

# Deep Learning Option Pricing with Market Implied Volatility Surfaces

Lijie Ding,<sup>1, a)</sup> Egang Lu,<sup>2</sup> and Kin Cheung<sup>3</sup>

<sup>1)</sup>*Neutron Scattering Division, Oak Ridge National Laboratory, Oak Ridge, TN 37831, USA*

<sup>2)</sup>*Independent researcher*

<sup>3)</sup>*Valkin Holdings, LLC, Rowland Heights, CA 91748, USA*

(Dated: 9 September 2025)

We present a deep learning framework for pricing options based on market-implied volatility surfaces. Using end-of-day S&P 500 index options quotes from 2018-2023, we construct arbitrage-free volatility surfaces and generate training data for American puts and arithmetic Asian options using QuantLib. To address the high dimensionality of volatility surfaces, we employ a variational autoencoder (VAE) that compresses volatility surfaces across maturities and strikes into a 10-dimensional latent representation. We feed these latent variables, combined with option-specific inputs such as strike and maturity, into a multilayer perceptron to predict option prices. Our model is trained in stages: first to train the VAE for volatility surface compression and reconstruction, then options pricing mapping, and finally fine-tune the entire network end-to-end. The trained pricer achieves high accuracy across American and Asian options, with prediction errors concentrated primarily near long maturities and at-the-money strikes, where absolute bid-ask price differences are known to be large. Our method offers an efficient and scalable approach requiring only a single neural network forward pass and naturally improve with additional data. By bridging volatility surface modeling and option pricing in a unified framework, it provides a fast and flexible alternative to traditional numerical approaches for exotic options.

## I. INTRODUCTION

Options are fundamental financial derivatives that enable investors to hedge risks, speculate on market movements, and manage portfolios in increasingly complex markets<sup>1</sup>. The global options market has grown significantly, driven by demand for sophisticated instruments like American puts and arithmetic Asian options<sup>2,3</sup>. However, pricing option portfolios often poses challenges. Traditional parametric models, such as the Black-Scholes framework, assume constant volatility and log-normal dynamics. Such assumptions necessitate model adjustments in the real-world with volatility smiles, term structure, and jumps<sup>4,5</sup>. Numerical methods, such as Monte Carlo simulations or finite difference schemes, offer flexibility for exotics but are computationally intensive, particularly when handling high-dimensional market-implied volatility surfaces calibrated from liquid listed European options<sup>6,7</sup>. As the complexity of these surfaces and the variety of exotic instruments grow, traditional approaches struggle to scale, limiting their efficiency for real-time pricing and risk assessment.

Recent advances in machine learning (ML)<sup>8,9</sup> have revolutionized financial modeling by offering data-driven, non-parametric alternatives that adapt to complex market patterns directly<sup>10-12</sup>. For options pricing, Physics-informed neural network<sup>13</sup> has been applied on solving the partial differential equations by including the Black-Scholes and related equations into the loss function<sup>14-17</sup>. Although this approach requires no pricing data for training, it is limited by a pre-fixed set of parameters for each

training. Other approaches use pricing data to train ML models such as neural network or Gaussian process regressor to learn the direct mapping from the pricing inputs (e.g. strike, time to maturity, volatility) to the pricing results<sup>18-21</sup>. However, these approaches have typically used flat volatility without accounting for the complexities of volatility surfaces observed from actual market data<sup>22-25</sup>, limiting their practicality for real world applications.

In this work, we introduce a variational autoencoder (VAE)<sup>26-28</sup>-based neural network framework to price options directly from complete sets of market-implied volatility surfaces. Using daily historical end-of-day S&P 500 European options data from 2018 to 2023, we construct arbitrage-free volatility surfaces on a  $41 \times 20$  grid of log-moneyness and time-to-maturity. The VAE compresses these high-dimensional surfaces into a 10-dimensional latent space, a multilayer perceptron (MLP) then maps these latent variables, along with strike and maturity, to prices for American puts and arithmetic Asian options<sup>29</sup>. This approach overcomes the limitations of parameterization models<sup>30,31</sup> by using a data-driven approach that bypasses the computationally intensive numerical solvers, thus offering an efficient, GPU-parallelization compatible pricing methodology that can scale further with increasingly available market data.

## II. METHOD

### A. Data preparation

To prepare the data for training our deep learning model, we first create implied volatility surface data from

<sup>a)</sup>Electronic mail: dingl1@ornl.gov

the end-of-day European options pricing data collected from the freely available optionsDX platform. We then use these volatility surfaces as inputs to generate prices for American puts and arithmetic Asian options using the corresponding pricing engines from the publicly available QuantLib library.<sup>32,33</sup>

For each business day in our data series, we generate the market implied volatility surface from a chain of options pricing data using the standard Black-Scholes pricing formula. For each options pricing data point, we compute the corresponding implied volatility  $\sigma_{BS}$  by solving the  $\sigma_{BS}$  from the pricing formula of European options:

$$\begin{aligned} c(K, T) &= S(0)N(d_1) - e^{-rT}KN(d_2) \\ p(K, T) &= e^{-rT}KN(-d_2) - S(0)N(-d_1) \\ d_1 &= \frac{\log(S/K) + (r + \sigma_{BS}^2/2)T}{\sigma_{BS}\sqrt{T}} \\ d_2 &= d_1 - \sigma_{BS}\sqrt{T} \end{aligned} \quad (1)$$

where  $c(K, T)$  and  $p(K, T)$  are the price of the call and put option, respectively, at strike  $K$  and time to maturity  $T$ .  $S(0)$  is the spot value of the underlying,  $r$  is the risk-free rate,  $N(\cdot)$  is the standard normal cumulative distribution function. For any strike  $K$  and time to maturity  $T$  where we have both put and call prices, we only use the implied volatility computed from the option that is out-of-the-money, i.e., we use the implied volatility of the put option for  $K < S(0)$  and that of the call option for  $K > S(0)$ , respectively, as that is usually the option that is more liquid.

To further standardize the format of the volatility surfaces for efficient machine training, we calculate the  $\sigma_{BS}(k, T)$  surface on a grid fixed in log moneyness  $k = \log(K/S_0)$  and time to maturity  $T$  by interpolation using neighboring data points, resulting in a uniform  $41 \times 20$  matrix for each business day's worth of data, with 41 log moneyness  $k \in [-0.3, 0.3]$  and 20 option maturities in years  $T \in [0.05, 1]$ . The scatter points in Fig. 1 shows two such sample implied volatility surfaces and the corresponding heat maps as observed from the market for SPX. Due to the limited data quality for some of the end-of-day quotes, some interpolated volatility surfaces appear not to be arbitrage-free. As a result, we have to filter out and eliminate these surfaces after carrying out test pricing on vanilla option valuation using QuantLib. In total, we have collected 1051 arbitrage-free volatility surfaces  $\mathbf{F} = \{\sigma_{BS}(k, T)\}$  between 2018 to 2023, and we subsequently divide  $\mathbf{F}$  into a training set  $\{\sigma_{BS}(k, T)\}_{train}$  containing 840 volatility surfaces and a testing set  $\{\sigma_{BS}(k, T)\}_{test}$  containing 211 volatility surfaces.

To generate pricing data for American puts and arithmetic Asian (call and put) options, we use the QuantLib library to compute option prices based on the market-implied volatility surfaces as ground truth valuations. We generate random combinations of log moneyness  $k$

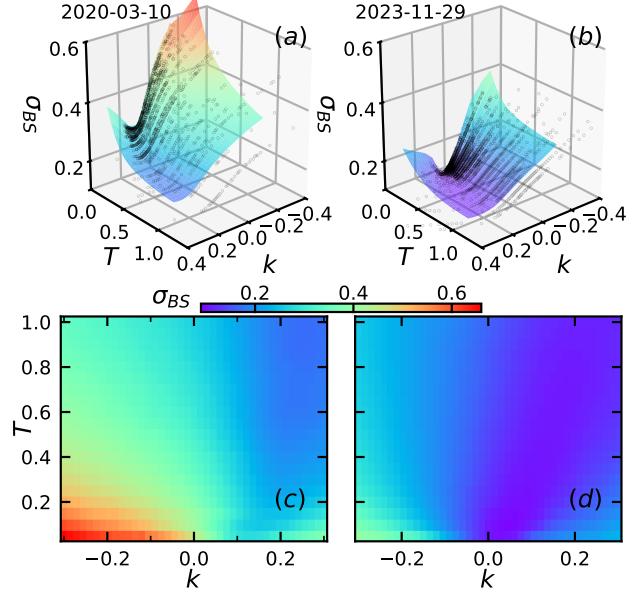


FIG. 1. Illustration of the market implied volatility surfaces for SPX (Standard and Poor's 500 index). (a) Black-Scholes volatility  $\sigma_{BS}$  versus log moneyness  $k = \log(K/S_0)$  and time to maturity in year  $T$  for Mar 10th, 2020, the day after the Covid crash. Gray dots are marker observed point, color surface is interpolated plane. (b) volatility surface at a random day. (c) and (d) are the heat map view of volatility surfaces in (a) and (b), respectively.

and time to maturity  $T$  from uniformly distributed domains of  $k \in [-0.3, 0.3]$  and  $T \in [0.05, 1]$  along with randomly chosen quote dates. Then the entire volatility surface for the corresponding date is passed onto the QuantLib pricer along with the strike  $K = S_0e^k$  and time to maturity  $T$ . For American puts, we have generated 20,000 price data using the volatility surface  $\sigma_{BS}(k, T)$  from the training set, and 4,000 pricing data using the  $\sigma_{BS}(k, T)$  from the testing set. For the arithmetic Asian options, we have prepared 10,000 pricing data for both calls and puts as training set, and 2,000 pricing data for each of calls and puts as testing data.

## B. Variational Autoencoder-based neural network

To learn the mapping between the option prices and the pricing inputs, which include the strike, the time to maturity, and the entire volatility surface (rather than a single interpolated volatility input), we designed a VAE-based neural network as illustrated in Fig. 2. As shown in sec III, this approach is based on the observation that even though the volatility surface representation  $\sigma_{BS}(k, T)$  is high dimensional in nature, the overall shape of the  $\sigma_{BS}(k, T)$  can be captured with a set of much lower dimensional latent variables.

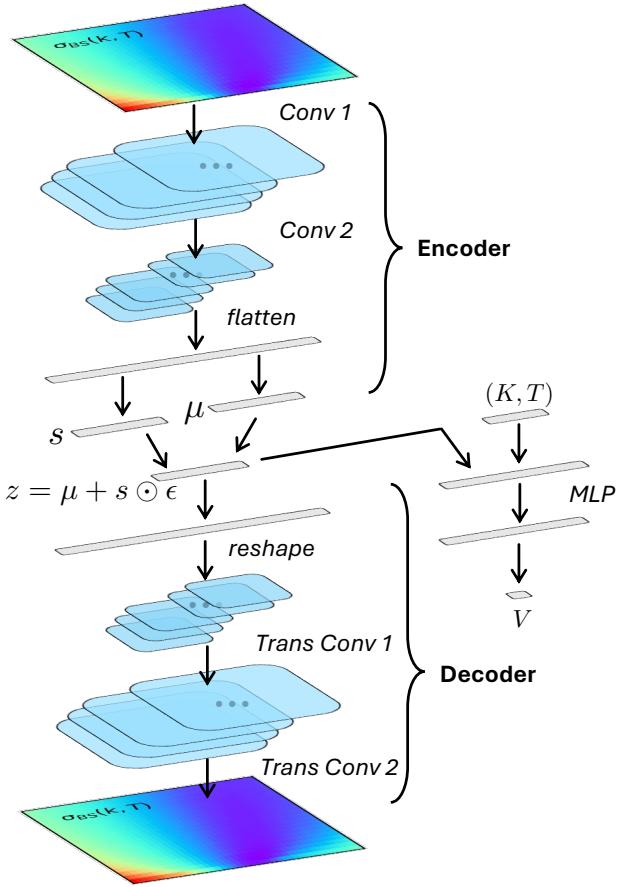


FIG. 2. Architecture of the variational autoencoder (VAE)-based neural network pricer. The neural network consists of two main parts: a VAE that extract the latent variables of the volatility surfaces, and a multilayer perception (MLP) that maps the instrument variables and the latent variables of the volatility surfaces to the option price. The VAE is made of an encoder of 2 convolutional layers, a latent space with 10 dimensions, followed by a decoder of 2 transposed convolutional layers symmetric to the encoder.

Based on this principle, the left part of the neural network is a VAE made of three main components: an encoder, a latent space, and a decoder. The encoder will compress each input volatility surface  $\sigma_{BS}(k, T)$ , a  $41 \times 20$  matrix, into a 10 dimensional latent space, and the decoder will reconstruct the volatility surface  $\sigma'_{BS}(k, T)$  back from the latent space. The encoder consists of two convolutional layers, each with kernel size 3 and strides 2. The first convolutional layer has 16 channels, the second one has 32 channels. After passing the two layers, the  $41 \times 20 \sigma_{BS}(k, T)$  will be transformed into a  $11 \times 5 \times 32 = 1760$  dimensional vector, which is then mapped onto the 10 latent variables, each with mean  $\mu_i$ 's and standard deviation  $s_i$ 's. These variables are rewritten as  $z_i = \mu_i + s_i \odot \epsilon$  with normally distributed variable  $\epsilon \sim \mathcal{N}(0, 1)$ . Values for the 10 latent variables  $z$  are obtained after randomly sampled from  $\epsilon$ , they will then go through the decoder which generates

a reconstructed volatility surface  $\sigma'_{BS}(k, T; \epsilon)$  as output, and this procedure is repeated 10 times, each time with values of  $z$ 's generated from another randomly sampled  $\epsilon$ . The volatility surface averaged over these 10 outputs  $\sigma'_{BS}(k, T) = \langle \sigma'_{BS}(k, T; \epsilon) \rangle_\epsilon$  is the reconstructed volatility surface used in the loss function. The VAE is then trained to minimize the following loss function that calculates the mean squared error between the input and output volatility surfaces, with  $N$  being the number of volatility surfaces:

$$L_{VAE} = \frac{1}{N} \sum_{\sigma_{BS}(k, T)} \langle [\sigma_{BS}(k, T) - \sigma'_{BS}(k, T)]^2 \rangle_{k, T} \quad (2)$$

To learn the price of the options for each set of inputs  $(\sigma_{BS}(k, T), K, T)$ , the volatility surface is fed into the encoder so that a set of latent variables is generated. These latent variables are then entered into the MLP on the right side of the neural network as in Fig. 2, along with the instrument parameters such as strike  $K$  and time to maturity  $T$ , and an option price  $V'$  is generated. The pricer MLP is then trained to minimize the following loss function:

$$L_{MLP} = \frac{1}{M} \sum_V (V' - V)^2 \quad (3)$$

where  $M$  is the number of pricing data corresponding to each combination of  $(\sigma_{BS}(k, T), K, T)$  and  $V$  is the ground truth option price as computed with the same inputs into QuantLib.

To train the entire neural network, we need to first train the VAE until it is able to reconstruct volatility surfaces satisfactorily. During this first stage of training, the MLP is not involved. In the second stage, We focus on training the MLP. During this process, we use the trained encoder to generate the latent variables to be fed into MLP, but the encoder itself is not part of the training, only the MLP is being trained. Lastly, in the fine tuning process, we train both the encoder and the MLP together as driven by MLP loss Eq. (3). In practice, the neural network is implemented using PyTorch and trained using Adam optimizer with CosineAnnealingLR scheduler. We train the VAE for 3,000 epoch, the MLP for 150 epoch and fine tune for another 50 epoch.

### III. RESULTS

We first demonstrate the feasibility of the dimension reduction of the market implied volatility surface data. We then discuss the training of our deep learning model and carry out analysis of the trained model. Finally, we apply our trained model as a pricer for American puts and arithmetic Asian options, both of which do not have closed form solutions.

### A. Feasibility of dimension reduction of volatility surfaces

We first inspect the dataset  $\mathbf{F} = \{\sigma_{BS}(k, T)\}$  of all 1,051 SPX volatility surfaces we collected. By rearranging each volatility surface  $\sigma_{BS}(k, T)$ 's  $41 \times 20$  matrix components into a column vector,  $\mathbf{F}$  becomes a  $1051 \times 820$  matrix with each column representing the volatility surface for a given date. We then carry out principal component analysis of  $\mathbf{F}$  using singular value decomposition such that  $\mathbf{F} = \mathbf{U}\Sigma\mathbf{V}^T$ , in which the  $\mathbf{U}$  is  $1051 \times 1051$ ,  $\Sigma$  is  $1051 \times 820$ , and  $\mathbf{V}^T$  is  $820 \times 820$ . The diagonal entries of the  $\Sigma$  are the singular values that determine the projection of  $\mathbf{F}$  onto the singular vector space  $\mathbf{V}$ .

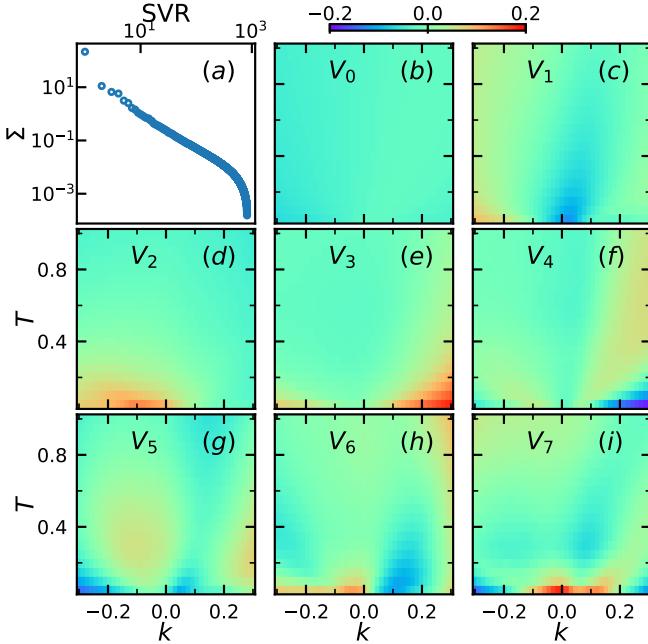


FIG. 3. Principal component analysis of the collected SPX volatility surfaces using singular value decomposition (SVD). (a) Decay of the singular value entry in  $\Sigma$  with  $\mathbf{F} = \mathbf{U}\Sigma\mathbf{V}^T$  versus the singular value rank (SVR). (b)-(i) The first 8 singular vectors of the volatility dataset  $\mathbf{F}$ .

As shown in the log-log scale plot in Fig. 3(a), the singular values decay rapidly as the rank increases, meaning that the higher rank singular vectors in  $\mathbf{V}$  becomes significantly less important in the representation of the volatility surfaces data  $\mathbf{F}$ , thus it is possible to perform dimensional reduction for the  $41 \times 20$  shape volatility surface data<sup>34</sup>. Fig. 3(b)-(i) shows the first 8 singular vectors in  $\mathbf{V}$ , reshaped back to the  $41 \times 20$  representation for easier comparison with the volatility surface data. The first and most dominant singular vector Fig. 3(b) is relatively uniform and flat across strikes and time to maturity, meaning it represents the overall parallel shifts of the entire volatility surface. The blue shade in Fig. 3(c) shows how volatility tends to decrease for higher strike options with time to maturity, and the brown shade shows the increases for the lower strikes options. Together they

can be interpreted as the opening and the closing of the volatility smile as a function of maturity. Fig. 3(d) and (e) mostly adjust the skew at the wings (high strike calls and low strike puts) of the volatility surface. And as the singular value rank increases, the pattern becomes more detailed, meaning each higher rank singular vector is representing ever more subtle aspects of the volatility surface, and are increasing less important. The rapid decay suggests 5–10 dimensions may suffice, motivating our VAE choice.

### B. Neural network training and analysis

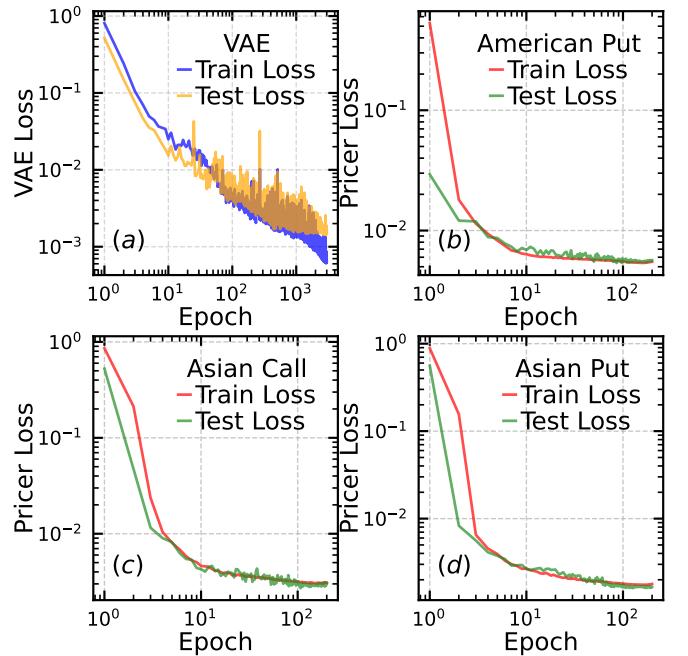


FIG. 4. Loss curves for the training of the neural-network model. (a) Train and test loss for training the variational autoencoder using only the volatility surfaces data. (b)-(d) Train and test loss for training the pricing network using the pricing data consist of both the volatility surface and instrument parameters.

In order to train the neural network model to price options using the entire volatility surface as input, we first need to achieve the dimension reduction of the volatility surface. As a result, we must first train the VAE on the left side of Fig. 2 until sufficiently accurate latent variables can be extracted from the encoder. Fig. 4(a) shows the loss curve of the VAE loss  $L_{VAE}$  versus training epoch. Training loss decreases steadily, while testing loss plateaus in the early thousands. We stopped at 3,000 epoch as the testing loss curve indicates that the network is well-trained without overfitting at that point. After the VAE is trained, we use the encoder to generate the latent variables that are fed into MLP along with  $K$  and  $T$  to train the option pricer. Fig. 4(b)-(d) show the loss

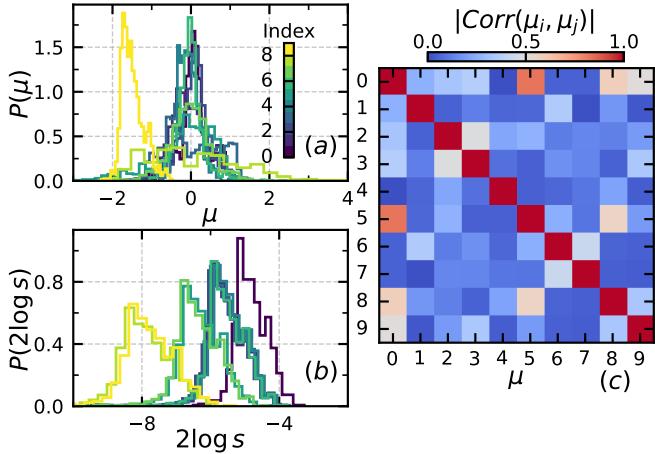


FIG. 5. Distribution and statistics of latent variable of the training set of volatility surface. (a) Density distribution  $P(\mu)$  of all 10 mean latent variables. (b) Density distribution of variance latent variables. (c) Absolute value of the correlation between all mean latent variables ( $\mu_i, \mu_j$ ).

curve for American put, Asian call, and Asian put, respectively. Again we stopped the training when the test loss curve seems to have reached a plateau. These loss curves are all well-behaved and indicate that our training steps are efficient and sufficient.

To further examine the trained neural network, we look at the distribution of the latent variables from all of the training volatility surfaces. Fig. 5(a) and (b) show the distribution of the mean and log variance latent variables. Most of the mean latent variables  $\mu$  are distributed around 0 with one around  $-1$ , while the log variance latent variable  $2\log s$  are mostly very small, at least smaller than  $-4$ , which results in nearly  $s \simeq 0$ . The main takeaway from this comparison is that most of the information is stored in the mean latent variables  $\mu$ 's as their magnitudes dominates those of the variances. In addition, when looking at the correlation of the 10  $\mu$ 's in Fig. 5(c), where the absolute value of the correlations between different indices of  $\mu$  is shown, the heat map shows emerging correlation between different indices of  $\mu$ , such as between  $\mu_0$  and  $\mu_5$ , indicating the number of latent space dimension is starting to saturate, and that our choice of having 10 variables appear sufficient for our dataset.

Furthermore, Fig. 6 demonstrates the reconstruction of the volatility surfaces using the trained VAE. The first row shows the original volatility surfaces  $\sigma_{BS}(k, T)$  for three different quote dates, including the day after the Covid crash (Fig. 6(a)). The second row shows the VAE-reconstructed volatility surfaces  $\sigma'_{BS}(k, T)$ , and the third row shows the difference  $\Delta\sigma_{BS} = \sigma'_{BS}(k, T) - \sigma_{BS}(k, T)$ . The differences  $\Delta\sigma_{BS}$  are relatively small in Fig. 6(h) and (i) as these two dates are fairly typical, while the differences  $\Delta\sigma_{BS}$  are slightly larger for Fig. 6(g). However, given the extreme volatile environment resulting from the Covid driven crash, the reconstructed volatility

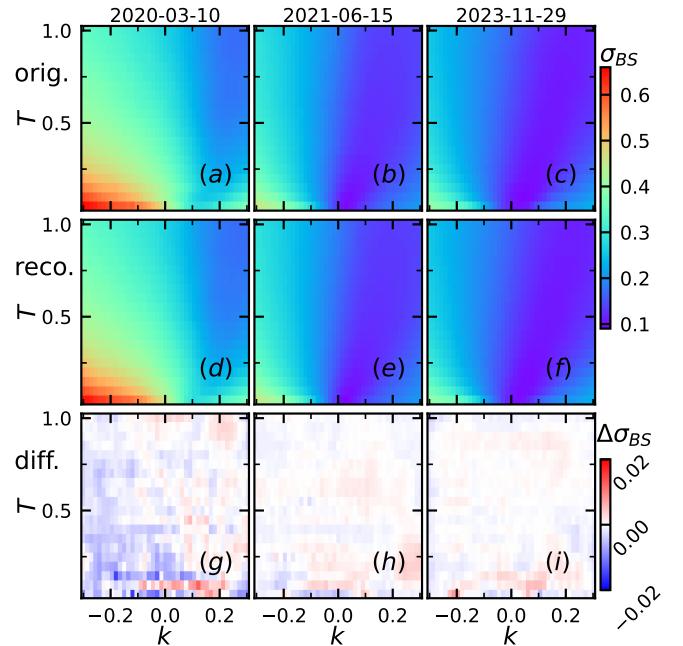


FIG. 6. Sample reconstruction of the volatility surface using the variational autoencoder network. (a)-(c) Original volatility surface of 3 different quote dates from the volatility dataset. (d)-(f) Reconstruction of the volatility surface for (a)-(c), respectively. (g)-(i) Difference  $\delta\sigma_{BS} = \sigma'_{BS} - \sigma_{BS}$  between the reconstruction  $\sigma'_{BS}$  and original volatility surface  $\sigma_{BS}$ .

surface is still quite accurate.

### C. Evaluation of neural network pricer

Finally, we apply the trained neural network pricer to price American Puts and arithmetic Asian options, both lacking closed-form solutions and must be computed numerically. Both the training and testing data sets are generated from historical volatility surfaces, and ground truth valuations are obtained from Quantlib. The training and testing sets are independent, i.e., the neural network was not trained with any volatility surfaces from the set of testing pricing data.

As shown in Fig. 7(a) for American Put prices, the neural network predicted prices align well with the ground truth prices, with all of the predicted prices distributing nicely around the diagonal line of perfect matches. Fig. 7(b) shows a detailed breakdown of the distribution of prediction errors  $Err = V' - V$  in the log moneyness  $k$  and time to maturity  $T$  plane. The errors are overall small, with the few larger errors observed along the longer expiries and close to at-the-money strikes, where absolute prices are low.

Moreover, Fig. 8(a) and (c) show the comparison between the neural network predicted prices versus the ground truth prices for arithmetic Asian call and put op-

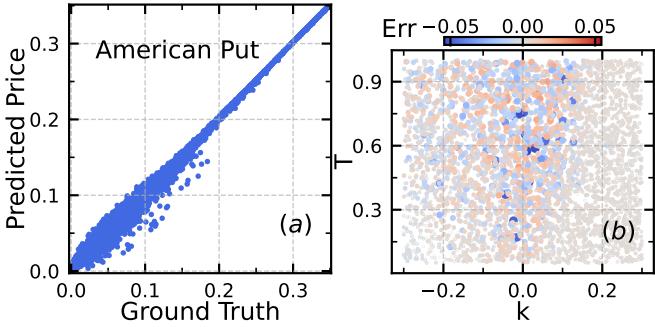


FIG. 7. Benchmark the performance of the neural network pricer for American Put. (a) Comparison between the ground truth price  $P$  and neural network-predicted price  $P'$ . (b) Distribution of  $Err = P' - P$  of all test pricing data in the log moneyness  $k$  and time to maturity  $T$  plane.

tions. For both types of Asian options, the prices again agree very well. Similarly, Fig. 8(b) and (d) show the error in the  $k$  and  $T$  plane. Similar to the American Puts, with the exception of a few outliers close to at-the-money strikes where the absolute prices are low, most of numerical errors are quite modest.

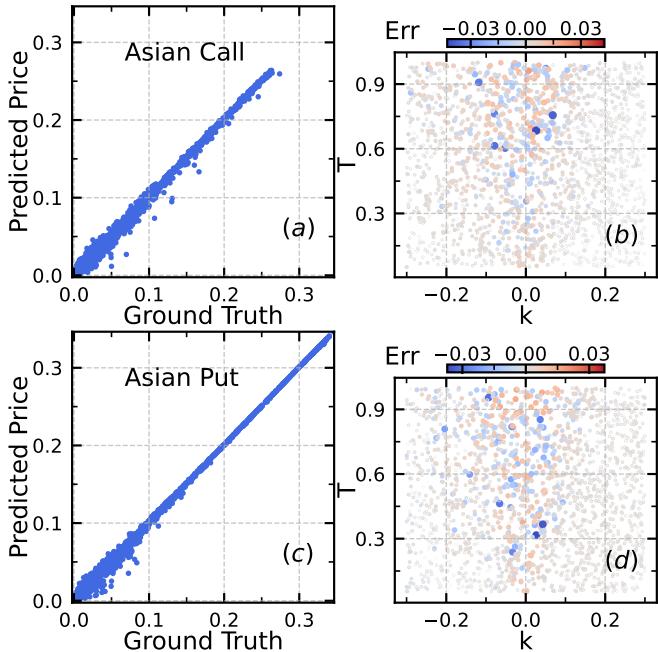


FIG. 8. Benchmark the performance of the neural network pricer for Asian call and Asian put. (a) Comparison between the ground truth price  $V$  and neural network-predicted price  $V'$  for Asian call. (b) Distribution of  $Err = V' - V$  of all test pricing data in the log moneyness  $k$  and time to maturity  $T$  plane for Asian call. (c)-(d) Similar to (a) and (b) but for Asian put.

#### IV. SUMMARY

In this work, we introduce a deep learning approach for options pricing that incorporates the entire volatility surface and bypasses traditional numerical methods. Our model is trained on market-implied volatility surfaces from SPX data, with pricing data generated using the QuantLib package. During the pricing, the entire volatility surface is used instead of using a single interpolated volatility input. We demonstrate that the two-dimensional volatility surface can be compressed into a lower-dimensional representation via singular value decomposition. Building upon this insight, we employ a VAE-based neural network to reduce each volatility surface specified by 820-dimensions into a 10-dimensional latent space represented by the neural network and the 10 sets of latent variables. After we have confirmed that the VAE is able to successfully and accurately reconstruct full volatility surfaces from the latent space, we use the encoder portion of the VAE to generate the latent variables, then together with option parameters such as strike and maturity, they are fed into an MLP to compute option prices. We have applied this methodology to American puts and arithmetic Asian options, whose valuations typically require lengthy numerical calculations. The results show that our neural network achieves high accuracy across all three types of options.

Our market data driven approach is efficient, scalable, and flexible. Unlike numerical methods, it computes exotic prices in a single forward pass and supports GPU-parallel processing of many trades. Demonstrated on end-of-day SPX data with fixed model size and latent dimensions, the architecture scales easily to broader resolutions by adjusting model size. Our deep learning framework will also improve as more data becomes available, making it increasingly accurate and adaptable over time. More importantly, this methodology can be used to compute portfolios of exotic options extremely quickly, thus providing traders with live risk management and intraday monitoring of profit and loss capabilities. Instead of using QuantLib, the architecture can easily accommodate proprietary pricing models to generate ground truth valuations during training such that the trained neural network can generate fast options valuations that are consistent with one's proprietary models, no matter how slow and complicated the proprietary models may be.

Looking ahead, our framework can be extended in several directions. One natural step is to broaden the dataset to include additional equity indices and single-stock options, as well as broaden the types of exotic options and structured products to be evaluated. Beyond equities, the same methodology can also be adapted to other asset classes such as commodities, foreign exchange, cryptocurrency and some interest rate derivatives where options pricing also relies similarly on using such volatility surfaces as inputs. Finally, the VAE-based approach itself offers a promising path toward building a

flexible, accurate and yet tractable volatility surface representation with substantially reduced dimensions<sup>35–37</sup>. It will be interesting to see if this approach can be applied to more complicated volatility inputs such as strike dependent swaption volatility surfaces<sup>38</sup> that are used with fixed income derivatives models such as SABR<sup>39</sup>.

## DATA AVAILABILITY

The code and data for this work are available at the GitHub repository [https://github.com/ljding94/VAE\\_pricing](https://github.com/ljding94/VAE_pricing)

## AUTHOR CONTRIBUTIONS

LD led the research. LD and EL prepared the market data. LD derived the theoretical framework, developed the code, generated and analyzed the data; and LD, EL, and KC wrote and edited the manuscript.

## ACKNOWLEDGMENT

This research was sponsored by the Laboratory Directed Research and Development Program of Oak Ridge National Laboratory, managed by UT-Battelle, LLC, for the U.S. Department of Energy.

## REFERENCES

- <sup>1</sup>F. Black and M. Scholes, “The pricing of options and corporate liabilities,” *Journal of Political Economy* **81**, 637–654 (1973).
- <sup>2</sup>P. Wilmott, *Paul Wilmott on quantitative finance* (John Wiley & Sons, 2013).
- <sup>3</sup>S. E. Shreve *et al.*, *Stochastic calculus for finance II: Continuous-time models*, Vol. 11 (Springer, 2004).
- <sup>4</sup>J. C. Hull, *Options, Futures, and Other Derivatives*, 9th ed. (Pearson, 2017).
- <sup>5</sup>R. Cont and P. Tankov, *Financial modelling with jump processes* (Chapman and Hall/CRC, 2003).
- <sup>6</sup>P. Glasserman, *Monte Carlo Methods in Financial Engineering* (Springer, 2004).
- <sup>7</sup>D. Duffie, *Dynamic asset pricing theory* (Princeton University Press, 2010).
- <sup>8</sup>K. P. Murphy, *Machine learning: a probabilistic perspective* (MIT press, 2012).
- <sup>9</sup>I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*, Vol. 1 (MIT press Cambridge, 2016).
- <sup>10</sup>M. F. Dixon, I. Halperin, and P. Bilokon, *Machine Learning in Finance: From Theory to Practice* (Springer, 2020).
- <sup>11</sup>S. Liu, C. W. Oosterlee, and S. M. Bohte, “Pricing options and computing implied volatilities using neural networks,” *Risks* **7**, 16 (2019).
- <sup>12</sup>A. Hirsa, T. Karatas, and A. Oskoui, “Supervised deep neural networks (dnns) for pricing/calibration of vanilla/exotic options,” *SSRN Electronic Journal* (2019).
- <sup>13</sup>M. Raissi, P. Perdikaris, and G. E. Karniadakis, “Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations,” *Journal of Computational physics* **378**, 686–707 (2019).
- <sup>14</sup>F. Gatta, V. S. Di Cola, F. Giampaolo, F. Piccialli, and S. Cuomo, “Meshless methods for american option pricing through physics-informed neural networks,” *Engineering Analysis with Boundary Elements* **151**, 68–82 (2023).
- <sup>15</sup>D. Hainaut and A. Casas, “Option pricing in the heston model with physics inspired neural networks,” *Annals of Finance* **20**, 353–376 (2024).
- <sup>16</sup>X. Wang, J. Li, and J. Li, “A deep learning based numerical pde method for option pricing,” *Computational economics* **62**, 149–164 (2023).
- <sup>17</sup>Y. Bai, T. Chaolu, and S. Bilige, “The application of improved physics-informed neural network (ipinn) method in finance,” *Nonlinear Dynamics* **107**, 3655–3667 (2022).
- <sup>18</sup>J. De Spiegeleer, D. B. Madan, S. Reyners, and W. Schoutens, “Machine learning for quantitative finance: fast derivative pricing, hedging and fitting,” *Quantitative Finance* **18**, 1635–1643 (2018).
- <sup>19</sup>P. Ndikum, “Machine learning algorithms for financial asset price forecasting,” *arXiv preprint arXiv:2004.01504* (2020).
- <sup>20</sup>R. M. Gaspar, S. D. Lopes, and B. Sequeira, “Neural network pricing of american put options,” *Risks* **8**, 73 (2020).
- <sup>21</sup>D. Anderson and U. Ulrych, “Accelerated american option pricing with deep neural networks,” *Quantitative Finance and Economics* **7**, 207–228 (2023).
- <sup>22</sup>J. Cao, J. Chen, J. Hull, and Z. Poulos, “Deep learning for exotic option valuation,” *arXiv preprint arXiv:2103.12551* (2021).
- <sup>23</sup>J. Ruf and W. Wang, “Neural networks for option pricing and hedging: a literature review,” *arXiv preprint arXiv:1911.05620* (2019).
- <sup>24</sup>D. A. Bloch, “Option pricing with machine learning,” Available at SSRN 3486224 (2019).
- <sup>25</sup>R. Culkin and S. R. Das, “Machine learning in finance: the case of deep learning for option pricing,” *Journal of Investment Management* **15**, 92–100 (2017).
- <sup>26</sup>C. Doersch, “Tutorial on variational autoencoders,” *arXiv preprint arXiv:1606.05908* (2016).
- <sup>27</sup>Y. Pu, Z. Gan, R. Henao, X. Yuan, C. Li, A. Stevens, and L. Carin, “Variational autoencoder for deep learning of images, labels and captions,” *Advances in neural information processing systems* **29** (2016).
- <sup>28</sup>M. Bergeron, N. Fung, J. Hull, and Z. Poulos, “Variational autoencoders: A hands-off approach to volatility,” *arXiv preprint arXiv:2102.03945* (2021).
- <sup>29</sup>J. Vecer, “A new pde approach for pricing arithmetic average asian options,” *Journal of computational finance* **4**, 105–113 (2001).
- <sup>30</sup>J. Gatheral and A. Jacquier, “Arbitrage-free svi volatility surfaces,” *Quantitative Finance* **14**, 59–71 (2014).
- <sup>31</sup>L. Ding, E. Lu, and K. Cheung, “Fast derivative valuation from volatility surfaces using machine learning,” *arXiv preprint arXiv:2505.22957* (2025).
- <sup>32</sup>J. R. Varma and V. Virmani, *Derivatives Pricing using QuantLib: An Introduction* (Indian Institute of Management, 2015).
- <sup>33</sup>J. R. Varma and V. Virmani, “Computational finance using quantlib-python,” *Computing in Science & Engineering* **18**, 78–88 (2016).
- <sup>34</sup>D. Ackerer, N. Tagasovska, and T. Vatter, “Deep smoothing of the implied volatility surface,” *Advances in Neural Information Processing Systems* **33**, 11552–11563 (2020).
- <sup>35</sup>B. Ning, S. Jaimungal, X. Zhang, and M. Bergeron, “Arbitrage-free implied volatility surface generation with variational autoencoders,” *SIAM Journal on Financial Mathematics* **14**, 1004–1027 (2023).
- <sup>36</sup>J. Wang, S. Liu, and C. Vuik, “Controllable generation of implied volatility surfaces with variational autoencoders,” *arXiv preprint arXiv:2509.01743* (2025).

- <sup>37</sup>B. T. Kelly, B. Kuznetsov, S. Malamud, and T. A. Xu, “Deep learning from implied volatility surfaces,” Swiss Finance Institute Research Paper (2023).
- <sup>38</sup>R. Fan, A. Gupta, and P. Ritchken, “Hedging in the possible presence of unspanned stochastic volatility: Evidence from swaption

markets,” *The Journal of Finance* **58**, 2219–2248 (2003).

- <sup>39</sup>P. Hagan, A. Lesniewski, and D. Woodward, “Probability distribution in the sabr model of stochastic volatility,” in *Large deviations and asymptotic methods in finance* (Springer, 2015) pp. 1–35.