Variational Autoencoders: 
A Hands-Off Approach to Volatility

Maxime Bergeron, Nicholas Fung, John Hull, Zissis Poulos, and Andreas Veneris

**KEY FINDINGS**

- We show how synthetic yet realistic volatility surfaces for an asset can be generated using variational autoencoders.
- We illustrate how variational autoencoders can be used to construct a complete volatility surface when only a small number of points are available, without making assumptions about the process driving the underlying asset or the shape of the surface.
- We empirically demonstrate our approach using foreign exchange data and show that pooling volatility surface data from multiple currency pairs improves results.

**ABSTRACT**

A volatility surface is an important tool for pricing and hedging derivatives. The surface shows the volatility that is implied by the market price of an option on an asset as a function of the option's strike price and maturity. Often, market data are incomplete, and it is necessary to estimate missing points on partially observed surfaces. In this article, the authors show how variational autoencoders can be used to model volatility surfaces. The first step is to train the model, deriving latent variables that can be used to construct synthetic volatility surfaces that are indistinguishable from those observed historically. The second step is to determine the synthetic surface generated by the latent variables that fits available data as closely as possible. The trained variational autoencoder can also be used to generate synthetic-yet-realistic surfaces, which can be used in stress testing, in market simulators for developing quantitative investment strategies, and for the valuation of exotic options.

The authors illustrate their procedure using foreign exchange market data.

The famous Black and Scholes (1973) formula does not provide a perfect model for pricing options, but it has been very influential in the way traders manage portfolios of options and communicate prices. The formula has the attractive property of involving only one unobservable variable: volatility. As a result, there is a one-to-one correspondence between the volatility substituted into the Black–Scholes formula and the option price. The volatility that is consistent with the price of an option is known as its implied volatility. Traders frequently communicate prices in the form of implied volatilities. This is convenient because implied volatilities tend to be less variable than the prices themselves.

A volatility surface shows the implied volatility of an option as a function of its strike price and time to maturity. If the Black–Scholes formula provided a perfect description of prices in the market, the volatility surface for an asset would be flat (i.e., implied volatilities would be the same for all strike prices and maturities) and
never change. However, in practice, volatility surfaces exhibit a variety of different shapes and vary through time.

Traders monitor implied volatilities carefully and use them to provide quotes and value their portfolios. When transactions for many different strike prices and maturities are available on a particular day, there is very little uncertainty about the volatility surface. However, in situations in which only a few points on the surface can be reliably obtained, it is necessary to develop a way of estimating the rest of the surface. We refer to this problem as completing the volatility surface.

Black–Scholes assumed that the asset price follows geometric Brownian motion. This leads to a lognormal distribution for the future asset price. Many other, more sophisticated models have been suggested in the literature in an attempt to fit market prices more accurately. Some, such as Heston (1993), assumed that the volatility is stochastic. Others, such as Merton (1976), assumed that a diffusion process for the underlying asset is overlaid with jumps. Bates (1996) incorporated both stochastic volatility and jumps. Madan, Carr, and Chang (1998) proposed a “variance-gamma” model in which there are only jumps. Recently, authors such as Gatheral, Jaisson, and Rosenbaum (2014) have proposed rough volatility models, in which volatility follows a non-Markovian process. One approach to completing the volatility surface is to assume one of these models and fit its parameters to the known points as closely as possible.

An alternative approach is to parameterize the volatility surface directly. The Gatheral (2004) model, and its time-dependent extension (Gatheral and Jacquier 2013), does this. Compared to stochastic volatility and jump models, these parametric representations are easier to calibrate and provide better fits to empirical data.

We propose an alternative deep learning approach using variational autoencoders (VAEs). The advantage of the approach is that it makes no assumptions about the process driving the underlying asset or the shape of the surface. The VAE is trained on historical data from multiple assets to provide a way in which realistic volatility surfaces can be generated from a small number of parameters. A volatility surface can then be completed by choosing values for the parameters that fit the known points as closely as possible. VAEs also make it possible to generate synthetic-yet-realistic surfaces, which can be used for other tasks (e.g., stress testing) and in market simulators for developing quantitative investment strategies. We illustrate our approach using data from foreign exchange markets.

Deep learning techniques are becoming widely used in the field of mathematical finance. Ferguson and Green (2018) pioneered the use of neural networks for pricing exotic options. Several works, such as those by Hernandez (2016), Horvath, Muguruza, and Tomas (2019), and Bayer et al. (2019), have used deep learning to calibrate models to market data. One advantage of these approaches is that, once computational time has been invested upfront in developing the model, valuations can be produced quickly. Our application of VAEs shares this advantage, but it also aims to empirically learn a parameterization of volatility surfaces.

Several works have used deep learning to model volatility surfaces directly. Ackerer, Tagasovska, and Vatter (2020) proposed an approach in which volatility is assumed to be a product of an existing model and a neural network. Chataigner, Crépy, and Dixon (2020) used neural networks to model local volatility using constraints inspired by existing models. A potential disadvantage of these approaches is that they train the neural network on each surface individually, which can be costly and impractical for real-time inference. Chataigner, Crépy, and Pu (2021) used deterministic autoencoders to compress and complete individual volatility surfaces, amortizing training over multiple surfaces. In contrast to their approach, we encode the space of volatility surfaces as a distribution taking full advantage of the regularization offered by stochasticity in variational autoencoders.
In other research, Ning et al. (2021) proposed a hybrid methodology that combines VAEs with stochastic differential equation models. Although this approach guarantees arbitrage-free volatility surfaces, its usefulness is limited because it does not fit to market data directly.

**VARIATIONAL AUTOENCODERS**

The architecture of a vanilla neural network is illustrated in Exhibit 1. There are series of hidden layers between the inputs (which form the input layer) and the outputs (which form the output layer). The value at each neuron of a layer (except the input layer) is $F(c + wv^T)$, where $F$ is a nonlinear activation function, $c$ is a constant, $w$ is a vector of weights, and $v$ is a vector of the values at the neurons of the immediately preceding layer. Popular activation functions are the rectified linear unit ($F(x) = \max(x, 0)$) and the sigmoid function ($F(x) = 1/(1 + e^{-x})$). The network’s parameters, $c$ and $w$, are in general different for each neuron. A training set consisting of inputs and outputs is provided to the network, and parameter values are chosen so that the network determines outputs from inputs as accurately as possible. Further details are provided by Goodfellow, Bengio, and Courville (2016).

An autoencoder is a special type of neural network in which the output layer is the same as the input layer. The objective is to determine a small number of latent variables that are capable of reproducing the inputs as accurately as possible. The architecture is illustrated in Exhibit 2. The encoding function, $E$, consists of a number of layers that produce a vector of latent variables, $z$, from the vector of inputs, $x$. The decoder function, $D$, attempts to reproduce the inputs from $z$. In the simple example in Exhibit 2, there are five input variables. These are reduced to two variables by the encoder, and the decoder attempts to reconstruct the original five variables from the latent variables. The parameters of the neural network are chosen to minimize the difference between $D(z)$ and $x$. Specifically, we choose the network’s parameters to minimize the reconstruction error (RE):

$$RE = \frac{1}{M} \sum_{j=1}^{M} (x_j - y_j)^2$$  \hspace{1cm} (1)
where $M$ is the dimensionality for the input and output, $x_i$ is the $i$th input value, and $y_i$ is the $i$th output value obtained by the decoder. Principal component analysis (PCA) is an alternative approach to dimensionality reduction and can be regarded as the use of an autoencoder with linear activation functions.\(^1\)

Although autoencoders are efficient at compressing data structures, they do not make any guarantees about how latent space is organized. Often the autoencoder will overfit to the training data, and decoding random samples from latent space will not generate realistic data points. A solution to this is to regularize the latent space, which can be done using VAEs (Kingma and Welling 2014). As its name suggests, a VAE is closely related to variational inference methods in statistics, which aim to approximate intractable probability distributions. Rather than producing latent variables in a deterministic manner, the latent variable is sampled from a distribution that is parameterized by the encoder. This regularizes the latent space, enabling the generation of synthetic data similar to the training data.

Although there are many potential prior distributions for the latent variables, a useful prior is the multivariate normal distribution, $\mathcal{N}(0, I)$, where the variables are uncorrelated, with a mean of zero and standard deviation of one. This is what we will use in what follows. Contrary to deterministic autoencoders, there are now two parts to the objective function that is to be minimized. The first part is the reconstruction loss in Equation 1. The second part is the Kullback–Leibler (KL) divergence between the parameterized normal distribution and $\mathcal{N}(0, I)$. That is,

$$\text{KL} = \frac{1}{2} \sum_{k=1}^{d} (-1 - \log \sigma_k^2 + \sigma_k^2 + \mu_k^2)$$

where $\mu_k$ and $\sigma_k$ are the mean and standard deviation of the $k$th latent variable. The objective function becomes

$$\text{RE} + \beta \text{KL}$$

where $\beta$ is a hyperparameter that tunes the strength of the regularization provided by the KL-divergence (Higgins et al. 2017). Note that in the limiting case where $\beta$ goes to 0, the VAE behaves like a deterministic autoencoder. The KL-divergence term

\(^1\) Avellaneda et al. (2020) provided a recent application of PCA to volatility surface changes.

**EXHIBIT 2**

Autoencoder Architecture
is introduced in the loss function to encourage the model to encode a distribution that is as close to normal as possible. This helps ensure stability during training and tractability during inference.

APPLICATION TO VOLATILITY SURFACES

Implied Volatility Surfaces

We now show how VAEs can be applied to volatility surfaces. As mentioned earlier, a volatility surface is a function of the strike price and time to maturity, in which the implied volatilities are obtained by inverting Black–Scholes on observed prices.
For a European call option with strike $K \geq 0$ and time to maturity $T > 0$, let $S$ denote the current price of the underlying asset, and let $r$ denote the (constant) risk-free rate. Let $C_{	ext{mkt}}(K, T)$ denote the market price of this option, and let $C_{\text{bs}}$ be the price of this option as predicted by the Black–Scholes formula (Black and Scholes 1973). The implied volatility $\sigma(K, T) \geq 0$ is implicitly defined by

$$C_{\text{mkt}}(K, T) = C_{\text{bs}} \left( S, K, T, r, \sigma(K, T) \right)$$

(4)

which can be solved using a root-finding method.

The moneyness of an option is a measure of the extent to which the option is likely to be exercised. A moneyness measure providing equivalent information to the strike price usually replaces the strike price in the definition of the volatility surface. One common moneyness measure is the ratio of strike price to asset price, $K/S$. Another is the delta of the option, $\Delta$. The delta is the partial derivative of the option price with respect to the asset price.\(^2\) Intuitively, the delta approximates the probability that an option expires in-the-money. For a call option on an asset, this varies from zero for a deep out-of-the-money option (high strike price) to one for a deep in-the-money option (low strike price). As per convention, we present results on foreign exchange rates using delta as a measure of moneyness.

Both the level of volatilities and the shape of the surface tend to change through time. However, implied volatility surfaces do not come in completely arbitrary shapes. Indeed, there are several restrictions on their geometry arising from the absence of (static) arbitrage (i.e., the existence of a trading strategy providing instantaneous risk-free profit). Lucic (2019) has provided a good discussion of approaches that can be used to understand such constraints. We include the conditions that we use to check for static arbitrage in the Appendix.

**Network Architecture**

Inspired by Bayer et al. (2019), we considered two methods for modelling volatility surfaces: the grid-based approach and the pointwise approach. Exhibit 3 provides an illustration of the differences between these approaches. In both, the input to the encoder is a volatility surface, sampled at $M$ prespecified grid points, which is then flattened into a vector, as shown in Exhibit 3, Panel A. Exhibit 3, Panel B illustrates the grid-based approach, which follows the same architecture as traditional VAEs, in which the decoder uses a $d$-dimensional latent variable to reconstruct the original grid points. Finally, the pointwise approach, as shown in Exhibit 3, Panel C, is an alternative architecture in which the option parameters (moneyness and maturity) are explicitly defined inputs to the decoder. Concretely, the input for the pointwise decoder is a single option’s parameters and the latent variable for the entire surface, and the output is a single point on the volatility surface. We can then use batch inference to output all points on the volatility surface in a single forward pass.

Whereas Bayer et al. opted to use the grid-based approach for their application, we choose the pointwise approach for greater expressivity. The pointwise approach has the advantage that interpolation is performed entirely by neural networks; therefore, the derivatives with respect to option parameters (the Greeks) can be calculated precisely and efficiently using backpropagation. This is not true for the grid-based approach, in which derivatives need to be approximated.

Throughout our investigation, we find that VAEs interpolated volatility surfaces quite well even in environments with limited data. However, as usual, if more data

\(^2\)The partial derivative is calculated using the Black–Scholes model with volatility set equal to the implied volatility.
are available, they should be used because this can improve results. We also experimented with VAEs that were penalized for constructing surfaces that exhibited arbitrage. Nevertheless, we find that this did not significantly improve results because the majority of surfaces produced by our VAEs did not exhibit arbitrage. This is not surprising in light of the fact that the market volatility surfaces we used as training data were arbitrage free to start.

Use Cases

Once the VAE has been trained, the network’s parameters can be fixed and used for inference tasks. During the calibration procedure, the goal is to identify the latent variables such that the outputs of the decoder match the market data as closely as possible. There are two methods for calibration. One method is to use the encoder to infer the latent variables corresponding to market data. This requires only a single pass through the network, making it the preferred choice when data are available. In the case in which data are sparse, the alternative is to use the decoder in isolation, minimizing the reconstruction loss using an optimizer such as L-BFGS (Zhu et al. 1997).

After the latent variables have been calibrated, the VAE can be used to infer unobserved option prices. The latent variables are simply decoded to obtain a fully interpolated surface. Although we focus on the use of VAEs for completing volatility surfaces, there are several other notable applications. In lieu of PCA, VAEs can be used for efficient nonlinear dimensionality reduction and analysis of the dynamics of volatility surfaces. Additionally, the model can be used to generate synthetic-yet-realistic volatility surfaces, which can be used in stress tests, or for inputs to other analyses (e.g., the valuation of exotic options).

EXPERIMENTAL RESULTS

Methodology

To test our methodology, we use over-the-counter option data from 2012–2020 for the AUD/USD, USD/CAD, EUR/USD, GBP/USD, and USD/MXN currency pairs, provided by Exchange Data International. We work with a prespecified grid consisting of 40 points, which are inputs to the encoder: eight times to maturity (one week, one month, two months, three months, six months, nine months, one year, and three years) and five different deltas (0.1, 0.25, 0.5, 0.75, and 0.9). Because prices are quoted for at-the-money, butterfly, and risk-reversal options, we use the equations provided by Reiswich and Wystup (2012) to obtain the implied volatilities for the call options (for further details, refer to Clark 2011).

The dataset is partitioned into a training set, which is used to train the VAE, and a validation set, which is used to evaluate performance. The partitions are split chronologically to prevent leakage of information. We use volatility surfaces from March 2020–December 2020 as the validation set, which corresponds to roughly 15% of all available data.

We find that the choice of network architecture makes a marginal difference to the results, so for the encoder and decoder we choose to use two hidden layers, with 32 neurons in each layer. We leave the latent dimension (i.e., the dimension of the encoder output) to be a variable in our experiments. To train our model, we minimize the objective function in Equation 3 using the Adam optimizer (Kingma and Ba 2017).
With an exponential number of hyperparameter combinations, we use a random grid search to identify suitable hyperparameter choices to optimally balance the reconstruction loss and KL divergence to ensure continuity in latent space.

Completing Volatility Surfaces

To evaluate the model’s ability to complete volatility surfaces, we randomly sample a subset of all options observed on a given day and assume that these provide the only known points on the volatility surface. We then use these points to calibrate our model. Rather than calibrating the latent variables deterministically, we use a stochastic algorithm that minimizes Equation 3 to approximate the distribution of the latent variable conditioned on the partially observed surface (Hoffman et al. 2013). By doing so, we are able to recover a distribution of volatility surfaces. Finally, we compare the average of the volatility surfaces to all 40 option prices. We vary the number of sample points and the number of latent dimensions in the trained VAEs to observe how our model performs in various conditions.

Initially, we trained VAEs on volatility surfaces from single currency pairs. However, we find that training using data from multiple currencies yielded more robust models. Exhibit 4 shows the mean absolute error on the AUD/USD validation set when the VAEs are trained using only the AUD/USD data, and Exhibit 5 shows the mean absolute errors when the VAEs are trained using volatility surfaces from all six currency pairs. In the majority of cases, better results are achieved by training the model on all currency pairs. This illustrates the existence of universal principles constraining the dynamics of volatility surfaces across different currency pairs.

To compare our results to a traditional volatility model with a similar number of parameters, we evaluate calibration performance with the Heston stochastic volatility model, which has five free parameters. Here, we perform a full surface calibration, where all 40 points are observed and fit to. The mean absolute error for each currency in the validation set is shown in Exhibit 6, where we compare Heston to our best performing model from Exhibit 5.

In addition to outperforming Heston in reconstructing volatility surfaces, there are practical benefits of using VAEs. A primary advantage of using the VAE is that it predicts prices significantly faster, which makes calibration much more efficient. Another advantage is that regularization during training encourages the latent space to be continuous; that is, small perturbations in latent space correspond to small perturbations in the volatility surface. This is not true for a model such as Heston, because the inverse map from market prices to model parameters can be multivalued. Finally, we highlight the flexibility of our approach. When extreme market conditions

---

**EXHIBIT 4**

**Mean Absolute Error across the AUD/USD Validation Set for Inferring Volatility Surfaces When Given Partial Observations**

<table>
<thead>
<tr>
<th>Latent Dimensions</th>
<th>Assumed Number of Known Points on Volatility Surface</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>87.2</td>
</tr>
<tr>
<td>3</td>
<td>77.2</td>
</tr>
<tr>
<td>4</td>
<td>73.2</td>
</tr>
</tbody>
</table>

**NOTES:** Each row contains a model with a different number of latent dimensions. Each model is trained on the AUD/USD data. Units are in basis points.
are encountered, a VAE can easily be retrained. In our experience, this can be done in just a few minutes using over 10,000 surfaces.

Although the majority of the predictions are within the mean bid–ask spread, we examine how the model performs with the outliers in our dataset. Naturally, we find that the errors come from surfaces with extremely high levels of volatility. For example, Exhibit 7, Panel B shows the interpolation of the AUD/USD volatility surface on March 24, 2020, the worst performing surface in our dataset. To test our model under realistic settings, we randomly sample 10 (of 40) points that are assumed to be observed, for which the delta is between 0.25 and 0.75 and the time to expiry is less than one year. We note that this model was never trained on markets in crisis conditions, but it can still provide reasonable estimates for the options expiring within a year. For comparison, we also show how our model performs in typical market conditions in Exhibit 7, Panel A.

Observing the distributions of errors, we find that options close to expiry exhibit the greatest error. This is not surprising because these options, particularly when they are close to the money, have the most volatile prices. We note that our models were trained using an equal weighting of all options; however, practitioners can easily alter the weights to suit their requirements.

### Generating Synthetic Surfaces

Aside from completing surfaces, VAEs can also be used for generating synthetic data (e.g., for stress testing a portfolio). To show that latent space yields a variety of synthetic surfaces, we interpolate between points in latent space and construct the corresponding surfaces. This is illustrated using a two-dimensional VAE in Exhibit 8. Although in general interpreting the latent dimensions of a VAE is a nontrivial task, we can observe how the direction of skew and the term structure of volatility vary across both dimensions, yielding a rich variety of volatility surface patterns.

One method to generate synthetic data with a distribution similar to a given dataset is to first infer the set of latent variables corresponding to that dataset, then perform a kernel density estimate (Scott 1992) on the latent variables, and finally decode the latent variables that are sampled from the estimated density. To show that this yields a variety of volatility surfaces similar to the original dataset,
EXHIBIT 7
Examples of Volatility Surfaces Interpolated Using a Four-Dimensional VAE

Panel A: The AUD/USD Volatility Surface on March 2, 2020—A Typical Surface from our Dataset

Panel B: The AUD/USD Volatility Surface on March 24, 2020—The Worst Performing Surface in our Dataset

(continued)
EXHIBIT 7 (continued)
Examples of Volatility Surfaces Interpolated Using a Four-Dimensional VAE

Notes: The error bars represent the bid and ask volatility. The predicted surface is inferred by observing only 10 (of 40) liquid midpoint prices, represented by the crosses.

EXHIBIT 8
Examples of Synthetic Volatility Surfaces Generated When Interpolating across Two Latent Dimensions

(continued)
we compare the Mahalanobis distance between pairs of real and synthetic surfaces. The Mahalanobis distance provides a scale-invariant metric on multidimensional data, which takes into account the correlations between features, making it more useful than other metrics such as Euclidean distance (Mahalanobis 1936). We calculate the Pearson correlation coefficient between Mahalanobis distances of the real and synthetic surfaces. The correlations are measured by pairing each synthetic surface with the real surface that is closest in latent space after encoding. A correlation of zero shows that the latent space is not expressive enough to replicate the dataset, whereas a correlation of one shows that the latent space is overfit to the data and unable to generalize. Ultimately, we find a correlation coefficient of 0.56 under this pairing scheme, demonstrating the VAE’s ability to organize latent space and generate synthetic-yet-realistic volatility surfaces.

CONCLUSIONS

Our results demonstrate that VAEs provide a useful approach to analyzing volatility surfaces empirically. We illustrated how a VAE can be trained on volatility surfaces and used foreign exchange markets, a realistic testing ground for our model. Our results show that volatility surfaces can be captured using VAEs with as few as three latent dimensions and that the resulting models can be used for practical purposes such as completing volatility surfaces and generating synthetic datasets.

APPENDIX

ARBITRAGE CONDITIONS

Following Gatheral and Jacquier (2013), we can specify static arbitrage conditions as follows: Let $F_t$ denote the forward price at time $t$ and $X = \log \frac{K}{F_t}$. Define $w(X, t) = t \sigma^2 (X, t)$ as the total implied variance surface. An implied volatility surface is free of calendar arbitrage if

$$\frac{\partial w}{\partial t} \geq 0 \quad (A1)$$
Let \( w' = \frac{\partial w}{\partial X} \) and \( w'' = \frac{\partial^2 w}{\partial X^2} \). Suppressing the arguments for \( w(X, t) \), the volatility surface is free of butterfly arbitrage if

\[
g(X,t) := \left(1 - \frac{Xw'}{2w}\right)^2 - \frac{w'}{4\left(\frac{1}{w} + \frac{1}{4}\right)} + \frac{w''}{2} \geq 0
\]  

(A2)

A volatility surface is said to be free of static arbitrage if the conditions are met in Equations A1 and A2. If we let

\[
L_{\text{cal}} = \left| \max\left(0, -\frac{\partial w}{\partial t}\right) \right|^2
\]  

(A3)

and

\[
L_{\text{but}} = \left| \max(0, -g) \right|^2,
\]  

(A4)

the loss function in Equation 3 can then be extended as follows:

\[
\text{RE} + \beta KL + \lambda_{\text{cal}} L_{\text{cal}} + \lambda_{\text{but}} L_{\text{but}}
\]  

(A5)

As the parameters \( \lambda_{\text{cal}} \) and \( \lambda_{\text{but}} \) are increased, the possibility of static arbitrage is reduced.

**ACKNOWLEDGMENTS**

We would like to thank Ryan Ferguson, Vlad Lucic, Ivan Sergienko, Gary Wong, and Nikola Pocuća for their interest in this work as well as their many helpful comments. We also thank Risk-fuel Analytics, NSERC, and Mitacs for providing financial support for this research.

**REFERENCES**


