systems (NIPS)*, 1–9.

SPX options data. DeltaNeutral; 2002-2019; SPX Options End of Day Data; http://www.deltaneutral.com

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*(56), 1929–1958.

Xiong, R., Nichols, E. P., & Shen, Y. (2016). Deep learning stock volatility with google domestic trends. arXiv preprint, arXiv:1512.04916, 1–6.

Zhou, D.-X. (2020). Universality of deep convolutional neural networks. *Applied and Computational Harmonic Analysis*, *48*(2), 787–794.

How to cite this article: Medvedev, N., & Wang, Z. (2022). Multistep forecast of the implied volatility surface using deep learning. *Journal of Futures Markets*, 42, 645–667. https://doi.org/10.1002/fut.22302