A Variational Autoencoder Approach to Conditional Generation of Possible Future Volatility Surfaces

Jacky Chen[*] John Hull[*] Zissis Poulos[*] Haris Rasul[*]
Andreas Veneris[†] Yuntao Wu[*]

November 9, 2023

Abstract

We develop a novel method to generate future possible implied volatility surfaces given a historical sequence of surfaces and extra features such as historical returns. The proposed model architecture is based on a conditional variational autoencoder (CVAE) that encodes historical data and a long short-term memory network (LSTM) that allows for the representation of sequences of observations. The architecture can be used to generate future surfaces conditional on any set of historical data. We apply the model to S&P500 data and show that the model is able to capture the real world dynamics.

Keywords: Implied volatility surfaces, variational autoencoders, long short-term memory

[*] Joseph L. Rotman School of Management, University of Toronto
[†] Department of Electrical and Computer Engineering, University of Toronto

1 Introduction

If the assumptions underlying the model developed by Black and Scholes [1973] and Merton [1973] held, a single volatility could be used to price all options dependent on a particular asset and that volatility would not change through time. In reality, the volatility that must be used in conjunction with the Black-Scholes-Merton model to price an option on an asset at any given time depends on the option's strike price and time to maturity. The volatility as a function of these parameters is referred to as the implied volatility surface (IVS). The IVS changes through time. A key question for a trader responsible for options dependent on a particular asset is therefore: "By how much can the asset price and the IVS change?" Every possible change gives rise to a gain or loss on the trader's portfolio. The trader must be satisfied that, with the hedges contemplated, the risks are acceptable.

There have been many attempts to model the dynamics of IVSs. Examples are Cont and Da Fonseca [2002], Cont et al. [2002], Carmona et al. [2017], Cuchiero et al. [2020], Cohen et al. [2020], Bloch and Böök [2021], Shang and Kearney [2022], Cont and Vuletić [2022], Francois et al. [2022], and Choudhary et al. [2023]. Some of these jointly model the asset price and the volatility surface while others model only the volatility surface.

In this research, we use a variational autoencoder (VAE) to model changes in the asset price and the IVS. This is, as far as we know, a different approach from that taken by other researchers. A VAE is an attractive tool because it requires no assumptions about the nature of the model. It uses historical data to produce a multivariate normal distribution for a number of latent variables that relate the changes in the asset price and IVS to recent asset price returns and recent IVSs. By sampling from the latent variables alternative future scenarios for the asset price and IVS are generated. We apply the methodology to options on the S&P500.

We define the IVS by 25 points: five moneyness levels combined with five times to maturity. Based on the assumptions by VAE models, the actual IVS and asset price should be a random sample from those generated by VAE. We test this by comparing a number of attributes of the actuals with those of the generated data. The attributes are: asset price, level, slope, and skew. The model proposed in this paper has shown to be both a viable and explainable approach for modelling a set of potential next-day volatility surfaces given some sequence of historical information. We show this by highlighting the mean value of generated surfaces can accurately capture the implied volatility of 1 year at-the-money options. We also show that the variation in the surfaces our model produces is directly related to next-day real market volatility conditions. The proposed model architecture is able to distinguish the evolution of IVSs and asset prices between recession and non-recession periods.

The rest of this paper is organized as follows. Section 2 explains the VAE methodology. Section 3 explains how we used daily data on S&P500 option trading to estimate implied volatilities for the 25 pre-determined moneyness/time-to-maturity combinations. Section 4 presents our VAE results for S&P500 data. Conclusions are in Section 5.

2 Methodology

2.1 Variational Auto-Encoders (VAEs)

The basic VAE architecture proposed by Kingma and Welling [2013] is an unsupervised generative model consisting of two parts: an encoder and a decoder.

The encoder infers from observed data, x, the latent factors, z, in the form of a probability distribution $q_\phi(z|x)$ with parameters, ϕ. We choose $q_\phi(z|x)$ to match a multivariate normal distribution, $p(z) = \mathcal{N}(0,1)$. We can ensure the matching of the two probability distributions by minimizing the Kullback-Leibler (KL) divergence:

$$\min_\phi D_{KL}(q_\phi(z|x)||p(z)). \tag{1}$$

The decoder takes the output, z, from the encoder to reproduce the observed data, x. Thus, a suitable objective function for the decoder part is to maximize the marginal log-likelihood of the observed data, x, in expectation over the distribution of the latent variable, z:

$$\max_\theta \mathbb{E}_{p_\theta(z)}(\log p_\theta(x|z)), \tag{2}$$

where θ denotes the parameters of the decoder.

The objective function of a VAE model is therefore:

$$\max_{\theta,\phi} \mathbb{E}_{q_\phi(z|x)}(\log p_\theta(x|z)) - D_{KL}(q_\phi(z|x)||p(z)). \tag{3}$$

Higgins et al. [2017] proposed a regularization factor on the KL divergence:

$$\max_{\theta,\phi} \mathbb{E}_{q_\phi(z|x)}(\log p_\theta(x|z)) - \beta D_{KL}(q_\phi(z|x)||p(z)). \tag{4}$$

Changing β changes the focus of training. A small value for β means that we focus on the accuracy of matching the generated data with observed data. A large β means that we focus more on enforcing structure in the space of hidden variables. This causes the decoder to smoothly interpolate between samples of hidden variables to obtain synthetic data that do not have extreme and unrealistic differences compared to observed data. In practice, an appropriate value of β can be found with hyperparameter tuning.

Typically, the decoder produces the mean value $\mu_{x|z}$, and we assume a unitary covariance. Therefore $\log p_\theta(x|z) = -\frac{1}{2}\|x - \mu_{x|z}\|_2^2$. Finally, the expectation can be approximated by averaging, so the maximum likelihood can be replaced by minimizing a mean squared error, which is the reconstruction loss.

Since we assume the prior $p(z) = \mathcal{N}(0, 1)$, the KL divergence can also be reduced to an exact expression:

$$D_{KL}(q_\phi(z|x)||p(z)) = \frac{1}{2}\sum_i \left(-1 - \log\sigma_i^2 + \sigma_i^2 + \mu_i^2\right), \tag{5}$$

where $z_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$ for each individual latent variable z_i in the vector z. The objective function can then be rewritten as:

$$\begin{aligned} \max\ & -\frac{1}{2}\|x - \mu_{x|z}\|_2^2 - \frac{\beta}{2}\sum_i \left(-1 - \log\sigma_i^2 + \sigma_i^2 + \mu_i^2\right), \text{ or} \\ \min\ & \frac{1}{2}\|x - \mu_{x|z}\|_2^2 - \frac{\beta}{2}\sum_i \left(1 + \log\sigma_i^2 - \sigma_i^2 - \mu_i^2\right). \end{aligned} \tag{6}$$

The basic VAE architecture generates new data samples unconditionally. Sohn et al. [2015] proposed a conditional VAE (CVAE), which generates data based on some given condition c. In this setting, the encoder maps inputs to latent factors according to the distribution $q_\phi(z|x, c)$ using both the observed data x and condition c. The decoder generates x by $p_\theta(x|z, c)$. This gives some control over the generated output. For instance, Ning et al. [2022] use CVAE to generate parameters for SDEs that characterize the behavior of IVS's conditional on observable market states such as spot rates or indices.

VAEs have also been applied to the task of timeseries generation (see Desai et al. [2021]). One-dimensional convolutional layers are used by the encoder to extract time-dependent features and construct the latent space, and transposed convolutional layers are used to reconstruct the timeseries. Trend blocks and seasonality blocks may be incorporated to improve the interpretability of such models. Apart from VAEs, other generative models such as Generative Adversarial Networks (GANs) have also been applied to generate price scenarios with particular focus on tail risk (see Cont et al. [2022]).

2.2 Long Short-Term Memory (LSTM)

The asset price and IVS changes we work with can be viewed as timeseries data. The asset price and IVS today can be dependent on those on yesterday. Recurrent neural networks (RNNs), such as LSTM originally proposed by Hochreiter and Schmidhuber [1997], incorporate the time evolution and can be useful in timeseries generation as shown by Siami-Namini et al. [2018]. Within each recurrent unit of the LSTM, a long term cell state c_t and a short term hidden state h_t are maintained by self-recurrence through four interacting neural network layers:

1) The **forget gate layer** which compares h_{t-1} and current input x_t and decides which elements in cell state c_{t-1} to keep and which to forget (f_t).

2) The **input gate layer** which first decides which cell unit to update (i_t) and then creates new candidate value $\tilde{c}_t$.

3) The **update layer** which updates the cell state c_t using f_t, i_t and $\tilde{c}_t$.

4) The **output layer** which calculates o_t based on x_t and h_{t-1}, and updates the short term memory h_t.

Let $\sigma(\cdot)$ be the sigmoid function. The LSTM model can be represented by the following equations:

$$\begin{aligned}
\text{Forget: }\; f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f), \\
\text{Input: }\; i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i), \\
\tilde{c}_t &= \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \\
\text{Update: }\; c_t &= f_t \times c_{t-1} + i_t \times \tilde{c}_t, \\
\text{Output: }\; o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o), \\
h_t &= o_t \times \tanh(c_t),
\end{aligned} \tag{7}$$

where × is the Hadamard product, *i.e.* element-wise product, and $W \cdot [h, x] = W_h h + W_x x$ is a tensor product of the weights with the concatenated vector $[h, x]$.

There are also several variants of LSTM, but Greff et al. [2017] show that these variants perform similarly. VAE and LSTM or other memory models have been combined to perform tasks in natural language generation (Bowman et al. [2016]) and anomaly detection in timeseries (Lin et al. [2020]).

2.3 Proposed Model

We first define some notation. Let x_t be the realized IV surface on day t, constructed as outlined in Sec. 3. Letting p_t be the asset price on day t, we use log-returns $r_t = \log\left(\frac{p_t}{p_{t-1}}\right)$ to model price changes. Let y_t be an extra feature of interest that can impact volatility surface dynamics, which includes underlying asset returns r_t. In what follows, the subscript t may be omitted whenever we refer to general operations that don't depend on specific days.

To use a VAE to model changes in asset prices and IVSs, we consider the problem: Given the context information, $\mathbf{x}_c = (..., x_{t-3}, x_{t-2}, x_{t-1})$ and $\mathbf{y}_c = (..., y_{t-3}, y_{t-2}, y_{t-1})$, which represents the historical timeseries of realized surfaces and extra features before day t respectively, can we generate possible surfaces and asset returns for day t and after, *i.e.* $\mathbf{x}_n = (x_t, x_{t+1}, x_{t+2}, ...)$ and $\mathbf{r}_n = (r_t, r_{t+1}, r_{t+2}, ...)$?

The basic VAE architecture learns an unconditional probability distribution based on the entire set of observations. It can only generate random samples without conditions, which means that the generated surfaces and extra features are difficult to control. The data points that occur more frequently across observations are the ones more likely to be generated. We are not able to know specifically for which day the surfaces/returns are generated. CVAEs could work by setting the historical timeseries as conditions. However, without controlling time steps, we could have a look-ahead bias when generating new surfaces. For example, a surface generated on day t could also use the latent values for day $t+1$, $t+2$ in the standard CVAE formulation. Secondly, as the length of historical content and length of future values to generate increase, we would need to encode more surfaces and extra features, meaning that the number of parameters will increase, because there is no recurrent units in the CVAE and inputs of each individual day require different parameters to encode. Finally, CVAEs using simple multilayer perceptron (MLP) or convolutional neural network (CNN) can only accept a fixed size input, and cannot generate surfaces based on variable length contexts without breaking the surface structures.

To address these challenges, we propose a new architecture which combines the advantages of CVAEs and LSTM and is suitable for our task. The architecture employs a CVAE as a backbone and contains three components: (1) The encoder that takes the historical context together with the future values $(\mathbf{x}_c, \mathbf{y}_c, \mathbf{x}_n, \mathbf{y}_n)$ to generate a distribution $\mathcal{N}(\mu_{t,i}, \sigma^2_{t,i})$ for each latent variable on each day. The latent representation $\mathbf{z} = (..., z_{t-1}, z_t, z_{t+1}, ...)$ can then be sampled from the distribution. Each latent variable on each day is sampled individually from the corresponding distribution $\mathcal{N}(\mu_{t,i}, \sigma^2_{t,i})$ in the training process. In the generation steps after the model is trained, the latent variables are sampled individually from the standard normal $\mathcal{N}(0, 1)$. (2) A context encoder that generates an efficient representation $\zeta = (..., \zeta_{t-3}, \zeta_{t-2}, \zeta_{t-1})$ of the historical context $(\mathbf{x}_c, \mathbf{y}_c)$. (3) Finally, a decoder that uses ζ as condition, and the sampled latent variables $\mathbf{z}$ to reconstruct the surfaces and returns in the future, $\hat{\mathbf{x}}_n$, $\hat{\mathbf{r}}_n$. The LSTM is a core part in each of the three components. With limited number of parameters and no look-ahead bias, it efficiently captures the time dependencies of the inputs when generating embeddings in the encoder and context encoder and generates outputs based on previous day conditions in the decoder, where both the inputs and outputs can be of any length. Also, with the addition of LSTM, we can separate the tasks of capturing temporal and spatial features.

We define the objective function for each day t:

$$L(\phi, \theta, x_t, y_t) = \frac{1}{HW}\|x_t - \hat{x}_t\|_2^2 + \alpha\|r_t - \hat{r}_t\|_2^2 - \frac{\beta}{2}\sum_{i=1}^{L}\left(1 + \log\sigma_{t,i}^2 - \sigma_{t,i}^2 - \mu_{t,i}^2\right), \qquad (8)$$

where H and W are the dimensions of the IVSs, L is number of latent variables. $\frac{1}{HW}\|x_t - \hat{x}_t\|_2^2$ is the mean squared error of points on the generated surface $\hat{x}_t$ and $\|r_t - \hat{r}_t\|_2^2$ is the mean squared error of generated underlying asset returns feature $\hat{r}_t$.[1] α is the regularization factor for the error on returns. Integrating historical returns in the loss function accounts for underlying asset dynamics, which aims to create more accurate volatility surface predictions that align with observed market trends. $-\frac{\beta}{2}\sum_{i=1}^{L}\left(1 + \log\sigma_{t,i}^2 - \sigma_{t,i}^2 - \mu_{t,i}^2\right)$ is the regularized KL divergence of the day. This objective function is computed for each generated day in a batch, and averaged across all days within a batch.

3 S&P500 Implied Volatility Surface Estimation

S&P500 Call Option data from January 2000 to February 2023 were downloaded from OptionMetrics.[2] We filtered the data in several ways. Firstly, we eliminated all options that didn't have a reported implied volatility in the database. Additionally, we retained only those options with an open interest greater than 0, moneyness (defined as the strike price divided by the index level K/S) ranging between 0.7 and 1.3, and a time to maturity from 1 month up to 2 years. We then established a 5 × 5 Time-to-Maturity (ttm) by Moneyness grid for the implied volatility surface, with ttm values set at 1, 3, 6, 12, and 24 months and Moneyness values of 0.7, 0.85, 1, 1.15, and 1.3. It's worth noting that the 5 × 5 implied volatility surface isn't directly observable from the available options on any given trading day due to variations in available strike prices and expiration dates. We used a search algo-

[1] We also experiment generating the complete extra feature vector y_t and compute the loss as $\frac{1}{E}\|y_t - \hat{y}_t\|_2^2$, with simple configuration, but it introduces bias towards specific surface properties and doesn't help with the performance.

[2] We gained the access through https://wrds-www.wharton.upenn.edu/pages/about/data-vendors/optionmetrics/

rithm outlined in Cao et al. [2021] to determine the implied volatilities for standard reference points.

We use $x[\mathrm{ttm}, \mathrm{Moneyness}]$ to index a single point on the surface, with ttm in the order of years. For example, $x[0.25, 1]$ means the implied volatility when ttm= 3 months and Moneyness $\frac{K}{S} = 1$.

4 Results

4.1 Extra feature definition

One of the extra features used is the daily log return of the asset price. We also define two other extra features. The skew is defined as

$$\mathrm{skew} = \frac{x\left[1, 0.85\right] + x\left[1, 1.15\right]}{2} - x\left[1, 1\right]. \tag{9}$$

and the slope is defined as

$$\mathrm{slope} = x\left[2, 1\right] - x\left[0.25, 1\right]. \tag{10}$$

These features are used in training and evaluation.

4.2 Training and Model Performance

We use data from 2000-01-03 to 2015-11-27 (4000 valid days) for training, data from 2015-11-30 to 2019-11-18 (1000 days) for validation and 2019-11-19 to 2023-02-24 (1000 days) for testing. The model uses 3 convolution layers of output size 5 for surface encoding, and the identity for the encoding of any extra features. The latent and context embedding dimensions are set to 5. We use 2 layer memory with 100 hidden units in each layer and apply a 0.2 dropout rate. The KL divergence weight is set to $\beta = 10^{-5}$. We train, validate and test on a variable sequence length between 4 and 10 inclusive, with context length being one less than the sequence length. The validation set is used to select the optimal model during 500 epoches of training. The model is defined and trained using PyTorch 1.13[3] on a RTX 3080 Ti linux machine[4]. All random seeds are reset to 0 before dataset preparation and before training. 64-bit floating points are used in calculation for a better precision. Several different feature configurations are tested, and we find that the best model is to use all returns, skews and slopes as extra features in addition to the surfaces, while minimizing loss on surfaces and returns. The model gets a 8.97×10^{-4} reconstruction loss on surface,

[3] The code is compatible with PyTorch 2.0, but the performance varies.

[4] `https://www.runpod.io/console/pods`

6.54×10^{-5} reconstruction loss on return and 2.15 KL divergence loss in validation set, and 2.43×10^{-2} reconstruction loss on surface, 1.86×10^{-4} reconstruction loss on return and 2.69 KL divergence loss in test set. We need to note that for the test set, data starting from around 2021-02-24 might have a higher loss as most of the grid points with a large time to maturity cannot be accurately calculated when constructing the dataset. We restrict the time range for evaluation in the next section to avoid the potential unwanted artifacts in the generated surfaces.

4.3 Evaluating Generated Scenarios

In order to leverage the model as a tool for decision-making, we would like to understand the variation in the outcomes it generates. The following tests are used to measure the range of surfaces the model can produce as well examine different scenarios that can explain instances of larger variation. The first set of tests involve developing histograms and z-score plots to visualize where realized values lie in a distribution produced by generated values. The plots are created for the single point on the surface $x[1, 1]$, the return, the skew and the slope. The plots use data in the training set only (first 4000 days). The surfaces and returns are generated conditioned on surfaces and extra features with 3 day context $t - 1$, $t - 2$ and $t - 3$, so there are 3997 distributions in total. The distribution of each day is based on 1000 samples of generated surfaces and returns on that day. We divide the distributions into quartiles (denoted Q1, Q2, Q3, and Q4), and build histograms based on the number of days on which the realized value lies in a specific quartile. The z-scores are estimated using sample mean and sample standard deviation, calculated based on the 1000 samples per day,

$$z = \frac{x - \bar{x}}{S}, \tag{11}$$

where x is the realized value, $\bar{x}$ is the mean of the generated values, and S is the standard deviation of the generated values. We would expect that the histogram plots are more concentrated towards Q2 and Q3 and the z-scores are centered around zero, preferably within ± 1.96 (95% confidence interval). The plots can be found in Fig. 1 and 2. From the histogram, the single point on the surface seems to be under-estimated by the model, as most of the realized points lie in the Q4. However, the z-scores are mostly concentrated within the range $[-2, 2]$, meaning that most of the realized values lie in 95% interval of the distribution. Most of the realized returns lie in Q2 and Q3, close to the mean. From the z-scores, most of the realized returns are in the positive side, meaning that the mean values of generated mean are under-estimating the realized returns. In terms of the additional surface features: skews and slopes. The skews are better learned than slopes, with a more even distribution, and fewer outliers. Also the z-scores are closer to zero for skews. Potentially, optimizing loss on the returns lowers the performance on the surfaces.

To examine the model's capability, particularly the context encoder, in differentiating volatility dynamics during distinct market regimes, we conduct principal component analysis (PCA). The focus of this analysis is to understand if the model is effective in distinguishing between the dynamics of NBER recession periods and dates where markets are more stable. First, we examine the context encoding process of real S&P500 surface data. Same as in the previous test, we use a 3 day context (x_{t-3}, y_{t-3}), (x_{t-2}, y_{t-2}), and (x_{t-1}, y_{t-1}) to generate the 3×5 embeddings $(\zeta_{t-3}, \zeta_{t-2}, \zeta_{t-1})$. These ζ values are then concatenated into a one dimensional vector χ, where the first 5 values are from ζ_{t-3}, the middle 5 values are from ζ_{t-2}, and the last 5 values are from ζ_{t-1}. This χ value is calculated for 5300 days from 2000-01-06 to 2021-02-02. We then apply PCA on $[\chi_1, \chi_2, ..., \chi_{5300}]$ to extract out the first and second principal components (FPC and SPC) for each day. We categorize the days according to if they were in a NBER recession period, and then plot the principal components. The NBER recession periods within our analysis are 2001-04-01 to 2001-11-30, 2008-01-01 to 2009-06-30 and 2020-03-01 to 2020-04-30[5]. The result is shown in the first image of Fig. 3. The majority of the stable market dates, represented by blue dots, cluster near the origin in the bottom left corner. In contrast, the principal components for the NBER recession dates, indicated by red triangles, are scattered to the top right corner. This noticeable separation indicates the model's ability to capture distinct volatility characteristics within varying market regimes and periods of recession. To further validate our VAE model, we apply this PCA test to a generated set of S&P500 surfaces to see if our model is producing similar outcomes to what was seen for real surface data. We conduct the same experiment now utilizing synthetic surfaces for the same range of dates used in the methodology described above. To do this, 1000 surfaces and returns are sampled for each day from 2000-01-06 to 2021-02-02, conditioned on surfaces and extra features with a 3 day context $t-3$, $t-2$ and $t-1$. We calculate the average of the 1000 surfaces and returns for each day. The skew and slope features are calculated based on the average surface of each day. Then, these generated values are used as inputs to the context encoder to get the $\hat{\zeta}$ and $\hat{\chi}$ values and the FPC, SPC. The plot is shown in the right image of Fig. 3. This plot is almost identical to the plot on the left. The generated values are capturing the realized dynamics with some variability.

In addition to the tests conducted on S&P500 data, we have developed a similar analysis on a more controlled environment using the SABR option price model seen in Appendix C which provides a well-defined universe of volatility surfaces. This additional study helps to investigate the explainability of the model's performance by controlling complexities that

[5] Monthly periods from `https://fred.stlouisfed.org/series/USREC`

may arise with real financial data.

4.4 Validation from Real Market Dynamics

To assess the accuracy of our VAE-generated surfaces in simulating real-world market dynamics, we considering the following regression analyses that compare our generated outputs to real market data:

$$x_t\left[\text{ttm}, \text{moneyness}\right] = \alpha + \beta_1 \mu\left(\hat{x}_t\left[\text{ttm}, \text{moneyness}\right]\right) + \epsilon_t, \quad (12)$$

$$\left|x_t\left[\text{ttm}, \text{moneyness}\right] - x_{t-1}\left[\text{ttm}, \text{moneyness}\right]\right| = \alpha + \beta_1 \sigma\left(\hat{x}_t\left[\text{ttm}, \text{moneyness}\right]\right) + \epsilon_t. \quad (13)$$

The mean value of generated surfaces should capture the actual level of each individual surface grid point, since in the training process, the mean squared error between the generated surfaces and the realized surfaces are optimized. The results are shown in Table 1 for all 25 grid points of surfaces we modelled. The coefficients β_1 are close to 1 for most of the surface grid points, with high R^2s, suggesting that the mean of our daily generated distributions are accurate representations of next-day surfaces. We use the standard deviation of generated surfaces as an insight into the challenges of predicting the next day's situation based on the historical context sequence. Larger standard deviations in generated values may suggest that the next day's realized surface values maybe significantly different from previous context history. The results are shown in Table 2. The coefficients β_1 are mostly positive with high significance, except for the ttm=3 month and $K/S = 0.7$ grid point. The positiveness of the coefficient shows that a higher standard deviation is correlated with a larger change in the realized values. The R^2 are ranging from 6% to 17.5% for $K/S = 0.85$, 1 and 1.15 points with 3 month, 6 month or 1 year maturity. The edge and corner points with 1 month/2 year time to maturity or $K/S = 0.7$, $K/S = 1.3$ have a lower performance. The reason could be that we use zero-padding in convolutions to preserve the spatial dimensions and the features of edge points are not well-exploited.

5 Conclusion

In this paper we propose a new model that incorporates the historical context and additional features to generate distributions of IV surfaces and corresponding returns on a single day. The same method can be easily extended to generate distributions on multiple days in the future, based on a history of any length. The performance is also consistent with a SABR dataset which is a well-defined simplified model for asset pricing and volatility.

We showed that the distributions learned by our VAE model with extra features using real data can be explained by market conditions and volatility indicators. This provides valuable insight into the model's ability to comprehend underlying market conditions.

The potential future work to expand on this analysis includes: 1) Generating surfaces and returns of multiple days in the future in one shot, and investigating how the errors are propagated. 2) Study how each point on the generated surfaces is affected by historical values. 3) Generalize the model so that it works on other existing datasets to compare its performance with CVAE or a conventional timeseries VAE.

Figure 1: Histogram Plot for S&P500 Surfaces and Returns

This figure shows the histogram plots of where the realized point lies in the VAE distribution. The first rows shows the distribution and z-scores for the levels. The second row shows the distribution and z-scores for the returns.

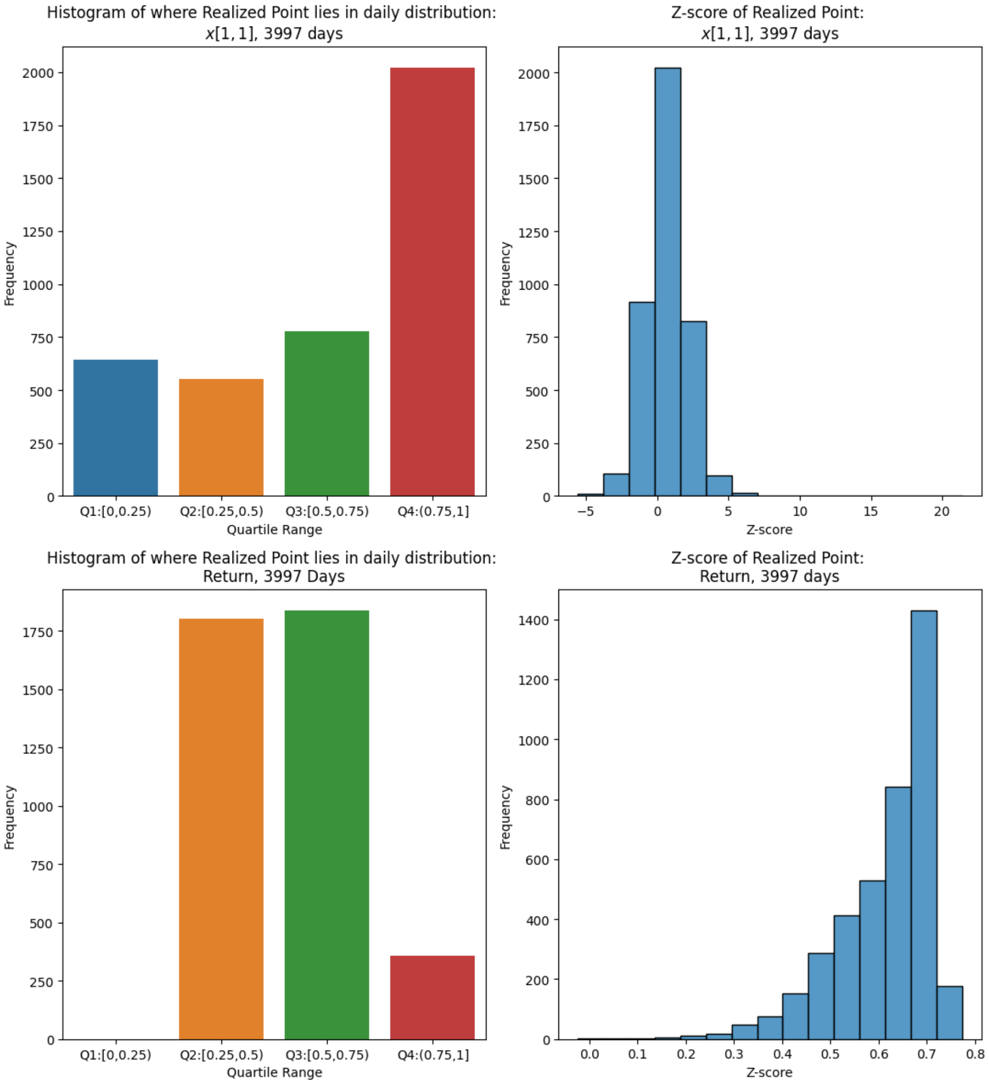


Figure 2: Histogram Plot for S&P500 Surface Structural Properties

This figure shows the histogram plots of how the structural properties of generated surfaces are distributed. The first rows shows the distribution and z-scores for the skews. The second row shows the distribution and z-scores for the slopes.

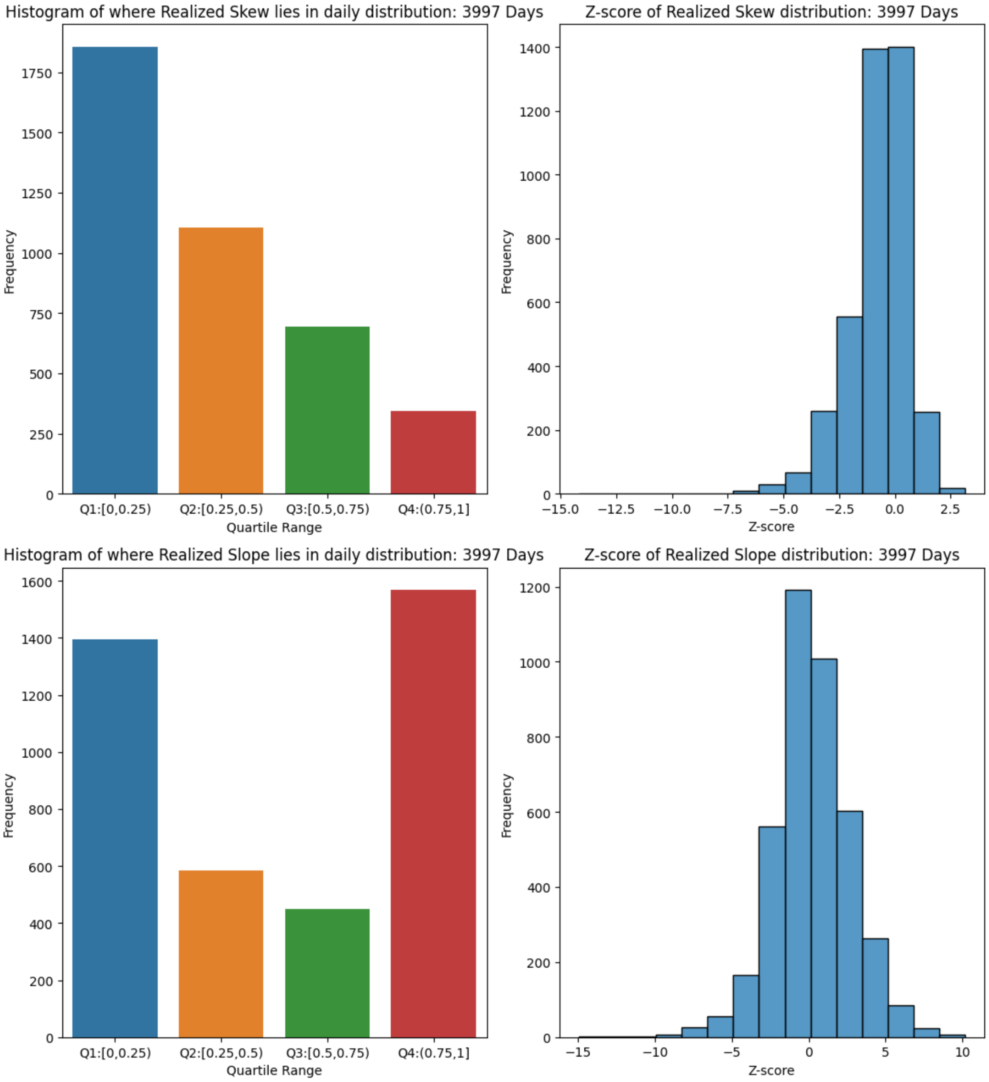


Figure 3: PCA of Context Embeddings

The left figure shows the first and second principal components of context embedding of the realized surfaces. The right figure shows the first and second principal components of context embedding of the mean of generated surfaces. In both figures, the horizontal axis represents the first principal component (FPC), and the vertical axis represents the second principal component (SPC). The data points in NBER recessions are plotted as red triangles. The data points in normal years are plotted as blue circles.

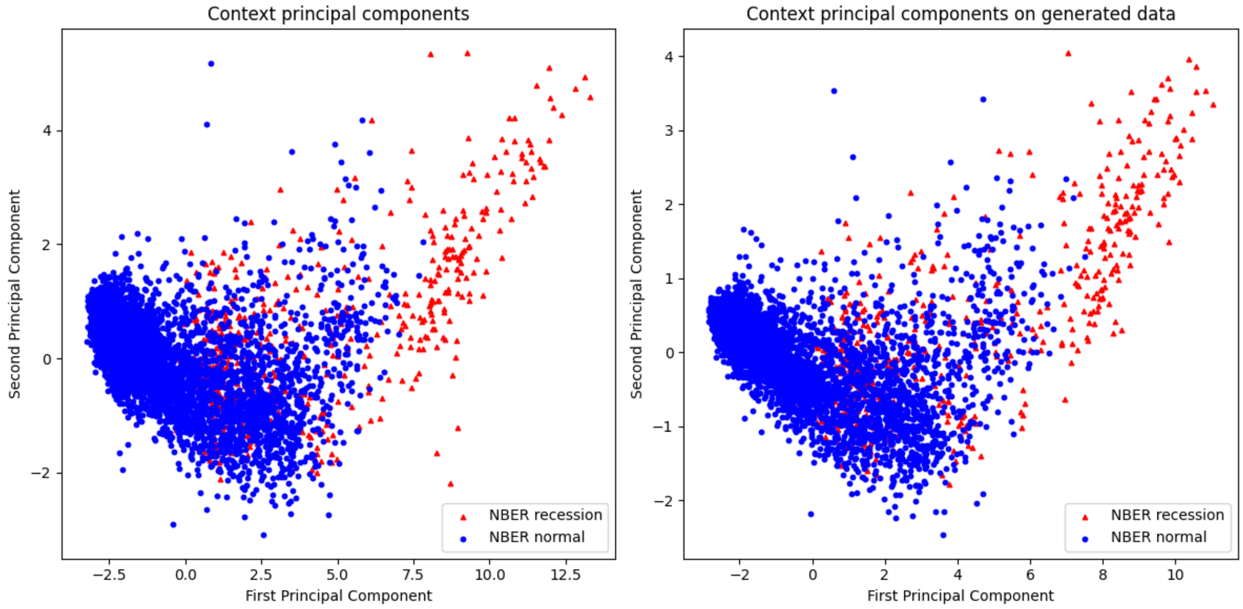


Table 1: Regression for IVSs with Mean Value

This table reports the results of following regression:

$$x_t\left[\text{ttm}, \text{moneyness}\right] = \alpha + \beta_1 \mu\left(\hat{x}_t\left[\text{ttm}, \text{moneyness}\right]\right) + \epsilon_t.$$

The regression is performed on all 25 points of the surfaces. The columns represent the moneyness grids. The rows represent the time-to-maturity grids. Panel A reports the coefficient β_1 for each grid point. *, **, *** denote the associated p-values below the 10%, 5%, 1% levels, respectively. Panel B reports the R^2 of each regression.

Panel A. coefficients

	K/S=0.7	K/S=0.85	K/S=1	K/S=1.15	K/S=1.3
1 month	2.3170***	1.2592***	1.1019***	0.9799***	1.3478***
3 month	1.6568***	1.1160***	1.0683***	1.0122***	1.0980***
6 month	1.3631***	1.1260***	1.0742***	1.0258***	0.9859***
1 year	1.2570***	1.0726***	1.0488***	1.0310***	0.9916***
2 year	1.1493***	1.0605***	1.4443***	0.9980***	0.9368***

Panel B. R^2

	K/S=0.7	K/S=0.85	K/S=1	K/S=1.15	K/S=1.3
1 month	0.243	0.674	0.876	0.852	0.421
3 month	0.260	0.874	0.935	0.921	0.683
6 month	0.626	0.915	0.954	0.939	0.476
1 year	0.703	0.940	0.960	0.935	0.866
2 year	0.553	0.907	0.046	0.897	0.825

Table 2: Regression for Change in IVSs with Standard Deviation

This table reports the results of following regression:

$$|x_t\,[\text{ttm}, \text{moneyness}] - x_{t-1}\,[\text{ttm}, \text{moneyness}]\,| = \alpha + \beta_1 \sigma\,(\hat{x}_t\,[\text{ttm}, \text{moneyness}]) + \epsilon_t.$$

The regression is performed on all 25 points of the surfaces. The columns represent the moneyness grids. The rows represent the time-to-maturity grids. Panel A reports the coefficient β_1 for each grid point. *, **, *** denote the associated p-values below the 10%, 5%, 1% levels, respectively. Panel B reports the R^2 of each regression.

Panel A. coefficients

	K/S=0.7	K/S=0.85	K/S=1	K/S=1.15	K/S=1.3
1 month	1.9927***	1.6149***	1.3298***	1.6641***	4.1280***
3 month	-0.2979***	1.3629***	1.6954***	1.6697***	2.7616***
6 month	0.9239***	1.3784***	1.4553***	1.2610***	3.5984***
1 year	1.8200***	1.2768***	0.9585***	1.1105***	0.5551***
2 year	1.1589***	1.1655***	5.3379**	1.0160***	0.2754***

Panel B. adjusted R^2

	K/S=0.7	K/S=0.85	K/S=1	K/S=1.15	K/S=1.3
1 month	0.010	0.036	0.190	0.160	0.099
3 month	0.001	0.059	0.168	0.143	0.105
6 month	0.026	0.080	0.165	0.175	0.043
1 year	0.019	0.107	0.141	0.170	0.014
2 year	0.031	0.059	0.001	0.026	0.002

Appendix

A Proposed Model Details

Let C be the context length (i.e., the number of days of history considered relevant), N be the number of days we want to generate, $T = C + N$ be the overall sequence length, and L be the number of latent variables. Let $x_t \in \mathbb{R}^{H \times W}$ be the realized IV surface on day t, where H represents time-to-maturity grid size, W represents moneyness (K/S) grid size. Let p_t be the asset price on day t. We use log-return $r_t = \log\left(\frac{p_t}{p_{t-1}}\right)$ to model price changes. Let $y_t \in \mathbb{R}^E$ be the extra features of interest, where E is the number of extra features. One of such extra feature can be r_t. Let $\mathbf{x}_c = (x_{t-C}, ..., x_{t-1}) \in \mathbb{R}^{C \times H \times W}$ be the historical surfaces from day $t-C$ to $t-1$, $\mathbf{x}_n = (x_t, ..., x_{t+N-1}) \in \mathbb{R}^{N \times H \times W}$ be the surfaces we want to generate for day t to day $t+N-1$. Let $\mathbf{y}_c = (y_{t-C}, ..., y_{t-1}) \in \mathbb{R}^{C \times E}$ be the extra features from from day $t-C$ to $t-1$, $\mathbf{y}_n = (y_t, ..., y_{t+N-1}) \in \mathbb{R}^{N \times E}$ be the extra features from from day t to $t+N-1$. If a, b are two constants, then $[a, b]$ is the usual interval notation. If $a = (a_1, ..., a_n)$, $b = (b_1, ..., b_m)$ are two vectors/tensors, then $[a, b] = (a_1, ..., a_n, b_1, ..., b_m)$ is the vector/tensor concatenation.

The model consists of three parts:

1. Encoder: takes $\mathbf{x} = [\mathbf{x}_c, \mathbf{x}_n] \in \mathbb{R}^{T \times H \times W}$ and $\mathbf{y} = [\mathbf{y}_c, \mathbf{y}_n] \in \mathbb{R}^{T \times E}$ and builds a distribution $\mathcal{N}(\mu_{t,i}, \sigma_{t,i})$ for each day t and each latent variable $z_{t,i}$. The latents $\mathbf{z} = (z_{t-C}, ..., z_{t+N-1}) \in \mathbb{R}^{T \times L}$ are then sampled individually from the corresponding distributions. $\forall i \in [t-C, t+N-1]$, the encoder generates the daily latent vector z_i by $q_\phi(z_i | \mathbf{x}_{[t-C,...,i]}, \mathbf{y}_{[t-C,...,i]}, i)$, where $\mathbf{x}_{[t-C,...,i]} = (x_{t-C}, ..., x_i)$.

2. Context Encoder: takes $\mathbf{x}_c$ and $\mathbf{y}_c$ and generates a compressed representation of the context $\zeta = (\zeta_{t-C}, ..., \zeta_{t-1}) \in \mathbb{R}^{C \times L}$.[6] $\forall i \in [t-C, t-1]$, $\zeta_i = \text{Embed}(\mathbf{x}_{c,[t-C,...,i]}, \mathbf{y}_{c,[t-C,...,i]})$, where $\mathbf{x}_{c,[t-C,...,i]} = (x_{t-C}, ..., x_i)$.

3. Decoder: takes $\mathbf{z}$ from the encoder and ζ from the context encoder and reconstructs $\hat{\mathbf{x}}_n = (\hat{x}_t, ..., \hat{x}_{t+N-1})$, $\hat{\mathbf{y}}_n = (\hat{y}_t, ..., \hat{y}_{t+N-1})$. Each $\hat{x}_i,\ \hat{y}_i$, $i \in [t, t+N-1]$ is reconstructed by sampling from $p_\theta(x_i, y_i | \zeta, \mathbf{z}_{[t-C,...,i]}, i)$, where $\mathbf{z}_{[t-C,...,i]} = (z_{t-c}, ..., z_i)$.

In each of the component, we use convolutional networks or multi-layer perceptrons to encode/decode the surface structures on each day, and LSTM to capture the time dependencies of any lengths. Graphical illustrations of each part and the complete model are shown in

[6] We can configure the context embedding size to values other than L, but we use L as embedding size here to be consistent with the main article.

Figure A1, A2, A3, A4. All convolutional/transpose convolutional/linear layers within the CNN/TCNN (transpose convolutional neueral network)/MLP blocks are followed by ReLU activation ($\max(x, 0)$), unless identity function (Id) or single linear layer is used in place of MLPs, or the layer is the last layer of TCNN/MLP in the decoder. The TCNN/MLP blocks used in the decoder are symmetric to the CNN/MLP blocks used in the encoder so that the shape of input and output as well as the process of encoding and decoding match. For all convolutional and transpose convolutional layers, the kernel size is fixed to 3×3 with stride 1 and equal size padding, so the grid size won't change after multiple convolutions. f_1, f_2, g_1, g_2 are embedding functions performed by these blocks in encoder and context encoder.

Figure A1: Encoder

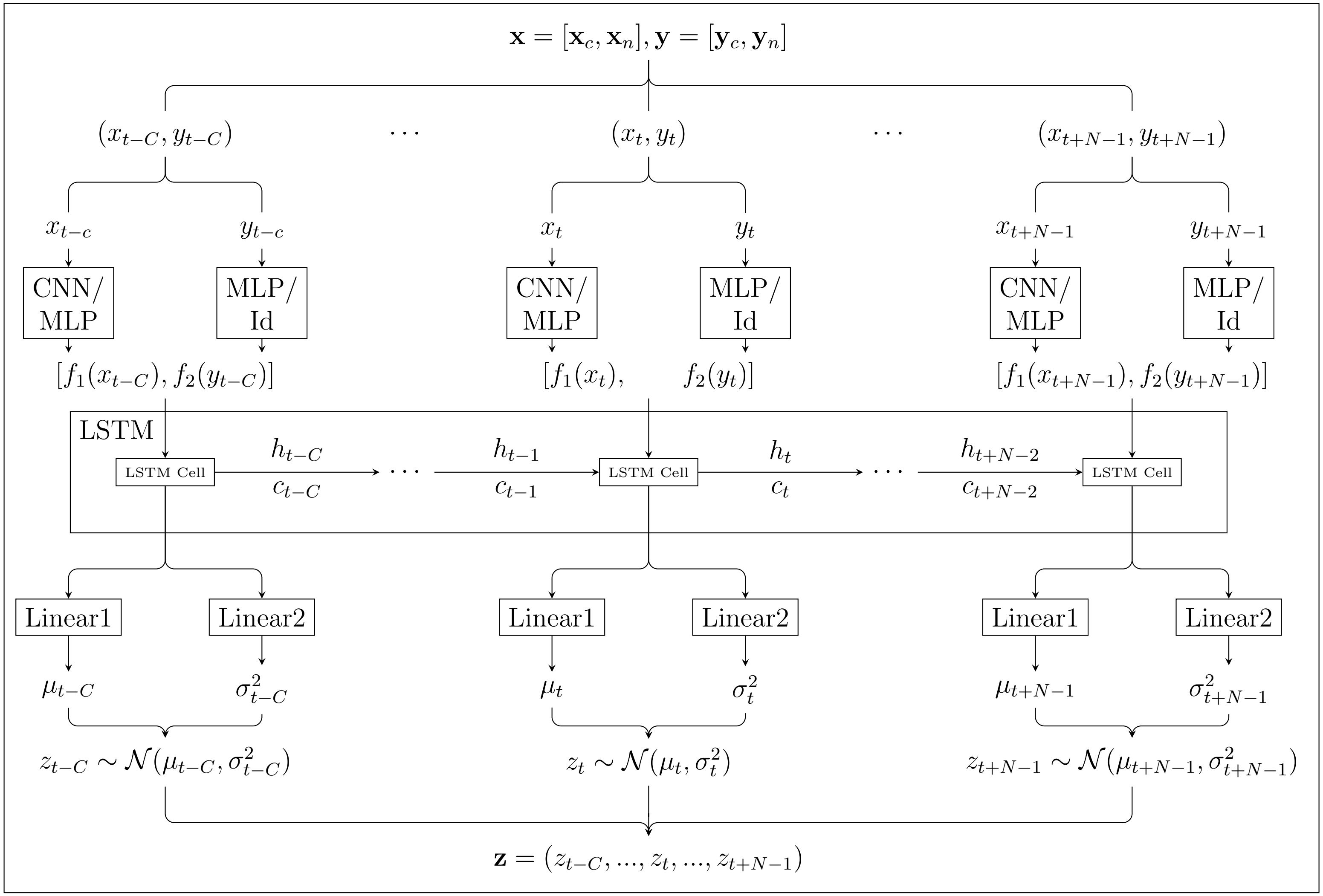


Figure A2: Context Encoder

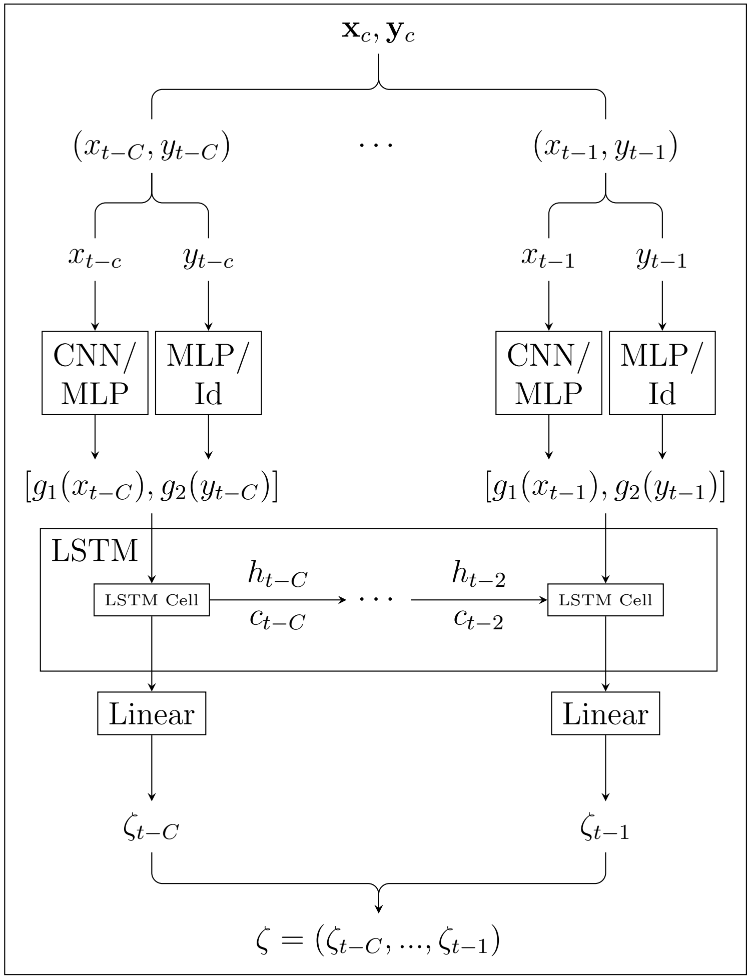


Figure A3: Decoder

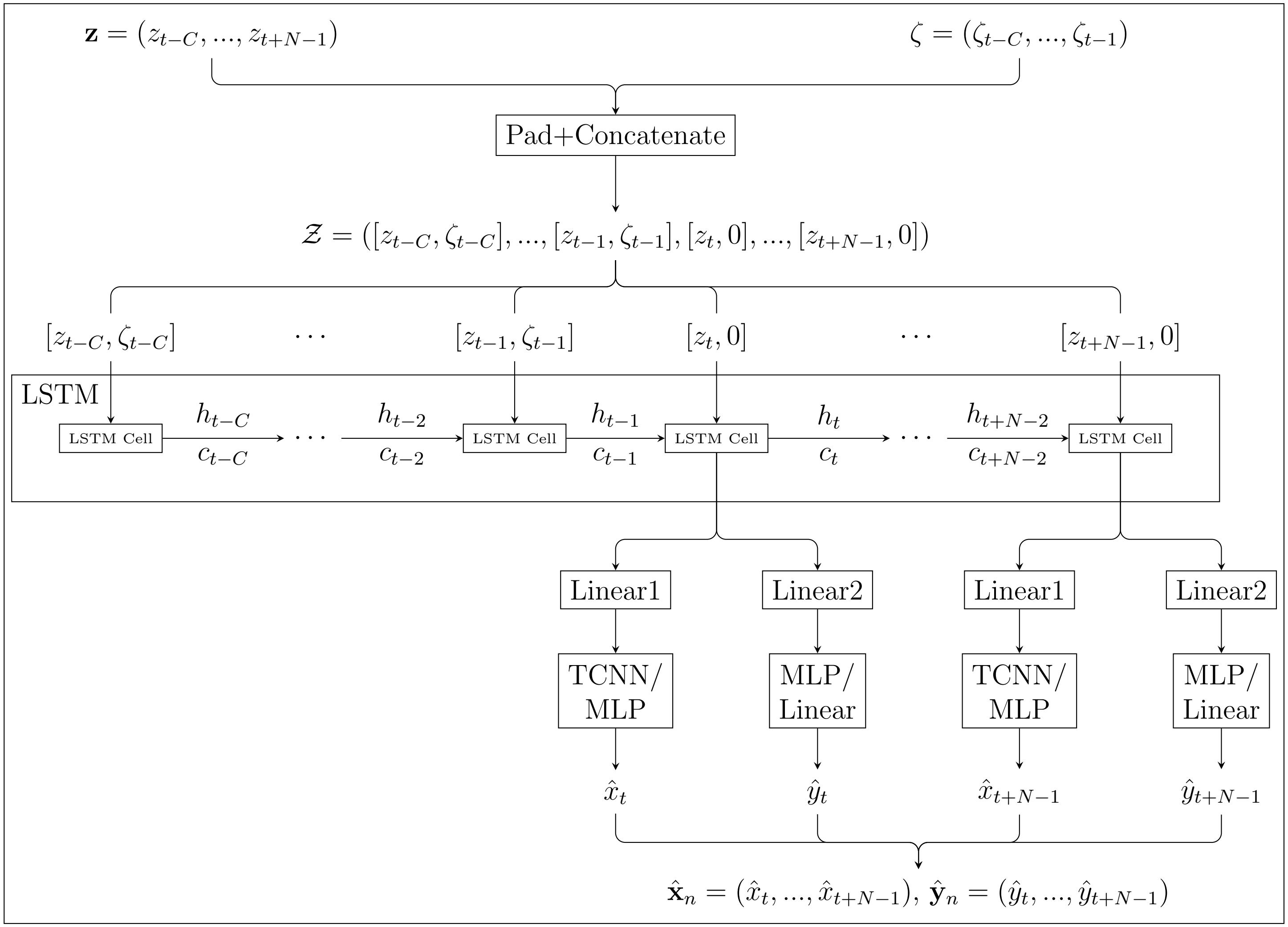
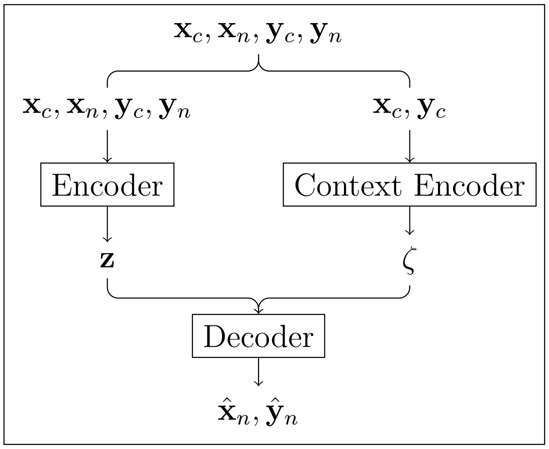


Figure A4: Overall Architecture


$\mathbf{x}_c, \mathbf{x}_n, \mathbf{y}_c, \mathbf{y}_n$
$\mathbf{x}_c, \mathbf{x}_n, \mathbf{y}_c, \mathbf{y}_n$
$\mathbf{x}_c, \mathbf{y}_c$
Encoder
Context Encoder
$\mathbf{z}$
ζ
Decoder
$\hat{\mathbf{x}}_n, \hat{\mathbf{y}}_n$

B Arbitrage Conditions

Following Gatheral and Jacquier [2013] and Bergeron et al. [2022], we specify static arbitrage conditions as follows: let $M = \frac{K}{S}$ be the moneyness, t be the time to maturity. Define $w(t, M) = t \cdot x(t, M)$ as the total implied variance surface, where x is the IVS on a single day. An IVS is free of *calendar arbitrage* if

$$\frac{\partial w}{\partial t} \geq 0. \tag{A1}$$

Let $w' = \frac{\partial w}{\partial M}$ and $w'' = \frac{\partial^2 w}{\partial M^2}$. The IVS is free of *butterfly arbitrage* if

$$\left(1 - \frac{Mw'}{2w}\right)^2 - \frac{w'}{4}\left(\frac{1}{w} + \frac{1}{4}\right) + \frac{w''}{2} \geq 0. \tag{A2}$$

An IVS is said to be free of static arbitrage if the conditions in Eq. A1 and A2 are satisfied.

We test the arbitrage conditions on the realized IVSs constructed by calibration on the available options. Among 5300 surfaces from 2000-01-06 to 2021-02-02, there are 4022 surfaces with at least one calendar arbitrage opportunity and 3859 surfaces with at least one butterfly arbitrage opportunity. The generated surfaces on each day can significantly reduce calendar arbitrage opportunities. The generated surfaces are unable to avoid butterfly arbitrage, but they generally do not create additional butterfly opportunities on each day.

It is possible to incorporate the calendar arbitrage condition into the calibration of IVSs on the available options we have, and the learned model can further reduce the calendar arbitrage opportunity with the better calibrated data. However, incorporating the butterfly arbitrage condition into the calibration may lead to realized market volatility being captured less well.

To further reduce the possibility of static arbitrage in the generated surfaces, we could extend our loss function in Eq. 8 similarly to Bergeron et al. [2022].

C Controlled SABR Environment

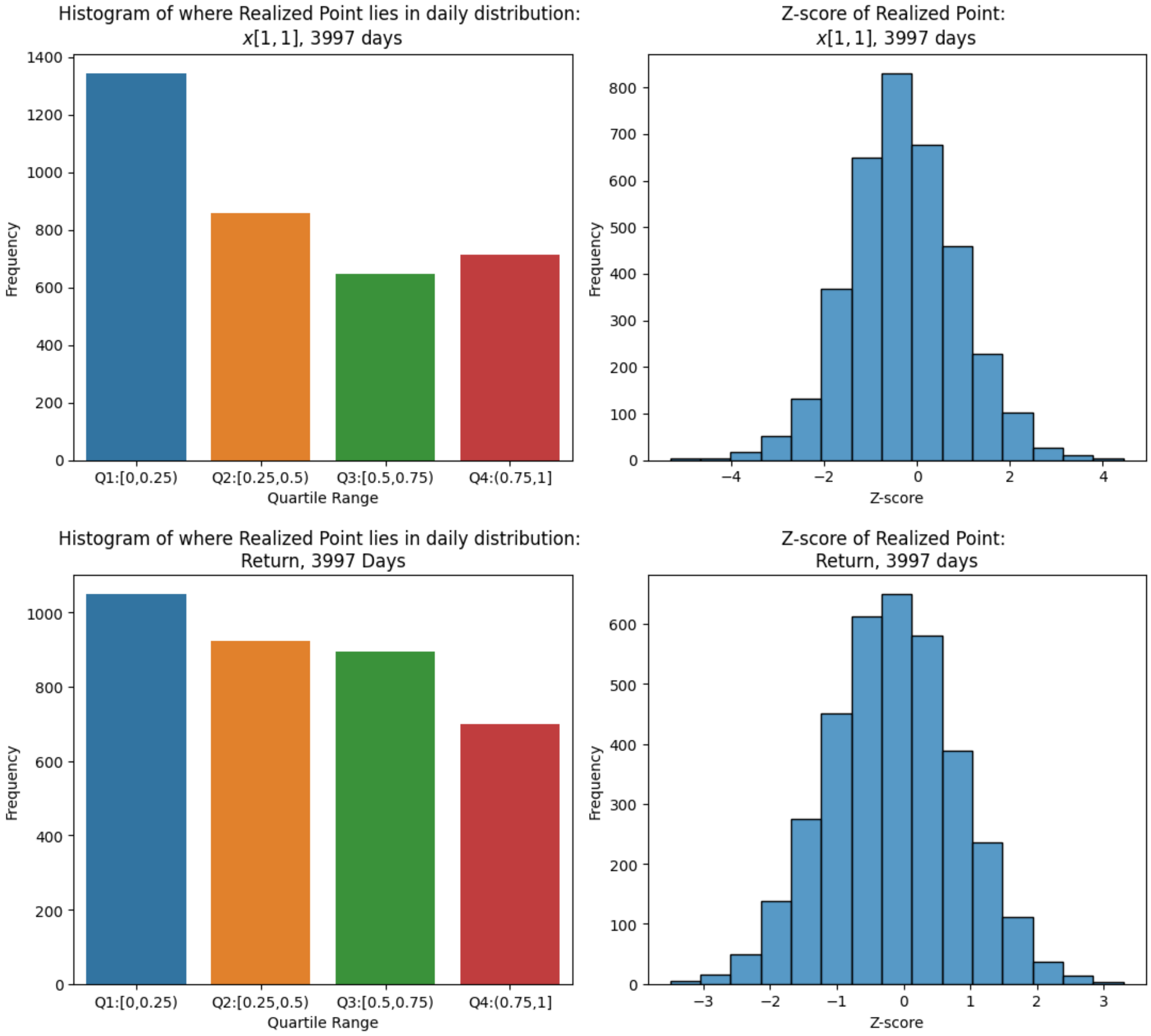
The histogram and latent variable tests were performed under a controlled environment in order to study the the model's capacity to understand latent features and patterns in path trajectories of price returns and volatility surface evolution. The SABR model developed by Hagan et al. [2002], was used to generate over 10,000 paths of evolving implied volatility surfaces given the same fixed grid of moneyness and time to maturity using General Brownian Motion for underlying asset price evolution. We define our SABR model with the following properties: A Skew parameter of $\beta = 1$, initial risk free rate of $r_f = 0$, volatility of volatility of $volvol = 0.3$, correlation parameter of $\rho =$ -0.7, and an initial underlying asset price of $S_0 = 10$. On the SABR data set, the model reported a reconstruction loss of 4.56×10^{-5} of surfaces, reconstruction loss of 3.97×10^{-5} on returns, and KL loss of 0.59 in validation; in testing, losses of 4.65×10^{-5} and 4.00×10^{-5} were reported for surface and returns reconstruction respectively with the same KL loss.

The model configuration used on S&P500 data yielded more balanced outcomes for this dataset in terms of volatility level and returns. Additionally, it produces a higher frequency of outcomes wherein realized additional features such as skew and slope are similar to the mean of daily generated normal distributions. This is evident through a greater number of outcomes falling within the Q2 and Q3 buckets, coupled with smaller daily z-scores, in our histogram tests as illustrated in Figure A5 and A6.

The SABR dataset has been valuable for enhancing the explainability our VAE model's latent space, as well as the influence of significant additional features like skew and slope. We investigated the effect of altering latent variables while fixing the others on two model variants. It is observed that including different features obtain distinct outcomes illustrated in Figure A7. The set of graphs in Panel A shows how the generated surfaces changes when we manipulate each latent value for a baseline model trained only on surface context, excluding any other extra features. The second set of graphs in Panel B shows the impacts of latent value manipulation for a model that trains losses on returns and includes the other extra features: skew and slope. When changing a latent value of the VAE model with skew and slope context information, there is reduced variation in the surface level and shape relative to the realized curve, in contrast to the model that excludes these features. Please note this figure illustrates changes in the first latent variable for ttm = 1 only, however this trend holds for all 5 latent factors in the model and all moneyness and time to maturity slices.

Figure A5: Histogram Plots of SABR Surfaces and Returns

This Figure shows the histogram plots for the best VAE model that includes skew and slope information. The top row shows the distribution and z-scores for the volatility levels. The second row shows the distribution and z-scores for the returns.



Histogram of where Realized Point lies in daily distribution: $x[1,1]$, 3997 days

Z-score of Realized Point: $x[1,1]$, 3997 days

Histogram of where Realized Point lies in daily distribution: Return, 3997 Days

Z-score of Realized Point: Return, 3997 days

Figure A6: Histogram Plots of SABR Surface Structural Properties

This Figure shows the histogram plots for the best VAE model that includes skew and slope information. The top row shows the distribution and z-scores for the skews. The second row shows the distribution and z-scores for the slopes.

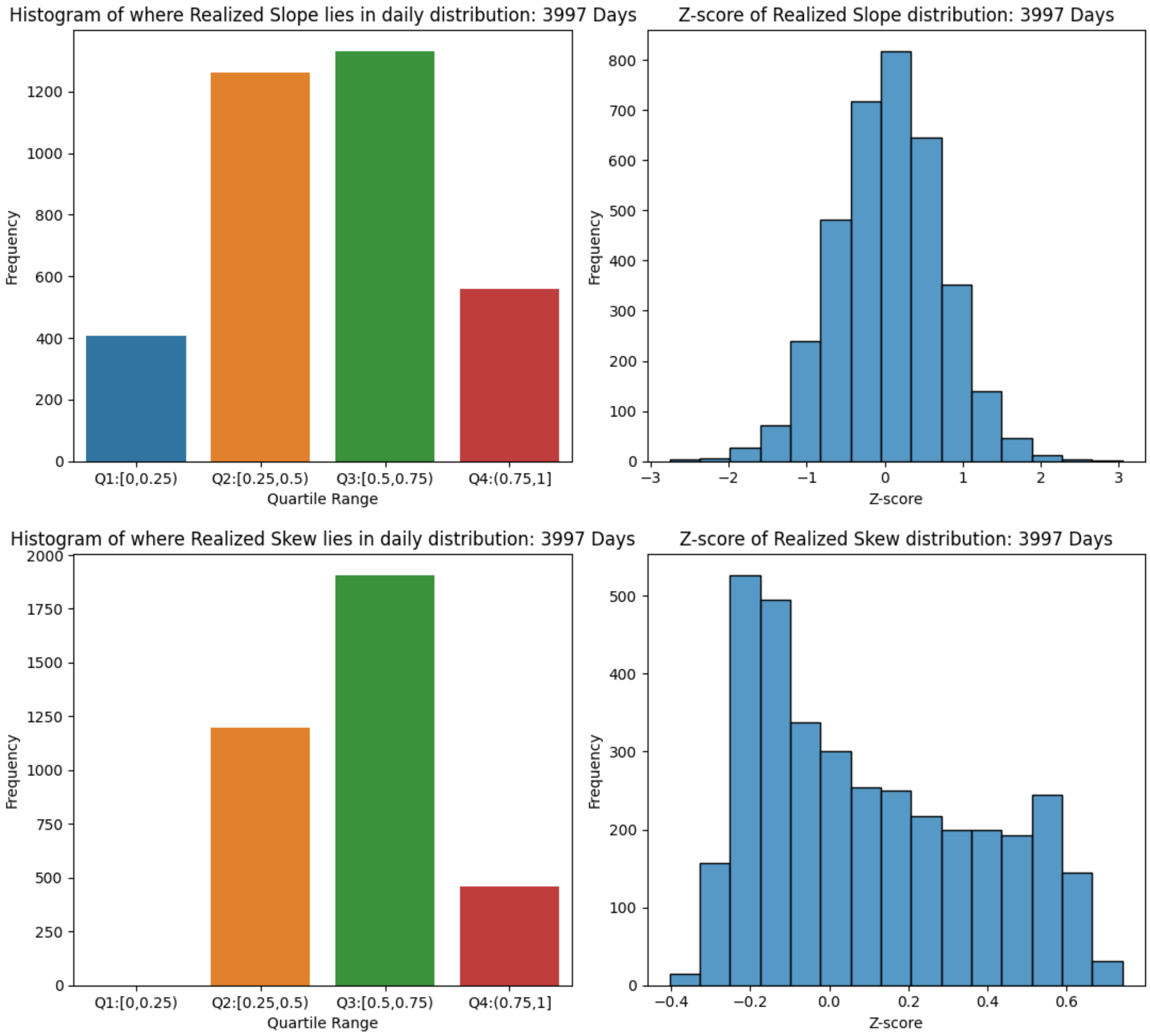
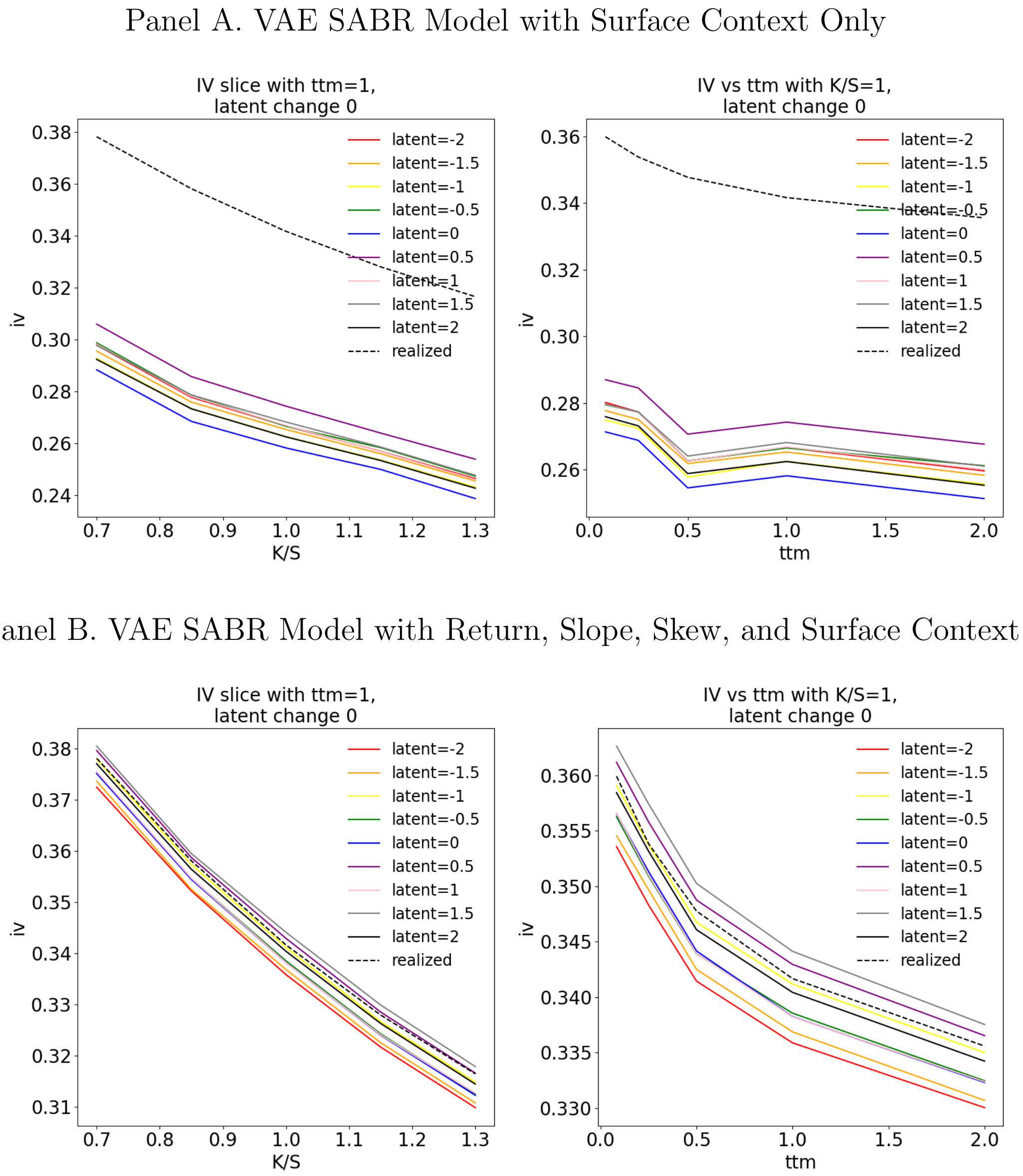


Figure A7: SABR Latent Manipulation



Panel A. VAE SABR Model with Surface Context Only

Panel B. VAE SABR Model with Return, Slope, Skew, and Surface Context

References

Maxime Bergeron, Nicholas Fung, John Hull, Zissis Poulos, and Andreas Veneris. Variational autoencoders: A hands-off approach to volatility. *Journal of Financial Data Science*, 4 (2):125–138, 2022.

Fischer Black and Myron Scholes. The pricing of options and corporate liabilities. *Journal of political economy*, 81(3):637–654, 1973.

Daniel Alexandre Bloch and Arthur Böök. Deep learning based dynamic implied volatility surface. October 2021. Available at SSRN: `https://ssrn.com/abstract=3952842` or `http://dx.doi.org/10.2139/ssrn.3952842`.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. *ArXiv*, abs/1511.06349, 2016.

Jay Cao, Jacky Chen, John Hull, and Zissis Poulos. Deep learning for exotic option valuation. *The Journal of Financial Data Science Winter 2022*, pages 4(1)41–53, 2021.

Rene Carmona, Yi Ma, and Sergey Nadtochiy. Simulation of implied volatility surfaces via tangent levy models. *SIAM Journal on Financial Mathematics*, 8:171–213, 2017.

Vedant Choudhary, Sebastian Jaimungal, and Maxime Bergeron. Funvol: A multi-asset implied volatility market simulator using functional principal components and neural sdes. March 2023. Available at SSRN: `https://ssrn.com/abstract=4377204` or `http://dx.doi.org/10.2139/ssrn.4377204`.

Sam N. Cohen, Christoff Reisinger, and Sheng Wang. Detecting and repairing arbitrage in traded option prices. *Applied Mathematical Finance*, 27:345–373, 2020.

Rama Cont and José Da Fonseca. Dynamics of implied volatility surfaces. *Quantitative Finance*, 2(1):45–60, 2002.

Rama Cont and Milena Vuletić. Simulation of arbitrage-free implied volatility surfaces. December 2022. Available at SSRN: `https://ssrn.com/abstract=4299363` or `http://dx.doi.org/10.2139/ssrn.4299363`.

Rama Cont, José Da Fonseca, and Valdo Durrleman. Stochastic models of implied volatility surfaces. *Economic Notes*, 31:361–377, 2002.

Rama Cont, Mihai Cucuringu, Renyuan Xu, and Chao Zhang. Tail-gan: Learning to simulate tail risk scenarios. *Available at SSRN 3812973*, 2022.

Christa Cuchiero, Walid Khosrawi, and Josef Teichmann. A generative adversarial network approach to calibration of local stochastic volatility models. *Risks*, 8(4):101, 2020. URL `https://doi.org/10.3390/risks8040101`.

Abhyuday Desai, Cynthia Freeman, Zuhui Wang, and Ian Beaver. Timevae: A variational auto-encoder for multivariate time series generation. *ArXiv*, abs/2111.08095, 2021.

Pascal Francois, Rémi Galarneau-Vincent, Genevieve Gauthier, and Frédéric Godin. Venturing into uncharted territory: An extensible parametric implied volatility surface model. February 2022. Available at SSRN: `https://ssrn.com/abstract=3888243` or `http://dx.doi.org/10.2139/ssrn.3888243`.

Jim Gatheral and Antoine Jacquier. Arbitrage-free svi volatility surfaces. *ArXiv*, abs/1204.0646, 2013.

Klaus Greff, Rupesh K. Srivastava, Jan Koutnik, Bas R. Steunebrink, and Jurgen Schmidhuber. LSTM: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10):2222–2232, oct 2017. doi: 10.1109/tnnls.2016.2582924. URL `https://doi.org/10.1109%2Ftnnls.2016.2582924`.

P. Hagan, D. Kumar, A. Lesniewski, and D. Woodward. Managing smile risk. *Wilmott*, pages 84–108, 2002.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-VAE: Learning basic visual concepts with a constrained variational framework. *International Conference on Learning Representations*, 2017. URL `https://openreview.net/forum?id=Sy2fzU9gl`.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9 (8):1735–1780, nov 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL `https://doi.org/10.1162/neco.1997.9.8.1735`.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *ArXiv*, abs/1312.6114, 2013.

Shuyu Lin, Ronald Clark, Robert Birke, Sandro Schönborn, Niki Trigoni, and Stephen Roberts. Anomaly detection for time series using vae-lstm hybrid model. *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4322–4326, 2020. doi: 10.1109/ICASSP40776.2020.9053558.

Robert C. Merton. Theory of rational option pricing. *The Bell Journal of Economics and Management Science*, 4(1):141–183, 1973.

Brian Ning, Sebastian Jaimungal, Xiaorong Zhang, and Maxime Bergeron. Arbitrage-free implied volatility surface generation with variational autoencoders. *SIAM Journal on Financial Mathematics*, 2022. URL `https://arxiv.org/abs/2108.04941`.

Han Lin Shang and Fearghal Kearney. Dynamic functional time-series forecasts of foreign exchange implied volatility surfaces. *Journal of Banking & Finance*, 125:106–120, 2022.

Sima Siami-Namini, Neda Tavakoli, and Akbar Siami Namin. A comparison of arima and lstm in forecasting time series. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1394–1401, 2018. doi: 10.1109/ICMLA.2018.00227.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. *Neural IPS*, pages 3483–3491, 2015. URL `http://papers.nips.cc/paper/5775-learning-structured-output-representation-using-deep-conditional-generative-models`.