Deep Learning Based Dynamic Implied Volatility Surface

Daniel Bloch [*], Arthur Book [†]
Quant Finance
Working Paper

12th of October 2021
Revised version : 1.0.0

Abstract

We propose to model the dynamics of the entire implied volatility surface (IVS) multi-step ahead by letting the parameters of a stochastic volatility model with an explicit expression for the smile be dynamically evolved. We assume that these model parameters are stochastic processes driven by some explanatory variables and use deep learning to infer their dynamics. For simplicity, we focus on the SVI model, let each model parameter have a term-structure, and learn to predict the future values of these parameters. The explanatory variables are the time series of the fitted model parameters and the time series of the forward prices. To capture the spatiotemporal relations of the IVS, we stack multiple convolutional LSTM (ConvLSTM) layers and form an encoding-forecasting structure, getting a network model capable of understanding the spatiotemporal relationships between strikes and time-to-maturities. However, this model is very sensitive to the term-structure of the model parameters and requires a very fine grid of volatility to converge. Thus, we simplify the model by considering a kernel of size one. The future smiles are reconstructed by using the parametric smile representation, where each parameter is replaced by its estimated value. We can then use the forecasted volatility surface for pricing and hedging options, performing risk analysis, as well as for volatility trading. We explore the performance of our model against a naive strategy by forecasting the volatility surface on the S&P 500 option prices several steps ahead, and computing some measures of accuracy. On average, our model systematically outperforms the naive approach at predicting long term forecasts for short to mid-range maturities. This shows that the dynamics of the IVS are dominated by trend and mean reversion, hence predictable.

Deep Networks, Convolutional LSTM, Dynamic Implied Volatility Surface, Option Pricing and Hedging, Risk Analysis, Volatility Trading

1 Introduction

Since the seminal article of Black et al. [1973] on option pricing, a large number of parametric and non-parametric volatility models were developed to capture the dynamics of the implied volatility surface (IVS). For instance, Harvey et al. [1992] propsed a linear model to predict the implied volatility surface. Campa et al. [1995] showed that implied volatilities are a function of time-to-maturity, while Dumas et al. [1998] let the volatilities be a function of the strike price and time-to-maturity. In the financial literature, some authors analysed the market and devised regimes of volatility (see Derman [1999], Daglish et al. [2006]), while others analysed the properties of a pricing model to deduce the dynamics of the smile inferred by

[*]Quant Finance, dan@quantfin.eu
[†]The author is expressing his own views which are not representative of his employer.

that model (see Hagan et al. [2002]). The former are deterministic implied volatility models, defined by assuming that either the per-delta or the per-strike implied volatility surface has a deterministic evolution. The latter assume a certain model for the joint dynamics of the volatility surface and the underlying asset. In the framework of non-parametric models, some authors such as Ait-Sahali [1998], Jackwerth [1999], proposed non-parametric models for the estimation of the risk-neutral marginal densities. These methods require a kernel regression of the market prices on some input variables. See Bondarenko [2003] for a review of these methods. However, non-parametric methods are less adapted to the extrapolation problem than the parametric ones, and they tolerate less control over the generated volatility surface.

Given a value for the spot price, the above parametric volatility models describe the way the volatility surface evolves over a short period of time. However, the estimated model parameters are highly unstable over time, and must therefore be modelled. Even though the implied volatility surface (IVS) characterises the agents belief of future evolution of the stock price returns, it is evident that today's market prices do not provide us with the right future anticipations of the stock price process. This is because the implied volatility surface is neither stationary nor Markovian, but stochastic (see Dumas et al. [1998], Cont et al. [2002b], Goncalves et al. [2006], Fengler et al. [2007]).

Given the high dimensionality of the problem, in order to estimate the future (short term forecast) behaviour of the volatility surface, one solution is to perform dimensionality reduction by projecting the surface into some basis functions (or risk factors) and assume they are driven by some dynamics. These stochastic risk factors form a vector basis for the implied volatility surface (IVS). Examples in the literature include the first three eigenmodes of a principal components analysis (PCA) decomposition, the parameters of a polynomial model, or, any other model parameters from a stochastic volatility model. Thus, the dynamics of the IVS are given by that of the volatility risk factors, which can be estimated with a statistical model, or, assumed to follow a specific stochastic process. For instance, Skiadopoulos et al. [2000] popularised the use of principal components analysis (PCA) in order to infer the dynamics of the volatility surface. Following this approach, Cont et al. [2002b] assumed an Ornstein-Uhlenbeck process for the dynamics of the factors to explain the deformation of the volatility surface. Goncalves et al. [2006] combined a cross-sectional approach to fitting a quadratic IVS (see Dumas et al.) with the application of vector autoregression (VAR) models to the (multivariate) time series of estimated cross-sectional coefficients. This is a two-step method where in the first step the IVS is expressed as a function of the moneyness and time-to-maturity, while in the second step, a time series model is constructed for the estimated coefficients of cross-sectional regression. Bernales et al. [2014] followed the same approach to study the predictability of the implied volatility surface of US stock options. Bedendo et al. [2009] and Chen et al. [2016] used a nonlinear Kalman filter as a combined approach to the two-steps model, which resulted in a more robust estimation of model parameters. Alexander et al. [2010] proposed a stochastic local volatility model where the volatility surface is fit with cubic polynomial expressed in terms of the forward moneyness. To account for the fact that the implied volatility was non-Markovian, they modified the IV sensitivity to the spot price, by letting the correction term be dependent on the regime of volatility observed over a recent period of historical data. At each time t, the correction term is computed ex-post the delta hedging error on a sample of option prices. Ladokhin [2009] focused on two different approaches to model the dynamics of the IV surfaces, one applicable to the Cubic and the Spline model, and the other one used for SABR models.

Rather than explicitly defining some stochastic processes to model the risk factors, or, having to rely on a statistical model to infer their dynamics, one approach is to let the risk factors be function of some explanatory variables. Bloch et al. [2002] assumed the parameters of a quadratic smile to be led by the spot process and some noise holding the whole market risk. This approach was further extended by Bloch et al. [2003] and Bloch [2012] to perform quantitative options relative value trading. In that setting, one can either use a statistical model, or, a neural network, to capture the relation between the volatility risk factors and their explanatory variables. By definition, the implied volatility map depends on time, the spot price, the strike and the maturity, denoted $(t, S_t; K, T)$, so that the spot price is the main driving factor of the IV surface. This result was confirmed by several statistical analysis (see Abergel et al. [2012], Romo [2012]). To a less extent, it is also driven by external factors, such as stochastic volatility and jumps (see Bates [1996], Bakshi et al. [1997]), and market factors, such as volume and return on the market portfolio. Further, Hasler et al. [2018] showed that the dynamics of the IVS was subject to economic and financial conditions. Using these results, Bloch [2020] let the explanatory variables be the spot price, the time-to-maturity, the trading volume, the estimated volatility, and financial indicators like the VIX. He proposed the NN-DIVS model which uses neural networks to capture the relation between the volatility risk factors and their explanatory variables. He modelled the dynamics of the IV surfaces by letting the parameters of a polynomial model, or, a stochastic volatility model with an explicit expression for the smile such as the SVI and the SABR model, be statistically evolved, by considering one neural network per parameter (risk factor). The volatility surface is reconstructed by using the polynomial model, or the parametric smile representation, where each parameter is replaced by a trained network.

Alternatively, non-parametric models such as neural networks (NNs) were developed, using market data to estimate the implicit stochastic process driving the spot price and its relationship with contingent claims. Some authors such as Malliaris et al. [1993], Hutchinson et al. [1994], Anders [1996], Yao et al. [2000b], Gencay et al. [2003], Bennell et al. [2004], Stafford [2018], among others, suggested using supervised learning for mapping market factors to market prices (or model prices) in view of estimating these prices when the factors evolve over time. Here, the explanatory variables considered are the underlying asset price, the traded volume, and some historically estimated volatility, so that the dynamics of the IVS is driven by all these factors and not just the spot price. Some authors also used neural networks for calibration purposes, by learning the map from the model parameters to the shapes of the full implied volatility surface. The input is the volatility surface and the output is its associated set of model parameters. Some recent work on model calibration with neural networks includes Hernandez [2017], Stone [2018], Bayer et al. [2018], McGhee [2018], Horvath et al. [2019], among others. Hosker et al. [2018] compared the performance of three existing forecasting models on the one-month VIX futures contract, 3/5 days ahead, against six different supervised learning models, including recurrent neural networks such as LSTM. Kim et al. [2021] used a self-attention mechanism of a transformer network to generate a smooth and robust volatility surface under the SABR model.

When it comes to predicting the dynamics of the implied volatility surface (IVS) with machine learning, one approach is to focus on the traded option prices and estimate their dynamics with historical data. For instance, Huynh [2018] showed that the multi-layer perceptron neural network was performing better than a dynamic factor autoregressive model and a random walk model at modelling and forecasting the implied volatility surface on the Swedish OMXS 30 index option market. He found that it was more difficult to predict in-the-money (ITM) options than out-of-the-money (OTM) ones, and that predicting short-term options was a more complicated task. Sun [2018] explored the abilities of different machine learning models to predict changes in the implied volatility surface over a one day to one month horizon in options on the S&P 500 index (SPX). Chen et al. [2019] introduced an attention mechanism into an LSTM model combined with a fully connected network layer to predict the IVS one step ahead. The forecast is on the traded discrete points of the implied volatility smile surface (45 points) with moneyness in the range $[0.80, 1.20]$ and maturities March, June, December, 18 and 24 months. At every time t, the explanatory variables are these observed implied volatilities plus a weekly and monthly average of each of the 45 points. Thus, the input feature is a matrix of size 45×3 and the target value is a matrix of size 45×1. The authors used the MSE on the training set to show convergence. Medvedev [2019] modelled the entire implied volatility surface using recurrent neural network architectures. He proposed a Convolutional LSTM (ConvLSTM) model to produce multivariate and multi-step forecasts of the S&P 500 implied volatility surface for a 1-day, 30-day, and 90-day horizon. The author considered four fixed quarterly contracts: March, June, September, December and used splicing technique [1] to construct the volatility surface. It led to unbalanced panel with unequal number of time series observations for each contract. As a solution, he considered a pivoting operation which requires the interpolation of a large number of missing volatility values. However, focusing only on the traded option prices is problematic in presence of sparse or erroneous data. Further, when performing market making on volatility, or, pricing and hedging an OTC option, we are exposed to inferring the probability distribution of returns from regions with few or no observations. This suppose synthetically generating data when market prices are missing (see Orosi [2012]). Ackerer et al. [2020] proposed neural networks to correct prior standard IVS methods in order to fit and interpolate-extrapolate implied volatility surface in such cases. They guarantee the absence of arbitrage opportunities by penalising the loss using soft constraints during training. Yet, this approach does not consider the dynamics of the volatility surface.

We address the above problems by evolving the entire implied volatility surface multi-step ahead for all strikes and maturities. We deal with the high dimensionality of that problem by projecting the surface into some basis functions as described above, getting a fixed grid volatility surface. For simplicity of exposition, we focus on the SVI model and let each model parameter have a term-structure. These model parameters are fitted to historical data for a fix set of strikes and a fix set of time-to-maturities. We propose to use an encoder-decoder approach to learn the dynamics of the implied volatility surface by learning the dynamics of these model parameters. We explore different Deep Recurrent Network architectures to represent the explanatory variables in a context vector which becomes the input of a Deep Network. However, as the spatiotemporal relations of the IVS are lost, we stack multiple convolutional LSTM (ConvLSTM) layers and form an encoding-forecasting structure, getting a network model capable of understanding the spatiotemporal relationships between strikes and time-to-maturities. Unfortunately, this model is very sensitive to the term-structure of the model parameters and requires a very fine grid of volatility to converge. Thus, we simplify the model by considering a kernel of size one. The volatility surface is reconstructed by using the parametric smile representation, where each parameter is replaced by its predicted value. Knowing the dynamics of the volatility surface, we can use it for pricing and hedging options, performing risk analysis and volatility trading. Once the networks have been trained, we use the SVI function to infer the volatility for a given strike and maturity and plug that volatility in the BS-formula to recover the market price. We explore the performance of our model against a naive strategy by forecasting the volatility surface on the S&P 500 option prices several steps ahead, and computing a measure of accuracy. In the cases of long term forecast for short to mid-range maturities we beat the benchmark.

2 Modelling the volatility surface

2.1 The implied volatility surface

2.1.1 Definition

The implied volatility (IV) is a mapping from time, spot prices, strike prices and expiry days to $\mathbb{R}^+$

$$\Sigma : (t, S_t, K, T) \to \Sigma(t, S_t, K, T) \tag{2.1}$$

Given the option price $C(t, S_t, K, T)$ at time t for a strike K and a maturity T, the market implied volatility satisfies

$$C(t, S_t, K, T) = C_{BS}(t, S_t, K, T; \Sigma(K, T))$$

where $C_{BS}(t, S_t, K, T; \sigma)$ is the Black-Scholes formula (see Black et al. [1973]) for a call option with volatility σ. In practise, we can only observe a few market prices from standard strikes and maturities with wide or

[1]The contracts are stitched head to toe on expiration to form continuous series.

narrow spreads depending on the liquidity on the market and the volume traded. As a result, the market is incomplete and there are more than one acceptable price (volatility) surface satisfying the no-arbitrage conditions. Hence, multiple risk-neutral distributions can fit the option prices so that one needs some additional criteria to generate a unique probability distribution function (pdf). To do so, one can either impose a functional form to the probability distribution and estimate its parameters using option data, or one can choose non-parametric methods obtaining perfect fit to market data. Since there is a one-to-one relation between the implied volatility surface (IVS) and the pdf, we will focus on modelling the IVS.

2.1.2 Constraints of no-arbitrage

The shape of the implied volatility surface (IVS) is constrained by absence of arbitrage, formally leading to the existence of no-arbitrage conditions (see Durrleman [2003], Lee [2005]). To summarise, the behaviour of the IVS is constrained as strikes tend to infinity (see Fengler [2005] for more detailed results) and as time-to-maturity tends to infinity (see Tehranchi [2009] for mode details).
For our purpose, we just need to introduce two of those limit behaviour, one for calendar arbitrage and one for infinite maturities. We define the forward moneyness as $\eta = \frac{K}{F(t,T)}$, where $F(t,T)$ is the forward price seen at time t for the maturity T, and let the total variance be given by $\nu^2(\eta,T) = \Sigma^2(\eta,T)(T-t)$. Assuming deterministic rates and dividend yield, if $\nu^2(\eta,T_i)$ is a strictly increasing function, for $i = 1,2$ with $T_1 < T_2$, then there is no calendar arbitrage.
Further, the IV surface flattens for infinitely large expiries. Given the forward log-moneyness $\overline{\eta} = \ln \eta$ (with no repo rate), for any $M > 0$ Tehranchi showed that

$$\lim_{T\to\infty} \sup_{\overline{\eta}_1,\overline{\eta}_2\in[-M,M]} |\Sigma(\overline{\eta}_2,T) - \Sigma(\overline{\eta}_1,T)| = 0$$

In fact, the flattening of the implied volatility smile is a universal property of all martingale models.

2.2 The choice of a volatility model

In the framework of parametric models, a natural choice is to perform a Taylor expansion up to the second order of the implied volatility surface around the money forward level (see Malz [1997], Daglish et al. [2006]). In general, the implied volatility is modelled with a functional form of the smile around the money forward $\eta = \frac{K}{S}$. For instance, Lewis [2000] described a second order polynomial for the smile as a function of log-moneyness $\overline{\eta} = \ln \frac{K}{S}$ given by

$$\Sigma(t,S_t;K,T) = a_0 + a_1\overline{\eta} + a_2(\overline{\eta})^2 + \epsilon \tag{2.2}$$

where a_0 is the intercept/level coefficient, a_1 is the skew, and a_2 is the curvature of the smile in the moneyness dimension, and ϵ is some external noise. In general, these parameters are estimate by ordinary least square (OLS). In order to avoid volatility explosion in the wings and satisfy the no-arbitrage conditions, this functional form is modified for high and low strikes. Among the polynomial models describing the whole volatility surface with one equation are the Cubic model and the Spline model. In the former the IV is a cubic function of moneyness $\overline{\eta}$ (or the time adjusted moneyness $\widehat{\eta} = \frac{1}{\sqrt{\tau}} \ln \frac{K}{S}$) and a quadratic function of time to expiry $\tau = T - t$. That is,

$$\Sigma(t,S_t;K,T) = a_0 + a_1\overline{\eta} + a_2(\overline{\eta})^2 + a_3(\overline{\eta})^3 + a_4\tau + a_5\tau^2 + \epsilon$$

where a_i, for $i = 0,..,5$, are parameters to be estimated. Dumas et al. [1998] assumed a quadratic model by setting a_3 to zero. Note, Goncalves et al. [2006] assumed that relation to hold for the logarithm of the volatility surface, that is,

$$\ln\Sigma(t,S_t;K,T) = a_0 + a_1\widehat{\eta} + a_2(\widehat{\eta})^2 + a_3\tau + a_4\tau\widehat{\eta} + \epsilon$$

where a_4 describes possible interactions between the moneyness and the time-to-maturity dimensions.

Alternatively, one can fit with little control a parametric form for the implied volatility derived from a model which is usually the result of an asymptotic expansion of a stochastic volatility model. For example, the SABR model of Hagan et al. [2002], or, the Stochastic Volatility Inspired (SVI) model introduced by Gatheral [2004], assume some behaviour of the underlying asset and connections with the values of the implied volatility. In the FX market, Castagna et al. [2007] proposed the Vanna-Volga (VV) method to construct the whole smile for a given maturity where volatilities are provided in terms of the Delta of the basic options. These models fit a wide range of smile patterns, both from empirical observations and those arising from several jump-diffusion models. However, being time slice parametric models, they present a number of difficulties when interpolating through time. For instance, the SABR model can be used on the whole volatility surface: one fit a SABR skew for each observed time to expiration, and then interpolate the values of implied volatility for any arbitrary. In that setting the model is called PSABR. However,direct interpolation and extrapolation of implied volatility surfaces does not guarantee a resulting smooth risk-neutral density, hence a proper local volatility surface.

For illustration purpose we follow the notation by Roper [2010] and give the formula for the SVI model. We let $x = \ln\left(\frac{K}{F(t,T)}\right)$ (this is $\overline{\eta}$ above), where $F(t,T)$ is the forward price, and define the time scaled implied volatility as

$$\begin{aligned} \Xi : \mathbb{R} \quad \times & \quad [0,\infty) \to [0,\infty] \\ & (x,T) \to \sqrt{T-t}\Sigma(F(t,T)e^x, T) \end{aligned} \tag{2.3}$$

That is, $\Xi(x,T) = \sqrt{\omega(\overline{\eta},T)}$ is the square root of the total variance $\omega(K,T) = (T-t)\Sigma^2(K,T)$. In that setting, the parametrisation of the SVI follows

$$\Xi^2_{SVI}(x,\tau) = a + b\Big(\rho(x-m) + \sqrt{(x-m)^2 + \sigma^2}\Big) \tag{2.4}$$

where $a \in \mathbb{R}$, $b \geqslant 0$, $|\rho| \leqslant 1$, $m \in \mathbb{R}$, and $\sigma > 0$. The right and left asymptotes are $\Xi^2_{SVI,RL}(x,\tau) = a \pm b(1 \pm \rho)(x-m)$, respectively. We have the condition $a + b\sigma\sqrt{1-\rho^2} \geqslant 0$ for the total variance Ξ^2_{SVI} to be positive. Roper [2010] showed that this volatility parametrisation was not arbitrage free for $x \in (-1,-0.5)$. This is not a problem as we will only consider cases where $x > -0.5$.
Changes in the parameters have the following effects:

- Increasing a increases the general level of variance, a vertical translation of the smile.
- Increasing b increases the slopes of both the put and call wings, tightening the smile.
- Increasing the correlation ρ decreases (increases) the slope of the left (right) wing, a counter-clockwise rotation of the smile.
- Increasing m translates the smile to the right.
- Increasing the volatility σ reduces the at-the-money (ATM) curvature of the smile.

An example of a set of parameters is: $a = 0.04$, $b = 0.4$, $\sigma = 0.1$, $\rho = -0.4$, and $m = 0.1$.

2.3 Modelling the dynamics of the volatility surface

We let $\mathcal{K} = \{K_1, ..., K_I\}$ be the set of strikes for the fixed strikes K_i, $i = 1, ..., I$. Given the time t, we let $\mathcal{T} = \{T_1 - t, ..., T_J - t\}$ be the set of time-to-maturities for the fixed maturities T_j, $j = 1, ..., J$. We assume a parametric volatility model to generate the volatility surface (see Section (2.2)), and consider one set of model parameters per fixed maturity T_j, $j = 1, ..., J$. We let $\mathcal{M}_t = \{M_{1,t}^{\mathcal{T}}, ..., M_{P,t}^{\mathcal{T}}\}$ be the set of model parameters for a given parametric volatility model seen at time t with time-to-maturities $\mathcal{T}$. We further let the p-th model parameter, $M_{p,t}$, have a term-structure denoted by $M_{p,t}^{\mathcal{T}} = \{M_{p,t}^{T_1-t}, ..., M_{p,t}^{T_J-t}\}$ for $p = 1, ..., P$. We can also represent the model parameters in space for a fixed time-to-maturity $T_j - t$ by $M_t^{T_j-t} = \{M_{1,t}^{T_j-t}, ..., M_{P,t}^{T_j-t}\}$, for $j = 1, ..., J$. In the special case where we focus on a single maturity T, we denote $\tau = T - t$ the time-to-maturity, and $M_{p,t}^{T-t}$ is the p-th model parameter for that maturity. To further simplify notation, we denote K a particular strike in the set $\mathcal{K}$.

We assume that these model parameters are stochastic processes driven by some explanatory variables ξ, denoted $\mathcal{M}_t(\xi)$, and that they define the dynamics of the implied volatility surface. We let the implied volatility surface seen at time t with the sets of time-to-maturities $\mathcal{T}$ and strikes $\mathcal{K}$ be approximated by the function

$$f(t, \mathcal{T}; \mathcal{M}(\xi_t); \mathcal{K})$$

In order to capture the dynamics of the whole volatility surface, we need to consider all the smiles in the set $\mathcal{T}$, and all the strikes in the set $\mathcal{K}$. At time t, we choose to estimate the future implied volatility surface at time $t + m$, $m > 0$, by computing the conditional expectation of the function f. That is,

$$\widehat{\Sigma}(t, S_t; t + m; K_i, T_j - t) = E[f(t + m, T_j - t; \mathcal{M}(\xi_{t+m}); K_i)|\ \xi_t]\ ,\ i = 1, ..., I \text{ and } j = 1, ..., J \tag{2.5}$$

Note, we are focusing on the dynamics of the smiles with the fixed time-to-maturity $T_j - t$, $j = 1, ..., J$, and fixed strikes K_i, $i = 1, ..., I$. However, at time $t + m$, the market quotes option prices at time-to-maturities $T_j - t - m$, for some index j, and not at time-to-maturities $T_j - t$. Thus, we must rely on some interpolation techniques in time and space to study the dynamics of the volatility surface with a fixed grid of points. This is discussed later in Section (2.4).
In the special case where we have a single maturity T and a strike K, the approximated function is given by $f(t, T - t; \mathcal{M}(\xi_t); K)$. Considering a single maturity T, the estimated IVS at time t for the future volatility surface at time $t + m$ is given by

$$\widehat{\Sigma}(t, S_t; t + m; K, T - t) = E[f(t + m, T - t; \mathcal{M}(\xi_{t+m}); K)|\ \xi_t] \tag{2.6}$$

As discussed above, there are several ways for solving this problem:

- We can forecast the value of the risk factors, $\mathcal{M}$, or that of the explanatory processes ξ.

  1. We can use a statistical model to forecast the value of the risk factors or that of the explanatory processes. For example,

  $$\widehat{M}_{p,t+m}^{T-t} = g(M_{p,t}^{T-t}) + \epsilon_t\ ,\ l = 1, ..., P\ ,\ \text{ or } \hat{\xi}_{t+m} = g(\xi_t) + \epsilon_t$$

  for some smooth function g. In that setting, we get

  $$\widehat{\Sigma}(t, S_t; t+m; K, T-t) \approx \begin{cases} f(t + m, T - t; \widehat{\mathcal{M}}; K) \text{ if we forecast the risk factors} \\ f(t + m, T - t; \mathcal{M}(\hat{\xi}_{t+m}); K) \text{ if we forecast the explanatory processes} \end{cases}$$

with $\widehat{\epsilon}_t = E_t[\epsilon_{t+m}]$ the estimated noise. For instance, Dumas et al. [1998] and Goncalves et al. [2006] constructed a time series model for the estimated coefficients of cross-sectional regression. Bernales et al. [2014] used a two-step method to predict the IVS of individual US stock options.

2. We can use machine learning to forecast the values of the risk factors, or, those of the explanatory processes. For example,

$$\widehat{M}_{p,t+m}^{T-t} \equiv F(M_{p,t}^{T-t};\theta)\ ,\ l = 1, ..., P \text{ or } \widehat{\xi}_{t+m} \equiv F(\xi_t;\theta)$$

for some neural network (or recurrent neural network) F with parameters θ. For example, McGhee [2018] calibrated the SABR stochastic volatility model with a neural network.

- We can improve an existing model with machine learning. For instance, Audrino et al. [**?**] proposed a semi-parametric methodology based on an additive expansion of simple fitted regression trees, which can be easily estimated using boosting techniques. They considered a starting model and a base learner, allowing for an arbitrary number of exogenous factors including the spot price, the bid-ask spread, trading volume, other stock or index returns, and interest rates. The volatility surface is defined by

$$\Sigma(t;\eta_t,\tau) = f_{\eta,\tau}(x^{pred}) + \epsilon_{\eta,\tau}$$

where x^{pred} is the predictor. The regression function f is a linear additive expansion of the form

$$F_0(x^{pred}) + \sum_{j=1}^{M} B_j(x^{pred})$$

where F_0 is the starting model and B_j is the base learner (regression trees). They tested several staring models including the quadratic model, a sticky moneyness model, a Bayesian vector autoregression model, a dynamic semiparametric factor model. The model is trained to minimise the residuals of observed and estimated implied volatility, improving the out-of-sample prediction of the surface.

- We can use neural networks to capture the change in volatility surface. For instance, Zeng et al. [2019] considered the varying values of implied volatility over time (on tick data from the S&P 500 options market) as the targets of an online adaptive primal SVR model. Cao et al. [2019] employed multilayer neural networks (3 hidden layers and 80 nodes per hidden layer) for understanding volatility surface movements over daily period of time ($m = 1$) on S&P 500 call options data. We let $F(\xi_t;\theta)$ be a network with input vector $\xi_t = \{r_t, \Delta_{BS}(t), T-t, V_t\}$ and model parameters θ. The inputs are the return $r_t = \frac{\Delta S_t}{S_{t-1}}$, the BS-delta $\Delta_{BS}(t)$, time-to-maturity $T-t$, and the VIX denoted V_t. Then, the estimated change in volatility is given by

$$E[\Delta\Sigma(t,S_t;t;K,T-t)] = \widehat{\Sigma}(t,S_t;t+1;K,T-t) - \Sigma(t,S_t;t;K,T-t) \equiv F(\xi_t;\theta)$$

- We can model the relation between the volatility risk factors and their explanatory variables $\mathcal{M}(\xi)$.

1. We can use a statistical model to capture the relation between the volatility risk factors and their explanatory variables. Bloch et al. [2002] [2003] modelled the dynamics of the IV surfaces by letting the parameters of a parametric model be statistically evolved with explanatory variables including the spot price and some noise.

2. We can use neural networks to capture the relation between the volatility risk factors and their explanatory variables. For instance, Bloch [2020] implicitly modelled the dynamics of the risk factors by letting the risk factors be function of some explanatory variables including the spot

price, volume, estimated volatility, financial indicators. He considered one neural network per risk factor. He let $\widehat{F}_t^p = h^p(\xi_t, \theta^p)$, $p = 1, .., P$, be the network with input (or feature) vector ξ_t and parameters θ^l. Then, the risk factors seen at time t for the time to maturity $T - t$ are expressed as

$$M_{p,t}^{T-t} \equiv \widehat{F}_t^p(\xi_t) \ , \ p = 1, ..., P$$

These factors are then replaced in the volatility function f to compute the estimated volatility surface.

2.4 Generating a fixed grid IVS

In Section (2.3), we have presented the implied volatility surface (IVS), seen at time t, with the fixed set of strikes $\mathcal{K} = \{K_1, ..., K_I\}$ and the fixed set of time-to-maturities $\mathcal{T} = \{T_1 - t, ..., T_J - t\}$. We have assumed a parametric volatility model and considered one set of model parameters per fixed maturity T_j, $j = 1, ..., J$, of the volatility surface. However, in financial markets, at each time $t \in [1, T]$, we have a collection of option prices with different expires T_j, $j = 1, ..., J_t$, and different strikes K_i, $i = 1, ..., I_t$. Since the number of expiries may vary over time, the set of time-to-maturities at time t is denoted $\mathcal{T}_t$, and the number of strikes may also vary, the set of strikes is denoted $\mathcal{K}_t$. We obtain the market IVS at time t by applying the inverse BS-formula to these observed option prices for each time-to-maturity $T_j - t$, $j = 1, ..., J_t$, and each strike K_i, $i = 1, ..., I_t$. That is,

$$\Sigma(t, T_j, K_i) = C_{BS}^{-1}\big(C_t(T_j, K_i); T_j, K_i\big) \ , \ j = 1, ..., J_t \ , \ i = 1, ..., I_t$$

where C_t is the market price at time t and C_{BS} is the Black-Scholes price.
We now need to create a fixed grid of IVS for the fixed sets of time-to-maturities $\mathcal{T}$ and strikes $\mathcal{K}$. To do so, we make sure that the no-arbitrage constraints discussed in Section (2.1) are satisfied. We rely on the fact that the chosen parametric volatility model is nearly arbitrage free in space so that we are left with enforcing the no calendar arbitrage.
At each observed time-to-maturity $T_j - t$, $j = 1, ..., J_t$, we fit a parametric volatility model

$$\mathcal{J}(M_t^{f,T_j-t}) = \sum_{i=1}^{I_t} w_i \big|\Sigma(t, T_j, K_i) - \widehat{\Sigma}(t, T_j, K_i; M_t^{f,T_j-t})\big| \ , \ j = 1, ..., J_t$$

which we use to infer the volatility for the fixed set of strikes $\mathcal{K}$. Then, for all the strikes K_i, $i = 1, ..., I$, we linearly interpolate the total variance in time for the fixed set of time-to-maturities $\mathcal{T}$. At last, for each time-to-maturity $T_j - t$, $j = 1, ..., J$, we fit the parametric volatility model to these interpolated/extrapolated implied volatilities,

$$\mathcal{J}(M_t^{T_j-t}) = \sum_{i=1}^{I} w_i \big|\Sigma(t, T_j, K_i) - \widehat{\Sigma}(t, T_j, K_i; M_t^{T_j-t})\big| \ , \ j = 1, ..., J$$

getting the set of model parameters $\mathcal{M}_t = \{M_{1,t}^{\mathcal{T}}, ..., M_{P,t}^{\mathcal{T}}\}$, where $M_{p,t}^{\mathcal{T}} = \{M_{p,t}^{T_1-t}, ..., M_{p,t}^{T_J-t}\}$ for $p = 1, ..., P$, is the term-structure for the p-th model parameter $M_{p,t}$. Thus, at each time t, we have a total of $P \times J$ model parameters per volatility surface. Further, we denote $F_t^{\mathcal{T}} = \{F_t^{T_1-t}, ..., F_t^{T_J-t}\}$ the forward price term-structure seen at time t for the set of time-to-maturities $\mathcal{T}$.

3 Deep learning architecture

3.1 Formulation of the problem

Existing machine learning (ML) models are not aware of the expiring nature of the option contracts (see Section (2.2)), and require continuous time series for constructing a balanced panel. We briefly define the dynamical system used to represent the time series that we need to model, and discuss the mathematical formulation of our problem.

3.1.1 Defining the dynamical system

As discussed in Section (1), there are a lot of variables explaining the dynamics of the implied volatility surface, such as the variables defining the contingent claim, stochastic volatility and jumps, market factors, and technical indicators. We let the explanatory variables ξ_t, seen at time t, be a history of all the fitted model parameters for the last n period of time together with the last n forward price term-structures for the J time-to-maturities. That is,

$$\xi_t = \{\mathcal{M}_t, \mathcal{M}_{t-1}, ..., \mathcal{M}_{t-n+1}, F_t^{\mathcal{T}}, F_{t-1}^{\mathcal{T}}, ..., F_{t-n+1}^{\mathcal{T}}\} \tag{3.7}$$

As a result, the total number of explanatory variables at time t is $n \times (P+1) \times J$. To simplify notation, we let $x = (x_1, ..., x_n)$ be the sequence of explanatory variables, where each element x_i, $i = 1, ..., n$, has dimension $d = (P+1, J)$. We choose to model the dynamics of the IVS by learning the dynamics of the explanatory variables with an encoder-decoder model. We consider three main types of architectures for the encoder:

1. A recurrent encoder with multiple layers of LSTM units. We flatten the dimension d of the explanatory variables so that it becomes a one-dimensional vector of size $(P+1) \times J$, and we let the matrix representing the dynamical system be of size (n, d). Since the inputs are flattened before being passed to the LSTM, the spatiotemporal relations of the IVS are lost.

2. A recurrent encoder with convolutional LSTM (ConvLSTM) units. We let the dimension d be the matrix of size $(P+1, J)$, and we let the tensor representing the dynamical system be of size (n, d). Hence, the ConvLSTM model is capable of understanding the spatiotemporal relationships between strikes and time-to-maturities.

3. A special case of [2] where we consider one LSTM per model parameter and time-to-maturity, $M_{p,t}^{T_j - t}$, $j = 1, .., J$ and $p = 1, ..., P$. Thus, we have a total of $P \times J$ independent LSTM units, which corresponds to the approach introduced by Bloch [2020]. In that setting, for one model parameter, the dimension d is the single-element vector of size 1, and the tensor representing the dynamical system is of size (n, d).

For the reasons discussed in [1], we focus on the architectures [2] and [3].

3.1.2 Mathematical formulation

We let $\mathcal{X}$ and $\mathcal{Y}$ be the input and output spaces, $\mathcal{D}$ the probability distribution, D the data set, H the set of hypotheses, $I(\cdot)$ the indicator function. We let $\mathscr{X} \in \mathcal{X}$ be the input tensor, and $\mathscr{Y} \in \mathcal{Y}$ be the output tensor. Following Shi et al. [2015], we consider the spatiotemporal sequence forecasting problem where we observe a dynamical system over a spatial region represented by an $(P+1) \times J$ grid which consists of $(P+1)$ rows and J columns. Inside each cell in the grid, there is one measurement, the value of the explanatory variable, which vary over time, so that the observation at any time can be represented by a tensor $\mathscr{X} \in \mathbb{R}^{(P+1) \times J}$.

Recording the observations periodically, at time $t \in [1, T]$, for the last n observations, the input variables are denoted by $\mathscr{X}_t, \mathscr{X}_{t-1}, ..., \mathscr{X}_{t-n+1}$ and the outputs are the future input values $\mathscr{Y}_{t+1} = \mathscr{X}_{t+1}, ..., \mathscr{Y}_{t+m} = \mathscr{X}_{t+m}$, $m > 0$, for every $t = 1, ..., T$. The spatiotemporal sequence forecasting problem consists in predicting the most likely length-m sequence in the future given the previous n observations including the current one. This is an arg max problem given by

$$\widehat{\mathscr{Y}}_{t+1}, ..., \widehat{\mathscr{Y}}_{t+m} = \arg \max_{\mathscr{Y}_{t+1},...,\mathscr{Y}_{t+m}} P\big(\mathscr{Y}_{t+1}, ..., \mathscr{Y}_{t+m} |\ \mathscr{X}_t, ..., \mathscr{X}_{t-n+2}, \mathscr{X}_{t-n+1}\big)$$

Here, the prediction target of the forecasting problem is a sequence which contains both spatial and temporal structures.
In the special case where we flatten the dimension d of the explanatory variables, we let $x \in \mathcal{X}$ be the input source, and $y \in \mathcal{Y}$ be the the future value. From a probabilistic approach, we want to find the future value y that maximises the conditional probability of y given a source sentence x, that is, $\arg\max_{y \in \mathcal{Y}} P(y|\ x)$.

Rather than directly predicting the future value $\mathscr{Y}$ from the input source $\mathscr{X}$, we choose to estimate this conditional probability with an encoder-decoder model, where the encoder represents the sequence as context tensor $\mathscr{Z}$, and where the decoder predicts the future value $\mathscr{Y}$. That is, we first encode the input source in some tensors which are then used as input in a decoder. Technically, we map the original input $\mathscr{X}$ to a sequence of feature representations $\mathscr{Z}$. That sequence can be aggregated to a regressor, or, one can expose it to non-linear element-wise transformations (such as self attention or maxpoolong) and use it as an input to another sequence-to-sequence feature mapping.
Mathematically, we get

$$\mathscr{Z}_t = f_E(\mathscr{X}_t, ..., \mathscr{X}_{t-n+2}, \mathscr{X}_{t-n+1})$$

and

$$\begin{aligned}
\widehat{\mathscr{Y}}_{t+1}, ..., \widehat{\mathscr{Y}}_{t+m} &= \arg \max_{\mathscr{Y}_{t+1},...,\mathscr{Y}_{t+m}} P\big(\mathscr{Y}_{t+1}, ..., \mathscr{Y}_{t+n} |\ \mathscr{X}_t, ..., \mathscr{X}_{t-n+2}, \mathscr{X}_{t-n+1}\big) \\
&\approx \arg \max_{\mathscr{Y}_{t+1},...,\mathscr{Y}_{t+m}} P\big(\mathscr{Y}_{t+1}, ..., \mathscr{Y}_{t+m} |\ \mathscr{Z}_t\big) \\
&\approx g_F\big(\mathscr{Z}_t\big)
\end{aligned}$$

where f_E is the encoding function and g_F is the forecasting function.

3.2 The models

We explore the effects that different architectures have on the dynamics of the implied volatility surface. We consider two main types of architecture, the hierarchical LSTM models and the convolutional LSTM models.

3.2.1 The hierarchical LSTM model

In order to encode the sequences x of explanatory variables into fixed-sized representations z, we consider a recurrent encoder with multiple layers of LSTM units. Both the number of layers and the number of nodes per layer are chosen by the user. We let $L \in \mathbb{N}$ denotes the number of layers (depth) of the recurrent encoder and N_o the number of nodes for each RNN unit. The lth RNN unit, F_l, outputs all the hidden states of the dynamical state model. We let $h_l = \{h_1^l, h_2^l, ..., h_n^l\}$ be the set of state variables of the dynamical system. Here, $h_i^l \in \mathbb{R}^d$, $i = 1, ..., n$, is the state variable for the ith token, so that h_l is a matrix of size (n, d).
In order to aggregate the hidden states of the stacked RNN units, we adapt the hierarchical bi-directional RNN max-pooling proposed by Talman et al. [2018] to sequence forecasting, and extend that model to

make it deep. We stack L layers of RNN units with a max-pooling mechanism in each layer that acts on the hidden states of the RNN layer. The recurrent encoder $H(F, \cdot) : \mathbb{R}^{N_o} \to \mathbb{R}^{N_o}$ is defined as the composition

$$H = H_L \circ \cdots \circ H_1$$

where each component is of the form

$$H_l = \Phi^l \circ F_l \tag{3.8}$$

where F_l is the lth RNN unit and the function $\Phi^l : \mathbb{R}^{n \times d} \to \mathbb{R}^{n \times d}$ is the max-pooling mechanism. Note, this max-pooling does not reduce the dimensionality of the problem. That is, we still get a (n, d) matrix after applying the max-pooling to the RNN unit. Here, we use the output of the lth max-pooling unit as input to the $(l+1)th$ RNN unit F_{l+1}. Eventually, the results from the L max-pooling mechanisms, $u = [u_1, ..., u_L]$, are concatenated, making up the final sequence representation. The latter will be flattened before being entered in the decoder. There are shortcut connections between the RNN layers, concatenating the sequence representations and the output hidden states of the subsequent layer, and feeding them as input to the next RNN layer of the network.
Once the sequences x have been embedded and vectorised, the decoder must use these input vectors to predict the next value y. Practically, the outputs from each RNN unit are concatenated, resulting in the final representation, u, that is going to be used for the prediction. This representation is then passed as input to deep network layers. Both the number of layers and the number of nodes per layer are chosen by the user.

3.2.2 The ConvLSTM models

When working with images, the best approach is to use a Convolutional Neural Network (CNN) architecture where the image passes through convolutional layers, in which several filters extract important features. Alternatively, when the data is a time series, we use recurrent neural networks (RNNs), such as the LSTM and GRU, where the model passes the previous hidden state to the next step of the sequence. However, when the data is in the form of multidimensional sequences, such as sequences of images, one approach is to combine the CNN architecture with RNNs. Shi et al. [2015] extended the fully connected LSTM (FC-LSTM) to have convolutional structures both in the input-to-state and state-to-state transitions. They used the convolution operators in addition to the Hadamard product. Viewing the states as the hidden representations of moving objects, a ConvLSTM with a larger transitional kernel ($n \times n$ with $n > 1$) captures faster motions than with a smaller one. The authors stacked multiple ConvLSTM layers and formed an encoding-forecasting structure, getting a network model for general spatiotemporal sequence forecasting problems. They considered a structure with two networks, an encoding network and a forecasting network as illustrated in Figure (1). Similarly to the LSTM, the ConvLSTM determines the future state of a certain cell in the grid by the inputs and past states of its local neighbours. This done by using a convolution operator in the state-to-state and input-to-state transitions. In our setting, all the inputs $\mathscr{X}_1, ..., \mathscr{X}_t$, the cell outputs $\mathscr{C}_1, ..., \mathscr{C}_t$, the hidden states $\mathscr{H}_1, ..., \mathscr{H}_t$, and the gates i_t, f_t, o_t of the ConvLSTM are 2D tensors with spatial dimensions (rows and columns).

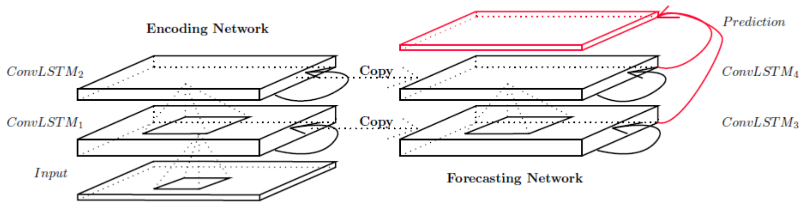

Figure 1: Encoding-forecasting ConvLSTM network.

From the mathematical formulation of the problem in Section (3.1.2), the initial states and cell outputs of the forecasting network are copied from the last state of the encoding network. The encoding LSTM compresses the whole input sequence into a hidden state tensor $\mathscr{H}$ and the forecasting LSTM unfolds this hidden state to give the final prediction. Here, the input and output elements are all 2D tensors, which preserve all the spatial information. The network has multiple stacked ConvLSTM layers leading to strong representational power suitable for predictions in complex dynamical systems.

When implementing the ConvLSTD 2D (it corresponds to spatial dimensions with rows and columns) with Keras, the inputs can be a 5D tensor and, if we select return_sequences, the output shape is a 5D tensor or a 4D one. Further, we need to set a few hyper parameters such as

- filters: Integer, the dimensionality of the output space. It is the number of output filters in the convolution.
- kernel_size: An integer or tuple/list of n integers, specifying the dimensions of the convolution window.
- strides: An integer or tuple/list of n integers, specifying the strides of the convolution.
- kernel_initializer: Initialiser for the kernel weights matrix, used for the linear transformation of the inputs.
- recurrent_initializer: Initialiser for the recurrent_kernel weights matrix, used for the linear transformation of the recurrent state.

The size of the kernel specifies the dimensions of the convolution window. In the special case of a 1×1 convolution, where the layers only have one channel, we just multiply every element in the previous layer by a number. In that setting, we recover the FC-LSTM network. This is because the inputs, cell outputs and hidden states of the traditional FC-LSTM can be seen as 3D tensors with the last two dimensions (rows and columns) being one. Here, all the features stand on a single cell. Note, larger state-to-state kernels are more suitable for capturing spatiotemporal correlations.

3.3 Generating sequences

When making m-steps ahead prediction with RNN models we need to define the way to produce such predictions. Several methods exists among which are the following two approaches:

1. Predict the next value and feed it back into the network for a number of m steps to produce m value predictions (autoregressive).
2. Predict all future time steps in one-go by having the number of LSTM / ConvLSTM layers l be equal to the number of m steps. Hence, we can simply use the output from each decoder LSTM / ConvLSTM cell as our prediction.

Since method [1] can produce any number of predictions in the future, without having to completely change the architecture, we will use it to forecast the dynamics of the implied volatility surface. Thus, we are now going to briefly discuss that method.

Recurrent neutral networks (RNNs) can be trained for sequence generation by processing real data sequences one step at a time and predicting what comes next. They learn to estimate the probability distribution of the next item in the sequence given all the previous ones. Assuming the predictions are probabilistic, they can generate new sequences by iteratively sampling from the output distribution of a trained network. The sample is fed as input at the next step. The obtained distribution is conditional, since the internal state of the network, and hence its predictive distribution, depends on the previous inputs. Predictions is made by using the RNNs internal representation to perform a high-dimensional interpolation between training examples.

Graves [2013] considered a deep recurrent neural network (RNN) composed of stacked LSTM layers, and explained how this network could be trained for the next-step prediction, and hence sequence generation. At time t, we get the input x_t, three hidden layers $h_{n,t}$, $n = 1, 2, 3$ and one output y_t. Here, the output y_t becomes the input x_{t+1} at time $t + 1$. The RNN prediction architecture is illustrated in Figure (2). In that setting, the output y_t is used to parameterise the predictive distribution $P(x_{t+1} \mid y_t)$ (since we are computing the next-step prediction of the input sequence).

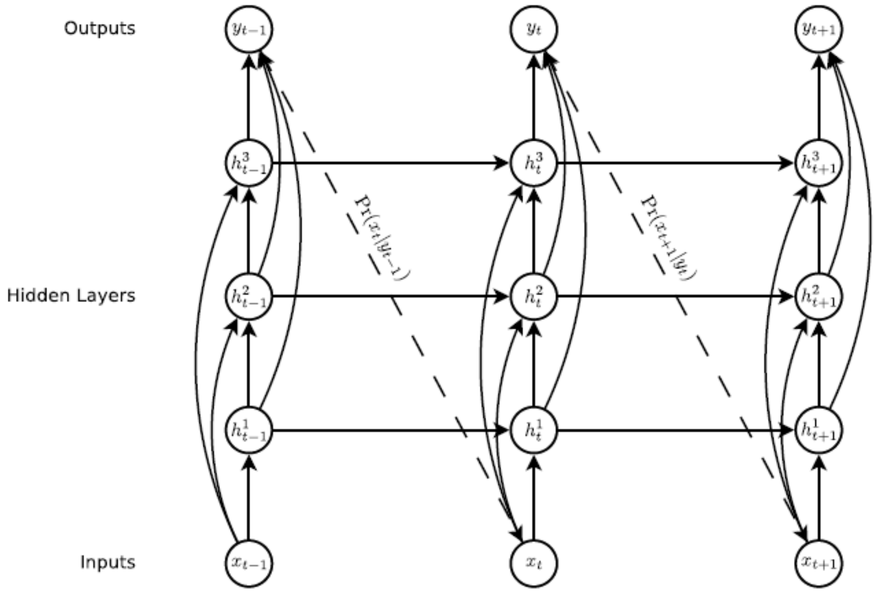

Figure 2: The circles mean network layers, the solid lines mean weighted connections and the dashed lines mean predictions.

Given an input vector sequence $x = (x_1, ..., x_T)$ with time length T, a stack of N recurrently connected hidden layers, the hidden vector sequences $h_n = (h_{n,1}, ..., h_{n,T})$, $n = 1, ..., N$, we compute the output vector sequence $y = (y_1, ..., y_T)$. Each output vector y_t is used to parameterise a predictive distribution $P(x_{t+1} \mid y_t)$ over the possible next inputs x_{t+1}. The hidden layer activations are computed by iterating equations from $t = 1$ to T and from $n = 2$ to N. These equations are

$$
\begin{aligned}
h_{1,t} &= f\big(W_{ih_1}x_t + W_{h_1h_1}h_{1,t-1} + b_{1,h}\big) \\
h_{n,t} &= f\big(W_{ih_n}x_t + W_{h_{n-1}h_n}h_{n-1,t} + W_{h_nh_n}h_{n,t-1} + b_{n,h}\big) \qquad (3.9)
\end{aligned}
$$

where W_{ih_n} is the weight matrix connecting the inputs to the nth hidden layer, $W_{h_1h_1}$ is the recurrent connection at the first hidden layer, and so on. Given the hidden sequences, the output sequence is given by

$$
\begin{aligned}
\hat{y}_t &= b_y + \sum_{n=1}^{N} W_{h_ny}h_{n,t} \\
y_t &= f_{out}(\hat{y}_t) \qquad (3.10)
\end{aligned}
$$

with a stack of N recurrently connected hidden layers.

4 The framework

4.1 Data preprocessing

We briefly discuss data scaling and data preparation to generate the discrete sets of option maturities $\mathcal{T}$ and strikes $\mathcal{K}$.

4.1.1 Data scaling

We let $\{x_t; t \in T\}$ be a real-valued stochastic process, and define the discrete time as $\{t_1, ..., t_n\} \subset T$, where $t_1 < t_2 < \cdots < t_n$. Further, we assume that the process $\{x_t\}$ is either serially uncorrelated, or, with minor lower order serial correlations, but it is dependent. We assume that the process x_t follows the dynamics

$$
x_t = \mu_t + a_t \ , \ x_0 = x \qquad (4.11)
$$

where a_t is the stochastic shock, or mean-corrected return (or innovation), of our process [2]. The conditional mean and conditional variance of x_t, given the filtration $\mathcal{F}_{t-1}$, are defined by

$$
\begin{aligned}
\mu_t &= E[x_t|\mathcal{F}_{t-1}] = G(\mathcal{F}_{t-1}) \\
\sigma_t^2 &= Var(x_t|\ \mathcal{F}_{t-1}) = E[(x_t - \mu_t)^2|\mathcal{F}_{t-1}] = H(\mathcal{F}_{t-1})
\end{aligned}
$$

where G and H are well defined functions with $H(\cdot) > 0$. The model for μ_t is the mean equation for x_t, and the model for σ_t^2 is the volatility equation for x_t. Both the functions G and H can be non-linear.

We standardise the process x by subtracting its expected value $E[x]$, and divide the difference by its standard deviation $\sigma_x = \sigma(x)$, getting

$$
\hat{x} = \frac{x - E[x]}{\sigma(x)}
$$

In our case, since we work with sample data, we subtract the population mean μ_x from an individual process value x_i and then divide the difference by the population standard deviation σ_x. That is, the standardised process, at state i, is given by

$$
\hat{x}_i = \frac{x_i - \mu_x}{\sigma_x}
$$

[2]since $a_t = x_t - \mu_t$

4.1.2 Constructing the IVS

The data corresponds to the S&P 500 bid-ask option prices for the past eight years, from the 1st of January 2012 until the 31st of December 2019. There is a large number of unique contracts series [3]: At each time $t \in [1, T]$, we have a collection of option prices with different expires T_j, $j = 1, ..., J_t$, up to 360 days, and different strikes K_i, $i = 1, ..., I_t$. Following Section (2.4), the method for constructing the implied volatility surface at time t is first to calculate the implied volatility discrete points for each listed option contract by using the inverse of the BS-formula at the mid option prices. Then, we fit the SVI functional form given in Equation (2.4) to these observed implied volatilities for each time-to-maturity $T_j - t$, $j = 1, ..., J_t$, with a Differential Evolution optimiser, obtaining the fitted model parameters $\mathcal{M}_t^f = \{M_{1,t}^{\mathcal{T}_t}, ..., M_{P,t}^{\mathcal{T}_t}\}$. Our row data is the set of all fitted model parameters $\mathcal{M}^f = \{\mathcal{M}_1^f, ..., \mathcal{M}_T^f\}$, where the set $\mathcal{M}_t^f$ is as above.

We create the fixed grid of IVS for the sets of time-to-maturities $\mathcal{T}$ and strikes $\mathcal{K}$ by using the SVI model in space and, for each strike, linearly interpolating the total variance in time. In our setting, at every time t, we consider $J = 8$ fixed pillars T_j, $j = 1, ..., J$, expressed in days where the set of time-to-maturities is given by $\mathcal{T} = \{7, 14, 21, 30, 60, 90, 120, 180\}$, $K = 50$ fixed strikes expressed in log-moneyness in the range $[-0.2, 0.2]$ where the set of strikes is given by $\mathcal{K} = \{-0.2, -0.192, ..., 0.2\}$. At last, we fit the SVI model on the fixed grid of IVS, getting the interpolated model parameters $\mathcal{M}_t = \{M_{1,t}^{\mathcal{T}}, ..., M_{P,t}^{\mathcal{T}}\}$. The fit of the SVI on the fixed pillar (seven days) is displayed in Figure (3). The plot in Figure (4) shows the interpolated SVI parameters over time for the set of time-to-maturities $\mathcal{T}$.

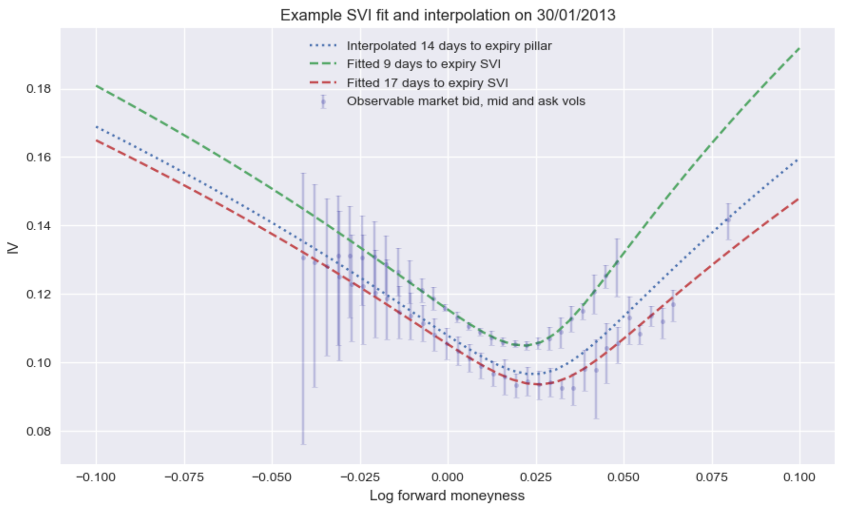

Figure 3: SVI fit on observed and interpolated implied volatilities.

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

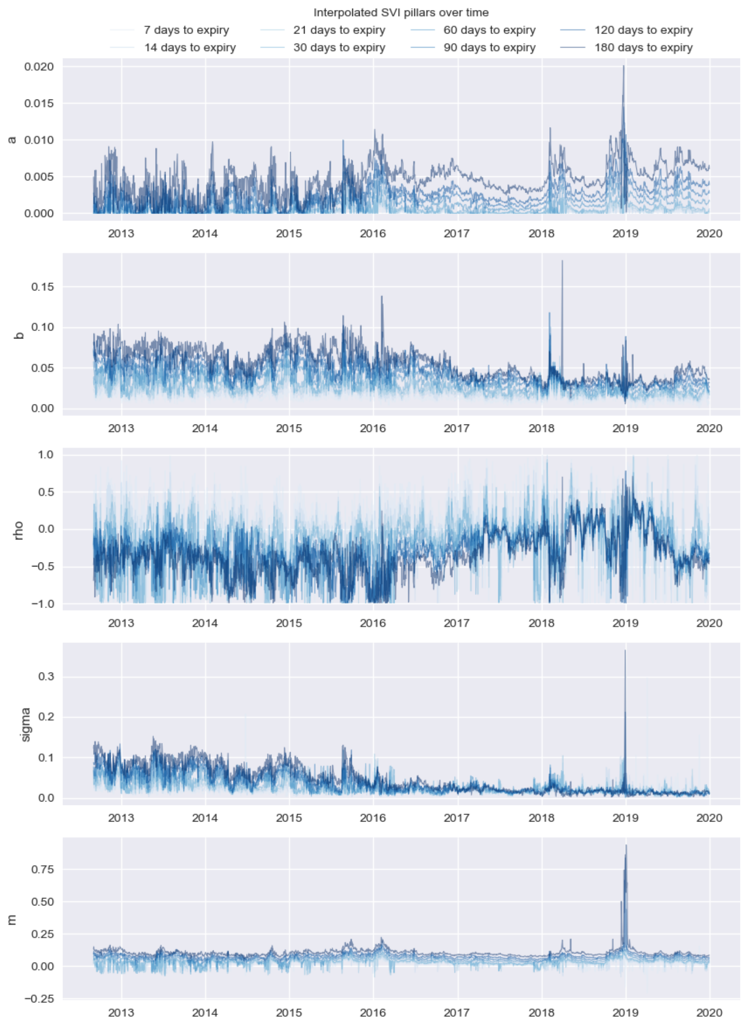

Figure 4: Interpolated parameters for the SVI model.

4.1.3 Volatility constraints

Before training the model, we need to make sure that the SVI parameters satisfy the constraints defined in Section (2.2). We use the notation in Section (3.2) and let z denote the output variables of the LSTM or ConvLSTM. We apply a softmax function to:

$$\phi_i = f_s(z_i^{\phi}) = \frac{e^{z_i^{\phi}}}{\sum_{j=1}^{K} e^{z_j^{\phi}}}$$

where z_i^ϕ is the corresponding network output. This way ϕ_i lies in the range $(0, 1)$ and sum to unity. For the standard deviation to remain positive, we use the exponential function

$$\sigma_i = e^{z_i^\sigma}$$

where z_i^σ is the corresponding network output. For parameters a and m taking values in $\mathbb{R}$, we let a_i and m_i be directly represented by the network outputs

$$a_i = z_i^a \text{ and } m_i = z_i^m$$

4.2 Model setup

4.2.1 Configuring the model

4.2.1.1 Defining the model To capture the dynamics of the IVS, we tested the ConvLSTM model described in Section (3.2.2) with dimension $d = (P, J)$ on different number of layers, number of neurons per layer, and kernel sizes. We first set the size of state-to-state convolutional kernel to one (1×1) to recover the FC-LSTM network and then gradually increased the size of the kernel. While in that setting we don't capture the spatiotemporal motion patterns, increasing the kernel sizes did not improve the performance of the model. This is probably due to the size of the dimension d, which is too small for the convolutional operator to properly operate, as well as to the fact that the model parameters are not correlated. Thus, we set the kernel to one and constructed different architectures of the ConvLSTM2D layers (looking at one, two, and three layers) with batch normalisation, followed with a Conv2D layer for the spaciotemporal outputs. We got the best results with two different architectures: an encoder-decoder with two ConvLSTM2D filters of size 8 each, and a simple encoder with two ConvLSTM2D layers, 64 neurons per layer, batch normalisation, and a final Conv2D layer.

4.2.1.2 Interpretation In CNN models, the purpose of doing convolution is to extract useful features from the input. Each type of filters helps to extract different aspects or features from the input. CNNs exploit the strong spatially local correlation present in natural images. While the pairs of implied volatility $\Sigma(K_i, T_j)$, $i = 1, ..., I$, $j = 1, .., J$ have strong spatial local correlation, this is not the case with the model parameters $\mathcal{M}_t$ (see Figure (4)). Thus, it make sense to use a ConvLSTM model to capture the spatiotemporal dynamics of the implied volatility surface, but not that of the model parameters. A classical LSTM per model parameter is sufficient.

4.2.1.3 Forecasting In order to forecast the model parameters at time $t + m$, we follow the approach described in Section (3.3) where we generate the forecasting sequence by processing real data sequences one step at a time and predicting what comes next. At each time t, we take as input the last $n = 10$ business days for the explanatory variables ξ_t in Equation (3.7), encodes these explanatory variables into a latent space, and predict the the SVI parameters and forward prices at time $t + 1$, denoted by the pair $\{\mathcal{M}_{t+1}, F_{t+1}^{\mathcal{T}}\}$. At time $t + 1$, we discard the last elements $\{\mathcal{M}_{t-n+1}, F_{t-n+1}^{\mathcal{T}}\}$ of the explanatory variables in Equation (3.7) and append the newly computed vales $\{\mathcal{M}_{t+1}, F_{t+1}^{\mathcal{T}}\}$. Then, the new explanatory variables ξ_{t+1} are fed as input in the network at the next step. This iterative process is repeated until we reach the time $t + m$.

4.2.2 Settings

4.2.2.1 Training and testing In order to forecast the implied volatility surface m steps ahead, we use a standard train-test split method that preserves temporal ordering of the time series. The research sample is divided into an in-sample training set with $t \in [1, T_I]$ and an out-of-sample validation set with $t \in [1, T_O]$. We tried different sample size for the training and testing sets:

1. we let the data from the 1st of January 2012 to the 31st of May 2018 be the training dataset and the remaining one and a half years of the data be the testing dataset, from 1st of June 2018 to 31st of December 2019.

2. we let the data from the 1st of January 2012 to the 31st of May 2016 be the training dataset and the remaining three and a half years of the data be the testing dataset, from 1st of June 2016 to 31st of December 2019.

Note, the testing set is deliberately chosen to increase in size in order to test the predictive performance of the model. In practice, one would use a sliding window and retrain the model in each window. We train the model to forecast the SVI parameters $M = \{a, b, \rho, \sigma, m\}$ (see Section (2.2)) at future time $t + m$, for $m = 1, ..., 20$ trading days, and for the set of time-to-maturities $\mathcal{T}$.

4.2.2.2 Model parameters In our framework, the optimiser selected is the Adam, the loss function selected is the mean absolute error (MAE), we do batch normalisation and the batch size is set to 64. The training data is further split in 80% training and 20% validation, where we test model performance in the latter. The number of training epoch is based on the loss function on the validation set: when it stops improving results we stop training the model to avoid over fitting the data. The average number of epochs is around 90. In our setting, we have, $P = 5$, model parameters and, $J = 8$, expiries, and the dimension is $d = P \times J$. The architecture of our model is displayed in Figure (5).

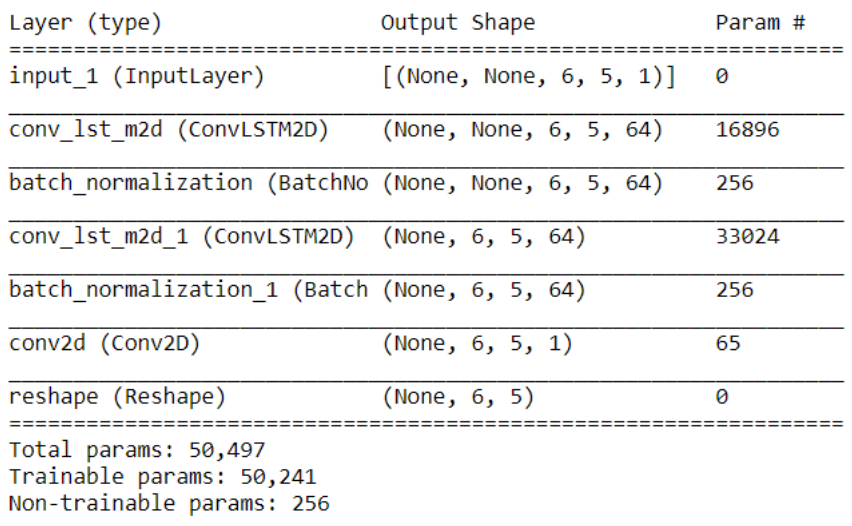
```
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, None, 6, 5, 1)]   0
_________________________________________________________________
conv_lst_m2d (ConvLSTM2D)    (None, None, 6, 5, 64)    16896
_________________________________________________________________
batch_normalization (BatchNo (None, None, 6, 5, 64)    256
_________________________________________________________________
conv_lst_m2d_1 (ConvLSTM2D)  (None, 6, 5, 64)          33024
_________________________________________________________________
batch_normalization_1 (Batch (None, 6, 5, 64)          256
_________________________________________________________________
conv2d (Conv2D)              (None, 6, 5, 1)           65
_________________________________________________________________
reshape (Reshape)            (None, 6, 5)              0
=================================================================
Total params: 50,497
Trainable params: 50,241
Non-trainable params: 256
```

Figure 5: Encoding-forecasting ConvLSTM network.

4.3 Testing the framework

4.3.1 Benchmark model

In addition to using metrics such as the MSE or the MAE on the training set to show convergence of our forecasting model, we choose to benchmark our model against a naive strategy. It consists in letting the prediction of the pth model parameter $M_{p,t+m}^{T_j-t}$ for the time-to-maturities $T_j - t$, $j = 1, ..., J$, be estimated by the realised value $M_{p,t}^{T_j-t}$ at time t and same time-to-maturities $T_j - t$, $j = 1, ..., J$. That is, the benchmark model assumes that the dynamics of the pth model parameter for the time-to-maturities $T_j - t$, $j = 1, ..., J$, satisfy the martingale

$$E_t[M_{p,t+m}^{T_j-t}] = M_{p,t}^{T_j-t} \ , j = 1, ..., J$$

We vary the future time $t+m$ for $m = 1, ..., 20$ trading days, and assess if the fitted model parameters are actual martingales or if they exhibit some trend. We also assess if our model can outperform this benchmark.

4.3.2 Measuring performance

We predict the implied volatility surface (IVS) at future time $t+m$ and, for all $t \in [1, T_O - m]$, compute the testing error of the IVS prediction against the realised market volatility, which we compare against that of the benchmark model. We display in a graph the plot of the two time series over the sample test, where the blue curve is the model prediction for future time $t+m$ and the green curve is the realised time series of the model parameters corresponding to this future time. We measure the performance of both the forecasting model and the benchmark model on the entire validation set by computing the MAE and RMSE on the implied volatility for the model prediction, the naive strategy, and their difference. That is, given the future index m, for all time $t \in [1, T_O - m]$ and for each time-to-maturity $T_j - t$, $j = 1, ..., J$, we compute the MAE for the model prediction as

$$\mathcal{E}_{MP}(M_{t+m}^{T_j-t}; t = 1, .., T_O - m) = \frac{1}{(T_O - m) \times I} \sum_{t=1}^{T_O-m} \sum_{i=1}^{I} \left| \Sigma(t+m, T_j, K_i) - \hat{\Sigma}(t; t+m; T_j, K_i; M_{t+m}^{T_j-t}) \right| \ , j = 1, ..., J$$

where $\Sigma(t+m, T_j, K_i)$ is the market volatility at time $t+m$ given in Equation (2.1) and $\hat{\Sigma}(t; t+m; T_j, K_i; M_{t+m}^{T_j-t})$ is the estimated future IVS computed at time t given in Equation (2.5). Similarly, we compute the MAE for the naive strategy as

$$\mathcal{E}_{NS}(M_t^{T_j-t}; t = 1, .., T_O - m) = \frac{1}{(T_O - m) \times I} \sum_{t=1}^{T_O-m} \sum_{i=1}^{I} \left| \Sigma(t+m, T_j, K_i) - \hat{\Sigma}(t; t+m, T_j, K_i; M_t^{T_j-t}) \right| \ , j = 1, ..., J$$

Note, $\hat{\Sigma}(t; t+m, T_j, K_i; M_t^{T_j-t}) \approx \Sigma(t, T_j, K_i)$ so that the MAE for the naive strategy simplifies to

$$\mathcal{E}_{NS}(M_t^{T_j-t}; t = 1, .., T_O - m) \approx \frac{1}{(T_O - m) \times I} \sum_{t=1}^{T_O-m} \sum_{i=1}^{I} \left| \Sigma(t+m, T_j, K_i) - \Sigma(t, T_j, K_i) \right| \ , j = 1, ..., J$$

Since the IVS is not stationary but stochastic, as m gets larger so does the MAE for the naive strategy. Then, we compute the difference between the performance measure of these two models

$$\mathcal{D}(j) = \mathcal{E}_{MP}(M_{t+m}^{T_j-t}; t = 1, .., T_O - m) - \mathcal{E}_{NS}(M_t^{T_j-t}; t = 1, .., T_O - m) \ , j = 1, ..., J$$

If that difference is negative, our forecasting model outperforms the naive strategy in the sample test.

5 Results

We train the model over different periods of time and predict the implied volatility surface (IVS) at future time $t+m$ for the set $\hat{\mathcal{T}} = [1, 3, 5, 10, 20]$. For each time $t+m$, we compute the RMSE and MAE presented in Section (4.3.2) for the model prediction, the naive strategy, and their difference.

The code that implements these deep neural architectures is available at:
https://github.com/ArthurBook/MachineSmiling.

5.1 Convergence

We display in Figure (6) the training development of the MAE against the number of epochs.

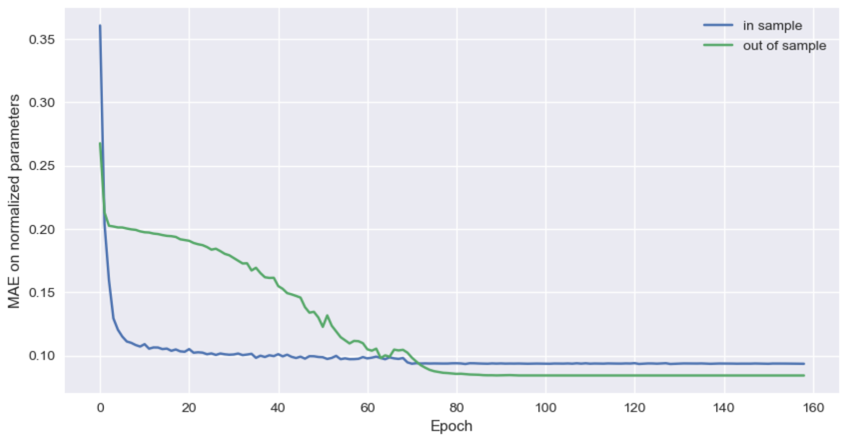

Figure 6: Training development.

This result shows that the RMSE of the training set within the sample as well as the validation set outside the sample gradually decrease and converge to a stable value as the number of epochs increases.

5.2 Predictions

We consider three datasets of different size: (1) train = (None,"2019-5-31") and test = ("2019-6-1",None), (2) train = (None,"2018-5-31") and test = ("2018-6-1",None), (3) train = (None,"2016-5-31") and test = ("2016-6-1",None).

5.2.1 Dataset 1

The RMSE on volatility for train = (None,"2019-5-31") and test = ("2019-6-1",None) are displayed in Table (1).

	Model - Naive				
	$t+1$	$t+3$	$t+5$	$t+10$	$t+20$
7	-0.004646	-0.002626	-0.008730	-0.021107	-0.015602
14	0.000383	0.000745	-0.003351	-0.008911	-0.002299
21	0.001811	0.001503	-0.000538	-0.002151	-0.000226
30	0.002967	0.001709	0.000316	-0.000423	-0.001409
60	0.003090	0.001830	0.000803	0.000104	-0.000579
90	0.002588	0.001312	0.000548	0.000403	0.000558
120	0.002346	0.001252	0.000611	0.000381	0.000260
180	0.002271	0.001133	0.000423	-0.000085	-0.000186

Table 1: RMSE on volatility for train = (None,"2019-5-31") and test = ("2019-6-1",None).

The MAE on volatility for train = (None,"2019-5-31") and test = ("2019-6-1",None) are displayed in Table (2).

	Model - Naive				
	$t+1$	$t+3$	$t+5$	$t+10$	$t+20$
7	-0.002154	-0.001443	-0.004807	-0.016251	-0.011870
14	0.002053	0.000878	-0.000811	-0.006952	-0.003690
21	0.002100	0.001389	0.001120	-0.002602	-0.000486
30	0.002820	0.000915	0.000910	-0.000733	-0.001230
60	0.002412	0.000789	0.001022	-0.000345	-0.001425
90	0.002072	0.000493	0.000677	-0.000284	-0.000049
120	0.001997	0.000687	0.000899	-0.000043	0.000527
180	0.001780	0.000654	0.000719	-0.000190	0.001125

Table 2: MAE on volatility for train = (None,"2019-5-31") and test = ("2019-6-1",None).

5.2.2 Dataset 2

The RMSE on volatility for train = (None,"2018-5-31") and test = ("2018-6-1",None) are displayed in Table (3).

	Model - Naive				
	$t+1$	$t+3$	$t+5$	$t+10$	$t+20$
7	-0.005747	-0.008865	-0.015143	-0.020370	-0.019728
14	0.002207	-0.000903	-0.004647	-0.007820	-0.008062
21	0.002889	-0.000224	-0.002709	-0.004261	-0.005049
30	0.003757	0.000496	-0.001248	-0.002475	-0.003952
60	0.004135	0.002083	0.000269	-0.001665	-0.003691
90	0.004205	0.002537	0.000628	-0.001406	-0.003576
120	0.003279	0.001717	0.000328	-0.001619	-0.003474
180	0.004584	0.003302	0.002068	-0.000003	-0.002121

Table 3: RMSE on volatility for train = (None,"2018-5-31") and test = ("2018-6-1",None).

The MAE on volatility for train = (None,"2018-5-31") and test = ("2018-6-1",None) are displayed in Table (4).

	Model - Naive				
	$t+1$	$t+3$	$t+5$	$t+10$	$t+20$
7	-0.003018	-0.006447	-0.012062	-0.015923	-0.017405
14	0.002474	-0.001022	-0.003747	-0.006156	-0.007271
21	0.002400	-0.000028	-0.001917	-0.002852	-0.004320
30	0.003107	0.000607	-0.000433	-0.001655	-0.003232
60	0.002813	0.001165	0.000291	-0.001217	-0.002693
90	0.002669	0.001169	0.000307	-0.000995	-0.002364
120	0.002664	0.001220	0.000352	-0.000810	-0.001727
180	0.002748	0.001847	0.001010	-0.000526	-0.001340

Table 4: MAE on volatility for train = (None,"2018-5-31") and test = ("2018-6-1",None).

5.2.3 Dataset 3

The RMSE on scaled volatility for train = (None,"2016-5-31") and test = ("2016-6-1",None) are displayed in Table (5).

	Model - Naive				
	$t+1$	$t+3$	$t+5$	$t+10$	$t+20$
7	-0.000435	-0.001085	-0.001770	-0.002369	-0.002463
14	0.000500	-0.000131	-0.000700	-0.001396	-0.001542
21	0.000522	0.000107	-0.000514	-0.000864	-0.001303
30	0.000704	0.000247	-0.000392	-0.000794	-0.001356
60	0.001358	0.000785	0.000139	-0.000354	-0.000629
90	0.001778	0.001202	0.00042	-0.000311	-0.000697
120	0.002225	0.001524	0.000888	0.000362	0.000142
180	0.002756	0.001932	0.001073	0.000125	-0.000643

Table 5: RMSE on scaled volatility for train = (None,"2016-5-31") and test = ("2016-6-1",None).

The RMSE on volatility for train = (None,"2016-5-31") and test = ("2016-6-1",None) are displayed in Table (6).

	Model - Naive				
	$t+1$	$t+3$	$t+5$	$t+10$	$t+20$
7	-0.003260	-0.007726	-0.013029	-0.018325	-0.019438
14	0.002291	-0.000798	-0.003926	-0.007905	-0.009484
21	0.002339	0.000874	-0.001742	-0.003623	-0.005766
30	0.002296	0.000663	-0.001582	-0.003305	-0.005895
60	0.003265	0.002072	0.000553	-0.000702	-0.001494
90	0.003579	0.002657	0.001214	-0.000279	-0.001241
120	0.004016	0.003043	0.002129	0.001327	0.001075
180	0.003922	0.003022	0.001931	0.000599	-0.000327

Table 6: RMSE on volatility for train = (None,"2016-5-31") and test = ("2016-6-1",None).

The MAE on volatility for train = (None,"2016-5-31") and test = ("2016-6-1",None) are displayed in Table (7).

	Model - Naive				
	$t+1$	$t+3$	$t+5$	$t+10$	$t+20$
7	-0.001196	-0.006673	-0.011169	-0.014574	-0.015373
14	0.002395	-0.001327	-0.003069	-0.005496	-0.006616
21	0.002202	0.000182	-0.001044	-0.001876	-0.002966
30	0.002482	0.000847	-0.000453	-0.001275	-0.002982
60	0.002481	0.001349	0.000692	0.000101	0.000239
90	0.002607	0.001820	0.001360	0.000636	0.000337
120	0.003320	0.002945	0.002866	0.002526	0.002728
180	0.002937	0.002211	0.001787	0.001323	0.001136

Table 7: MAE on volatility for train = (None,"2016-5-31") and test = ("2016-6-1",None).

5.3 Findings

From the results in Section (5.2), we observe that, on average, our model systematically outperforms the naive approach to forecast the implied volatility surface at future time $t+m$ as m gets larger, but not for small m in the range $m = [1, 3]$ day(s) forecast. Further, the model does better at forecasting shorter maturities than it does at forecasting longer ones. This suggest that the long term dynamics of the IVS for short to mid-range maturities are dominated by trend and mean reversion. As a result, part of the dynamics of the IVS is predictable: long term short maturities. It can be explained by the rising volume of options traded in the market and the rising number of algorithmic trading that facilitates automated trading across all asset classes and market segments. In 2020, 25.55 billion futures contracts were traded worldwide, up from 12.13 billion in 2013. The number of options contracts traded increased from 9.42 to 21.22 billion contracts in the same period (according to Statista Research Department, Mar 15, 2021). The total number of futures and options traded on exchanges worldwide reached a record level of 46.77 billion contracts in 2020, up 35.6% from 2019 (according to FIA, Jan 21, 2021). Algorithmic trading contributed nearly $60 - 73\%$ of all US equity trading in 2018 (according to (Business Wire). In Europe and the US, 10% of the hedge funds used algorithms to trade over 80% of their value in 2020 (according to The TRADE's Algorithmic Trading Survey 2020).
On the other hand, our results show that short horizon forecast is dominated by noise when using daily data. In that case, getting intra day data would help at forecasting the volatility surface over short period of time. These experimental results show that the deep learning system presented in this paper is very effective at forecasting part of the implied volatility surface over different training period. In addition, getting a training dataset covering a long period of time helps improving accuracy measures. As a result, we can roll the training dataset in time and use the forecasting model for pricing and hedging options, performing risk analysis, or, for volatility trading.

In addition to considering accuracy measures such as RMSE and MAE, we need to get an understanding of the forecasting power of the model directly expressed in terms of implied volatility. We display in Figure (7) both the realised and predicted at-the-money forward volatilities for train = (None,"2016-5-31") and test = ("2016-6-1",None).

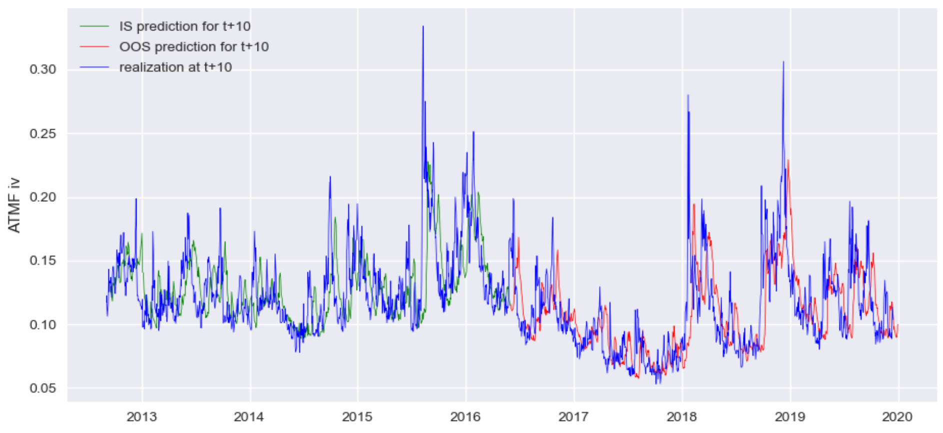

Figure 7: SVI prediction of the at-the-money forward implied volatilities.

To better understand the predictive power of the model, we let time t be "2017-5-31" with a spot price at 2412, and time $t+m$ be "2017-06-13" with a spot price at 2440, that is, ten business days. We display in Figure (8) the predicted SVI smile and the one obtained from the naive approach with model parameters set at time t versus the realised SVI smile at $t+m$.

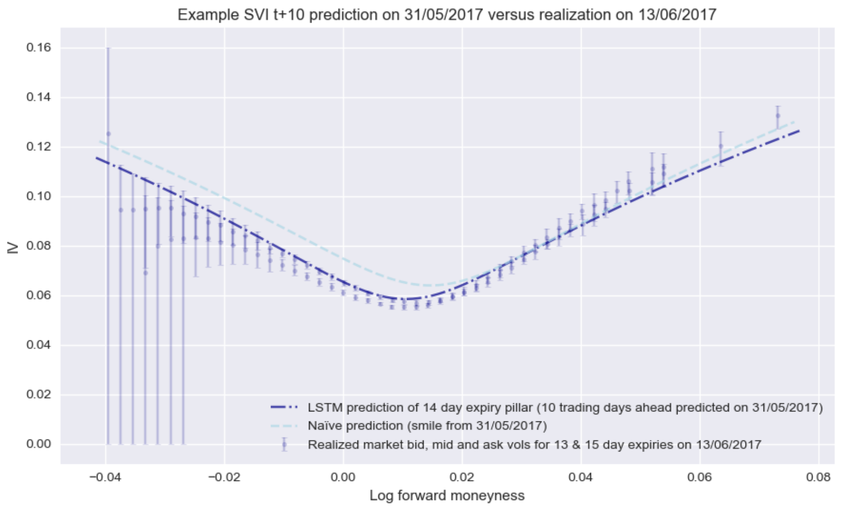

Figure 8: SVI fit and prediction of implied volatilities for $t+10$ and ten days maturity.

We observe that the predicted skew decreased, matching the realised skew, in accordance with the increase by 28 points of the SPX during that period of time. Further, the at-the-money forward volatility falls, which is consistent with the market rally.

5.4 Future work

Repeat the prediction tests performed in Section (5.2) with intra day market data for the future time $t + m$, with $m = [1, 3]$.

6 Conclusion

We considered several architectures of convulational LSTM networks for learning the dynamics of the implied volatility surface by performing a dimensionality reduction and learning the main volatility risk factors from the data. We assumed that these factors were stochastic processes driven by some explanatory variables. We modelled the dynamics of the IV surfaces by letting the parameters of a stochastic volatility model with an explicit expression for the smile be statistically evolved. We focused on the SVI model and learned to predict the future values of the model parameters. The chosen explanatory variables are the time series of the fitted model parameters and the time series of the forward prices. We tested our model against a naive strategy by forecasting the volatility surface on the S&P 500 option prices several steps ahead, and computing some measures of accuracy. On average, our model systematically outperformed the naive approach on long term forecasts for short to mid-range maturities. This showed that the IVS has dynamics of its own, dominated by trend and mean reversion. We demonstrated the predictive power of our model on different datasets and showed that, on average, it was capable of correctly predicting the movements of the market smiles.

References

[2012] Abergel F., Zaatour R., What drives option prices? *Journal of Trading*, **7**, (3), pp 12–28.

[2020] Ackerer D., Tagasovska N., Vatter T., Deep smoothing of the implied volatility surface. arXiv:1906.05066v3.

[1998] Ait-Sahalia Y., Lo A.W., Nonparametric estimation of state-price densities implicit in financial asset prices. *Journal of Finance*, **53**, pp 499–547.

[2004] Alexander C., Nogueira L.M., Hedging with stochastic local volatility. ISMA, Centre Discussion Paper in Finance, University of Reading **2004-11**.

[2010] Alexander C., Rubinov A., Kalepky M., Leontsinis S., Hedging with stochastic local volatility. ICMA, Centre Discussion Paper in Finance, University of Reading.

[1996] Anders U., Korn O., Schmitt C., Improving the pricing of options: a neural network approach. ZEW Discussion Papers, pp 96–104.

[2019] Babbar K., McGhee W.A., A deep learning approach to exotic option pricing under LSVol. Working Paper.

[1997] Bakshi G., Cao C., Chen Z., Empirical performance of alternative option pricing models. *Journal of Finance*, **52**, pp 2003–2049.

[1996] Bates D., Jumps and stochastic volatility: Exchange rate processes implicit in deutschemark options. *Review of Financial Studies*, **9**, pp 69–108.

[2018] Bayer C., Stemper B., Deep calibration of rough stochastic volatility models. Preprint, arXiv:1810.03399.

[2009] Bedendo M., Hodges S.D., The dynamics of the volatility skew: A Kalman filter approach. *Journal of Banking and Finance*, **33**, (6), pp 1156–1165.

[2009] Bengio Y., Learning deep architectures for AI. Now Publishers Inc.

[2004] Bennell J., Sutcliffe C., Black-Scholes versus artificial neural networks in pricing FTSE 100 options. *Intelligent Systems in Accounting*, Finance and Management, **12**, pp 243–260.

[2014] Bernales A., Guidolin M., Can we forecast the implied volatility surface dynamics of equity options? Predictability and economic value tests. *Journal of Banking and Finance*, **46**, pp 326–342.

[2006] Bishop C.M., Pattern recognition and machine learning. Springer Verlag.

[1973] Black F., Scholes M., The pricing of options and corporate liabilities. *Journal of Political Economics*, **81**, pp 637–659.

[2002] Bloch D.A., Miralles P., Statistical dynamics of the smile. Technical Report, DrKW.

[2003] Bloch D.A., Aube J-D., A Statistical deterministic implied volatility model. Technical Report, Credit Lyonnais.

[2010a] Bloch D., A note on calibration of Markov processes. *Wilmott Journal*, **2**, (2), pp 66–96.

[2010] Bloch D.A., A practical guide to implied and local volatility. Working Paper, University of Paris 6, SSRN-id1538808.

[2012a] Bloch D., From implied to local volatility surface. Working Paper, University of Paris 6 Pierre et Marie Curie.

[2012] Bloch D., From implied volatility surface to quantitative options relative value trading. Working Paper, University of Paris 6 Pierre et Marie Curie.

[2018] Bloch D., Recipe for quantitative trading with machine learning. Working Paper, University of Paris 6 Pierre et Marie Curie.

[2019] Bloch D., Machine learning: Models and algorithms. Working Paper, University of Paris 6 Pierre et Marie Curie.

[2019b] Bloch D., Option pricing: Theory and applications. Working Paper, University of Paris 6 Pierre et Marie Curie.

[2020] Bloch D., Neural networks based dynamic implied volatility surface. Working Paper, University of Paris 6 Pierre et Marie Curie.

[2020] Bloch D., Book A., Predicting future implied volatility surface using TDBP-learning. Working Paper, University of Paris 6 Pierre et Marie Curie.

[2003] Bondarenko O., Estimation of risk-neutral densities using positive convolution approximation. *Journal of Econometrics*, **116**, pp 85–112.

[2018] Brostrom A., Kristiansson R., Exotic derivatives and deep learning. Degree Project in Mathematics, KTH Royal Institute of Technology, Stockholm, Sweden.

[1995] Campa J.M., Chang K., Testing the expectations hypothesis on the term structure of volatilities. *Journal of Finance*, **50**, pp 529–47.

[1997] Campbell J.Y., Lo A.W., MacKinlay A.C., The econometrics of financial markets. Princeton University Press, New jersey.

[2019] Cao J, Chen J., Hull J., A Neural network approach to understanding implied volatility movements. Working Paper, Joseph L. Rotman School of Management University of Toronto.

[2007] Castagna A, Mercurio F., The vanna-volga method for implied volatilities, *Risk*, January.

[2013] Cavallo E, Galiani S., Noy I., Pantano J., Catastrophic natural disasters and economic growth. *Review of Economics and Statistics*, **95**, (5), pp 1549–1561.

[2016] Chen S, Zhou Z., Li S., An efficient estimate and forecast of the implied volatility surface: A nonlinear Kalman filter approach. *Economic Modelling*, **58**, pp 655–664.

[2019] Chen S, Zhang Z., Forecasting implied volatility smile surface via deep learning and attention mechanism. Working Paper.

[2015] Clevert D-A., Unterthiner T., Hochreiter S., Fast and accurate deep network learning by exponential linear units (elus). Working Paper, ArXiv e-prints.

[2002] Cont R., Tankov P., Calibration of jump-diffusion option-pricing models: A robust non-parametric approach. Ecole Polytechnique, CMAP, September.

[2002b] Cont R., da Fonseca J., Dynamics of implied volatility surfaces. *Quantitative Finance*, **2**, pp 45–60.

[2017] Culkin R., Das S.R., Machine learning in finance: The case of deep learning for option pricing. Working Paper, Santa Clara University.

[1988] Cybenko G., Continuous valued neural networks with two hidden layers are sufficient. Technical Report, Department of Computer Science, Tufts University, Medford, MA.

[1989] Cybenko G., Approximations by superpositions of sigmoidal functions. *Mathematics of Control, Signals, and Systems*, **2**, (4), pp 303–314.

[2006] Daglish T., Hull J., Suo W., Volatility surfaces: Theory, rules of thumb and empirical evidence. Working Paper, August.

[2014] Deng L., Yu D., Deep learning: Methods and applications. Now Publishers.

[1999] Derman E., Regimes of volatility. *Risk*, **4**, pp 55–59.

[1998] Dumas B., Fleming J., Whaley R., Implied volatility functions: Empirical tests. *Journal of Finance*, **53**, (6), pp 2059–2106.

[2003] Durrleman V., A note on initial volatility surface. Working Paper.

[2005] Fengler M.R., Semiparametric modeling of implied volatility. Springer Finance, Springer-Verlag Berlin Heidelberg.

[2007] Fengler M., Hardle W., Mammen E., A semiparametric factor model for implied volatility surface dynamics. *Journal of Financial Econometrics*, **5**, pp 189–218.

[1989] Funahashi K., On the approximate realization of continuous mappings by neural networks. *Neural Network*, **2**, (3), pp 183–192.

[2004] Gatheral J., A parsimonious arbitrage-free implied volatility parameterization with application to the valuation of volatility derivatives. Presentation at Global Derivatives.

[2003] Gencay R., Salih A., Degree of mispricing with the Black-Scholes model and nonparametric cures. *Annals of Economics and Finance*, **4**, pp 73–101.

[2006] Goncalves S., Guidolin M., Predictable dynamics in the S&P 500 index options implied volatility surface. *The Journal of Business*, **79**, (3), pp 1591–1635.

[2013] Graves A., Generating sequences with recurrent neural networks. Working Paper, arXiv:1308.0850.

[2002] Hagan P.S., Kumar D., Lesniewski A.S., Woodward D.E., Managing smile risk. *Wilmott Magazine*, pp 84–108.

[2013] Hahn J.T., Option pricing using artificial neural networks: An Australian perspective. Bond University, Faculty of Business, PhD.

[1992] Harvey C.R., Whaley R.E., Market volatility prediction and the efficiency of the S&P 100 index option market. *Journal of Financial Economics*, **31**, (1), pp 43–73.

[2018] Hasler M., Jeanneret A., The dynamics of the implied volatility surface. Working Paper.

[2017] Hernandez A., Model calibration with neural networks. Risk.

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

[1989] Hornik K., Stinchcombe M., White H., Multilayer feedforward networks are universal approximators. *Neural Networks*, **2**, (5), pp 359–366.

[1990] Hornik K., Stinchcombe M., White H., Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural Networks*, **3**, (5), pp 551–560.

[1991] Hornik K., Approximation capabilities of multilayer feedforward networks. *Neural Networks*, **4**, (2), pp 251–257.

[2019] Horvath B., Muguruza A., Tomas M., Deep learning volatility: A deep neural network perspective on pricing and calibration in (rough) volatility models. Preprint, arXiv:1901.09647v1.

[2018] Hosker J., Djurdjevic S., Nguyen H., Slater R., Improving VIX futures forecasts using machine learning methods. *SMU Data Science Review*, **1**, (4), Article 6.

[1994] Hutchinson J.M., Lo A.W., Poggio T., A nonparametric approach to pricing and hedging derivative securities via learning networks. *Journal of Finance*, **49**, (3), pp 851–889.

[2018] Huynh S., Modelling and forecasting implied volatility using neural network. Thesis, Department of Finance, Hanken School of Economics.

[1999] Jackwerth J.C., Option-implied risk-neutral distributions and implied binomial trees: A literature review. *Journal of Derivatives*, **7**, (2), pp 66–81.

[2019] Karatas T., Oskoui A., Hirsa A., Supervised deep neural networks for pricing/calibration of vanilla/exotic options under various different processes. Working Paper.

[2020] Kim A., Deep learning for uncertainty measurement. PhD Thesis, School of Business and Economics, Humboldt-Universitat zu Berlin.

[2021] Kim H., Park K., Jeon J., Song C., Bae J., Kim Y., Kang M., Candidate point selection using a self-attention mechanism for generating a smooth volatility surface under the SABR model. *Expert Systems with Applications*, **173**, 114640.

[2009] Ladokhin S., Volatility modeling in financial markets. Master Thesis, VU University Amsterdam.

[2005] Lee R.W., Implied volatility: Statistics, dynamics, and probabilistic interpretation. *Recent advances in applied probability*, Springer, New York, pp 241–268.

[2000] Lewis A.L., Option valuation under stochastic volatility. Finance Press.

[2019] Liu S., Oosterlee C.W., Bohte S.M., Pricing options and computing implied volatilities using neural networks. Working Paper.

[1993] Malliaris M., Salchenberger L., A neural network model for estimating option prices. *Journal of Applied Intelligence*, , pp 193–206.

[1997] Malz A., Estimating the probability distribution of the future exchange rate from option prices. *Journal of Derivatives*, Winter, pp 18–36.

[2018] McGhee W.A., An artificial neural network representation of the SABR stochastic volatility model. Preprint.

[2019] Medvedev N., Multi-step forecast of the implied volatility surface using deep learning. Electronic Theses and Dissertations, 3647, South Dakota State University.

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

[2012] Romo J.M., Dynamics of the implied volatility surface: Theory and empirical evidence. Working Paper.

[2010] Roper M., Arbitrage free implied volatility surfaces. Working Paper, School of Mathematics and Statistics, The University of Sydney, Australia.

[2015] Shi X., Chen Z., Wang H., Yeung D-Y., Wong W-K., Woo W-C., Convolutional LSTM network: A machine learning approach for precipitation nowcasting. arXiv:1506.04214v2.

[2000] Skiadopoulos G., Hodges S., Clewlow L., The dynamics of the S&P 500 implied volatility. *Review of Derivatives Research*, **3**, pp 263–282.

[2014] Srivastava N., Hinton G.E., Krizhevsky A., Sutskever I., Salakhutdinov R., Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, **15**, (1).

[2015] Srivastava N., Mansimov E., Salakhutdinov R., Unsupervised learning of video representations using lstms. In *ICML*.

[2018] Stafford D., Machine learning in option pricing. University of Oulu, Oulu Business School, Master Thesis.

[2017] Stark L., Machine learning and options pricing: A comparison of Black-Scholes and a deep neural network in pricing and hedging DAX 30 index options. Department of Finance, Aalto University School of Business.

[2018] Stone H., Calibrating rough volatility models: A convolutional neural network approach. Preprint, arXiv:1812.05315.

[2018] Sun K., Using machine learning methods to predict implied volatility surfaces for SPX options. Thesis, Operations Research and Financial Engineering, Princeton University.

[2018] Talman A., Yli-Jyra A., Tiedemann J., Natural language inference with hierarchical BiLSTM max pooling architecture. arXiv preprint arXiv:1808.08762.

[2009] Tehranchi M.R., Asymptotics of implied volatility far from maturity. *Journal of Applied Probability*, **46**, (3), pp 629–650.

[2004] Wang Y., Yin H., Qi L., No-arbitrage interpolation of the option price function and its reformation, *Journal of Optimization Theory and Applications*, **120** (3), 627–649.

[2000] Yao J.T., Tan C.L., Time dependent directional profit model for financial time series forecasting. *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks*, July 2000, Como, Italy, **5**, pp 291–296.

[2000b] Yao J.T., Tan C.L., Option price forecasting using neural networks. *Omega*, **28**, pp 455–466.

[2019] Zeng Y., Klabjan D., Online adaptive machine learning based algorithm for implied volatility surface modeling. Working Paper.