South Dakota State University

Open PRAIRIE: Open Public Research Access Institutional Repository and Information Exchange

Electronic Theses and Dissertations

2019

Multi-Step Forecast of the Implied Volatility Surface Using Deep Learning

Nikita Medvedev
South Dakota State University

Follow this and additional works at: https://openprairie.sdstate.edu/etd

Part of the Finance Commons

MULTI-STEP FORECAST OF THE IMPLIED VOLATILITY SURFACE USING DEEP

LEARNING

BY

NIKITA MEDVEDEV

A thesis submitted in partial fulfilment of the requirements for the

Master of Science

Major in Economics

South Dakota State University

2019

THESIS ACCEPTANCE PAGE

Nikita Medvedev

This thesis is approved as a creditable and independent investigation by a candidate for the master's degree and is acceptable for meeting the thesis requirements for this degree. Acceptance of this does not imply that the conclusions reached by the candidate are necessarily the conclusions of the major department.

Zhiguang Wang

Advisor   Date

Eluned Jones

Department Head   Date

Dean, Graduate School   Date

ACKNOWLEDGMENTS

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

MULTI-STEP FORECAST OF THE IMPLIED VOLATILITY SURFACE USING DEEP

LEARNING

NIKITA MEDVEDEV

2019

Implied volatility is an essential input to price an option. Machine learning architectures have shown strengths in learning option pricing formulas and estimating implied volatility cross-sectionally. However, implied volatility time series forecasting is typically done using the univariate time series and often for short intervals. When a univariate implied volatility series is forecasted, important implied volatility properties such as volatility skew and the term structure are lost. More importantly, short term forecasts can't take advantage of the long term persistence in the volatility series.

The thesis attempts to bridge the gap between machine learning-based implied volatility modeling and multivariate multi-step implied volatility forecasting. The thesis contributes to the literature by modeling the entire implied volatility surface (IVS) using recurrent neural network architectures. I implement Convolutional Long Short Term Memory Neural Network (ConvLSTM) to produce multivariate and multi-step forecasts of the S&P 500 implied volatility surface. The ConvLSTM model is capable of understanding the spatiotemporal relationships between strikes and maturities (term structure), and of modeling volatility surface dynamics non-parametrically.

I benchmark the ConvLSTM model against traditional multivariate time series Vector autoregression (VAR), Vector Error Correction (VEC) model, and deep learning-based Long-Short-Term Memory (LSTM) neural network. I find that the ConvLSTM significantly outperforms traditional time series models, as well as the benchmark Long Short Term Memory(LSTM) model in predicting the implied volatility surface for a 1-day, 30-day, and 90-day horizon, for out-of-the-money and at-the-money calls and puts.

1   INTRODUCTION

1.1   BACKGROUND

The typical performance of a financial asset is measured via its return. The fluctuations in the returns, as well as their randomness, is described by the asset's volatility. Volatility modeling has been extensively researched in the areas of finance, risk management, and policymaking. Volatility is an essential component of financial derivatives; models and forecasts are particularly important for the institutions involved in derivative trading. Good forecasts of the volatility of asset returns enable institutions to asses their investment risk and optimize their investment portfolios. Two measures of volatilities exist: 1) historical (realized) volatility, which can be observed from the historical data and realized at a certain point in time; 2) implied volatility (IV), which is not directly observable, but instead is an investor's forward-looking view on the underlying asset's future returns. Market expectations, macroeconomic conditions, and general market supply and demand are common factors that drive future volatility. Implied volatility is typically derived through a closed-form formula. Changes in implied volatility are market-driven and dynamic, which makes predicting future implied volatility a challenging task.

1.2   PROBLEM

As identified in the survey by Samsudin and Mohamad (2016), the vast majority of volatility forecasting models fall into two major categories: 1) option-implied volatility models and 2) historical time series models. The former represents traders' forward-looking view on the future direction of an asset's volatility throughout the life-cycle of the contract. Because the market attempts to predict the future expected volatility of the underlying asset that changes dynamically, the incorrect parameters to the closed-form option pricing formula lead to a degree of overpricing of the options as the time to maturity of the contract increases (Hull and White, 1987). On the other hand,
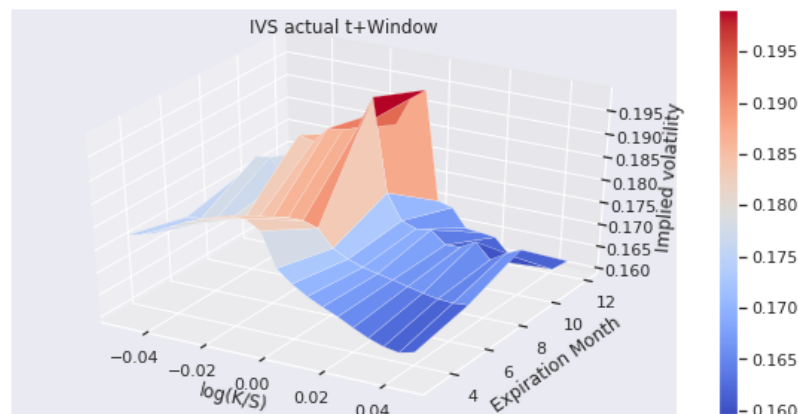
Figure 1: Average interpolated Implied Volatility Surface (IVS) between 2007-08-17 and 2007-12-31.

historical time series tries to express future price trends relying on past price trends. This approach doesn't take into consideration the market's expectations of future returns. Likewise, time series methods tend to ignore the underlying asset's fundamental information, such as earnings, debt, revenues, or other news.

Implied volatility (IV) has two important empirically observed properties: 1) IV term structure; 2) IV skew. The first property implies that the future IV is typically higher than today's IV because of the large degree of uncertainty in future returns. The second property implies that the market sentiment and investor preferences can cause option strikes to have differences in IV levels. The voluminous literature has popularized the Skew following empirical studies such as Rubinstein (1985), Dupire (1994), and Dumas, Fleming, and Whaley (1998) that collectively bring out that IV's vary with strike prices and time-to-maturities, in contrast with the assumptions of the (Black and Scholes, 1973) option pricing model (BSM), which assumes constant volatility and log-normal price distribution of the underlying asset. IV skew and IV term structure can be captured in a dynamic 3D figure known as the implied volatility surface (IVS). In the short term, deviations, from the classical IVS shape can be observed, such as upward skew for some strikes, or downward term structure. For instance, in Figure 1 between 2007-08-17 and 2007-12-31, it is observed that the IV is higher for OTM puts than OTM calls and that IVS

has a downward IV term structure. Much of the previous research on option-implied volatility modeling and forecasting focuses on producing cross-sectional results spearheaded by the non-parametric methods for volatility pricing and not time series forecasting. This is an area where machine learning architectures have shown significant strengths. Adaptation of these data-driven machine learning techniques in volatility research tends to focus on three major areas: 1) forecasting future historical volatility of an underlying asset; 2) non-parametric framework or solvers for option or pricing; 3) time series forecasting on the option implied volatility, volatility products (e.g., VIX), or volatility distributions. The term structure dimension used to be a secondary priority for earlier researchers but is thoroughly addressed in the more modern research (Chalamandaris and Tsekrekos, 2011). The thesis contributes to the third major subsection of the literature that focuses on implied volatility forecasting and proposes a data-driven machine learning technique for pricing IV and multi-step forecast of both IV skew and IV term structure through the IVS.

1.3   PRELIMINARIES

Forecasting historical volatility means predicting future volatility using historical volatility. A number of studies had been conducted over the years using the financial time series or econometric models to address this. One of the assumptions of a typical regression model is constant variance (random noise) of the residuals, also known as homoscedasticity. However, it is well known that financial time series tend to experience unequal differences in means, medians, and interquartile ranges across different periods, and thus, the residuals follow a mixture of distributions, also known as heteroscedasticity. Up until more recent years, autoregressive conditional heteroscedasticity (ARCH) model (Engle, 1982) where the lagged variance of the series is expressed as a function of time, and generalized autoregressive conditional heteroscedasticity (GARCH) family models (Bollerslev, 1986), were among the best models in forecasting realized volatility.

However, GARCH family models tend to produce a term structure that reverts to the long-term mean (Sinclair, 2013) and fails to capture the volatility skew (Dupire, 1994). Throughout the years, many GARCH family models have evolved to address the failure of capturing the IV skew. However, Hansen and Lunde (2005), tested over 330 GARCH family models, seeking to identify advantages of the more sophisticated models to a standard GARCH(1,1) model. The research found no evidence that more sophisticated GARCH models significantly outperform the standard GARCH(1,1) model.

The results of a number of empirical studies suggest that option-implied volatility can be used to predict future volatility. Time series predictions of IV can outperform historical volatility (HV) as a predictor of future realized volatility (RV) (Samsudin and Mohamad, 2016; Chalamandaris and Tsekrekos, 2011). More-so forecasted IV and historical IV are more closely related than RV and IV, and the conditions in the options market can impact this relationship (Zumbach, 2009). More recently, studies had emerged that utilize a hybrid approach by combining various iterations of neural networks with GARCH models, specifically to address the relative weakness of capturing the IV skew. Neural networks have shown advantages in estimating complex non-linear 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 non-parametric 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 non-parametric financial derivative research that predominantly focuses on providing efficient solutions to the partial differential equations (PDEs). Often the use of the non-parametric algorithms and techniques, such as forests, boosting, support vectors, and neural networks for volatility estimation can lead to far more superior results (Park, Kim, and Lee, 2014). The usage of the artificial neural networks (ANNs) primarily focuses on development of efficient alternatives to the time

consuming numerical solvers which fail to deliver fast on-line solutions (Liu, Oosterlee, and Bohte, 2019). Because of this performance bottleneck, the ANNs are trained to approximate the PDEs. The solutions are expressed as a set of matrix multiplication, and the workload distributed to the graphical processing units (GPUs) and run in parallel, which significantly improves the computational performance. Multiple researchers have noted significant performance gains from applying these techniques to the widely recognized options pricing models. Culkin and Das (2017) applied the simplest form of ANN, multi-layer perceptron (MLP) neural network to the BSM option pricing model. They note that deep neural networks can serve as a universal approximator for almost any function. In particular, the results were significant when applied to the BSM model. Liu, Oosterlee, and Bohte (2019) did the same for both BSM and Heston stochastic volatility (Heston, 1993) models, which confirmed the findings of Culkin and Das (2017).

The IV skew and the term structure can be captured in the implied volatility surface (IVS). The IVS is used by the market participants to price the options. However, because option volume is supply and demand-driven, markets often do not have quotes for all of the strikes and maturities, so the IVS is interpolated to fill the missing quotes (Orosi, 2012), which is why forecasting and interpolation techniques are especially important for the market makers. For example, CBOE's Volatility Index (VIX) is a measure of expected annualized volatility implied by S&P 500 options that are measured and published by the Chicago Board of Exchange (CBOE). The VIX itself is a measure derived from the near term SPX option contract. VIX, due to its popularity, has been a research topic of many studies. Traders consider VIX a mean-reverting asset, and thus a variation of mean-reverting time series models (e.g., GARCH family) seems to be a natural choice for a large number of studies for forecasting VIX. Although the calculation of the VIX is transparent, the spot VIX is difficult to replicate because of the square root function. So because of the difficulty to replicate VIX and VIX covering only the near term contract, the forecast of the IVS from a traders perspective is of great importance.

1.4   GOAL

Neural networks can learn IV equations accurately, and parametric time series models are able to capture temporal relationships. The thesis bridges the gap between these two methodologies by applying two new neural network architectures for multi-step time series forecasting of the IVS that also takes advantage of non-parametrically learned BS-IV function. The thesis contributes to the literature of modeling and forecasting implied volatility using entirely machine learning techniques. The goal of this thesis is to answer three overarching research questions: 1) Do the cointegrated relationships significantly impact the multi-step forecast of the IVS? Cointegration is relevant because of the short-run stochastic dynamics of volatility; 2) Can the recurrent neural network architecture significantly outperform traditional time series models in a multi-step out-of-sample forecast of the IVS? While MLP's have shown strength in option pricing and volatility estimation, supervised time-series predictions are computationally expensive; 3) 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. Lastly, I will select the IVS forecasting methodology with the lowest forecasting error for multiple time horizons. I aim to demonstrate that the combination of these factors can lead to an overall improvement in forecasting the option based implied volatility over the traditional time series approaches.

2   LITERATURE REVIEW

In this section, I will briefly discuss the geometric Brownian motion (GBM), which is a core component of three widely popular option pricing models. I discuss the Black-Scholes model and briefly mention other methods for option pricing and implied volatility modeling. I will also review econometric and parametric methods that are

relevant for volatility time-series forecasting. Lastly, I will review some of the more recent methodologies used for option implied volatility forecasting and option pricing, with the help of traditional Artificial Neural Networks and more recent deep learning architectures.

2.1   IMPLIED VOLATILITY

The BS model (Black and Scholes, 1973) undoubtedly had a significant impact on the world of finance and provided a theoretical framework for options trading and hedging. One of the main advantages of the model is that the BS formula can be inverted to produce implied volatility and IVS.

The key underlying assumption of the model is that an underlying asset follows the stochastic process S_t such that:

$$dS_t = \mu S_t dt + \sigma S_t dW_t \tag{1}$$

where both μ or the drift and σ volatility are constant and W_t is a Brownian motion. Using Ito's lemma, the BS equation for the European call option can be denoted as:

$$\frac{\partial C}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 C}{\partial C^2} + rS\frac{\partial C}{\partial S} = rC \tag{2}$$

where t is time to maturity, risk-free rate is r. The Equation 2 serves as a closed form solution for pricing European call and put options:

$$C(S, t) = N(d_1)S - N(d_2)Ke^{-rt}$$
$$d_1 = \frac{1}{\sigma\sqrt{t}}\left[\ln\left(\frac{S}{K}\right) + t\left(r + \frac{\sigma^2}{2}\right)\right]$$
$$d_2 = d_1 - \sigma\sqrt{t} \tag{3}$$
$$N(x) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{x} e^{-\frac{1}{2}z^2}dz$$

where S is stock price, K is strike, $N(x)$ is normal distribution.

Although BS makes assumptions on the underlying asset that don't hold empirically such as log return of stock price with constant volatility it's still an incredibly useful and almost a universally used mechanism for extracting IV given the option market data (Homescu, 2011).

$$\sigma(K, t) = BS^{-1}(C, S, K, t, r) \tag{4}$$

BS volatility σ can be derived by solving Equation 4, where BS^{-1} denotes inverse BS function. The inverse Equation 4 can be solved numerically, with an iterative technique that solves Newton-Raphson formula, since BS equation guarantees σ such that $\sigma \in [0, \infty]$[1]: solve for σ_{imp}:

$$\sigma_{imp} = \sigma_n - \frac{BS(\sigma_n) - P}{\nu(\sigma_n)} \tag{5}$$

where σ_{imp} is the IV, P is option price, v is the volatility derivative vega. Initially an arbitrary σ_0 is guessed, and given the market data, the iterative solution will converge to the optimal σ_{imp}.[2] The key limiting factor of the BS model is the constant volatility, which fails to capture the skew (smile) present in the market data. Hull and White (1987) identified that BS model under-prices at-the-money options and overprices deep-out-of money options, and the effect is exaggerated as the time to maturity increases. Extensions to the BS model had been made to account for the smile effect, but one has to break the intuition behind Equation 2 that the options risk can be fully hedged by trading the underlying. This adds non-tradable risk sources such as jump risk, stochastic volatility, or transaction costs (Dupire, 1994; Eraker, Johannes, and Polson, 2003). Furthermore, volatility is strike dependent and it's attributed to an option trader's belief in leptokurtosis of an underlying asset and expectations of larger price swings than assumed by the GBM,

---

[1] $BS(\sigma)$ for European options has a closed-form volatility derivative (vega) which is non-negative.
[2] Solution works well with European payoffs. Exotic payoffs require applying secant optimization method.

which would increase the market prices of the out-of-money options relative to the BS prices (Alexander, 2004). This is because of the risk premium caused by the changes in volatility (Heston, 1993).

Heston (1993) proposed a widely popular model that allows for a stochastic correlation between underlying asset returns and its volatility. The resulting two-factor stochastic volatility model (SV) can be used to explain the presence of the implied volatility smile, especially in the long time to maturity options. The model is often used as a benchmark for other option pricing models, or to extrapolate implied volatility, when constructing the IVS. Orosi (2012) benchmarks Heston model against other methods of constructing and predicting the IVS, and proposes a spline model, which has superior performance compared with the benchmark models proposed in the study. Most notably, the interpolation techniques are especially important for the market makers to price the illiquid options.

2.2   HISTORICAL VOLATILITY FORECASTING

In contrary to the theoretical models, time series models seek to explain the movement in volatility using some auto-regressive 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 implied volatility forecasting. Hansen and Lunde (2005) benchmarked over 339 types of volatility models, to the standard ARCH(1) and GARCH(1,1) for on-day ahead forecast of realized volatility. 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, Gavala, and Jiang (2006) proposed several parametric and non-parametric 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 implied volatility has a longer memory than realized volatility. The forecasts were performed one-step ahead for both realized and implied volatility. They find evidence that implied volatility contains valuable information about realized volatility, which, if included, can significantly improve the performance of predicting realized volatility over a long time horizon.

Xiong, Nichols, and Shen (2016) proposed a new methodology for predicting future realized volatility of the S&P 500 index, using macroeconomic factors obtained from Google Trends. A 25-dimensional vector of 25 trend series representing a selected number of keywords, beginning January 1st 2004, is fed into the Long-Short-Term Memory (Hochreiter and Schmidhuber, 1997) neural network (LSTM) with 1 hidden LSTM cell. 1-step forecast of future realized volatility 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 realized volatility for multi-step out of sample forecast on high-frequency data. The proposed model has accuracies 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 5day to 22-day forecast. However, the random forest model was able to outperform the benchmark Heterogeneous Autoregressive (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 realized volatility 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 2-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.

2.3   IMPLIED VOLATILITY FORECASTING

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 6 different supervised learning models, including the recurrent neural network (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 implied volatility estimation. For instance, Liu, Oosterlee, and Bohte (2019) proposes a machine learning technique for efficiently computing implied volatility of 3 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 and Das (2017), where a simple NN is trained on a dataset 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 realized volatility forecasting. The domain of non-parametric 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 implied volatility 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 thesis aims to bridge the gap between forecasting and non-parametric modeling of IV. I 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 U.S. stock market index S&P 500 and its most popular financial product offered through the Chicago Board Options Exchange (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 2002-02-05 and 2007-12-31. The historical open-high-low-close(OHLC) data for IRX - The U.S. treasury bill tracking index and SPX - S&P 500 options index for the same periods are collected from Yahoo Finance. Delta Neutral data contains the end of day quotes, implied volatility, and sensitivity information

for the traded SPX options.

3.2   DATA PREPARATION

There are a total of 4765 unique contracts series[3] in the sample. Due to the computational constraints, I begin from down-sampling the 4765 contracts to 4 quarterly contracts: March, June, September, December. By down-sampling, the number of unique contracts is reduced to 1962.
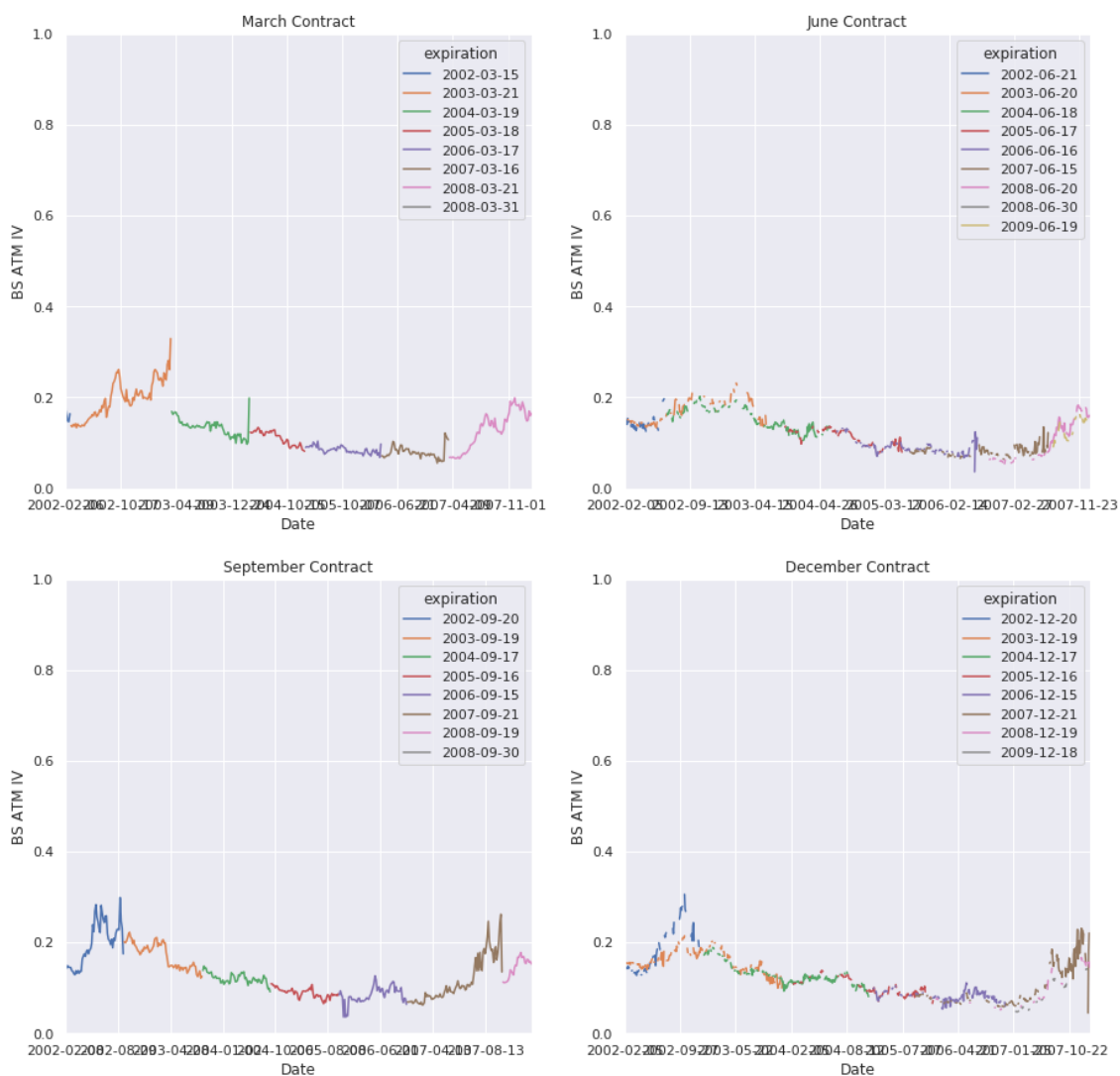
Figure 2: Selected at-the-money (ATM) IV for March, June, September and December contracts and expirations.

[3]Series are labeled unique when a series has a unique strike-expiration combination

I 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 the thesis are not aware of the expiring nature of the contracts, and expect a continuous input.

Because the dataset contains nonstandardized daily option chains and only includes historical option contracts that had been traded on a trading day, I am left with 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, it's then required to interpolate a total of 388,545 missing values out of 517,446 total synthetic observations, where 128,901 values are not missing. The missing values appear after producing a daily balanced panel for each of the target moneyness bins. To address this, I elected a two way (forward and backward direction) linear interpolation to fill the missing BS-IV values. I found that in many cases, the implied volatility 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 3 steps: 1) Interpolate based on the implied volatility 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 implied volatility observations are summarized in Table 1. The step-by-step methodology along with the the IV binning technique discussed in Section 3.4 is displayed in Algorithm 1.

The series follow an identical format where the panel is expressed as either $m \times n$ 2d matrix or 3d 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:

Table 1: Descriptive statistics of implied volatility observations.

Descriptive statistics for the train and test set, grouped by the contract expiration month, option type and set type. 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 defined in Section 3.4. Train set is for the period 2002-02-05 to 2007-08-17 and Test set is for the period 2007-08-17 to 2007-12-31.

Expiration Month	type	Train/Test	count	mean	std	min	25%	50%	75%	max
3	ATM	Test	270.0	0.173517	0.020578	0.130264	0.159058	0.171543	0.186004	0.232255
		Train	4113.0	0.137501	0.044594	0.018430	0.105920	0.124671	0.155511	0.503159
	OTM calls	Test	720.0	0.168977	0.019177	0.130264	0.154741	0.168666	0.180663	0.227763
		Train	10968.0	0.135211	0.042906	0.030436	0.104773	0.121936	0.153075	0.370673
	OTM puts	Test	810.0	0.183476	0.021089	0.144728	0.167457	0.181057	0.196362	0.241240
		Train	12339.0	0.143171	0.045793	0.032555	0.110591	0.131850	0.159565	0.573669
6	ATM	Test	270.0	0.172121	0.018115	0.136246	0.159129	0.171054	0.184637	0.215517
		Train	4113.0	0.137857	0.044923	0.035999	0.110621	0.127786	0.154821	1.219848
	OTM calls	Test	720.0	0.169626	0.016832	0.136761	0.157402	0.168287	0.181189	0.214050
		Train	10968.0	0.134474	0.038502	0.027980	0.107923	0.123425	0.152467	0.673974
	OTM puts	Test	810.0	0.177085	0.018836	0.144185	0.162872	0.175271	0.191154	0.220612
		Train	12339.0	0.142827	0.039726	0.038961	0.115236	0.133663	0.158523	1.199098
9	ATM	Test	270.0	0.176224	0.033696	0.065167	0.155639	0.174869	0.191138	0.401705
		Train	4113.0	0.137959	0.089537	0.022246	0.106320	0.123153	0.152594	3.910069
	OTM calls	Test	720.0	0.171379	0.024505	0.061908	0.154092	0.172667	0.187088	0.307736
		Train	10968.0	0.138194	0.171103	0.041281	0.104927	0.120159	0.149237	7.190991
	OTM puts	Test	810.0	0.187652	0.037848	0.065499	0.166425	0.187107	0.203250	0.567468
		Train	12339.0	0.142431	0.044617	0.032246	0.110954	0.131188	0.158022	0.383376
12	ATM	Test	270.0	0.167583	0.031822	0.034859	0.148593	0.166525	0.181346	0.448458
		Train	4113.0	0.145827	0.120620	0.025532	0.112641	0.132496	0.157582	3.125669
	OTM calls	Test	720.0	0.163037	0.024563	0.044491	0.145840	0.162217	0.176002	0.297113
		Train	10968.0	0.141859	0.099006	0.026296	0.111144	0.127803	0.154656	2.875060
	OTM puts	Test	810.0	0.183424	0.033181	0.033641	0.162127	0.178440	0.200243	0.466328
		Train	12339.0	0.158823	0.175566	0.030437	0.118292	0.139863	0.161711	3.221428

$$X_{i,j,k} = \begin{bmatrix} x_{0,0,0.10} & x_{0,0,0.15} & x_{0,0,0.20} & \cdots & x_{0,0,k} \\ x_{1,0,0.10} & x_{1,0,0.15} & x_{1,0,0.20} & \cdots & x_{1,0,k} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{i,j,0.10} & x_{i,j,0.15} & x_{i,j,0.20} & \cdots & x_{i,j,k} \end{bmatrix} \tag{6}$$

3.3 BS PDE SOLVER

I follow a traditional method for estimating the Black-Scholes implied volatility extracted from the option market prices and inverting BS PDE for IV.

$$\sigma(K,t) = BS^{-1}(C, S, K, t, r) \tag{7}$$

C = Option price

K = Contract strike

S = Daily SPX index closing prices

t = (Contract Expiration-Closing day)/252

r = Daily IRX index closing prices (interest rate)

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[4] 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.

---

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

3.4   IV LOG-MONEYNESS BINNING

A common problem with the financial time-series data is the stochastic trend and non-stationarity. In particular, option contracts are strike dependant and have a finite time-to-maturity. A number of techniques for parametric time series 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 non-stationarity 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 7 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 and Das (2017); 2) Standardized forward moneyness: $m = \dfrac{ln(S/K)}{\sigma\sqrt{t}}$ 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 I can generate a fixed range of buckets for each expiration month of the contract. For each contract month I 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 $\longrightarrow$ σ_{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** For each day in days return (bucket labels, expiration months, σ_{BS}) like Matrix 6.

3.5 STATIONARITY AND COINTEGRATION TESTS

The purpose of the statistical tests is to build a proper time series model for implied volatility forecasting. The preliminary analysis includes the Augmented Dickey-Fuller (ADF) Test, (Kwiatkowski et al., 1992) (KPSS) test, (Maddala and Wu, 1999) Unit-Root test (MADWU) 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.

In particular, I intend to address two questions: 1) Do the option implied IV series follow a stationary AR process? 2) Does the IV panel exhibit any meaningful short and long-run cointegrated relationships?

In this case, unit-root tests are used to determine if the time-series should be first differenced $I(1)$ or regressed $I(0)$. I prepare 3 variations of the series, similar to that of Gospodinov, Gavala, and Jiang (2006). $\bar{V}_t$ represents BS-IV series, $ln(\bar{V}_t)$ represents a log-transformed series and $ln(\bar{V}_t^{1/2})$. MADWU test for panel data stationarity, and is used to address the lack of power of the ADF test in distinguishing unit root of the panel data, because under ADF different null hypothesis are tested for each series (Maddala and

Wu, 1999). For simplicity, consider IV series V_t decomposition:

$$\bar{V}_t = T_t + z_t$$

$$T_t = \beta_0 + \beta_t \tag{8}$$

$$z_t = \phi z_{t-1} + \eta_t' \sim WN(0, \sigma^2)$$

where T_t is a deterministic linear trend and z_t is a common AR(1) process. If $\phi < 1$, then $z_t = z_0 + \sum_{j=1}^{t} \eta_j$, is a stochastic process and V_t is IV series with drift.

The ADF tests the null hypothesis of the presence of the unit root against an alternative hypothesis of series being stationary, where the ADF t-statistic is the result of the least square estimates for the regression in Equation 8.

$$ADF_t = \frac{\hat{\phi} - 1}{SE(\phi)} \tag{9}$$

When the ADF test is applied to the original non-differenced series, I fail to reject the null-hypothesis, implying that series are non-stationary $I(0)$ in most of the cases and need to be differenced. Non-stationarity is quite common for financial series. The original series are first differenced, and the ADF test is applied to the $I(1)$ series, I reject the null hypothesis for the majority of the series for 2002-2007 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 implied volatility 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 non-differenced IV series follow a non-stationary process are somewhat inconsistent with the earlier research. Although sampling methodology and sampling window or even the IV instrument can
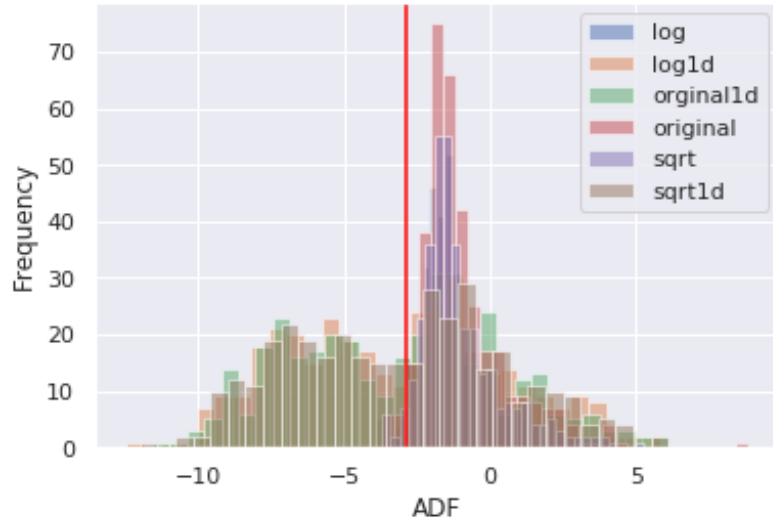
Figure 3: ADF Test for 2002-2007.

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, Gavala, and Jiang, 2006). As the result of these findings, I continue treating the volatility series as $I(1)$ process

As a complement to the ADF test, I conduct the KPSS test (Kwiatkowski et al., 1992) to test whether series are level stationery. KPSS test decomposes IV series V_t into deterministic, random walk and stationary error term:

$$y_t = T_t + z_t + \eta_t \tag{10}$$

where T_t is a deterministic linear trend, z_t is a random walk with the properties like Equation 8 and $\eta \sim N(0, \sigma^2)$ is the stationary error. The partial sum of OLS residuals S_t from Equation 10 is constructed to obtain the KPSS test statistic:

$$KPSS = n^{-2} \sum_{t=1}^{n} \frac{S_t^2}{\hat{\sigma^2}} \tag{11}$$

where $\hat{\sigma^2}$ is the long-run variance of the estimate of the residual. The null hypothesis for
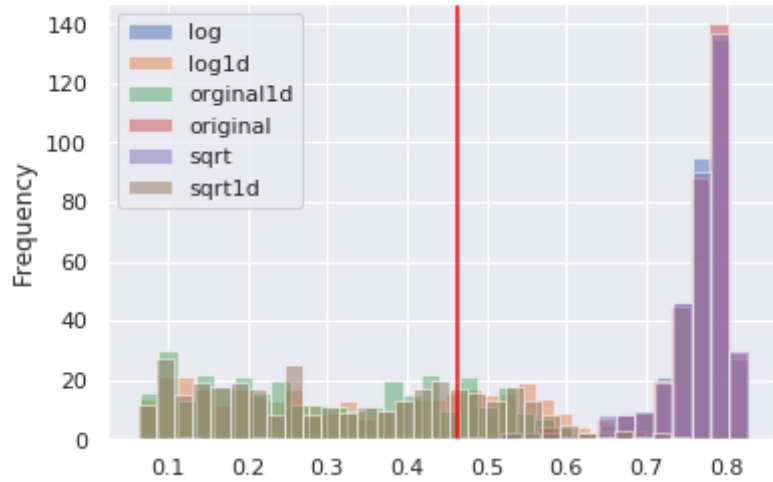
Figure 4: KPSS Test for 2002-2007.

the 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 the original series are first differenced, I fail to reject the null hypothesis for the majority of the series, concluding that the series are level stationary for the 2002-2007 sample.

Although it's 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, I utilize Madwu fisher type test:

$$MADWU = P = -2 \sum_{t=1}^{N} \ln p_i \tag{12}$$

where p_i is the p-value from the unit-root root test, that results in a χ^2 distribution P. 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, I 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 chi-squared distribution, I can conclude that although some series tend to exhibit unit-root, these series don't significantly impact the overall stationarity of the panel, yielding critical values 8462.8 and 9436.2 respectively.

Now that I've established that the panel exhibits stationarity when first differenced. I am also interested in exploring a presence of any spurious correlation between groups, i.e 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 equilibriums 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. I utilize the urca[5] 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. Based on the the critical values of the trace statistic in Table 2, I 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.

---

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

Table 2: Johansen test trace statistic critical values.

Relations	Critical Value
r ≤ 19	138.466
r ≤ 18	1622.217
r ≤ 17	3722.018
r ≤16	5904.430
r ≤ 15	8191.562
r ≤ 14	10493.895
r ≤ 13	12835.898
r ≤ 12	15191.440
r ≤ 11	17551.433
r ≤ 10	19918.851
r ≤ 9	22286.508
r ≤ 8	24656.662
r ≤ 7	27028.054
r ≤ 6	29408.080
r ≤ 5	31795.009
r ≤ 4	34190.643
r ≤ 3	36605.766
r ≤ 2	39054.031
r ≤ 1	41648.846
r ≤ 0	47320.253

3.6 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 non-paramterically estimate the cointegrated behavior of the panel 3) accounts for long term error correction (or long memory of the series). The model evaluation process is based on $y_{t+1}...y_{t+30}$ 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. I summarize the findings in Chapter 5.

3.6.1 LONG-SHORT TERM MEMORY MODEL

Long Short Term Memory neural network (Hochreiter and Schmidhuber, 1997) is a type of recurrent neural network (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 5. Forget gate and output gates are
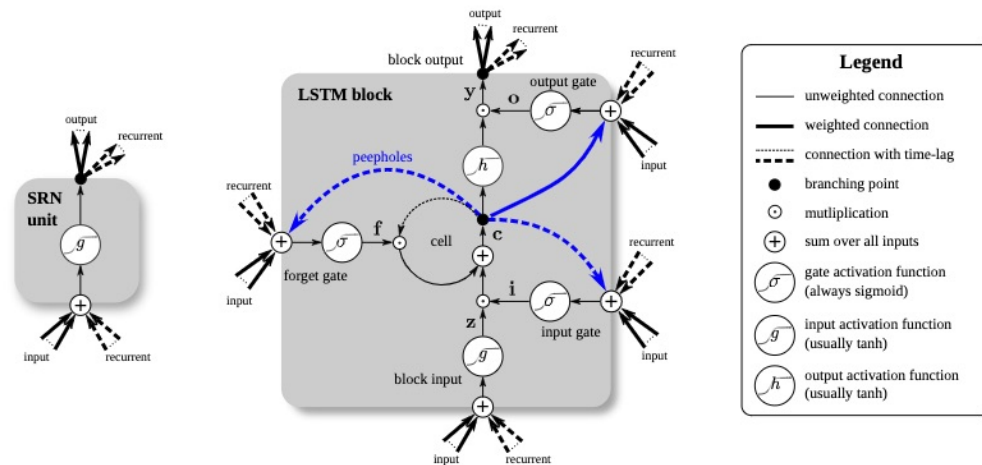
Figure 5: Simple RNN cell and LSTM memeory cell as presented in Greff et al. (2017).

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 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[6] and the flow for the models is generated in Tensorboard[7] and visualised in Figure 6. 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.

---

[6]https://keras.io/
[7]https://www.tensorflow.org/tensorboard
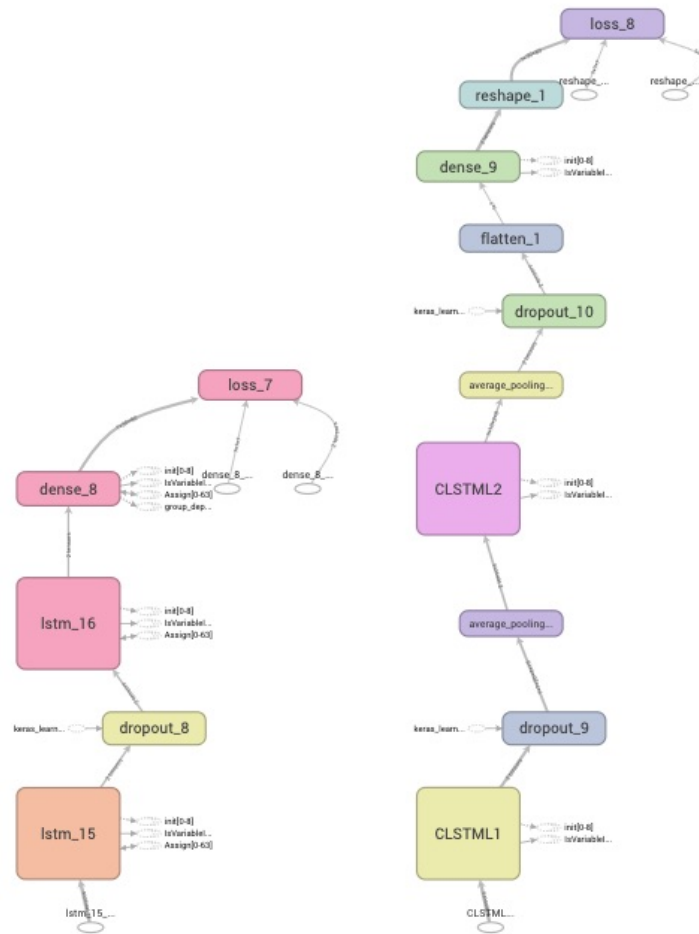
Figure 6: LSTM and ConvLSTM model flowcharts.

3.6.2 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 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 7.

As shown by Shi et al. (2015) the ConvLSTM is shown to consistently outperform the fully connected LSTM on multiple spatiotemporal datasets, on various configurations.
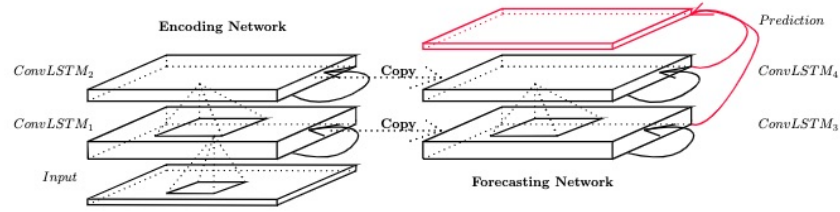
Figure 7: Forecasting ConvLSTM as presented in Shi et al. (2015).

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 non-linearity, 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 non-linear 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 6. 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, features), where rows and columns represent the grid as shown in Figure 7.

I 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, I 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, December).

3.6.3   VECTOR AUTOREGRESSION MODEL

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

$$\hat{y}_t = A_1 \boldsymbol{y}_t + \ldots + A_k \boldsymbol{y}_{t-p} + \boldsymbol{u}_t \tag{13}$$

where $\hat{y}_t$ is a $K \times 1$ vector of either forecast $yt + i$ or lagged observation $y_t, y_{t-1}, ..., y_{t-p}$, $A_1, ..., A_k$ are 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 cointegrated 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.

3.6.4   VECTOR ERROR CORRECTION MODEL

As observed in Johansen procedure, the volatility series experience long-run cointegrated behavior, so the Vector Error Correction Model (VECM) is a restricted VAR model, where the large swings in the short term can be restricted and converged to their cointegrated (long-term) relation. So Equation 13 can be rewritten to include the cointegrated transformation:

$$\Delta \hat{y}_t = \Pi y_{t-1} + \sum_{i=1}^{p-1} \Gamma_i \Delta y_{t-1} + u_t \tag{14}$$

where $\Pi = I - \Pi_1 + ... + \Pi$ is a long-run relationship $k \times k$ matrix and $\Gamma = \Gamma_1 + ... + \Gamma_i$ is a short run relationship.

4 ALGORITHMS AND OPTIMIZATIONS

I 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: VAR and VEC training and validation algorithm.

> train ⟵ Select all from prepared dataset where day < 2007-08-17
> test ⟵ Select all from prepared dataset where day > 2007-08-17
> **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**

Algorithm 2 is modified to interface with R to produce the VAR and VEC models using tsDyn[8] package in R.

In addition to the algorithm implementation with the deep learning framework Keras, a number of modifications to the LSTM and ConvLSTM Algorithm 3 were done: The data was transformed into a supervised learning problem where the data is reorganized into 2 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

---

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

Algorithm 3: LSTM and ConvLSTM training and validation algorithm.

train ⟵ Select all from prepared dataset where day < 2007-08-17
test ⟵ Select all from prepared dataset where day > 2007-08-17
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

---

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's important to carry forward volatility relationships for the lengthy periods of time as discussed in Section 3.5, 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 series. I begin by first scaling the features into a range of -1 and -1 due to the hyperbolic tangent activation function in the LSTM and ConvLSTM hidden layers, before training.

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, I 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 1 hour and 20 minutes on Google's Research Colab[9] GPU environment.

The ConvLSTM was trained for 851 epochs, triggering an early stop after approximately 1 hour and 30 minutes. The training process is visualized in the Figure 8. 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 tradeoff. 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

---

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

loss
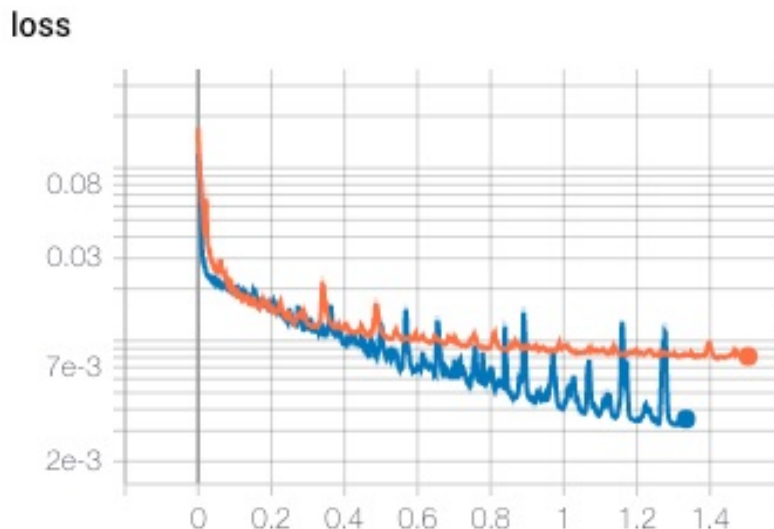
Figure 8: LSTM (blue) vs ConvLSTM (orange) training process with minimizing scaled IV MSE.

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 tradeoff, 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 implied volatility forecasting (Liu, Oosterlee, and Bohte, 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 4. The output vector is a 30-day prediction window for the March, June, September, December month contracts for each of the 20 log-moneyness categories, 10 for the OTM puts, and 10 for the OTM calls.

5   RESULTS AND DISCUSSION

In this section, I 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-day, 30-day, and 90-day windowed forecast. To support the findings, I conduct the (Diebold and Mariano, 1995) (DM) test for the equal predictive accuracy of the models 2) Which of the models can correctly model and predict the IVS dynamics. I conduct a Welch's t-test to test whether the distribution of the predictions is significantly different and test whether the models can capture the skew and the term structure of the IVS. 4 models are compared side-by-side, and the evaluation is based on the 90 trading-day hold out period from 2007-08-17 to 2007-12-31 over a 1-day, 30-day and 90-day prediction windows. In this section, I present findings that convolution operation in the LSTM memory cell, significantly improves modeling and forecasting future implied volatility and outperforms traditional time-series approaches for long-horizon predictions.

5.1   MODEL EVALUATION CRITERIA

In Section 3.1, I 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 non-differenced 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 1370 days. One of the

---

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

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-day, 30-day, and 90-day windows. As the evaluation criteria I choose:

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

where the RMSE metric is expressed in the actual BS IV volatility level, and:

$$MAPE = \frac{100\%}{n} \sum_{t=1}^{n} \left| \frac{\sigma_{actual} - \sigma_{forecast}}{\sigma_{actual}} \right|$$

MAPE is expressed as a real percentage point difference between the actual and the predicted BS-IV. I 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 (Gospodinov, Gavala, and Jiang, 2006; Xiong, Nichols, and Shen, 2016; Culkin and Das, 2017; Luong and Dokuchaev, 2018; Liu, Oosterlee, and Bohte, 2019).

Lastly, I conduct the Welch's t-test to test whether the distribution of the predictions is significantly different across groups, and the DM test for significance of the forecasts at various time horizons.

Table 3: Mean in-sample performance, mean out-of-sample performance and Savitzky-Golay filter smoothed prediction performance on the out-of-sample set.

Model	In Sample		Out-of-Sample		Out-of-Sample(filter smoothed)	
	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
VAR	0.194	102.89	0.041	19.28	0.04	18.87
VEC	0.193	102.88	0.041	19.13	0.04	18.75
LSTM	0.020	3.49	0.068	20.07	0.06	18.60
ConvLSTM	0.045	4.29	0.030	12.28	0.03	11.35

Table 4: Mean out-of-sample RMSE and MAPE grouped by expiration month and contract type.

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.

Expiration Month	type	RMSE				MAPE			
		ConvLSTM	LSTM	VAR	VEC	ConvLSTM	LSTM	VAR	VEC
3	ATM	0.019	0.035	0.024	0.024	8.802	14.56	12.35	12.39
	OTM calls	0.019	0.034	0.023	0.023	8.701	14.77	11.91	11.94
	OTM puts	0.018	0.035	0.023	0.023	7.314	12.97	11.22	11.27
6	ATM	0.016	0.032	0.019	0.02	7.101	12.98	9.915	10.45
	OTM calls	0.015	0.028	0.02	0.02	6.85	11.36	10.34	10.45
	OTM puts	0.016	0.03	0.021	0.021	6.559	11.7	10.21	10.45
9	ATM	0.04	0.057	0.057	0.058	16.38	25.23	29.52	29.59
	OTM calls	0.038	0.055	0.048	0.048	16.99	25.7	26.5	26.59
	OTM puts	0.043	0.057	0.075	0.075	17.61	25.55	40.55	40.65
12	ATM	0.038	0.129	0.041	0.041	18.21	29.94	21.71	22.64
	OTM calls	0.03	0.077	0.034	0.032	14.99	25.73	19.91	17.99
	OTM puts	0.043	0.136	0.049	0.047	18.58	31.64	23.92	23.16

5.2 RESULTS

In Table 3, I summarize the mean in-sample, out-of-sample, and out-of-sample errors with the Savitzky-Golay filter applied. Much lower in-sample RMSE and MAPE for the ConvLSTM and LSTM are observed, which could be attributed to an ability of the neural networks to accurately approximate the BS-IV (Liu, Oosterlee, and Bohte, 2019; Culkin and Das, 2017). Liu, Oosterlee, and Bohte (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 and

Table 5: Average out-of-sample forecast, for 1-day, 30-day, 60-day, and 90-day horizons.

Model	horizon	RMSE	MAPE
VAR	h=1	0.020	6.06
	h=30	0.042	20.46
	h=60	0.044	22.10
	h=90	0.041	19.28
VEC	h=1	0.019	5.95
	h=30	0.042	20.41
	h=60	0.044	21.97
	h=90	0.041	19.13
LSTM	h=1	0.077	38.32
	h=30	0.033	11.96
	h=60	0.041	17.73
	h=90	0.068	20.07
ConvLSTM	h=1	0.025	7.80
	h=30	0.030	11.14
	h=60	0.027	11.18
	h=90	0.030	12.28

Das (2017) found that ANN's can learn to price options with the BS equation and produce very low error rates.

The out of sample forecasts are divided into multiple time horizons, which are summarized in Table 5. The horizon represents a daily mean IV across all log-moneyness and maturity groups. It's observed that LSTM and ConvLSTM have overall higher RMSE and MAPE for a 1-day prediction horizon when compared to VAR and VEC models, and for the LSTM the RMSE and MAPE are reduced for a 30-day forecasting horizon, the errors become substantially larger for a longer forecast. For the ConvLSTM, the error increases slightly from a 1-day forecast but remains near-constant for multiple forecasting horizons. 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 implied volatility 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.
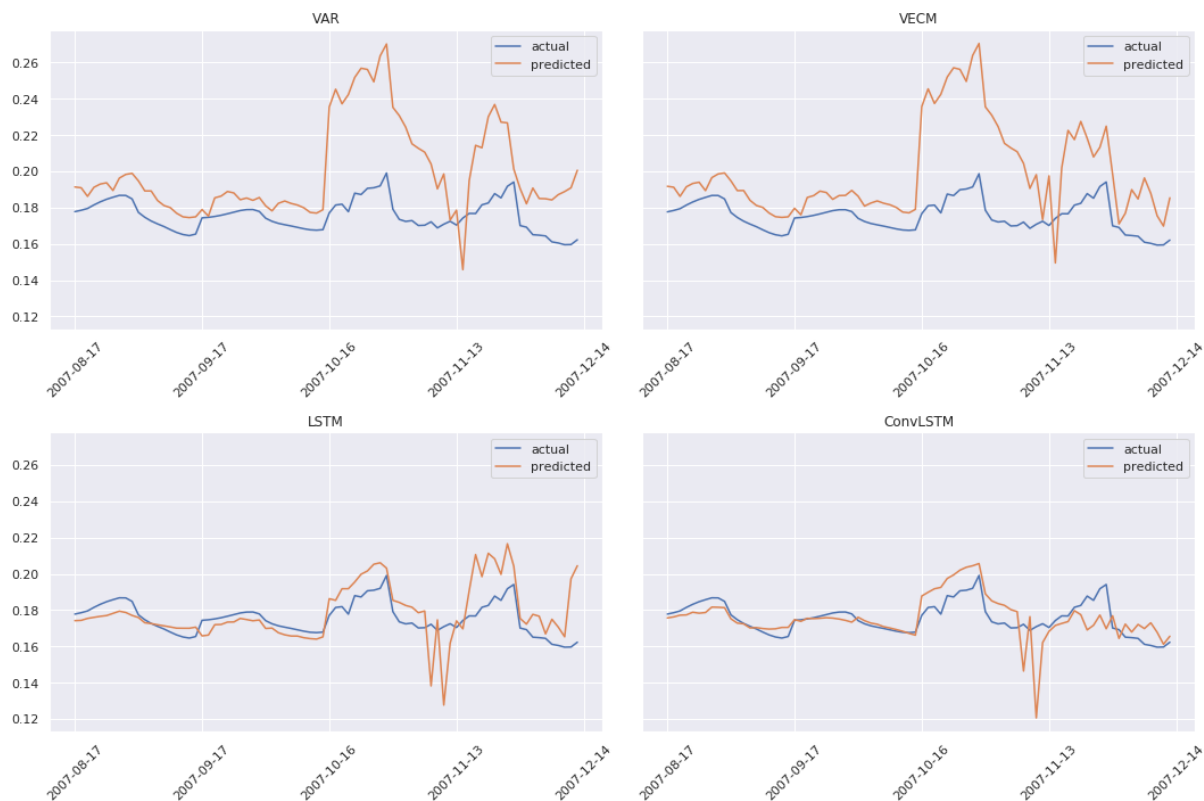
Figure 9: Average out-of-sample predicted IV.

Average predictions against true values are visualized in Figure 9. It is observed that all 4 models seemingly predict the average IV pattern but overestimate the IV jumps and magnitude of these jumps in the longer-term. 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 performs better than the VAR model due to the cointegrated relationships between moneyness groups and maturities that are significantly impacting the performance of the model. Because machine learning models use a different architecture, the output IVS can be produced given the input IVS. The LSTM produces a smoother forecast but experiences a period of instability from 2007-11-1 to 2007-12-14, as seen in Figure 9. This behavior can be observed for other
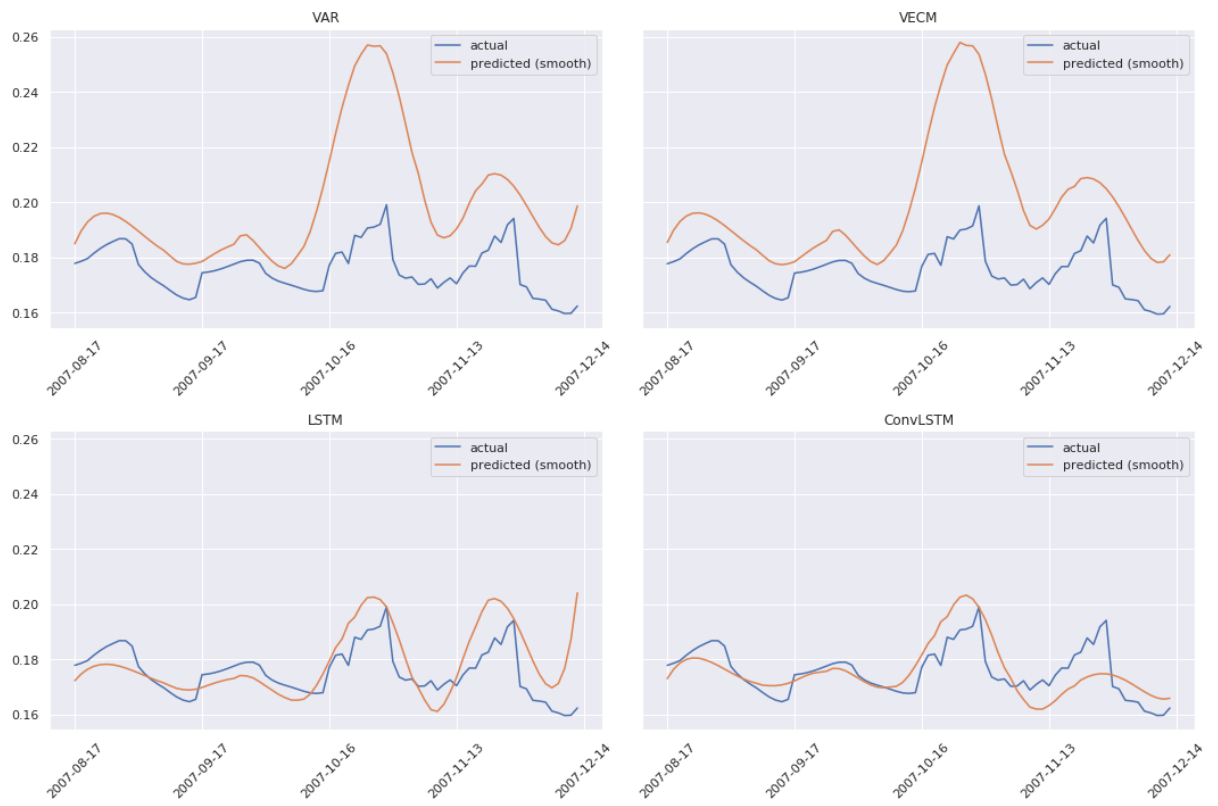
Figure 10: Average out-of-sample predicted Savitzky-Golay smoothed IV.

models as well, but machine learning models show better overall forecasting performance. ConvLSTM produces a lower degree of under-estimation for long-term contracts. ConvLSTM 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. The significance test for the forecasts at multiple time horizons and the side-by-side comparison between the models is summarized in Table 6.

To mitigate the effect of the forecast instability, I 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 10. Similar to that of the unsmoothed forecast, VAR and VEC 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. By applying the Savitzky-Golay smoothing filter, I saw a reduction in RMSE and MAPE cross-sectionally for all models, and summarize the results in Table 3.

I 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 11e shows mean actual IVS is based on the same time-frame of 2007-08-17 to 2007-12-31 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 11e. As discussed in Section 3.4, 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

(a) VAR average predicted IVS.

(b) VEC average predicted IVS

(c) LSTM average predicted IVS

(d) ConvLSTM average predicted IVS

(e) Actual average IVS
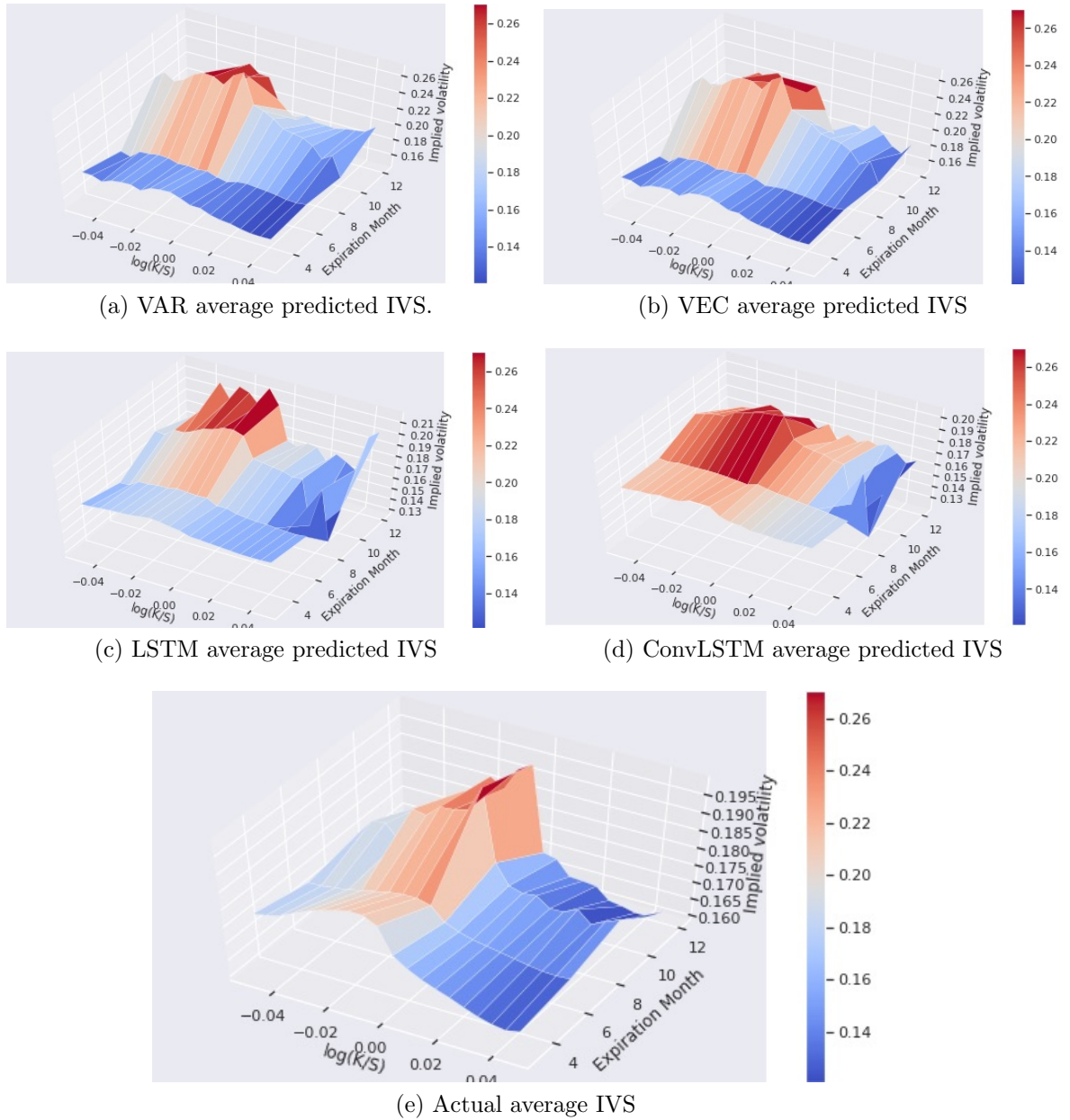
Figure 11: Average and predicted IVS for the 90 day out-of-sample period

(a) VAR average residuals.

(b) VEC average residuals.

(c) LSTM average residuals.

(d) ConvLSTM average residuals.
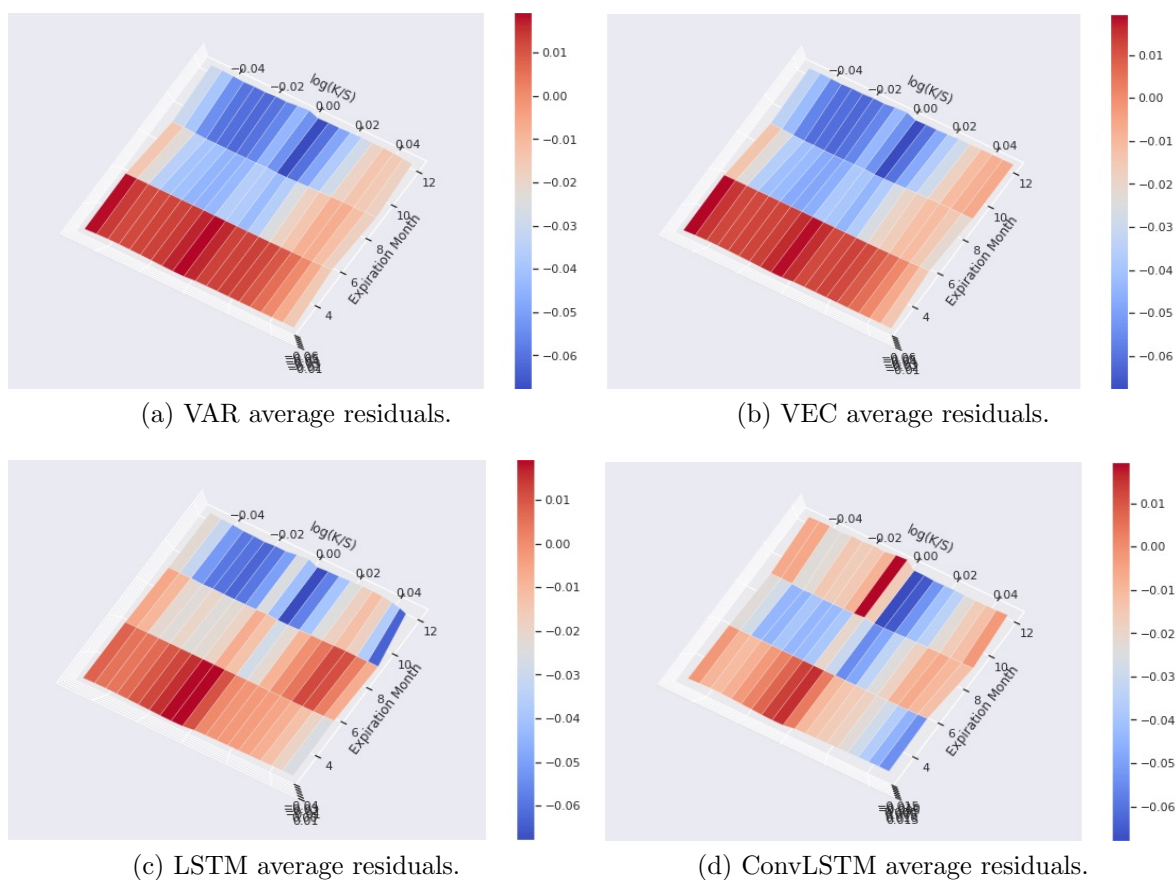
Figure 12: Average IV residual for the 90 day out-of-sample period.

11 shows the mean smoothed predictions for the entire holdout period for each model. Figure 12 shows the residual plotted in 3D, constructed similarly to the IVS, where the residual is a difference between mean true and mean predicted values for each moneyness group.

In general, all 4 models were able to capture the presence of the skew and the term structure component of the IV series. However, VAR, VEC, LSTM, and ConvLSTM tend to overestimate the true IV levels for the near-term contracts and to underestimate the long term contracts. The IVS of the VAR model in Figure 11a, overestimates the short-term IV for the nearby contracts than actual, and a flat skew for the mid-term(Match) contract, when normal IV skew is observed. The degree of overestimation of the IV is significantly larger for the nearby (September and December) contracts by over 0.05 IV points when

visualized in Figure 12a, as well as for the long term (March and June) contracts. VEC model sees similar results and higher forecasting error for the nearby contracts, however when compared to the VAR model, the degree of overestimation is smaller. The VEC model is also able to capture both the skew and the term structure more accurately than the VAR model. This again confirms that the long-run EC can be used to improve the forecasts of the IVS. The LSTM sees a similar pattern to that of the VEC model, where the predicted OTM put IV is often higher than the actual IV level, but the error reduction is achieved through the overall narrower range of the residuals. This could be attributed to an ability of the LSTM to more accurately represent complex non-linear relationships, through the use of the activation and memory gates that can't be modeled using a constant ERC matrix of the VECM model. Similar to VEC and VAR models, the LSTM underestimates the long-term put implied volatility, as well as overestimates the short-term implied volatility, which indicates difficulty of capturing the term structure component of the IVS, similar to that of the VAR and VEC models. The residuals of the LSTM model are smaller for the June contract when compared to VAR and VEC models, the same is observed for the ConvLSTM, which unlike other models predicts higher IV values for the OTM calls of the near terms contracts and consistently predicts lower IV values for the near-the-money puts. However, because of the convolutional layer, the model is able to capture skew better than other 3 models.

In Table 4 I summarize the out-of-sample predictions for each contract month, where similar dynamics to Figure 12a are observed. 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 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 other models.

I conduct the Welch's t-test to compare the mean prediction significance between time-series and machine learning models, and summarize the findings for the LSTM vs VEC in Table 7, ConvLSTM vs VEC in Table 8 and ConvLSTM vs LSTM in Table 9. On average, the LSTM significantly outperforms, $(\alpha = 0.05)$ for all moneyness groups and months, except for a number of groups in the December contracts. I find that ConvLSTM also significantly outperforms the VEC model for all groups and months. Lastly, I compare ConvLSTM vs. LSTM and find mixed results for the significance of the mean forecast window. So to test the significance of the prediction over multiple forecasting horizons, the DM test for predictive accuracy between two models is chosen. I conduct a side-by-side comparison, which is summarized in Table 6. I use a DM test of the R package forecast[11], which implements a modified DM test proposed by Harvey, Leybourne, and Newbold (1997). 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 1 is significantly more accurate than the predictions of the model 2 over a particular time horizon.

I first convert the multivariate forecast into a mean forecast across all log-moneyness groups and expirations, which converts a multivariate series into a uni-variate series, like displayed in Figure 9, and collect standard errors for the forecasts. I then run the test recursively for the time horizons of 1 trading day, 30 trading days, and 90 trading days to obtain the DM summary statistics and the associated p-values.

I confirm that ConvLSTM significantly outperforms VAR for 1 trading, and 90 trading day horizon, as well as the 30-day horizon but at a lower significance level $\alpha = 0.14$, with similar results when compared to the VEC model. ConvLSTM has better predictive accuracy for 1 day and 30-day horizon, but similar accuracy for a full 90 trading day forecast when compared to LSTM. I find that VEC model significantly

---

[11]https://cran.r-project.org/web/packages/forecast/index.html

Table 6: DM Test for 1, 30, 90 trading day horizon.

Rows represent the baseline model and columns represent the model that the baseline model is compared to. Cells contain DM test statistic and p-values of the one-sided DM test.

Model	h	VAR	VEC	LSTM	ConvLSTM
VAR	h=1	-	DM = -2.8103 p-value=0.997	DM = 0.74196 p-value =0.23	DM = -4.891, p-value = 1
	h=30	-	DM = -3.643 p-value=0.9998	DM = 1.1824, p-value =0.1201	DM = -1.0305, p-value = 0.8472
	h=90	-	DM = -0.69006 p-value=0.754	DM = 0.40869, p-value = 0.3419	DM = -4.891, p-value = 1
VEC	h=1	DM = 2.9527, p-value = 0.002014	-	DM = 0.78582 p-value=0.217	DM = -4.8521, p-value = 1
	h=30	DM = 2.4494, p-value =0.008133	-	DM = 1.3725, p-value=0.08668	DM = -1.0232, p-value = 0.8455
	h=90	DM = 4.8521, p-value = 2.584e-06	-	DM = 0.42568, p-value = 0.3357	DM = -4.8521, p-value = 1
LSTM	h=1	DM = -0.74196, p-value = 0.77	DM = -0.78582, p-value = 0.783	-	DM = -3.9166, p-value = 0.9999
	h=30	DM = -1.1824, p-value =0.8799	DM = -1.3725, p-value = 0.9133	-	DM = -3.9166, p-value =0.9999
	h=90	DM = -0.15803, p-value = 0.5626	DM = -0.42568, p-value = 0.6643	-	DM = -0.35655, p-value= 0.6389
ConvLSTM	h=1	DM = 4.891,p-value = 2.213e-06	DM = 4.8521, p-value = 2.584e-06	DM = 3.9166, p-value = 8.786e-05	-
	h=30	DM = 1.0305, p-value = 0.1528	DM = 1.0232, p-value = 0.1545	DM = 3.9166, p-value = 8.786e-05	-
	h=90	DM = 4.8521, p-value = 2.584e-06	DM = 4.8521, p-value =2.584e-06	DM = 0.35655, p-value = 0.3611	-

outperforms VAR model in 1 trading day (p-value = 0.002014), 30 trading days (p-value =0.008133) and 90 days (p-value = 2.584e-06) forecasts, which confirms my earlier findings of significance of the error-correlation terms. When the LSTM is compared to VAR and VEC models, I fail to reject then null hypothesis and conclude that the LSTM and VAR and VEC models have similar predictive accuracy for all 3-time horizons. This is somewhat inconsistent with the findings summarized in Table 5, where I observe much lower RMSE and MAPE for a 30 day and 60-day horizons and might be attributed to the sampling methodology of the forecasting errors (small sample problem).

5.3   CONCLUSION

In this paper, I apply and analyze two new neural network architectures for forecasting time-varying multi-step evolution of the implied volatility surface and benchmark them against traditional time-series VAR and VEC models. I apply a new methodology for encoding IV term-structure as a discrete dimension, which allows to apply two new types of the recurrent neural network architectures for modeling and multi-step forecasting of the IVS. The dataset in this thesis contains SPX call and put options traded between 2002-02-05 and 2007-12-31. I evaluate the out-of-sample forecasting performance on a 90 trading day holdout dataset from 2007-08-17 to 2007-12-31 using root-mean-squared-error (RMSE) and mean-absolute-percentage-error (MAPE). I conduct Welch's t-test and one-sided DM test for statistical significance of the forecasts, between time series and machine learning models for multiple time horizons. Lastly, 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. I 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.

The first research question was to explore whether cointegrated relationships serve a significant role in forecasting implied volatility. I find that including cointegrated relationships, such as having an error correction term (EC) to adjust short term dynamics to their long-term equilibrium, significantly reduces the forecasting error. In particular, I found that including the EC, not only reduced the RMSE and MAPE for the multi-step out-of-sample forecast of the IVS from 0.0441 RMSE to 0.032 RMSE but was also able to capture the dynamics of the IV skew.

The second question was whether recurrent neural network architecture could significantly outperform traditional time series models in the multi-step out-of-sample

forecast of the IVS. I find that ConvLSTM produces the lowest overall out-of-sample error RMSE (0.0301) and MAPE(12.12%) 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. I find that ConvLSTM produces cross-sectionally lower RMSE and MAPE for at-the-money (ATM), out-of-the-money(OTM) calls, and OTM puts for all expiration months when compared to other models. I also specifically find that ConvLSTM produces the lowest RMSE and MAPE for longer-term (March and June) contracts, yielding mean RMSE of 0.0186 and MAPE of 8.27% for March contract, and mean RMSE of 0.0156 and MAPE of 6.84% for June contract. However, similar to previous research, I find that all models experience a higher degree of overestimation when predicting IV for the near-term contracts and underestimation when predicting IV for the long term expiration, which remains to be a challenge for this topic.

The third research question was whether incorporating historical spatial dynamics of implied volatility term structure and implied volatility skew, can significantly improve the multi-step forecast of the IVS. I 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.

The thesis makes multiple contributions to the current research on modeling and forecasting implied volatility. Firstly, the thesis proposes a different methodology for encoding implied volatility term-structure as a separate discrete dimension. Previously, the IVS term-structure and implied volatility 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 discussed in the thesis 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 dataset. Secondly, I apply Convolutional LSTM (ConvLSTM) to the domain of modeling and forecasting IVS. The convolutional neural networks have shown good results in the domain of computer vision, handwriting recognition, and modeling spatial relationships, but hadn't found much application in the domain of modeling and forecasting IVS. Lastly, I 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, I 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 non-parametric IVS. Another way could be training a fully connected neural network similar to that of Liu, Oosterlee, and Bohte (2019) or Culkin and Das (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 implied volatility term-structure as a separate discrete dimension. I 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). Another suggestion would be conducting hyperparameter tuning (such as Grid Search) as discussed by Liu, Oosterlee, and Bohte (2019) for the models, which could significantly improve the results.

APPENDIX

A   STATISTICAL RESULTS

Table 7: LSTM forecast vs VEC forecast Welch's significance tests

month	moneyness	t	p-value
3	-0.0490	-5.3696	0.0000
	-0.0440	-6.5335	0.0000
	-0.0390	-7.3608	0.0000
	-0.0340	-7.8681	0.0000
	-0.0290	-8.0969	0.0000
	-0.0240	-8.0961	0.0000
	-0.0190	-7.9158	0.0000
	-0.0140	-7.6044	0.0000
	-0.0090	-7.2074	0.0000
	-0.0040	-6.7648	0.0000
	0.0010	-6.3104	0.0000
	0.0060	-5.8708	0.0000
	0.0110	-5.4657	0.0000
	0.0160	-5.1095	0.0000
	0.0210	-4.8118	0.0000
	0.0260	-4.3700	0.0000
	0.0310	-3.9521	0.0002
	0.0360	-3.7199	0.0003
	0.0410	-3.6758	0.0004
	0.0460	-3.7695	0.0003

6	-0.0490	-3.9032	0.0002
	-0.0440	-4.4132	0.0000
	-0.0390	-4.9668	0.0000
	-0.0340	-5.5267	0.0000
	-0.0290	-6.0047	0.0000
	-0.0240	-6.5317	0.0000
	-0.0190	-7.5557	0.0000
	-0.0140	-8.0647	0.0000
	-0.0090	-7.3245	0.0000
	-0.0040	-6.7420	0.0000
	0.0010	-6.4237	0.0000
	0.0060	-6.0709	0.0000
	0.0110	-6.0113	0.0000
	0.0160	-6.1951	0.0000
	0.0210	-7.4528	0.0000
	0.0260	-8.8083	0.0000
	0.0310	-9.4584	0.0000
	0.0360	-10.0375	0.0000
	0.0410	-10.7352	0.0000
	0.0460	-11.3451	0.0000
9	-0.0490	-12.0276	0.0000
	-0.0440	-12.7410	0.0000
	-0.0390	-13.5965	0.0000
	-0.0340	-13.4764	0.0000
	-0.0290	-14.4175	0.0000
	-0.0240	-13.5172	0.0000
	-0.0190	-13.5281	0.0000

	-0.0140	-12.8990	0.0000
	-0.0090	-13.0803	0.0000
	-0.0040	-13.2178	0.0000
	0.0010	-12.5916	0.0000
	0.0060	-12.2468	0.0000
	0.0110	-11.3835	0.0000
	0.0160	-10.6296	0.0000
	0.0210	-9.9581	0.0000
	0.0260	-9.7551	0.0000
	0.0310	-9.0176	0.0000
	0.0360	-9.0301	0.0000
	0.0410	-8.1983	0.0000
	0.0460	-5.9014	0.0000
12	-0.0490	-3.9624	0.0001
	-0.0440	-2.5794	0.0115
	-0.0390	-1.8518	0.0673
	-0.0340	-1.3125	0.1927
	-0.0290	-0.7524	0.4538
	-0.0240	-0.5989	0.5507
	-0.0190	-0.5919	0.5554
	-0.0140	-0.6755	0.5011
	-0.0090	-0.8629	0.3905
	-0.0040	-1.1884	0.2378
	0.0010	-1.7183	0.0892
	0.0060	-2.5611	0.0121
	0.0110	-3.8121	0.0003
	0.0160	-5.1533	0.0000

0.0210	-5.3749	0.0000
0.0260	-3.9376	0.0002
0.0310	-1.8210	0.0719
0.0360	-0.1978	0.8436
0.0410	0.7016	0.4847
0.0460	1.1589	0.2496

Table 8: ConvLSTM forecast vs VEC forecast Welch's significance tests

month	moneyness	t	p-value
3	-0.0490	-6.3338	0.0000
	-0.0440	-7.5088	0.0000
	-0.0390	-8.4413	0.0000
	-0.0340	-9.0782	0.0000
	-0.0290	-9.4299	0.0000
	-0.0240	-9.5412	0.0000
	-0.0190	-9.4661	0.0000
	-0.0140	-9.2539	0.0000
	-0.0090	-8.9457	0.0000
	-0.0040	-8.5741	0.0000
	0.0010	-8.1643	0.0000
	0.0060	-7.7357	0.0000
	0.0110	-7.3030	0.0000
	0.0160	-6.8776	0.0000
	0.0210	-6.4686	0.0000
	0.0260	-5.8006	0.0000
	0.0310	-4.9655	0.0000

	0.0360	-4.4849	0.0000
	0.0410	-4.2424	0.0001
	0.0460	-4.2045	0.0001
6	-0.0490	-4.2564	0.0001
	-0.0440	-4.7970	0.0000
	-0.0390	-5.4252	0.0000
	-0.0340	-6.1044	0.0000
	-0.0290	-6.8368	0.0000
	-0.0240	-7.5057	0.0000
	-0.0190	-8.7931	0.0000
	-0.0140	-9.2308	0.0000
	-0.0090	-8.6238	0.0000
	-0.0040	-7.9427	0.0000
	0.0010	-7.2052	0.0000
	0.0060	-6.5461	0.0000
	0.0110	-5.9833	0.0000
	0.0160	-5.7288	0.0000
	0.0210	-6.7578	0.0000
	0.0260	-8.1973	0.0000
	0.0310	-9.3268	0.0000
	0.0360	-10.7155	0.0000
	0.0410	-12.2382	0.0000
	0.0460	-13.4621	0.0000
9	-0.0490	-14.5498	0.0000
	-0.0440	-15.5603	0.0000
	-0.0390	-17.3110	0.0000
	-0.0340	-16.5635	0.0000

	-0.0290	-17.8931	0.0000
	-0.0240	-16.8115	0.0000
	-0.0190	-16.2710	0.0000
	-0.0140	-15.3894	0.0000
	-0.0090	-15.8603	0.0000
	-0.0040	-16.4228	0.0000
	0.0010	-16.6469	0.0000
	0.0060	-17.0883	0.0000
	0.0110	-17.5637	0.0000
	0.0160	-17.7725	0.0000
	0.0210	-18.1808	0.0000
	0.0260	-19.2013	0.0000
	0.0310	-18.4107	0.0000
	0.0360	-17.3913	0.0000
	0.0410	-16.7383	0.0000
	0.0460	-16.3732	0.0000
12	-0.0490	-15.6219	0.0000
	-0.0440	-15.0617	0.0000
	-0.0390	-14.3601	0.0000
	-0.0340	-13.0505	0.0000
	-0.0290	-11.6275	0.0000
	-0.0240	-11.1859	0.0000
	-0.0190	-10.6810	0.0000
	-0.0140	-10.2698	0.0000
	-0.0090	-9.9202	0.0000
	-0.0040	-9.6040	0.0000
	0.0010	-9.2937	0.0000

0.0060	-8.9606	0.0000
0.0110	-8.5750	0.0000
0.0160	-8.1127	0.0000
0.0210	-7.5701	0.0000
0.0260	-6.9837	0.0000
0.0310	-6.4351	0.0000
0.0360	-6.0144	0.0000
0.0410	-5.7569	0.0000
0.0460	-5.6289	0.0000

Table 9: ConvLSTM forecast vs LSTM forecast Welch's significance tests

month	moneyness	t	p-value
3	-0.0490	19.4779	0.0000
	-0.0440	11.3172	0.0000
	-0.0390	5.1866	0.0000
	-0.0340	1.3916	0.1675
	-0.0290	-0.3500	0.7272
	-0.0240	-0.4201	0.6754
	-0.0190	0.7981	0.4269
	-0.0140	2.9018	0.0046
	-0.0090	5.4814	0.0000
	-0.0040	8.1794	0.0000
	0.0010	10.7433	0.0000
	0.0060	13.0323	0.0000
	0.0110	14.9883	0.0000
	0.0160	16.5972	0.0000

	0.0210	17.8581	0.0000
	0.0260	18.8654	0.0000
	0.0310	19.5299	0.0000
	0.0360	19.7532	0.0000
	0.0410	18.4332	0.0000
	0.0460	17.2811	0.0000
6	-0.0490	14.7838	0.0000
	-0.0440	12.7852	0.0000
	-0.0390	11.4371	0.0000
	-0.0340	9.2359	0.0000
	-0.0290	6.7930	0.0000
	-0.0240	4.4499	0.0000
	-0.0190	2.9274	0.0039
	-0.0140	3.3891	0.0009
	-0.0090	5.1916	0.0000
	-0.0040	2.2904	0.0232
	0.0010	2.4834	0.0140
	0.0060	2.8968	0.0043
	0.0110	4.2217	0.0000
	0.0160	5.5032	0.0000
	0.0210	3.9640	0.0001
	0.0260	7.9040	0.0000
	0.0310	7.1076	0.0000
	0.0360	2.5655	0.0117
	0.0410	-3.9339	0.0001
	0.0460	-9.2072	0.0000
9	-0.0490	-14.5065	0.0000

-0.0440	-14.3058	0.0000
-0.0390	-17.7877	0.0000
-0.0340	-19.6364	0.0000
-0.0290	-19.6904	0.0000
-0.0240	-25.1214	0.0000
-0.0190	-27.2626	0.0000
-0.0140	-27.6970	0.0000
-0.0090	-26.9621	0.0000
-0.0040	-26.1503	0.0000
0.0010	-21.1726	0.0000
0.0060	-15.0577	0.0000
0.0110	-6.4500	0.0000
0.0160	4.1745	0.0000
0.0210	21.5505	0.0000
0.0260	29.4432	0.0000
0.0310	30.6871	0.0000
0.0360	29.8073	0.0000
0.0410	27.2271	0.0000
0.0460	23.8184	0.0000

12	-0.0490	20.4734	0.0000
	-0.0440	18.2313	0.0000
	-0.0390	14.4601	0.0000
	-0.0340	12.7698	0.0000
	-0.0290	9.8831	0.0000
	-0.0240	8.1081	0.0000
	-0.0190	6.6584	0.0000
	-0.0140	5.4527	0.0000

-0.0090	4.4849	0.0000
-0.0040	3.7708	0.0002
0.0010	3.3527	0.0010
0.0060	3.3032	0.0012
0.0110	3.7230	0.0003
0.0160	4.7293	0.0000
0.0210	6.4315	0.0000
0.0260	8.9002	0.0000
0.0310	12.1120	0.0000
0.0360	15.7757	0.0000
0.0410	18.9909	0.0000
0.0460	20.3883	0.0000

REFERENCES

Alexander, Carol (2004). "Normal mixture diffusion with uncertain volatility: Modelling short- and long-term smile effects". In: *Journal of Banking & Finance* 28(12), 2957â2980. ISSN: 03784266. DOI: 10.1016/j.jbankfin.2003.10.017.

Black, Fischer and Myron Scholes (1973). "The Pricing of Options and Corporate Liabilities". In: *Journal of Political Economy* 81(3), pp. 637–54. URL: https://EconPapers.repec.org/RePEc:ucp:jpolec:v:81:y:1973:i:3:p:637-54.

Bollerslev, Tim (1986). "Generalized autoregressive conditional heteroskedasticity". In: *Journal of Econometrics* 31(3), pp. 307–327. URL: https://ideas.repec.org/a/eee/econom/v31y1986i3p307-327.html.

Chalamandaris, Georgios and Andrianos E. Tsekrekos (2011). "How important is the term structure in implied volatility surface modeling? Evidence from foreign exchange options". In: *Journal of International Money and Finance* 30(4), 623â640. ISSN: 02615606. DOI: 10.1016/j.jimonfin.2011.02.001.

Corsi, F. (2008). "A Simple Approximate Long-Memory Model of Realized Volatility". In: *Journal of Financial Econometrics* 7(2), 174â196. ISSN: 1479-8409, 1479-8417. DOI: 10.1093/jjfinec/nbp001.

Culkin, Robert and Sanjiv R Das (2017). "Machine Learning in Finance: The Case of Deep Learning for Option Pricing". In: p. 15.

Diebold, Francis and Roberto Mariano (1995). "Comparing Predictive Accuracy". In: *Journal of Business & Economic Statistics* 13(3), pp. 253–63. URL: https://EconPapers.repec.org/RePEc:bes:jnlbes:v:13:y:1995:i:3:p:253-63.

Dumas, Bernard, Jeff Fleming, and Robert E. Whaley (1998). "Implied Volatility Functions: Empirical Tests". In: *Journal of Finance* 53(6), pp. 2059–2106. URL: https://EconPapers.repec.org/RePEc:bla:jfinan:v:53:y:1998:i:6:p:2059-2106.

Dupire, Bruno (1994). "Pricing with a Smile". In: *Risk* 7(1), 18â20.

Engle, Robert F (1982). "Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflation". In: *Econometrica* 50(4), pp. 987–1007. URL: https://ideas.repec.org/a/ecm/emetrp/v50y1982i4p987-1007.html.

Eraker, BjÃ²rn, Michael Johannes, And Nicholas Polson (2003). "The Impact of Jumps in Volatility and Returns". In: *The Journal of Finance*, p. 32.

Gospodinov, Nikolay, Athanasia Gavala, and Deming Jiang (2006). "Forecasting volatility". In: *Journal of Forecasting* 25(6), 381â400. ISSN: 0277-6693, 1099-131X. DOI: 10.1002/for.993.

Greff, Klaus et al. (2017). "LSTM: A Search Space Odyssey". In: *IEEE Transactions on Neural Networks and Learning Systems* 28(10). arXiv: 1503.04069, 2222â2232. ISSN: 2162-237X, 2162-2388. DOI: 10.1109/TNNLS.2016.2582924.

Hansen, Peter R. and Asger Lunde (2005). "A forecast comparison of volatility models: does anything beat a GARCH(1,1)?" In: *Journal of Applied Econometrics* 20(7), 873â889. ISSN: 0883-7252, 1099-1255. DOI: 10.1002/jae.800.

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

Hochreiter, Sepp and JÃ¼rgen Schmidhuber (1997). "Long Short-Term Memory". In: *Neural Computation* 9(8), 1735â1780. DOI: 10.1162/neco.1997.9.8.1735.

Homescu, Cristian (2011). "Implied Volatility Surface: Construction Methodologies and Characteristics". In: *arXiv:1107.1834 [q-fin]*. arXiv: 1107.1834. URL: http://arxiv.org/abs/1107.1834.

Hosker, James et al. (2018). "Improving VIX Futures Forecasts using Machine Learning Methods". In: 1(4), p. 91.

Hull, John C and Alan White (1987). "The Pricing of Options on Assets with Stochastic Volatilities". In: *Journal of Finance* 42(2), pp. 281–300. URL: https://EconPapers.repec.org/RePEc:bla:jfinan:v:42:y:1987:i:2:p:281-300.

Johansen, Soren (1991). "Estimation and Hypothesis Testing of Cointegration Vectors in Gaussian Vector Autoregressive Models". In: *Econometrica* 59(6), p. 1551. ISSN: 00129682. DOI: 10.2307/2938278.

Kwiatkowski, Denis et al. (1992). "Testing the null hypothesis of stationarity against the alternative of a unit root". In: *Journal of Econometrics* 54(1â3), 159â178. ISSN: 03044076. DOI: 10.1016/0304-4076(92)90104-Y.

Liu, Shuaiqiang, Cornelis W. Oosterlee, and Sander M. Bohte (2019). "Pricing options and computing implied volatilities using neural networks". In: *arXiv:1901.08943 [cs, math, q-fin]*. arXiv: 1901.08943. URL: http://arxiv.org/abs/1901.08943.

Luo, Rui et al. (2018). "A Neural Stochastic Volatility Model". In: *arXiv:1712.00504 [cs, q-fin, stat]*. arXiv: 1712.00504. URL: http://arxiv.org/abs/1712.00504.

Luong, Chuong and Nikolai Dokuchaev (2018). "Forecasting of Realised Volatility with the Random Forests Algorithm". In: *Journal of Risk and Financial Management* 11(4), p. 61. ISSN: 1911-8074. DOI: 10.3390/jrfm11040061.

Maddala, G S and Shaowen Wu (1999). "A Comparative Study of Unit Root Tests with Panel Data and a New Simple Test". In: p. 22.

Majmudar, Utkarsh and Arnab Banerjee (2004). "VIX Forecasting". In: *SSRN Electronic Journal*. DOI: 10.2139/ssrn.533583.

Orosi, Greg (2012). "Empirical performance of a spline-based implied volatility surface". In: *Journal of Derivatives* 18, p. 16.

Park, Hyejin, Namhyoung Kim, and Jaewook Lee (2014). "Parametric models and non-parametric machine learning models for predicting option prices: Empirical comparison study over KOSPI 200 Index options". In: *Expert Systems with Applications* 41(11), 5227â5237. ISSN: 09574174. DOI: 10.1016/j.eswa.2014.01.032.

Rubinstein, Mark (1985). "Nonparametric Tests of Alternative Option Pricing Models Using All Reported Trades and Quotes on the 30 Most Active CBOE Option Classes from August 23, 1976 Through August 31, 1978". In: p. 27.

Samsudin and Azhar Mohamad (2016). "Implied Volatility Forecasting in the Options Market: A Survey". In: *Sains Humanika* 8, pp. 9–18. DOI: 10.11113/sh.v8n2.721.

Shi, Xingjian et al. (2015). "Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting". In: p. 9.

Sinclair, Euan (2013). *Volatility trading.* John Wiley & Sons, Inc.

Xiong, Ruoxuan, Eric P. Nichols, and Yuan Shen (2016). "Deep Learning Stock Volatility with Google Domestic Trends". In: *arXiv:1512.04916 [q-fin].* arXiv: 1512.04916. URL: http://arxiv.org/abs/1512.04916.

Zumbach, Gilles (2009). "Volatility forecasts and the at-the-money implied volatility: a multi-components ARCH approach and its relation with market models". In: *arXiv:0901.2275 [q-fin].* arXiv: 0901.2275. URL: http://arxiv.org/abs/0901.2275.