Variational Autoencoders: A Hands-Off Approach to Volatility

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

† Riskfuel Analytics
‡ University of Toronto
*Corresponding author: John Hull (hull@rotman.utoronto.ca)

Variational Autoencoders: A Hands-Off Approach to Volatility

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 is incomplete and it is necessary to estimate missing points on partially observed surfaces. In this paper, we show how variational autoencoders can be used for this task. The first step is to derive 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 synthetic surfaces produced in the first step can also be used in stress testing, in market simulators for developing quantitative investment strategies, and for the valuation of exotic options. We illustrate our procedure using foreign exchange market data.

Keywords: Derivatives; Implied volatility surfaces, Unsupervised learning; Variational autoencoders

JEL Classification: G10, G20

Variational Autoencoders: A Hands-Off Approach to Volatility

1. Introduction

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 that it involves only one unobservable variable: volatility. As a result, there is a one-to-one correspondence between the volatility substituted into Black-Scholes 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 where 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 assumes 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), assume that the volatility is stochastic. Others, such as Merton (1976), assume that a diffusion process for the underlying asset is overlaid with jumps. Bates (1996) incorporates both stochastic volatility and jumps. Madan *et al.* (1998) propose a "variance-gamma" model where there are only jumps. Recently, rough volatility models have been proposed by authors such as Gatheral *et al.* (2014). In these, 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.

Parametric models are another way to complete volatility surfaces. The popular stochastic volatility inspired representation (Gatheral 2004), as well as its time dependent extension (Gatheral and Jacquier 2013), characterizes the geometry of surfaces directly through each of its parameters. Compared to stochastic volatility models, parameteric 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 such as 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 researchers, such as Hernandez (2016), Horvath *et al.* (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 also aims to empirically learn a parameterization of volatility surfaces.

Some research uses deep learning to model volatility surfaces directly. Ackerer *et al.* (2020) propose an approach where volatility is assumed to be a product of an existing model and a neural network. Chataigner *et al.* (2020) use 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 *et al.* (2021) use 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

We conclude this introduction with a brief outline of the paper. Section 2 introduces variational autoencoders. Section 3 describes how variational autoencoders can be applied to volatility surfaces. Section 4 presents experimental results. Finally, conclusions are presented in Section 5.

2. Variational Autoencoders

The architecture of a vanilla neural network is illustrated in figure 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 *et al.* (2017).

An autoencoder is a special type of neural network where 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 figure 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 figure 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

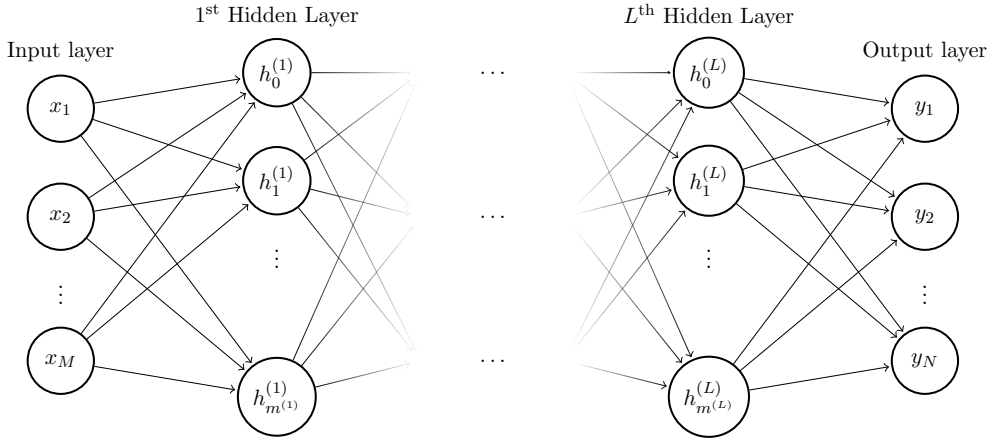


Figure 1. A neural network with L hidden layers, with M inputs and N outputs. The i^{th} hidden layer contains $m^{(i)}$ neurons, and $h_k^{(i)}$ is the value at the k^{th} neuron of hidden layer i.

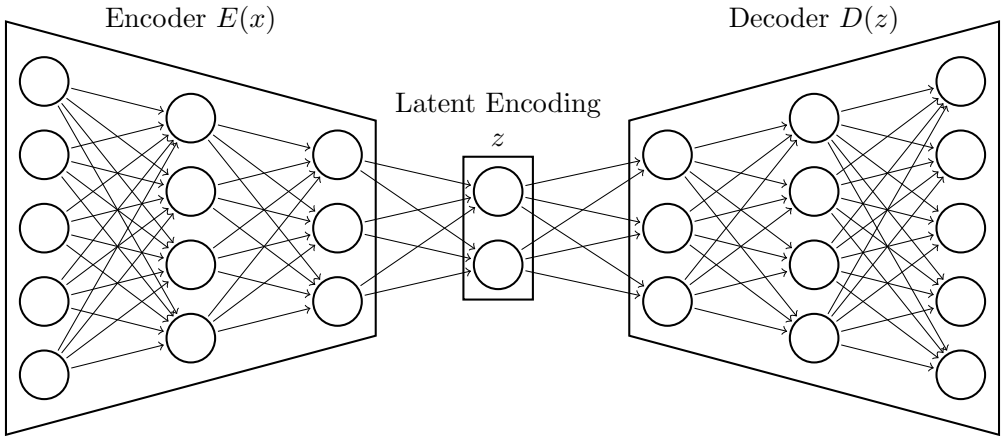


Figure 2. An autoencoder can be split into encoder and decoder networks. Note that the dimensionality of the latent encoding is typically smaller than the original input dimension.

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):

$$\mathrm{RE} = \frac{1}{M}\sum_{i=1}^{M}(x_i - y_i)^2 \tag{1}$$

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]

A useful extension of autoencoders is the *Variational Autoencoder* (VAE), which was introduced by Kingma and Welling (2014). As its name suggests, the VAE is closely linked 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. By sampling from the distribution, synthetic data similar to the input data can be generated. A useful prior distribution for the latent variables is a multivariate normal distribution, $\mathcal{N}(0, I)$, where the variables are uncorrelated with mean zero and standard deviation one. This is what we will use in what follows. Contrary to deterministic autoencoders, there are now two parts to the objective function which is to be minimized. The first part is the loss function in equation (1). The second part is the Kullback-Leibler (KL) divergence between the parameterized distribution and $\mathcal{N}(0, I)$. That is:

$$\mathrm{KL} = \frac{1}{2}\sum_{\mathrm{k}=1}^{\mathrm{d}}(-1 - \log\sigma_{\mathrm{k}}^2 + \sigma_{\mathrm{k}}^2 + \mu_{\mathrm{k}}^2) \tag{2}$$

where μ_k and σ_k are the mean and standard deviation of the k^{th} latent variable. The objective function is:

$$\mathrm{RE} + \beta\mathrm{KL} \tag{3}$$

[1] Avellaneda et al (2020) provides a recent application of PCA to volatility surface changes.

where β is a hyperparameter that tunes the strength of the regularization provided by KL. Note that in the limiting case where β goes to 0, the VAE behaves like a deterministic autoencoder. The reason for introducing the KL divergence term in the loss function is 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.

3. Application for Volatility Surfaces

3.1. *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, where 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 $C_{mkt}(K, T)$ denote the market price of this option, and let C_{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:

$$C_{mkt}(K, T) = C_{BS}\left(S, K, T, r, \sigma(K, T)\right), \tag{4}$$

which can be solved using a root-finding method.

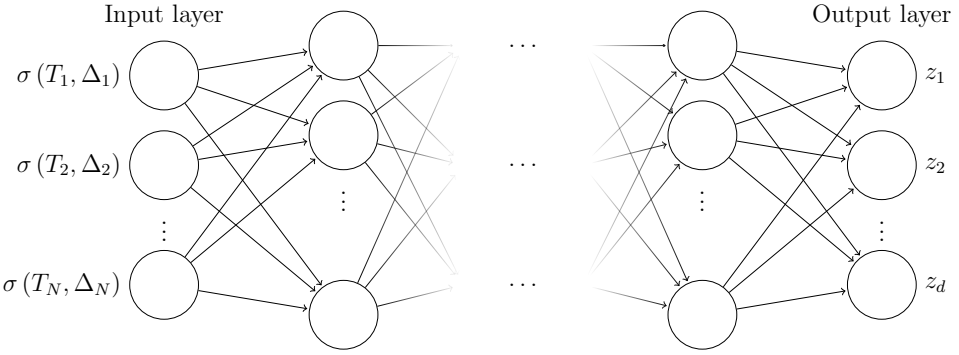
The *moneyness* of an option is a measure of the extent to which the option is likely to be exercised. A moneyness measure providing equivalent information to the strike price usually replaces the strike price in the definition of the volatility surface. One common moneyness measure is the ratio of strike price to asset price, K/S. Another is the delta of the option, Δ. The delta is the partial derivative of the option price with respect to the asset price.[1] 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 tends 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, that is, the existence of a trading strategy providing instantaneous risk-free profit. Lucic (2019) provides a good discussion of approaches that can be used to understand such constraints.[2]

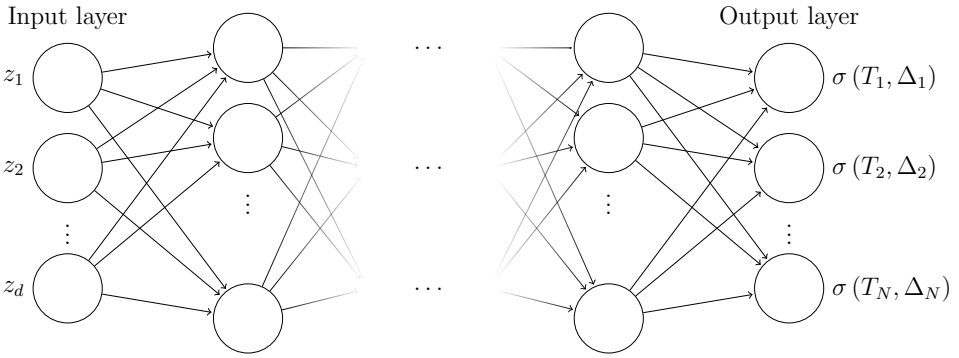
3.2. *Network Architecture*

Inspired by Bayer *et al.* (2019), we considered two methods for modelling volatility surfaces: the *grid-based* approach, and the *pointwise* approach. Figure 3 provides an illustration of the differences between these approaches. In both approaches, the input to the encoder is a volatility surface, sampled at N prespecified grid points, which is then flattened into a vector, as shown in figure 3(a). Figure 3(b) illustrates the grid-based approach, which follows the same architecture as traditional VAEs, where the decoder uses a d-dimensional latent variable to reconstruct the original grid points. Finally, the pointwise approach, as shown in figure 3(c), is an alternative architecture where 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

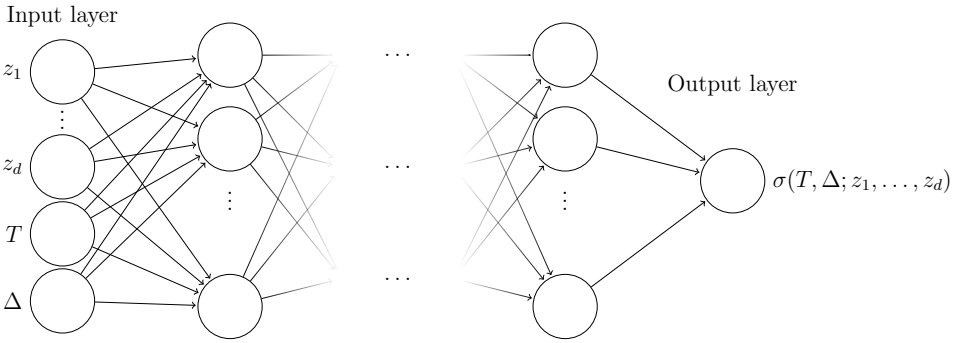
[1] The partial derivative is calculated using the Black-Scholes model with volatility set equal to the implied volatility.
[2] For convenience, we also include the conditions that we use to check for static arbitrage in Appendix A.



(a) The encoder architecture for a deterministic autoencoder. For a variational autoencoder the output is the mean and standard deviation of z.



(b) The decoder architecture for the grid-based training approach.



(c) The decoder architecture for the pointwise training approach.

Figure 3. An illustration of the grid-based and pointwise architectures.

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.

While Bayer *et al.* opt 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 and 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, where 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 is available it should be used since it 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, as 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 begin with. For further details, we refer the reader to Appendix A.

3.3. *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 is available. In the case where data is sparse, the alternative is to use the decoder in isolation, minimizing the reconstruction loss using an optimizer such as LFBGS (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 non-linear 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 such as the valuation of exotic options.

4. Experimental Results

4.1. *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: 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). As prices are quoted for at-the-money, butterfly, and risk-reversal options, we use the equations provided in Reiswich and Wystup (2012) to obtain the implied volatilities for the call options (for further details refer to Clark (2010)).

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 15% of available data as the validation set, which contains data from March 2020 – December 2020.

We find that the choice of network architecture makes a marginal difference to the results so we choose to use two hidden layers in the encoder and decoder, with 32 units 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 from 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.

4.2. *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

Table 1. The mean absolute error across the AUD/USD validation set for inferring volatility surfaces when given partial observations. Each row contains a trained model with a different number of latent dimensions. *Models are trained using only AUD/USD volatility surfaces.* Units are in basis points.

	Assumed Number of Known Points on Volatility Surface							
Latent Dimensions	5	10	15	20	25	30	35	40
2	87.2	77.7	75.9	74.5	73.1	73.0	72.9	72.7
3	77.2	66.4	62.4	60.0	59.0	58.0	57.5	57.2
4	73.2	57.8	53.7	50.1	48.5	47.0	46.9	46.5

Table 2. The mean absolute error across the AUD/USD validation set for inferring volatility surfaces when given partial observations. Each row contains a trained model with a different number of latent dimensions. *Models are trained using all available currency pairs.* Units are in basis points.

	Assumed Number of Known Points on Volatility Surface							
Latent Dimensions	5	10	15	20	25	30	35	40
2	107.6	82.5	71.4	64.2	63.9	63.5	63.5	63.3
3	75.9	53.8	49.8	48.2	47.0	46.6	46.5	46.3
4	61.1	41.5	37.7	35.9	34.7	34.2	34.1	33.6

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. 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 see how our model performs in various conditions.

Initially, we trained VAEs on volatility surfaces from single currency pairs. However, we find that training models using data from multiple currencies yielded more robust models through transfer learning. Table 1 shows the mean absolute error when the models are trained using only the AUD/USD data, while table 2 shows the mean absolute errors when VAEs are trained using volatility surfaces from all six currency pairs. It can be seen that in the majority of cases better results are achieved by training the model on all six 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 perform calibration using Heston. The mean absolute error for each currency in the validation set is shown in table 3, where we compare Heston to our best performing model from table 2. In addition to outperforming Heston in reconstructing volatility surfaces, there are some additional practical benefits from 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, *i.e.,* small perturbations in latent space result in small perturbations in the volatility surface. This is not true for a model such as Heston, as the inverse map from market prices to model parameters can be multivalued. Finally we highlight the flexibility of our approach. When extreme market conditions are encountered, a

Table 3. Mean absolute error when calibrating using Heston and a four-dimensional VAE. All 40 points of the volatility surface are observed for calibration. Units are in basis points.

Currency Pair	Heston	VAE
AUD/USD	56.6	33.6
USD/CAD	35.3	32.5
EUR/USD	32.2	30.9
GBP/USD	47.6	34.0
USD/JPY	58.5	38.2
USD/MXN	92.2	56.7

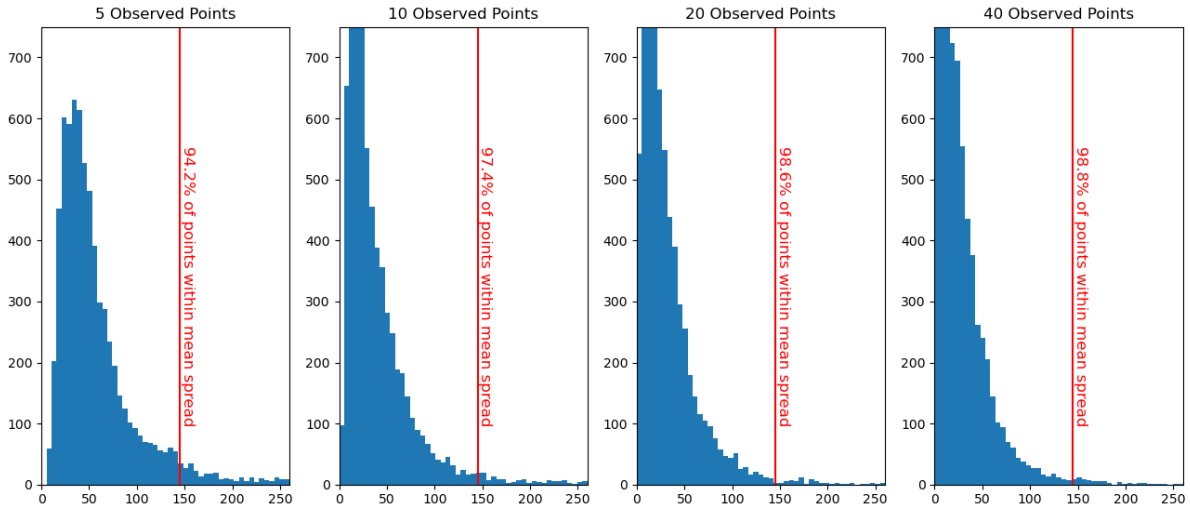


Figure 4. The error distributions across the AUD/USD validation set when inferring volatility surfaces given partial observations using a four-dimensional VAE. The red line represents the mean bid-ask spread across all liquid options. Units are in basis points.

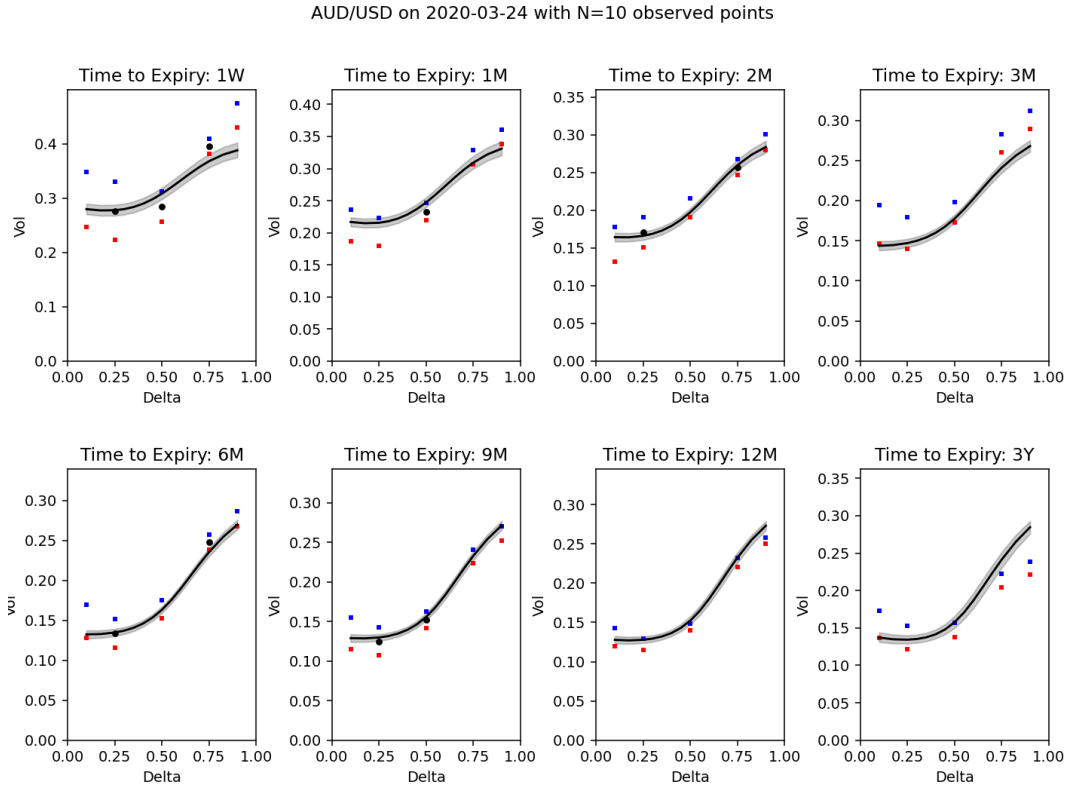
VAE can easily be retrained. In our experience, this can be done in only a few minutes using over 10,000 surfaces.

To supplement our findings, we also examine the distributions of errors. To isolate the most important options, we restrict our attention to prices whose bid-ask spread is smaller than the 90% quantile of the spreads for the given currency. The distribution of errors when inferring incomplete AUD/USD surfaces is presented in figure 4.

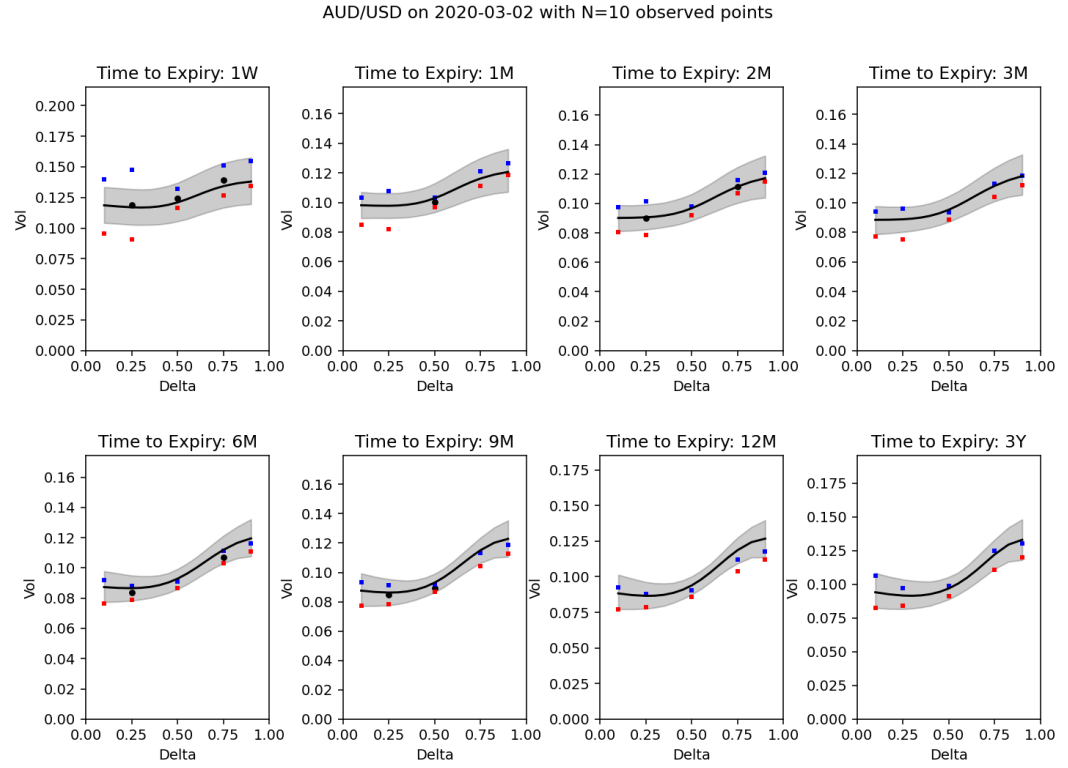
While the majority of the predictions are within the mean bid-ask spread, we examine how the model performs with the outliers in our dataset. We find that the errors come from surfaces with extremely high levels of volatility. For example, figure 5(a) shows the interpolation of the AUD/USD volatility 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, where 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 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 figure 5(b).

To investigate where our model performs the best, we calculate the mean absolute error for individual grid points. We find that options close to expiry have the greatest error. This is not surprising as these options, particularly when they are close to the money, exhibit 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.

As mentioned, VAEs can be useful for generating synthetic volatility surfaces (e.g. for stress testing a portfolio) as well as for completing partially observed surfaces. A straightforward approach is to sample random points in latent space and use the decoder to construct volatility surfaces. To show that this would yield a variety of volatility surfaces, we interpolate between points in latent space and construct the corresponding surface. This is illustrated in the case of a two-dimensional VAE in figure 6. While, in general, interpreting the latent dimensions of a VAE is a non-trivial task, we can observe how the direction of skew and the term structure of volatility varies across both dimensions. A rich variety of volatility surface patterns are obtained.



(a) The AUD/USD volatility surface on March 24, 2020 - the worst performing surface in our dataset.



(b) The AUD/USD volatility surface on March 2, 2020 - a typical surface from our dataset.

Figure 5. Examples of volatility surfaces interpolated using a four dimensional VAE. The red and blue points represent the bid and ask volatility. The predicted surface is inferred by observing only 10 (of 40) liquid mid point prices, represented by the black points. The shaded area represents the 90% confidence interval after decoding multiple samples from latent space.

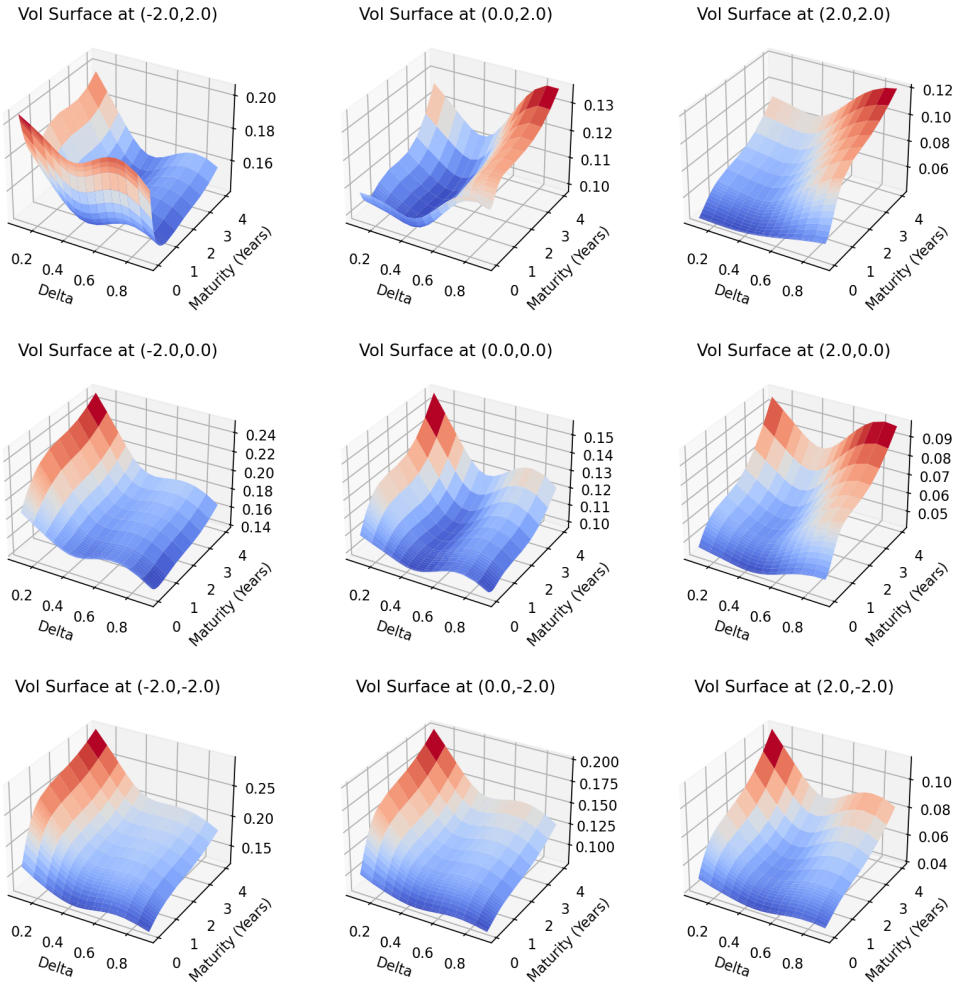


Figure 6. Examples of synthetic volatility surfaces generated when interpolating across two latent dimensions.

We may wish to observe the behavior of volatility surfaces in specific scenarios. Figure 7 shows the encoded latent variables for the volatility surfaces in the validation set. While the majority of the points are clustered near the origin, we notice many outliers which correspond to volatility surfaces observed at the beginning of the international pandemic in March 2020. By sampling and decoding latent variables in these outlying regions, we can simulate extreme scenarios that may occur.

5. Conclusion

Our results demonstrate that VAEs provide a useful approach to analyzing volatility surfaces empirically. We have described how a VAE can be trained, and used the context of foreign exchange markets as a realistic testing ground. Our results show that volatility surfaces can be captured using VAEs with as few as two latent dimensions and that the resulting models can be used for practical and exploratory purposes. For the sake of concreteness, we limited the scope of this paper to VAEs with Gaussian priors. Future work could determine if a VAE model with non-Gaussian priors is able to learn an even wider range of market behaviors while retaining the stability of our model.

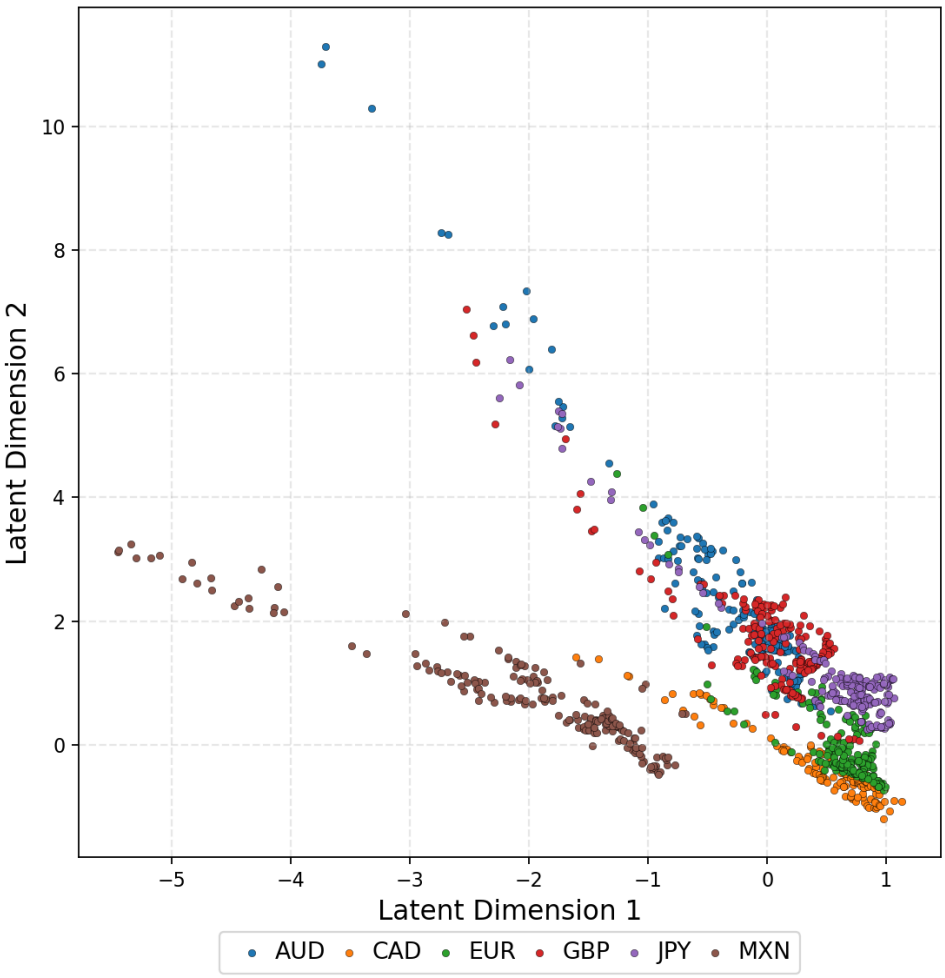


Figure 7. Latent encodings of implied volatility surfaces (colour coded)

References

Ackerer, D., Tagasovska, N. and Vatter, T., Deep Smoothing of the Implied Volatility Surface. *arXiv:1906.05065 [cs, q-fin, stat]*, 2020 ArXiv: 1906.05065.

Bates, D.S., Jumps and Stochastic Volatility: Exchange Rate Processes Implicit in Deutsche Mark Options. *The Review of Financial Studies*, 1996, **9**, 69–107.

Bayer, C., Horvath, B., Muguruza, A., Stemper, B. and Tomas, M., On deep calibration of (rough) stochastic volatility models. *arXiv:1908.08806 [q-fin]*, 2019 ArXiv: 1908.08806.

Black, F. and Scholes, M., The Pricing of Options and Corporate Liabilities. *The Journal of Political Economy*, 1973, **81**, 637–654.

Chataigner, M., Crépey, S. and Dixon, M., Deep Local Volatility. *arXiv:2007.10462 [q-fin]*, 2020 ArXiv: 2007.10462.

Chataigner, M., Crépey, S. and Pu, J., Nowcasting networks. *Journal of Computational Finance*, 2021, **24**, 1–39.

Clark, I.J., Foreign Exchange Option Pricing. *Foreign Exchange Option Pricing*, 2010, p. 300.

Ferguson, R. and Green, A., Deeply Learning Derivatives. *arXiv:1809.02233 [cs, q-fin]*, 2018 ArXiv: 1809.02233.

Gatheral, J., A parsimonious arbitrage-free implied volatility parameterization with application to the valuation of volatility derivatives. , 2004, p. 41.

Gatheral, J. and Jacquier, A., Arbitrage-free SVI volatility surfaces. *arXiv:1204.0646 [q-fin]*, 2013 ArXiv: 1204.0646.

Gatheral, J., Jaisson, T. and Rosenbaum, M., Volatility is rough. *arXiv:1410.3394 [q-fin]*, 2014 ArXiv: 1410.3394.

Goodfellow, I., Bengio, Y. and Courville, A., *Deep Learning*, 2017.

Hernandez, A., Model Calibration with Neural Networks. SSRN Scholarly Paper ID 2812140, Social Science Research Network, Rochester, NY, 2016.

Heston, S.L., A Closed-Form Solution for Options with Stochastic Volatility with Applications to Bond and Currency Options. *The Review of Financial Studies*, 1993, **6**, 327–343 Publisher: [Oxford University Press, Society for Financial Studies].

Horvath, B., Muguruza, A. and Tomas, M., Deep Learning Volatility. *arXiv:1901.09647 [q-fin]*, 2019 ArXiv: 1901.09647.

Kingma, D.P. and Ba, J., Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*, 2017 ArXiv: 1412.6980.

Kingma, D.P. and Welling, M., Auto-Encoding Variational Bayes. *arXiv:1312.6114 [cs, stat]*, 2014 ArXiv: 1312.6114.

Lucic, V., Volatility Notes. SSRN Scholarly Paper ID 3211920, Social Science Research Network, Rochester, NY, 2019.

Madan, D.B., Carr, P.P. and Chang, E.C., The Variance Gamma Process and Option Pricing. *Review of Finance*, 1998, **2**, 79–105.

Merton, R.C., Option pricing when underlying stock returns are discontinuous. *Journal of Financial Economics*, 1976, **3**, 125–144.

Reiswich, D. and Wystup, U., FX Volatility Smile Construction. *Wilmott*, 2012, **2012**, 58–69.

Zhu, C., Byrd, R.H., Lu, P. and Nocedal, J., Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software*, 1997, **23**, 550–560.

Appendix A: 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 $w(X,t) = t \cdot \sigma^2(X,t)$ as the total implied variance surface. An IVS is free of *calendar arbitrage* if

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

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

$$g(X,t) := \left(1 - \frac{Xw'}{2w}\right)^2 - \frac{w'}{4}\left(\frac{1}{w} + \frac{1}{4}\right) + \frac{w''}{2} \geq 0. \tag{A2}$$

A volatility surface is said to be free of static arbitrage if the conditions in equations (A1) and (A2). If we let

$$L_{cal} = \left\| \max\left(0, -\frac{\partial w}{\partial t}\right) \right\|^2, \tag{A3}$$

and

$$L_{but} = \left\| \max(0, -g) \right\|^2, \tag{A4}$$

the loss function in equation (3) can then be extended as follows:

$$\mathrm{RE} + \beta \mathrm{KL} + \lambda_{\mathrm{cal}} \mathrm{L}_{\mathrm{cal}} + \lambda_{\mathrm{but}} \mathrm{L}_{\mathrm{but}}. \tag{A5}$$

As the parameters λ_{cal} and λ_{but} are increased, the possibility of static arbitrage is reduced.