DIFFUSION PROBABILISTIC MODEL FOR IMPLIED VOLATILITY SURFACE GENERATION AND COMPLETION

Xiaochan MA

to obtain the degree of Master of Science
in Applied Mathematics
at the Delft University of Technology,

Student number:	5462533
Project duration:	February 2023 – December 2023
Thesis committee:	Dr. Shuaiqiang Liu, TU Delft & ING Bank, supervisor
	Prof.Dr. Cornelis Vuik, TU Delft
	Prof.Dr. Papapantoleon, TU Delft

TU Delft

Acknowledgement

I would like to acknowledge my supervisor, Dr. Shuaiqiang Liu. His unwavering commitment and dedication have been a beacon of guidance throughout my thesis journey. Despite his busy schedule, Dr. Liu consistently made time to mentor me almost every week, meticulously reviewing and improving my work with an exceptional level of detail and care. His valuable insights, critical comments, and constructive suggestions have significantly shaped my research and writing, instilling in me a pursuit of excellence and thoroughness. His high standards and relentless dedication have not only made me better at what I do but have also deeply inspired me in my academic endeavors.

I also extend my heartfelt thanks to Dr. Kees Vuik and Dr. Antonis Papapantoleon for their guidance and willingness to serve as my thesis defense examiners. Their expertise and feedback have been invaluable in shaping my thesis and aiding my graduation journey.

I would also like to give special thanks to my boyfriend, and my family as a whole for their continuous support and understanding when undertaking my research and writing my report.

Last, but not least, I want to express my deepest gratitude and appreciation to my friends and classmates who helped me in one way or another during my study and thesis work.

ABSTRACT

Implied volatility surfaces are integral to option pricing and risk management but often display missing data. Prior research has typically engaged mathematical models or data-driven methods for generating or completing these surfaces. Given the similarity between implied volatility surfaces and images, our paper introduces the Denoising Diffusion Probabilistic Model (DDPM), a novel deep learning image generation model, for this task. A distinctive aspect of DDPM's training involves progressively adding noise to the surfaces until they resemble pure noise, and then learning to denoise back to the original surfaces. We employ the Heston model to simulate implied volatility surfaces, then train the DDPM using this synthetic data. Additionally, a Variational Autoencoder model (VAE) is implemented as a comparative benchmark for assessing DDPM's efficacy. Our experiments demonstrate that DDPM excels in generating and reconstructing missing areas in implied volatility surfaces, highlighting its potential in this field. Looking to the future, combining DDPM with VAE could provide more interpretable results, enhancing the model's utility and applicability in financial analysis.

Contents

1 Introduction

1.1. Problem description

Implied volatility plays a crucial role in the financial industry, with its applications spanning from options pricing to risk management. This indicator is an indispensable part of financial markets and provides a unique window into market expectations regarding future price fluctuations. In the options market, implied volatility impacts option prices and the choice of trading strategies since it is one of the core factors in options pricing. Furthermore, financial institutions and investors rely on implied volatility to assess and manage the risk in their portfolios, with high implied volatility suggesting increased market uncertainty and low implied volatility potentially reflecting market stability and optimism.

When dealing with European options, a holder possesses the right, though not the obligation, to exercise the contract upon reaching maturity time T. The fair price $V(t,S)$ can be determined by the Black-Scholes (BS) equation, and computing the implied volatility is searching for volatility for the Black-Scholes pricing model so that the model can provide an option price that matches the observed option price. We will discuss more details about the BS model in chapter 2.

Unfortunately, a direct formulaic representation of the implied volatility is unavailable. It requires the use of numerical calculation methods, such as the bisection method or the Newton-Raphson method, to approximate the solution.

In reality, the implied volatility also experiences variations across strike prices and maturity periods. This leads to the formation of implied volatility surfaces, symbolized by $\sigma^* := \sigma^*(T-t, K)$. Such volatility surfaces are extensively utilized in pricing financial derivatives and find profound significance in hedging and risk management. The complexity of these surfaces as in Figure 1.1 is evident in patterns like the volatility smile/skew and term structure. However, clear-cut formulas that capture the essence of these volatility surfaces are also absent.

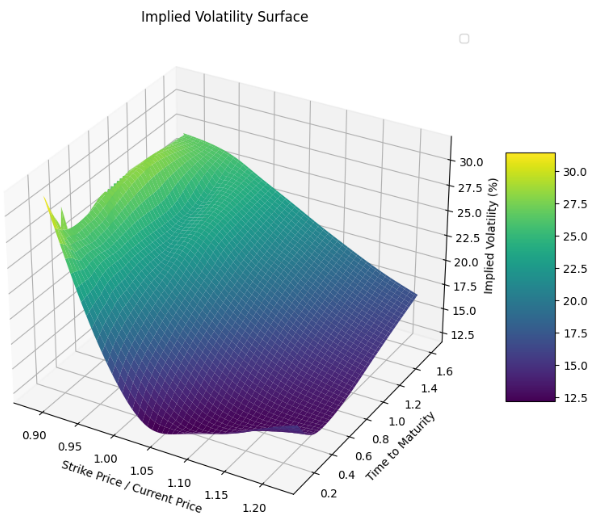


Figure 1.1: Implied volatility surface S&P-500 options, November 5th 2023.

There have been a lot of efforts in the literature to model implied volatility surfaces, mainly categorized into two approaches: using mathematical models for fitting and utilizing data-driven methods. In the latter approach, many researchers have applied various machine learning models to estimate implied volatility surfaces. We aim to explore a novel model and compare its performance with existing models. Through this approach, we aim to advance the research on implied volatility surface estimation, potentially providing more accurate and reliable tools for forecasting market volatility.

1.2. BACKGROUND

An option is a financial derivative that gives the holder the right, but not the obligation, to buy or sell an underlying asset at a predefined price within a set period. When two parties enter into an option contract, the buyer acquires the right to choose to exercise it or not. The seller of the option is obligated to honor the contract if the buyer decides to exercise it. As a result of this imbalance between rights and obligations, the buyer pays a premium to the seller for the rights granted by the option. The value of this premium is influenced by several factors, including the current price of the underlying asset, the option's strike price, the time until expiration, and notably, the expected volatility of the underlying asset's price. This expected volatility embedded in the option's market price is referred to as the 'implied volatility'.

In real markets, implied volatility is a dynamic quantity that reflects traders' changing expectations and sentiments about future price movements. Since it is impossible to determine the future volatility of an asset, implied volatility is an important consensus of market expectations. This consensus is particularly important because it helps determine the risk and reward trade-offs of options and influences the decisions of individual and institutional investors.

In addition, it is important to understand that implied volatility is not uniform across strike prices and expiry dates. This variation in implied volatility is often visually represented using an implied volatility surface. The implied volatility surface provides a comprehensive picture of how the market views future price movements under different scenarios, making it a valuable tool for traders, risk managers and financial strategists.

1.3. OVERVIEW OF METHODS FOR IVS

Implied volatility surface modeling is a key component in the pricing and risk management of financial derivatives. Due to its significance, researchers have proposed various methods to model and predict implied volatility. Broadly speaking, these methods can be categorized into two main types, mathematical modeling and data-driven approaches.

1.3.1. MATHEMATICAL MODELING

Mathematical modeling primarily relies on specified mathematical models to describe the price dynamics of the underlying asset. The core idea of this approach is to capture and describe the variations in asset prices through specific mathematical formulas.

For example, the Heston stochastic volatility model presumes volatility to be a stochastic process, and its dynamics can be described by certain stochastic differential equations [1]. Researchers use historical market data to estimate the parameters of these mathematical models, hoping to more accurately capture the characteristics of volatility in the market. Building on the foundational Heston model, other mathematical approaches have been developed to enhance the modeling of implied volatility surface. Affine jump-diffusion models introduced by [2] provide a rich framework for capturing the term structure of implied volatility surface. Non-parametric models offer flexibility without the constraint of predefined functional forms, with spline-based methods ensuring an arbitrage-free surface as demonstrated by [3].

The challenge of calibrating these models to market data has led to various numerical techniques. Fast calibration methods using Fourier transforms, as shown by [4], have improved the efficiency of the Heston model calibration. [5] have provided insights into the calibration of affine jump-diffusion models, which are crucial for capturing the dynamics of implied volatility surface.

The main advantage of the mathematical modeling method is its interpretability since models are often based on principles from economics or physics, offering a deeper understanding of market volatility changes. However, this method also has its drawbacks. The most apparent limitation is that no single model can perfectly fit all volatility surfaces. Even if a model performs well during a specific period or market environment, it might fail under other conditions. This is because market behaviors and dynamics are intricate and ever-changing, influenced by numerous unpredictable external factors.

1.3.2. DATA-DRIVEN MODELING

On the other hand, data-driven approaches focus more on the data itself rather than relying on any predetermined mathematical model. In the domain of implied volatility surface modeling, traditional data-driven approaches have been foundational. Principal Component Analysis (PCA) is one such technique that has significantly contributed to the field. Alexander's work [6] on PCA has been instrumental in distilling the essential factors that influence the volatility surface. By reducing the dimensionality of the data, PCA helps in identifying the major movements of the surface, which is beneficial for simplifying the complex structure of the financial markets. Time series analysis represents another cornerstone of implied volatility surface modeling. The study by Cont and da Fonseca using autoregressive models [5] is a prime example of this approach. By capturing the evolution of the implied volatility surface over time, these models reveal persistent and mean-reverting features of volatility.

With the advent of machine learning, particularly deep learning, new methodologies have been developed to address the intricate patterns found in financial markets. For instance, when using deep learning to model the volatility surface, researchers might design a neural network model and train it with vast amounts of market data, enabling it to predict future volatilities. In recent years, researchers have begun to experiment with the application of the Variational Autoencoder (VAE) in financial modeling, particularly in the modeling of implied volatility surfaces. This trend highlights the attractiveness of combining traditional quantitative financial modeling with modern data-driven approaches.

Ning et al. [7] is an early exploration of using VAE to generate arbitrage-free implied volatility surfaces. Initially, they used a stochastic differential equation (SDE) model to fit parameters and then used VAE to generate them directly. In doing so, they have innovatively combined the mathematical rigor of SDEs with the adaptability of data-driven VAEs, providing a powerful framework for modeling volatility surfaces well suited to real-world financial market requirements.

In contrast, other researchers have chosen to use a purely data-driven approach. For example, Bergeron et al. [8] have innovated a "hands-off" approach. This approach emphasizes the minimal manual intervention required for VAE, in contrast to traditional models that often require calibration and fine-tuning. In particular, their work highlights the adaptability of VAEs to the multifaceted nature of fluctuating surfaces, demonstrating their resilience. Similarly, Dierckx, Davis and Schoutens [9] harnessed the VAE to autonomously learn volatility characteristics, surpassing conventional volatility modeling techniques. Their study underscores its practical implications for the financial sector through synthetic surface generation, portfolio stress-testing, and anomaly detection.

On the other hand, unlike the Gaussian distribution used by most, Borovkova and van den Oever [10] introduce a completely new perspective by integrating the Student-t distribution into a latent space of VAE tailored for large portfolios. This approach is particularly important because the Student-t distribution is adept at capturing the "fat tails" that are common in financial returns, ensuring that VAE models remain resilient even under extreme market conditions.

The primary advantage of the data-driven method is its flexibility since it does not require any explicit assumptions about the dynamics of the underlying asset price, allowing it to adapt better to market changes. However, a significant downside of data-driven approaches is their lack of interpretability. Given that these models often function as a "black box," it is challenging to understand their inner workings or how they make predictions.

Deep learning techniques have shown promising results in generating arbitrage-free implied volatility surfaces, as demonstrated in recent research. Two distinct methodologies emerge from the literature: penalization in training neural networks and combining stochastic differential equation models with variational autoencoders.

The first method, explored in [11] and [8], involves the incorporation of penalization terms during the training of Generative Adversarial Networks (GANs) and VAEs, respectively. In GANs, the generator network is trained specifically to minimize arbitrage violations, assisted by a discriminator network. For VAEs, the approach is slightly different. Although initially penalization for constructing arbitrage surfaces was considered, it was found to be largely unnecessary as VAEs naturally tended to produce arbitrage-free surfaces.

On the other hand, the second method, as presented in [7], diverges from the concept of penalization. Instead, it integrates model-free VAEs with stochastic differential equation (SDE) driven models, including regime switching models and Lévy additive processes. This approach involves projecting historical volatility surfaces onto the SDE models' parameter space and then training the VAEs on these parameters. The key here is the combination of VAEs' generative capabilities with the robustness of SDE models, ensuring that the resulting implied volatility surfaces are consistent with historical data and inherently free from arbitrage.

1.4. RESEARCH OBJECTIVES

This study aims to delve deeper into data-driven methods. Specifically, it attempts to apply one of the emerging models in image generation, namely the Denoising Diffusion Probabilistic Model (DDPM), to the modeling of implied volatility surfaces. We aim to provide a more stable and precise method for predicting and modeling implied volatility surfaces.

The main applications center on two areas, the generation of new implied volatility surfaces and the completion of these surfaces. Generating new implied volatility surfaces is particularly crucial for financial institutions. In practical operations, by generating new volatility surfaces, investors and traders can predict and evaluate potential future market trends, offering them valuable decision-making insights. Given the demonstrated excellence of the DDPM model in image generation, we have reason to believe that it could hold similar potential for generating implied volatility surfaces.

The significance of surface completion lies in the fact that in real market scenarios, implied volatility surfaces might exhibit partial data missing. These omissions could arise due to insufficient trading volumes, market anomalies, or other reasons. However, when engaging in asset pricing, risk management, or other financial analyses, a complete implied volatility surface is essential. By employing the DDPM model for surface completion, we can efficiently bridge these data gaps, offering financial professionals a more comprehensive and accurate implied volatility reference. This not only aids them in better understanding the current market situation but also bolsters their decision-making.

1.5. CONTRIBUTIONS

As previously mentioned, our contributions, primarily focusing on the data-driven approach, can be detailed as follows:

1. Implementation of DDPM on Implied Volatility Surfaces: We endeavored to apply the DDPM model to implied volatility surface modeling. With its notable success in the image generation domain, we aim to leverage DDPM's capabilities for generating new volatility surfaces and completing existing ones with potentially improved stability and precision.

2. Reproduction and Enhancement of Existing Method: We reproduced the existing VAE method applied to the implied volatility surface. Further, we use the VAE model for the generation of new surfaces and surface completion. Notably, for surface completion, we optimized by calibrating the latent space. This calibration offers a thorough adjustment that potentially better models and predicts missing parts of the implied volatility surface.

3. Comparative analysis of DDPM and VAE: We compare the results obtained using the DDPM and VAE methods. Using a variety of metrics and qualitative assessments, we evaluate the performance, stability, and accuracy of both models in generating and completing implied volatility surfaces. Our findings provide valuable insights into the strengths and limitations of each methodology, providing practitioners in the field with a more informed decision-making process.

In summary, our work not only reproduced and enhanced the already prevailing models but also integrated newer models. This improved implied volatility surface modeling and forecasting accuracy and reliability in real financial applications. This innovative integration and application of data-driven approaches aim to provide advanced insights and viable solutions to the challenges faced in modeling implied volatility surfaces.

1.6. THESIS OUTLINE

This thesis extensively explores the implications of generative deep learning models on the implied volatility surface. Beginning with an introduction that frames the problem, the work offers an overview of current methods applied to implied volatility surfaces and a literature review of recent research. Subsequently, there is an in-depth look into the fundamentals of the implied volatility surface, highlighting models like Black–Scholes and the Heston model in Chapter 2. Chapter 3 dives deep into generative deep learning, elucidating concepts of deep learning, generative models, the Variational Autoencoder, and the Denoising Diffusion Probabilistic Model. Chapter 4 discusses how these models interact with implied volatility surfaces, including data generation techniques, training methodologies, and various applications. Several robust evaluation metrics are presented to assess the performance of these methods. Chapter 5 provides a detailed analysis of the numerical results, comparing the results of VAE with DDPM, and giving a comparative analysis. Chapter 6 finally offers a summary of the insights gained from the research and points toward future areas of exploration in this domain.

2

Implied Volatility Surface

2.1. Related Research

Research into implied volatility traces back many years. It is an indispensable concept in financial mathematics, providing insight into the market's expectations of future asset price fluctuations [12]. Implied volatility is not derived from historical data, as historical volatility is, but is inferred directly from the market price of an option. It represents the market's general expectation of the future level of volatility of the underlying asset. The interest in implied volatility stems from its forward-looking nature, which makes it the indicator of choice for traders and investors who are more concerned with future uncertainty than past movements.

At the center of modern financial theory is the Black-Scholes equation. Proposed by Fischer Black, Myron Scholes and Robert Merton, this differential equation provided the first continuous-time mathematical model for option pricing[12, 13]. Their framework made several simplifying assumptions, including persistent volatility. Although it has had a significant impact on derivatives trading by providing a structured approach to valuation, real-world differences have been observed, leading to further research and model enhancements.

The difference between the theoretical price of the Black-Scholes model and the actual market option price has given rise to the implied volatility surface [14]. This graphical representation shows how implied volatility varies with different strike prices and expiry dates. It provides valuable insights into market anomalies, sentiment or external factors that may not be immediately apparent from price data alone. The complexities and shapes found in the implied volatility surface, such as the "volatility smile", have led to further research into more advanced models.

The Heston model is one of such advances. Recognizing the limitations of the assumption of constant volatility in the Black-Scholes model, Steven Heston introduced a model in which volatility itself follows a stochastic process [1]. The innovation of the model is its ability to account for the clustering of volatility observed in real-world financial markets, where periods of high volatility tend to be accompanied by periods of similar volatility and vice versa. The flexibility and adaptability of Heston's model made it more consistent with observed market behavior and cemented its position as the model of choice for many financial practitioners.

2.2. BLACK–SCHOLES MODEL

The Black-Scholes model is a landmark in the option pricing field. It provides a theoretical framework for calculating the value of European-style options. The model takes into account a variety of factors including the current price of the asset, the strike price of the option, the risk-free rate, the time to expiration, and the volatility of the asset price. In other words, the Black-Scholes model provides a quantitative method of measuring the value of an option, allowing traders to have a clearer picture of their investment.

Then we turn to see how can we get the framework of the Black-Scholes model. Generally speaking, option pricing models like the Black-Scholes model are based on the concept of a portfolio of hedged securities. An investor can create a portfolio of options and their underlying stocks to ensure a certain return. In equilibrium, this guaranteed return must be equal to the risk-free rate. The nature of option pricing is consistent with the principle of no-arbitrage pricing. This principle states that any zero-cost investment can only earn zero return, and any non-zero-cost investment can only earn a return corresponding to its inherent risk, without generating excess profits. The derivation of the Black-Scholes option pricing model clearly shows that option pricing is fundamentally no-arbitrage pricing.

The BS model makes several key assumptions. First, stock price movements are random and follow a lognormal distribution. Second, the risk-free rate, the stock's expected return variable, and price volatility remain constant over the life of the option. Further, the market is frictionless, which means there are no taxes or transaction costs. Fourth, the stock does not pay dividends or any other income during the option term, an assumption that can be relaxed. Fifth, the option is a European-style option, which means it cannot be exercised before expiration. Sixth, there are no risk-free arbitrage opportunities in financial markets. Finally, trading in financial assets can occur continuously and allows short selling with the entire proceeds of the financial asset.

The Black-Scholes model is fundamentally based on the concept of geometric Brownian motion (GBM), which describes the random movement of an asset's price over time. GBM is a continuous-time stochastic process in which the logarithm of the asset price follows a Brownian motion with drift. Mathematically, it is expressed as

$$\mathrm{dS} = \mu S\mathrm{d}t + \sigma S\mathrm{dW},$$

where S is the asset price, μ is the drift rate, σ is the volatility, and dW is the increment of a Wiener process or standard Brownian motion. This model assumes that the percentage change in the asset price is normally distributed, making GBM a suitable model for the random behavior of financial markets.

With this foundation, the BS model can be derived using a no-arbitrage argument, which establishes the relationship between an option and its underlying asset. By constructing a portfolio (Π) consisting of a position in an option (V) and a position in the underlying asset (S), one can hedge away the risks. The portfolio's value is defined as

$$\Pi = V(S,t) - \Delta S,$$

where Δ represents the number of shares of the underlying. The differential change in this portfolio, $\mathrm{d}\Pi$, can be expressed as

$$\mathrm{d}\Pi = \frac{\partial V}{\partial t}\mathrm{dt} + \frac{\partial V}{\partial S}\mathrm{dS} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2}\mathrm{dt} - \Delta\mathrm{dS}.$$

To create a risk-free portfolio, the term involving dS is set to zero, resulting in

$$\Delta = \frac{\partial V}{\partial S}.$$

Subsequently, the risk-free portfolio change is

$$\mathrm{d}\Pi = \left(\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2}\right)\mathrm{dt}.$$

Equating this to the risk-free rate return

$$\mathrm{d}\Pi = r\Pi\mathrm{dt}.$$

Upon integrating, we arrive at the Black-Scholes partial differential equation,

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0.$$

This equation provides the theoretical foundation for the Black-Scholes option pricing formula. In this formula, we have understood the meaning of all symbols except for σ. This remaining σ represents the volatility of the underlying asset's returns. It is a measure of the asset's future uncertainty and is a crucial input in the option pricing formula. However, σ is not directly observable in the market. What traders and analysts do observe are the prices at which options are traded. This is where the concept of implied volatility (σ) comes in. Instead of inputting the volatility into the Black-Scholes model to find the option price, market practitioners use the observed market price of the option and infer the volatility. This reverse-engineered volatility, σ, represents the market's aggregate expected volatility of the underlying asset for the remaining life of the option.

The process of deriving σ_{IV} involves using the Black-Scholes model in conjunction with market prices, and iteratively adjusting σ until the output price of the model matches the market price. The σ that achieves this matching is implied volatility, a key parameter used by market participants to measure market sentiment and make trading decisions.

2.3. IMPLIED VOLATILITY SURFACE

As we mentioned before, one of the key assumptions of the Black-Scholes model is that the volatility of the underlying asset remains constant. However, in the real world, this assumption is clearly invalid or flawed. Numerous empirical studies have shown that the volatility of financial asset prices is not a fixed constant, but rather a process that changes over time. As a result, the implied volatility derived from option market prices and the BS formula exhibits two main patterns: the "volatility smile" and the volatility term structure.

First, the volatility smile refers to the phenomenon that implied volatility varies with the strike price of the option. Since implied volatility is a function of strike price and expiry date, it is worth noting that implied volatility is at its lowest level when the strike price is equal to the initial stock price of S_0. As the strike price deviates from S_0, implied volatility increases, creating what is commonly referred to as a "volatility smile". In addition, the implied volatility increases as the maturity date increases. This means that for the same underlying product with a fixed residual maturity, the implied volatility will vary with its strike price. The curve plotted on the axis of the strike price and implied volatility forms a volatility "smile". The existence of the volatility smile suggests that the assumptions underpinning the BS pricing model are only partially validated in real financial markets. The shape of implied volatility is not always the same for different financial options. In general, the implied volatility of currency options is roughly U-shaped. Flat options have the lowest volatility, while the volatility of real and imaginary options increases with their currency and appears somewhat symmetric on both sides. On the other hand, the implied volatility of stock options is usually L-shaped, sloping to the lower right. Volatility decreases as the strike price increases, and options with lower strike prices imply significantly higher volatility than options with higher strike prices. The slope of the stock volatility smile is usually negative due to the significant negative skewness of stock returns.

Volatility Term Structure refers to the phenomenon where the implied volatility changes with different option expiration dates. Specifically, with all other factors remaining constant, it represents the variation in implied volatility corresponding to at-the-money options due to different expiration dates. In general, the term structure's exact shape differs for various underlying assets.

In the long term, volatilities tend to mean-revert. This means that as the expiration date approaches, the changes in implied volatility become more pronounced. As the time to maturity extends, the implied volatility gradually moves closer to the historical volatility's average value.

The shape of the volatility smile is also influenced by the option's expiration date. Generally, the closer the expiration date, the more pronounced the volatility "smile." For longer maturities, the difference in implied volatilities across various strike prices diminishes, approaching a constant.

Combining the volatility smile with the volatility term structure, we can derive the volatility surface, which provides insights into the market's expectations of the future distribution of the asset. This volatility surface is also known as the volatility matrix. Suppose for a specific stock S, the options market has effective quotes for a set of options with tenures T_i (where $i = 1,2,3,...I$) and strike prices K_j (where $j = 1,2,...J$, represented as $C_{ij} = C(T_i, K_j)$). We can then calculate the corresponding implied volatilities σ_{ij}. This allows us to form a surface representing the implied volatilities.

2.4. HESTON MODEL

In our exploration of option pricing models, it becomes evident that to fully capture the intricacies of the implied volatility surface observed in markets, a more complex model is required. While the Black-Scholes model has its merits, it assumes constant volatility, an oversimplification that fails to capture the dynamics seen in real markets. One solution to this limitation is to introduce stochastic volatility models, which allow volatility to be a random process.

In the Black-Scholes model, the only source of randomness is the stock price, so it can be hedged with the underlying stock. However, in the stochastic volatility model, volatility is also stochastic, which means that the portfolio needs to be hedged against two sources of randomness: stock price and volatility. Assume both the stock price and its instantaneous volatility satisfy the SDEs:

$$\begin{aligned}
dS_t &= \mu_t S_t dt + \sqrt{v_t} S_t dZ_1,\\
dv_t &= \alpha(S_t, v_t, t)\, dt + \eta\beta(S_t, v_t, t)\sqrt{v_t} dZ_2,\\
\langle dZ_1 dZ_2\rangle &= \rho dt.
\end{aligned}$$

To achieve a risk-free investment, we need to construct a portfolio with two hedging terms, a position in the underlying stock with quantity $-\Delta$, which means that the portfolio is delta-hedged against changes in the stock price, and a position in a volatility-dependent asset with quantity $-\Delta_1$, which means that the portfolio is vega-hedged against changes in implied volatility. By doing so, we can eliminate the risk associated with both the stock price and the volatility, and create a risk-free investment

$$\Pi = V(S, v, t) - \Delta S - \Delta_1 V_1(S, v, t).$$

The change in the portfolio value can be written as

$$\begin{aligned}
d\Pi =& \left\{\frac{\partial V}{\partial t} dt + \frac{\partial V}{\partial S} dS + \frac{\partial V}{\partial v} dv + \frac{1}{2}\frac{\partial^2 V}{\partial S^2}(dS)^2 + \frac{1}{2}\frac{\partial^2 V}{\partial v^2}(dv)^2 + \frac{\partial^2 V}{\partial S \partial v} dS dv\right\} - \Delta dS\\
&- \Delta_1\left\{\frac{\partial V_1}{\partial t} dt + \frac{\partial V_1}{\partial S} dS + \frac{\partial V_1}{\partial v} dv + \frac{1}{2}\frac{\partial^2 V_1}{\partial S^2}(dS)^2 + \frac{1}{2}\frac{\partial^2 V_1}{\partial v^2}(dv)^2 + \frac{\partial^2 V_1}{\partial S \partial v} dS dv\right\}.
\end{aligned}$$

Here we use Ito's lemma to derive the dV,dV_1. Then we can substitutes the expressions for dS, dv and $\langle dZ_1 dZ_2\rangle = \rho dt$ from the given stochastic process. Thus, we have

$$\begin{aligned}
d\Pi =& \left\{\frac{\partial V}{\partial t} + \frac{1}{2} v S^2 \frac{\partial^2 V}{\partial S^2} + \rho\eta v\beta S \frac{\partial^2 V}{\partial v \partial S} + \frac{1}{2}\eta^2 v\beta^2 \frac{\partial^2 V}{\partial v^2}\right\} dt\\
&- \Delta_1\left\{\frac{\partial V_1}{\partial t} + \frac{1}{2} v S^2 \frac{\partial^2 V_1}{\partial S^2} + \rho\eta v\beta S \frac{\partial^2 V_1}{\partial v \partial S} + \frac{1}{2}\eta^2 v\beta^2 \frac{\partial^2 V_1}{\partial v^2}\right\} dt\\
&+ \left\{\frac{\partial V}{\partial S} - \Delta_1 \frac{\partial V_1}{\partial S} - \Delta\right\} dS + \left\{\frac{\partial V}{\partial v} - \Delta_1 \frac{\partial V_1}{\partial v}\right\} dv.
\end{aligned}$$

To ensure that the portfolio is risk-free, we need to eliminate the stochastic terms dS and dv, i.e. $\frac{\partial V}{\partial S} - \Delta_1 \frac{\partial V_1}{\partial S} - \Delta = 0$ and $\frac{\partial V}{\partial v} - \Delta_1 \frac{\partial V_1}{\partial v} = 0$.

This leaves us with

$$\Delta_1 = \frac{\partial V_1}{\partial v} / \frac{\partial V}{\partial S}, \tag{2.1}$$

and

$$\begin{aligned} d\Pi = &\left\{ \frac{\partial V}{\partial t} + \frac{1}{2} v S^2 \frac{\partial^2 V}{\partial S^2} + \rho \eta v \beta S \frac{\partial^2 V}{\partial v \partial S} + \frac{1}{2} \eta^2 v \beta^2 \frac{\partial^2 V}{\partial v^2} \right\} dt \\ &- \Delta_1 \left\{ \frac{\partial V_1}{\partial t} + \frac{1}{2} v S^2 \frac{\partial^2 V_1}{\partial S^2} + \rho \eta v \beta S \frac{\partial^2 V_1}{\partial v \partial S} + \frac{1}{2} \eta^2 v \beta^2 \frac{\partial^2 V_1}{\partial v^2} \right\} dt \\ &. \end{aligned} \tag{2.2}$$

In addition, since the risk-free portfolio has a return at the risk-free rate r, we have

$$d\Pi = r\Pi dt = r\left(V - \Delta S - \Delta_1 V_1\right) dt \tag{2.3}$$

where the left-hand side represents the change in the value of the portfolio Π over a small time increment dt. The right-hand side represents the rate at which the value of the portfolio is expected to grow over time due to the risk-free interest rate r.

Combining equations (2.1), (2.2), (2.3), we get

$$\begin{aligned} &\frac{\frac{\partial V}{\partial t} + \frac{1}{2} v S^2 \frac{\partial^2 V}{\partial S^2} + \rho \eta v \beta S \frac{\partial^2 V}{\partial v \partial S} + \frac{1}{2} \eta^2 v \beta^2 \frac{\partial^2 V}{\partial v^2} + rS \frac{\partial V}{\partial S} - rV}{\frac{\partial V}{\partial v}} \\ = &\frac{\frac{\partial V_1}{\partial t} + \frac{1}{2} v S^2 \frac{\partial^2 V_1}{\partial S^2} + \rho \eta v \beta S \frac{\partial^2 V_1}{\partial v \partial S} + \frac{1}{2} \eta^2 v \beta^2 \frac{\partial^2 V_1}{\partial v^2} + rS \frac{\partial V_1}{\partial S} - rV_1}{\frac{\partial V_1}{\partial v}} \end{aligned}$$

The left-hand side is a function of V only and the right-hand side is a function of V_1 only. The only way that this can be done is for both sides to be equal to some function f of the independent variables S, v and t. We deduce that

$$\frac{\partial V}{\partial t} + \frac{1}{2} v S^2 \frac{\partial^2 V}{\partial S^2} + \rho \eta v \beta S \frac{\partial^2 V}{\partial v \partial S} + \frac{1}{2} \eta^2 v \beta^2 \frac{\partial^2 V}{\partial v^2} + rS \frac{\partial V}{\partial S} - rV = -(\alpha - \phi \beta \sqrt{v}) \frac{\partial V}{\partial v} \tag{2.4}$$

where, without loss of generality, we have written the arbitrary function f of S, v, and t as $\alpha - \phi\beta\sqrt{v}$.

Among various stochastic volatility models, the Heston model stands out as one of the most prominent. We assume that the Heston process generates the risk-neutral measure, so the market price of volatility risk ϕ in the general valuation equation (2.4) is set to zero now. Then (2.4) becomes

$$\frac{\partial V}{\partial t} + \frac{1}{2} v S^2 \frac{\partial^2 V}{\partial S^2} + \rho \eta v S \frac{\partial^2 V}{\partial v \partial S} + \frac{1}{2} \eta^2 v \frac{\partial^2 V}{\partial v^2} + rS \frac{\partial V}{\partial S} - rV = \lambda(v - \bar{v}) \frac{\partial V}{\partial v}.$$

Then suppose that we consider only the future value, we derive the European option pricing formula under the Heston model by replacing some variables, $x = log(\frac{F_{t,T}}{K})$ and

$\tau = T - t$.

$$-\frac{\partial C}{\partial \tau} + \frac{1}{2} v C_{11} - \frac{1}{2} v C_1 + \frac{1}{2}\eta^2 v C_{22} + \rho \eta v C_{12} - \lambda(v - \bar{v}) C_2 = 0 \tag{2.5}$$

The solution of (2.5) is

$$C(x, v, \tau) = K\left\{e^x P_1(x, v, \tau) - P_0(x, v, \tau)\right\},$$

where the first term P_1 represents the pseudo-expectation of the final index level given that the option is in-the-money, similar to the $N(d_1)$ in the BS model, and the second term represents the pseudo-probability of exercise $P(S_T > K)$, similar to the $N(d_1)$ in the BS model. By processing the complex variable integral, we can obtain

$$P_j(x, v, \tau) = \frac{1}{2} + \frac{1}{\pi}\int_0^\infty du \operatorname{Re}\left\{\frac{\exp\left\{C_j(u,\tau)\bar{v} + D_j(u,\tau)v + iux\right\}}{iu}\right\}.$$

Finally, the simulation of the Heston model can be obtained by discretizing the iterative relationship equation. Common methods include Eulerian discretization (negative variance problem), Milstein discretization, etc.

3 Generative Deep Learning Models

This chapter mainly provides background knowledge of generative deep learning models related to our research. Section 3.1 offers a brief introduction to deep learning, introducing its foundational concepts and several neural network architectures needed in our explorations. Moving forward, Section 3.2 provides an overview of generative models, with a specific focus on their application in image generation. In Section 3.3 and Section 3.4, we respectively discuss the mechanics, architectures and training processes of DDPM and VAE models.

3.1. Deep learning framework

Deep Learning is a research direction in the field of Machine Learning. It was introduced into machine learning to bring it closer to its initial goal, Artificial Intelligence.

Deep Learning is about understanding the intrinsic patterns and hierarchical representations in sample data. The information acquired during these learning processes greatly assists in interpreting data such as text, images, and sounds. Its ultimate goal is to endow machines with the ability to analyze and learn, akin to human capabilities, and to recognize data like text, images, and sounds. Deep Learning is a sophisticated machine learning algorithm. The results it achieves in speech and image recognition far surpass those of previous related technologies.

In our research, we mainly employed theories and algorithms from the field of image generation in deep learning. Before delving into specific details, we need first to understand some basic neural network architectures.

Fully Connected Network The Fully Connected network structure (FC) is the most basic layer in neural networks. Each node in a fully connected layer is connected to all nodes of the previous layer. In the early days, the fully connected layer was primarily used for classifying extracted features. However, since every output in the fully connected layer is connected to every input, the fully connected layer typically has a pretty high number of parameters. This requires a significant amount of storage and computational space.

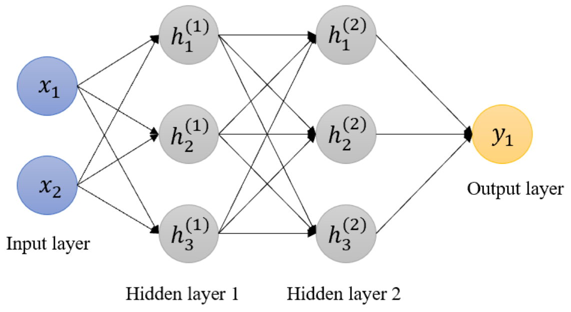


Figure 3.1: A sample structure of FC with two hidden layers.

The problem of parameter redundancy means that conventional neural networks, consisting purely of FC layers, are rarely applied in more complex scenarios. Conventional neural networks are typically used in simple scenarios that depend on all features. For example, the house price prediction model and the online advertising recommendation model both use relatively standard fully connected neural networks.

Convolutional Neural Network The Convolutional Neural Network (CNN) is also a type of neural network specifically designed to handle data with a grid-like structure, such as image data (which can be viewed as a two-dimensional pixel grid). Unlike FC, not all neurons in the upper and lower layers of a CNN are directly connected. Instead, they are connected through convolutional kernels as intermediates, and then sharing of these kernels to greatly reduce the number of parameters in the hidden layer.

A basic CNN consists of a series of layers, and each layer transforms one neuron into another through a differentiable function. These layers primarily include the convolutional layer, pooling layer, and fully connected layer.

The convolutional layer is the core building block of a CNN. The primary role of the convolutional layer is to extract features from images. It achieves this through the learning of convolutional kernel weights according to an objective function to capture the required features. The convolutional kernel is capable of dimension reduction or elevation, facilitating cross-channel interactions and information integration. Additionally, the kernel can deepen and widen the depth of learning network layers at a minimal parameter cost.

The pooling layers follow the convolutional layers and are responsible for reducing the spatial size of the convolved featured. It keeps feature invariance, enabling the model to prioritize certain features over their exact spatial placement within the image, thus helping to screen essential features. In addition, through feature dimensionality reduction, the model is able to extract a wide range of feature information. The pooling layer can also effectively prevent overfitting and enhance the generalization ability of the model.

The fully connected layer mainly plays the role of a classifier in deep learning. After previous convolution, pooling and activation layer processing, the raw data has been mapped to the feature space of the hidden layer. The task of the fully connected layer is to map these distributed features to the label space of the samples. However, due to the problem of parameter redundancy in the fully connected layer, many recent high-performance network models, such as ResNet and GoogLeNet, employ global average pooling instead of the traditional fully connected layer in order to fuse deep features.

CNNs have shown excellent performance in many application domains, especially in large-scale image-processing scenarios. They are the backbone of most computer vision systems for tasks like image classification, image generation, object detection, and segmentation. The hierarchical nature of CNNs ensures that they can learn patterns from data at various levels of abstraction.

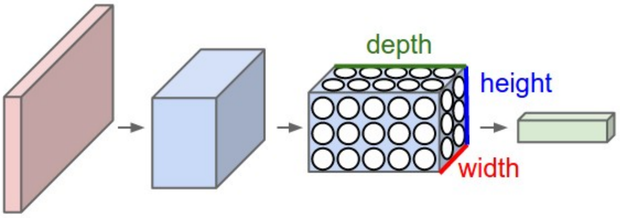


Figure 3.2: A ConvNet arranges its neurons in three dimensions (width, height, depth), as visualized in one of the layers [15].

Figure 3.2 shows the structural format of a CNN, where neurons are arranged in three dimensions (width, height, and depth) to form the convolutional neural network. As demonstrated by one of the layers in the figure, each layer of the CNN transforms a 3D input into a 3D output.

As for general structure, CNNs typically use an architecture resembling autoencoders. Such architecture consists of an encoding part, a bottleneck that represents high-level features in the latent space, and a decoder that synthesizes the output using transposed convolutions, also known as deconvolutions. The distinction of transposed convolution is the application of strides on the output instead of the input.

Notably, there is a specialized CNN architecture called U-Net from [15], which incorporates direct skip connections between down-sampling and up-sampling layers of the same level. These connections are particularly beneficial when the input and output share abundant low-level structures. In our research on the DDPM, we primarily employ the U-Net structure for the model's architecture.

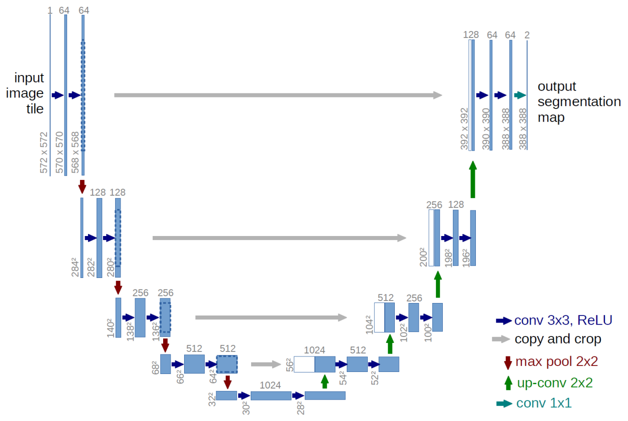


Figure 3.3: U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations [15].

3.2. GENERATIVE MODEL

Generative models, literally speaking, can be understood as models designed to generate samples. Conceptually, the process begins with a given dataset. Based on this dataset, a model is built, and model parameters are learned from the data. Using the learned parameters, new data can be constructed using the model.

Generative models occupy a notable place in the field of machine learning, aspiring to model the underlying distribution of data to generate new, similar instances [16]. These models are designed based on the characteristics of the sample distribution and aim to generate images based on user needs. On the one hand, research on image generation helps build large data sets required for deep learning training and helps computers enhance their understanding of image information. On the other hand, advances in image generation have led to critical applications such as image editing, image restoration, and text-to-image conversion, offering huge potential in practical applications.

These models are broadly categorized into various types such as Variational Autoencoder (VAE), Generative Adversarial Network (GAN), Diffusion models, and Autoregressive models like PixelRNN and PixelCNN, each boasting unique mechanisms and strengths [17, 18, 19, 20]. Their utilities extend across a diverse spectrum of applications, not limited to synthesizing images, but also encompassing tasks like drug discovery, text-to-speech conversion, and reinforcement learning environments [21, 22, 23]. Essentially, these models endeavor to decipher and embody the intrinsic structure and regularities within a dataset, enabling the generation of samples that are not merely imitative but hold the statistical essence of the original data [24]. Our research primarily focuses on VAE and DDPM, and the following sections will discuss these two models in detail.

Variational Autoencoder (VAE) was introduced by Kingma and Welling in 2013. By combining deep learning with probabilistic graphical models, VAE establishes a balance between data representation and generation. This paved the way for its use from unsupervised learning to data generation [25].

Over time, various improvements and extensions have emerged that have enhanced its applicability [26]. For instance, [27] exploring the structure of the VAE model, a novel anomaly detection technique based on reconstruction probabilities was introduced. It provides a more principled measure of anomalies compared to traditional reconstruction error methods. This approach integrates anomaly detection into VAE by primarily utilizing data from typical instances and positioning low reconstruction probability data points as criteria. Further advances in the field of VAE have proposed models such as β-VAE and CVAE. The β-VAE introduced by [28] employs the hyperparameter β to reconcile the reconstruction error with the KL divergence, which affects the quality of the latent space representation. On the other hand, the CVAE proposed by [29] provides a mechanism in which the generative model is conditioned on input observations, and the input is modulated with the a priori Gaussian latent variables responsible for the output.

Further contributions of VAE have been observed in various fields. For example, molecular design has been automated using VAE,[30] and similarity metrics beyond pixels have been explored [31]. Other research has dissected sources of disentanglement in VAEs [32], developed neural audio synthesis [33], and even ventured into collaborative filtering [34].

The evolution of diffusion models in the domain of deep learning traces back to the work on Denoising Diffusion Probabilistic Models (DDPM) presented in 2020 [19]. Such models take inspiration from the theory of stochastic processes [35] and advancements in nonequilibrium thermodynamics [36]. These models offered a robust framework where data generation can be perceived as a reversed noise addition process. Following this, Song et al. proposed the Denoising Diffusion Implicit Model (DDIM) [37] which introduced novel techniques for accelerating the diffusion sampling procedure and demonstrated the importance of variance schedule selection.

Building upon these foundational contributions, a lot of recent works have emerged, pushing the boundaries of diffusion models in various dimensions. Among these, the GLIDE model [38] has made strides towards achieving photorealistic image generation and editing, guided by textual inputs. Similarly, DALL·E2 [39] and Imagen [40] have shown the versatility of text-conditional image generation, by integrating CLIP latent for hierarchical image synthesis and deep language understanding. Furthermore, the diffusion model with latent space was explored by Rombach et al. [41], named the latent diffusion model. This model greatly accelerates the diffusion process. It also introduces the notion of conditioning, specifically translating textual inputs to images, tapping into the area of AI painting. Today, the confluence of diffusion models, latent spaces, and conditional mechanisms has made them a cornerstone in AI painting, signifying their indispensable role in modern generative methods.

3.3. DENOISING DIFFUSION PROBABILISTIC MODEL (DDPM)

Diffusion denoising relies on the concept of diffusion, inspired by the natural process where molecules move from areas of high to low concentration. In the context of data denoising, especially images, the idea is to smooth out noise while preserving key features. More specifically, the DDPM defines a Markov chain of diffusion steps to slowly add random noise to data and then learn to reverse the diffusion process to construct desired data samples from the noise. Unlike VAE, diffusion models are learned with a fixed procedure and the latent variable has the same dimension as the original data. As shown in 5.8, generating the original image from Gaussian random noise is a reverse process, and vice versa is a forward process.

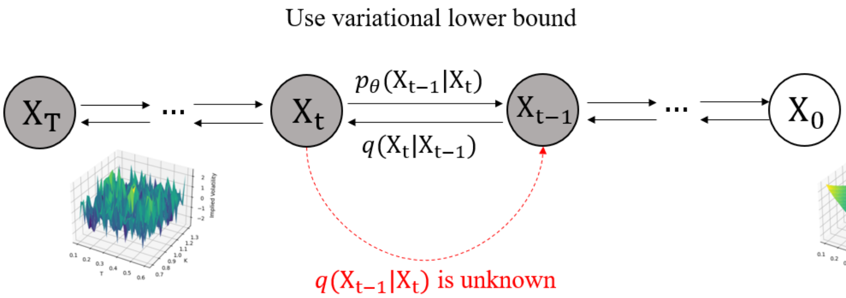


Figure 3.4: The Markov chain of forward (reverse) diffusion process.

3.3.1. FORWARD DIFFUSION PROCESS

The so-called forward process is the process of adding noise to the image step by step. Although this process cannot directly generate images, it is a crucial part of understanding the diffusion model. Beginning with an image represented as $x_0 \sim q(x)$, this image serves as the origin of the diffusion process. As part of the diffusion forward process, Gaussian noise is systematically applied over T iterations, leading to a series of transformed images: $x_1, x_2, \ldots, x_T$. This series can be visualized in the q process as shown in 5.8.

To determine the nature of the Gaussian noise at each iteration, a set of hyperparameters is defined. Specifically, $\{\beta_t \in (0,1)\}_{t=1}^{T}$ outlines the variances of the Gaussian distributions at each step. One key aspect of this process is its Markovian nature. The image at stage t is influenced solely by its predecessor at stage $t-1$. This can be mathematically expressed as

$$q(x_t \mid x_{t-1}) = \mathcal{N}\left(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t \mathbf{I}\right), \quad q(x_{1:T} \mid x_0) = \prod_{t=1}^{T} q(x_t \mid x_{t-1}) \tag{3.1}$$

where $\mathbf{I}$ represents the identity matrix. As the iterations t increases, the image x_t becomes more dominated by noise. Given a sufficiently large T, x_T approaches an isotropic Gaussian distribution. The reparameterization trick mentioned in Section 3.4 helps in understanding the noise evolution. For simplification, some terms can be represented as $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{i=1}^{T} \alpha_i$. Then the relations rewrite the reparameterization of x_t as

$$
\begin{aligned}
x_t &= \sqrt{\alpha_t}x_{t-1} + \sqrt{1-\alpha_t}\epsilon_1, && \text{where} \quad \epsilon_1, \epsilon_2, \ldots \sim \mathcal{N}(0, \mathbf{I}) \\
&= \sqrt{\alpha_t}\left(\sqrt{\alpha_{t-1}}x_{t-2} + \sqrt{1-\alpha_{t-1}}\epsilon_2\right) + \sqrt{1-\alpha_t}\epsilon_1 \\
&= \sqrt{\alpha_t\alpha_{t-1}}x_{t-2} + \left(\sqrt{\alpha_t(1-\alpha_{t-1})}\epsilon_2 + \sqrt{1-\alpha_t}\epsilon_1\right) \\
&= \sqrt{\alpha_t\alpha_{t-1}}x_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\bar{\epsilon}_2, && \text{where} \quad \bar{\epsilon}_2 \sim \mathcal{N}(0, \mathbf{I})(*) \\
&\vdots \\
&= \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\bar{\epsilon}_t
\end{aligned}
\tag{3.2}
$$

The inherent Gaussian properties of the noise in this process enable the application of Gaussian properties. The additive property suggests that the sum of two independent Gaussian noises gives another Gaussian distribution with a variance equal to the sum of their individual variances, i.e.

$$\sqrt{\alpha_t(1-\alpha_{t-1})}\epsilon_2 + \sqrt{1-\alpha_t}\epsilon_1 \sim \mathcal{N}(0, [\alpha_t(1-\alpha_{t-1}) + (1-\alpha_t)]\mathbf{I}) = \mathcal{N}(0, (1-\alpha_t\alpha_{t-1})\mathbf{I}). \tag{3.3}$$

Merging two Gaussian distributions yields a consolidated Gaussian distribution with a modified standard deviation. This observation guarantees the fact that $\bar{\epsilon}_2$ remains Gaussian in (3.2). Consequently, any x_t in our process can be expressed as

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)\mathbf{I})$$

This mathematical representation emphasizes our ability to generate random noise directly from the starting image.

In summary, the forward diffusion process offers a clear method to progressively transform an initial image into its noisy counterpart.

3.3.2. REVERSE DIFFUSION PROCESS

If the forward diffusion process is characterized as the procedure of noise addition, the reverse diffusion process can be understood as a denoising inference mechanism of diffusion. This process essentially reverses the original data, denoted as x_0, from its final noisy representation, which can be considered as $x_T \sim \mathcal{N}(0, \mathbf{I})$.

It has been proven in [42] that if the forward transition $q(x_t \mid x_{t-1})$ follows a Gaussian distribution and the noise parameter β_t is sufficiently small, then the reverse transition $q(x_{t-1} \mid x_t)$ retains its Gaussian nature. However, a straightforward inference of $q(x_{t-1} \mid x_t)$ is complex. As a resolution, deep learning models, specifically the U-Net model in our paper, are employed to predict this inverse transition

$$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}\left(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)\right), p_\theta(x_{0:T}) = p(x_T)\prod_{t=1}^{T} p_\theta(x_{t-1} \mid x_t). \tag{3.4}$$

Even though directly computing the reverse distribution $q(x_{t-1} \mid x_t)$ remains intractable, possessing knowledge of x_0 permits the computation of $q(x_{t-1} \mid x_t, x_0)$ as

$$q\left(x_{t-1} \mid x_t, x_0\right) = \mathcal{N}\left(x_{t-1}; \tilde{\mu}\left(x_t, x_0\right), \tilde{\beta}_t \mathbf{I}\right). \tag{3.5}$$

More specifically, we can prove it using Bayes' rule

$$\begin{aligned}
q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0\right) &= q\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{x}_0\right) \frac{q\left(\mathbf{x}_{t-1} \mid \mathbf{x}_0\right)}{q\left(\mathbf{x}_t \mid \mathbf{x}_0\right)} \\
&\propto \exp\left(-\frac{1}{2}\left(\frac{\left(\mathbf{x}_t - \sqrt{\alpha_t}\mathbf{x}_{t-1}\right)^2}{\beta_t} + \frac{\left(\mathbf{x}_{t-1} - \sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0\right)^2}{1-\bar{\alpha}_{t-1}} - \frac{\left(\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0\right)^2}{1-\bar{\alpha}_t}\right)\right) \\
&= \exp\left(-\frac{1}{2}\left(\frac{\mathbf{x}_t^2 - 2\sqrt{\alpha_t}\mathbf{x}_t\mathbf{x}_{t-1} + \alpha_t\mathbf{x}_{t-1}^2}{\beta_t} + \frac{\mathbf{x}_{t-1}^2 - 2\sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0\mathbf{x}_{t-1} + \bar{\alpha}_{t-1}\mathbf{x}_0^2}{1-\bar{\alpha}_{t-1}} - \frac{\left(\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0\right)^2}{1-\bar{\alpha}_t}\right)\right) \\
&= \exp\left(-\frac{1}{2}\left(\left(\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}}\right)\mathbf{x}_{t-1}^2 - \left(\frac{2\sqrt{\alpha_t}}{\beta_t}\mathbf{x}_t + \frac{2\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}}\mathbf{x}_0\right)\mathbf{x}_{t-1} + C\left(\mathbf{x}_t, \mathbf{x}_0\right)\right)\right).
\end{aligned} \tag{3.6}$$

$C\left(\mathbf{x}_t, \mathbf{x}_0\right)$ is a function not involving $\mathbf{x}_{t-1}$, so we can omit it. (3.6) transforms the entire reverse process back to the forward process. Meanwhile, following the standard Gaussian density function, the mean and variance can be calculated as

$$\tilde{\beta}_t = 1/\left(\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}}\right) = 1/\left(\frac{\alpha_t - \bar{\alpha}_t + \beta_t}{\beta_t\left(1-\bar{\alpha}_{t-1}\right)}\right) = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\cdot\beta_t; \tag{3.7}$$

$$\begin{aligned}
\tilde{\boldsymbol{\mu}}_t\left(\mathbf{x}_t, \mathbf{x}_0\right) &= \left(\frac{\sqrt{\alpha_t}}{\beta_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}}\mathbf{x}_0\right) / \left(\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}}\right) \\
&= \left(\frac{\sqrt{\alpha_t}}{\beta_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}}\mathbf{x}_0\right) \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\cdot\beta_t \\
&= \frac{\sqrt{\alpha_t}\left(1-\bar{\alpha}_{t-1}\right)}{1-\bar{\alpha}_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0.
\end{aligned} \tag{3.8}$$

Based on reparameterization, we can express x_0 in terms of x_t as $x_0 = \frac{1}{\sqrt{\bar{a}_t}}\left(x_t - \sqrt{1-\bar{a}_t}\bar{\epsilon}_t\right)$. Then incorporating this into (3.8), we could derive

$$\tilde{\mu}_t = \frac{1}{\sqrt{a_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{a}_t}}\bar{\epsilon}_t\right).$$

In this framework, the Gaussian distribution $\bar{\epsilon}_t$ symbolizes the noise predicted by the deep learning model for denoising purposes. This can be equated to $\epsilon_\theta\left(x_t, t\right)$, which gives the final expression

$$\mu_\theta\left(x_t, t\right) = \frac{1}{\sqrt{a_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{a}_t}}\epsilon_\theta\left(x_t, t\right)\right). \tag{3.9}$$

As such, the inference of each step in DDPM can be summarized as

1. At each time step, the Gaussian noise $\epsilon_\theta\left(x_t, t\right)$ is predicted based on x_t and t. Subsequently, the mean $\mu_\theta\left(x_t, t\right)$ is obtained according to equation (3.9).

2. The variance $\Sigma_\theta(x_t, t)$ is then determined. In DDPM, an untrained $\Sigma_\theta(x_t, t) = \tilde{\beta}_t$ is used, and it is believed that the results of $\tilde{\beta}_t = \beta_t$ and $\tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \cdot \beta_t$ are similar.
3. Based on equation (3.5), $q(x_{t-1} \mid x_t)$ is obtained. Then, x_{t-1} is derived using reparameterization.

3.3.3. TRAINING

So far, we have discussed the entire process of diffusion. The next step is to train a neural network to obtain reliable values for $\mu_\theta(x_t, t)$ and $\Sigma_\theta(x_t, t)$. This is achieved by maximizing the log-likelihood of the model's predicted distribution under the true data distribution, specifically by optimizing the cross entropy of $p_\theta(x_0)$ under $x_0 \sim q(x_0)$:

$$\mathscr{L} = \mathbb{E}_{q(x_0)}\left[-\log p_\theta(x_0)\right] \tag{3.10}$$

As seen in 5.8, this process is similar to VAE, implying that the negative log-likelihood can be optimized using the Variational Lower Bound (VLB). Since the KL divergence is non-negative, we have

$$\begin{aligned}
-\log p_\theta(x_0) &\leq -\log p_\theta(x_0) + D_{KL}\left(q(x_{1:T} \mid x_0) \| p_\theta(x_{1:T} \mid x_0)\right) \\
&= -\log p_\theta(x_0) + \mathbb{E}_{q(x_{1:T}\mid x_0)}\left[\log \frac{q(x_{1:T} \mid x_0)}{p_\theta(x_{0:T}) / p_\theta(x_0)}\right] \\
&= -\log p_\theta(x_0) + \mathbb{E}_{q(x_{1:T}\mid x_0)}[\log \frac{q(x_{1:T} \mid x_0)}{p_\theta(x_{0:T})} + \log p_\theta(x_0)] \\
&= \mathbb{E}_{q(x_{1:T}\mid x_0)}\left[\log \frac{q(x_{1:T} \mid x_0)}{p_\theta(x_{0:T})}\right].
\end{aligned} \tag{3.11}$$

where $\quad p_\theta(x_{1:T} \mid x_0) = \frac{p_\theta(x_{0:T})}{p_\theta(x_0)}$.

Taking the expectation of both sides of (3.11) for $\mathbb{E}_{q(x_0)}$ and applying Fubini's theorem in the double integral, we then get

$$\mathscr{L}_{VLB} = \mathbb{E}_{q(x_0)}\left(\mathbb{E}_{q(x_{1:T}\mid x_0)}\left[\log \frac{q(x_{1:T} \mid x_0)}{p_\theta(x_{0:T})}\right]\right) = \mathbb{E}_{q(x_{0:T})}\left[\log \frac{q(x_{1:T} \mid x_0)}{p_\theta(x_{0:T})}\right] \geq \mathbb{E}_{q(x_0)}\left[-\log p_\theta(x_0)\right]$$

Therefore, minimizing $\mathscr{L}_{VLB}$ suffices to minimize our target loss $\mathscr{L}$ in (3.10). Further deriving $\mathscr{L}_{VLB}$, we can get the accumulation of entropy and multiple KL divergences

$$\begin{aligned}
L_{\mathrm{VLB}} &= \mathbb{E}_{q(\mathbf{x}_{0:T})}\left[\log \frac{q(\mathbf{x}_{1:T} \mid \mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})}\right] \\
&= \mathbb{E}_q\left[-\log p_\theta(\mathbf{x}_T) + \sum_{t=1}^{T} \log \frac{q(\mathbf{x}_t \mid \mathbf{x}_{t-1})}{p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)}\right] \\
&= \mathbb{E}_q\left[\log \frac{q(\mathbf{x}_T \mid \mathbf{x}_0)}{p_\theta(\mathbf{x}_T)} + \sum_{t=2}^{T} \log \frac{q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)} - \log p_\theta(\mathbf{x}_0 \mid \mathbf{x}_1)\right] \\
&= \mathbb{E}_q[D_{\mathrm{KL}}\left(q(\mathbf{x}_T \mid \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_T)\right) + \sum_{t=2}^{T} D_{\mathrm{KL}}\left(q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)\right) - \log p_\theta(\mathbf{x}_0 \mid \mathbf{x}_1)].
\end{aligned} \tag{3.12}$$

Since there are no learnable parameters in the forward process q, X_T is pure Gaussian noise. In the derived results, the first term, that is, the entropy term, can be regarded as a constant and ignored; and Every KL term compares two Gaussian distributions and therefore they can be computed in closed form, i.e.

$$\begin{aligned}
D_{\mathrm{KL}}\left(q\left(\mathbf{x}_t \mid \mathbf{x}_{t+1}, \mathbf{x}_0\right) \| p_\theta\left(\mathbf{x}_t \mid \mathbf{x}_{t+1}\right)\right) &= \mathbb{E}_{x_0, \bar{\epsilon}_t}\left[\frac{1}{2\left\|\Sigma_\theta\left(x_t, t\right)\right\|_2^2}\left\|\tilde{\mu}_t\left(x_t, x_0\right)-\mu_\theta\left(x_t, t\right)\right\|^2\right] \\
&= \mathbb{E}_{x_0, \bar{\epsilon}_t}\left[\frac{\beta_t^2}{2 \alpha_t\left(1-\bar{\alpha}_t\left\|\Sigma_\theta\right\|_2^2\right)}\left\|\bar{\epsilon}_t-\epsilon_\theta\left(x_t, t\right)\right\|^2\right] \\
&= \mathbb{E}_{x_0, \bar{\epsilon}_t}\left[\frac{\beta_t^2}{2 \alpha_t\left(1-\bar{\alpha}_t\left\|\Sigma_\theta\right\|_2^2\right)}\left\|\bar{\epsilon}_t-\epsilon_\theta\left(\sqrt{\bar{\alpha}_t} x_0+\sqrt{1-\bar{\alpha}_t} \bar{\epsilon}_t, t\right)\right\|^2\right].
\end{aligned} \tag{3.13}$$

Finally, based on [19], DDPM further simplifies the loss by omitting the weight. Then we can get the simplifying form of loss L_{VLB},

$$L^{\text{simple}} = \mathbb{E}_{x_0, \bar{\epsilon}_t}\left[\left\|\bar{\epsilon}_t-\epsilon_\theta\left(\sqrt{\bar{\alpha}_t} x_0+\sqrt{1-\bar{\alpha}_t} \bar{\epsilon}_t, t\right)\right\|^2\right]+C, \tag{3.14}$$

where C is a constant. So in fact the core of the training is to minimize the MSE between the Gaussian noise $\bar{\epsilon}_t$ and ϵ_θ.

3.3.4. Latent Diffusion Model (LDM)

As mentioned before, it can be observed that DDPM suffers from high computational cost issues. Generally speaking, the reverse process of DDPM requires thousands of sampling processes. The trained neural network is used to predict noise in each single process, which leads to high memory consumption and long computation times. To solve this, [43] proposed to train an Encoder-decoder (in the form of an Autoencoder) using methods like VAE or GAN. Subsequently, the Diffusion Model can be modeled in the latent space of this generator.

For a LDM, initially, an autoencoder model containing an encoder, denoted as $\mathscr{E}$, and a decoder, denoted as $\mathscr{D}$, is trained. This enables image compression via the encoder. Subsequently, diffusion operations are performed in the latent representation space, and the original pixel space is recovered using the decoder. This methodology is referred to as Perceptual Compression.

Introducing Perceptual Compression means processing the original image through autoencoder models like VAE. It ignores the high-frequency information in the image and retains only essential, foundational features. As mentioned in [43], the advantage of this approach is that it greatly reduces the computational complexity during the training and sampling stages. This allows tasks like text-to-image generation to produce images on consumer-grade GPUs within a time frame of about 10 seconds, significantly lowering the barriers to practical application. Therefore, such an approach that compresses high-dimensional features to a lower dimension and conducts operations in this low-dimensional space possesses universality. It can be easily generalized to various domains, including text, audio, and video.

Executing diffusion operations in the latent representation space largely mirrors the standard diffusion model. However, a significant distinction lies in introducing Conditioning Mechanisms for the diffusion operation. By employing a cross-attention scheme, multi-modal training is realized, enabling the implementation of conditional image generation tasks.

As for the training objective function, in our previous work, we obtained the objective function for the DDPM model as (3.14). It can also be represented in the following form,

$$L_{DDPM} = \mathbb{E}_{x,\epsilon\sim\mathcal{N}(0,1),t}\left[\|\epsilon - \epsilon_\theta\left(x_t, t\right)\|_2^2\right], \tag{3.15}$$

where t is uniformly sampled from $\{1,\ldots,T\}$.

In the LDM, we introduced a pre-trained perceptual compression model, which consists of an encoder $\mathcal{E}$ and a decoder $\mathcal{D}$. This allows us to utilize the encoder to obtain z_t during training, enabling the model to learn in the latent representation space. The corresponding objective function can be written as

$$L_{LDM} = \mathbb{E}_{\mathcal{E}(x),\epsilon\sim\mathcal{N}(0,1),t}\left[\|\epsilon - \epsilon_\theta\left(z_t, t\right)\|_2^2\right] \tag{3.16}$$

In summary, when compared to the DDPM, the LDM offers significant improvements. However, its advantages are more pronounced in more complex cases. As the datasets become increasingly complex and pixel-dense, the LDM becomes more advantageous, providing an efficient model that capitalizes on its latent representation space to enhance the diffusion process. Regarding our ongoing experiments, the current scales and complexities of the implied volatility surfaces do not necessarily need a reduction into the latent space. Looking ahead, if there's a pivot towards applying the DDPM model to larger, potentially more complex surfaces, adopting the LDM would be a strategically sound choice.

3.4. VARIATIONAL AUTOENCODER (VAE)

3.4.1. AUTOENCODER (AE)

Before understanding VAE, it is essential to first get acquainted with the Autoencoder (AE). An autoencoder is a neural network that learns to encode input data into a compressed representation and then directly decode it back to its original form.

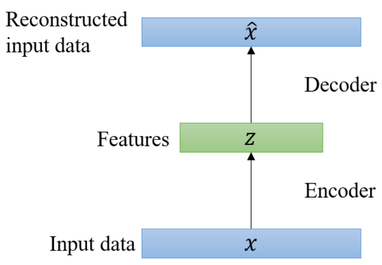


Figure 3.5: The architecture of AE.

As illustrated in 3.5, the AE, through self-supervised training, is capable of taking the raw input feature x and, after encoding via an encoder, obtaining the latent feature representation z. This process achieves automated feature engineering and accomplishes both dimensionality reduction and generalization. Subsequently, by decoding z, we can reconstruct the output $\hat{x}$. An optimal state for an AE is when the decoder's output can perfectly or nearly restore the original input, i.e., $\hat{x} \approx x$.

For this purpose, the loss function required to train an AE is

$$\text{loss} = \|x - \hat{x}\|.$$

The core of AE is on reconstruction. So the decoding result, based on the training objective, will be identical to the input if the loss is sufficiently small. From this perspective, the value of decoding has no practical significance, except to add error to smoothen some initial zeros or have slight utility.

It is evident that the entire process from input to output in AE is based on the mapping of existing training data. Even though the dimension of the hidden layer is often much smaller than the input layer, the probability distribution of the hidden layer still depends only on the distribution of training data. This leads to a discontinuous distribution in the hidden state space. It only sparsely records the one-to-one correspondence between the input samples and generated images. Therefore, if we randomly generate states in the hidden layer, its decoding will likely no longer retain the characteristics of input features. As a result, using the decoder to generate data is quite challenging.

3.4.2. ARCHITECTURE OF VAE

Compared to the AE, the VAE introduces a probabilistic interpretation of the latent space representation of the input. Instead of mapping an input sample directly to a discrete latent value, the VAE models the latent variables as a probabilistic distribution, $p(z|x)$, the conditional distribution of latent variables given the input data. In practice, this distribution is often assumed to be Gaussian, with the encoder network producing the mean μ and variance σ^2. Instead of using a fixed point in the latent space, the VAE samples a feature from this Gaussian distribution defined by μ and σ^2. The decoder then utilizes this sampled feature to try to reconstruct the output.

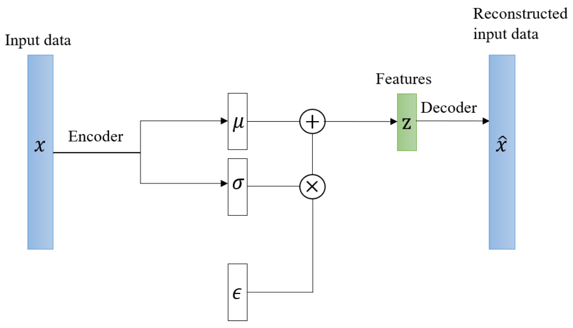


Figure 3.6: The architecture of VAE.

Referring to 3.6, we can observe that the VAE process consists of three stages. First, the input data is processed by the encoder to produce μ and σ. Next, features are sampled, and finally, the features are transformed by the decoder to get the reconstructed data. It is noteworthy that the second step not only requires μ and σ but also a random variable ϵ which follows the standard Gaussian distribution. This implementation detail is referred to as the *reparameterization trick*.

To sample z from $p(z|x)$, although we recognize that $p(z|x)$ is a Gaussian distribution, both the mean and variance of it are determined by the model. The challenge is that we need to reverse this sampling process, which is non-differentiable, to optimize the model that determines these values. However, the outcome of the sampling process is differentiable. This leads to using a known fact: sampling a Z from $N(\mu,\sigma^2)$ is equivalent to sampling an ϵ from $N(0,1)$ and then calculating $z=\mu+\sigma\cdot\epsilon$.

By this transformation, sampling from $N(\mu,\sigma^2)$ is effectively replaced with sampling from $N(0,1)$, followed by a parameter transformation to obtain the result of the original sampling process. As a consequence, the sampling operation no longer participates in gradient descent. Instead, the result of sampling is involved, rendering the entire model trainable.

Beyond the loss of AE, VAE adds a regularization term, the KL divergence between the encoding inference distribution and a standard Gaussian distribution. The reason for adding this regularization term is to prevent the model from degrading into a regular AE. During the neural network training process, to minimize reconstruction error, the variance is inevitably reduced to zero. This means there would be no random sampling noise, and the VAE would gradually turn into a standard AE.

Therefore, the loss function required to train a VAE is

$$\text{loss} = \|x-\hat{x}\| + \text{KL}(N(\mu,\sigma), N(0,1)).$$

In summary, for an input x, the VAE generates a latent probability distribution $p(z|x)$. It then randomly samples from this distribution, creating a continuous and complete latent space, which addresses the problem in AE where it can not be used for generation.

3.4.3. TRAINING

The above subsection only provides an intuitive explanation of VAE. Next, we will delve deeper into VAE from the perspective of Bayesian probability theory.

Firstly, assume that we have a batch of data samples $X=\{x_1,x_2,\dots,x_n\}$, which are independent and identically distributed (i.i.d.). Our goal is to generate more data that also follows this distribution, so our task is to estimate the distribution of these sample data. But for a set of arbitrary data, we might not know the parametric form of its distribution, making a direct estimation almost impossible. We denote the data distribution as $x_i \sim p(x;\theta)$ or $p_\theta(x)$, where θ represents the parameters we aim to estimate.

Naturally, we can use maximum likelihood estimation to solve for it, expressing the likelihood function as

$$p(x;\theta) \approx p(X;\theta) = \prod_{i=1}^{n} p(x_i;\theta).$$

In practice, we often minimize the negative log-likelihood as

$$\theta^* = \arg\min_{\theta} - \sum_{i=1}^{n} \log p(x_i;\theta).$$

However, the parameter θ here can be quite complex. We consider it as also following a new distribution $\theta \sim p(\theta;\phi)$. Here, we use the marginal likelihood, which is an integration over the parameter θ:

$$p(x;\phi) = \int_{\theta} p(x;\theta)p(\theta;\phi)d\theta.$$

So, when estimating the data distribution model, we first turn to estimate $p(x;\phi)$, then consider the model parameters θ.

Directly solving the above formulas is intractable. Thus, we introduce a new variable to assist in the solution, referring to it as the latent variable model. Leveraging the Bayesian formula, we get:

$$p(x;\phi) = \int p(x,z;\phi)\,dz = \int p(x|z;\phi)p(z)\,dz.$$

This breaks down the generation process into two steps, firstly sampling a z from the prior distribution $p(z)$, and then generating a sample through the conditional probability $p(x|z;\phi)$.

Now if we Look back at 3.6, $p(z)$ is easy to understand — it is the prior distribution for the features, and $p(x|z;\phi)$ represents the effect of the decoder. We use $q(z|x;\theta)$ to depict the effect of the encoder. The final generated result is denoted as $p(x)$.

Next, we can express the following two equations

$$q_\theta(x,z) = q(z|x)p(x);$$
$$p_\theta(x,z) = p(x|z)p(z).$$

If the above two equations could be equal, we could then use $q(x,z)$ to approximate $p(x,z)$ and calculate $p(x)$. Here we introduce the KL divergence to quantify the similarity between those two probability distributions,

$$\begin{aligned}
KL(q(x,z)||p(x,z)) &= \iint q(x,z)\log\frac{q(x,z)}{p(x,z)}\,dz\,dx \\
&= \iint q(z|x)p(x)\log\frac{q(z|x)p(x)}{p(x|z)p(z)}\,dz\,dx \\
&= \int p(x)\left[\int q(z|x)\log\frac{q(z|x)p(x)}{p(x|z)p(z)}\,dz\right]dx \\
&= \mathbb{E}_{p(x)}\left[\int q(z|x)\left(\log p(x) + \log\frac{q(z|x)}{p(x|z)p(z)}\right)dz\right] \\
&= \mathbb{E}_{p(x)}\left[\log p(x)\int q(z|x)\,dz\right] + \mathbb{E}_{p(x)}\left[\int q(z|x)\log\frac{q(z|x)}{p(x|z)p(z)}\,dz\right] \\
&= \mathbb{E}_{p(x)}[\log p(x)] + \mathbb{E}_{p(x)}[KL(q(z|x)||p(z)) - \mathbb{E}_{q(z|x)}[\log p(x|z)]].
\end{aligned} \tag{3.17}$$

Rearranging the terms, we then get

$$\mathbb{E}_{p(x)}[\log p(x)] - KL(q(x,z)||p(x,z)) = \mathbb{E}_{p(x)}[\mathbb{E}_{q(z|x)}[\log p(x|z)] - KL(q(z|x)||p(z))].$$

The first term on the left side is to maximize the likelihood, and the second term is also to be maximized (minimizing KL divergence taken as negative). Therefore, maximizing the left side is equivalent to maximizing the right side, and thus our optimization objective becomes

$$\max \mathbb{E}_{p(x)}[\mathbb{E}_{q(z|x)}[\log p(x|z)] - KL(q(z|x)||p(z))]$$

Here, we got the final objective function of VAE, also known as Evidence Lower Bound (ELBO),

$$ELBO := \mathbb{E}_{q(z|x)}[\log p(x|z)] - KL(q(z|x)||p(z)) \tag{3.18}$$

4 Methodology

After developing a thorough understanding of the underlying theoretical knowledge of generative models, our focus will turn to how to apply these models to the implied volatility surface. In Section 4.1, we will explore why generative models can be applied to the implied volatility surface; Section 4.2 then elaborate on The setting of the training process; Section 4.3 will delve into the specific application scenarios of the model; and in Section 4.4, we will introduce the indicators used to evaluate the performance of the model.

4.1. Generative Models on Implied Volatility Surfaces

Before we focus on the application of generative models on implied volatility surfaces, it is essential to clarify why image-based generative models are applicable to these surfaces. Here are three primary reasons.

Structural Similarity While implied volatility surfaces might not be as complex as standard images, they both have a two-dimensional framework, and each two-dimensional point contains specific information. For images, each point represents its pixel value. In contrast, for implied volatility surfaces, each point signifies the implied volatility at a given expiration date and strike price. This structural alignment allows image-based generative models to be directly employed in the creation and modification of implied volatility surfaces.

Features Learning and Generation Image generative models inherently try to learn the internal distribution of an image or its latent features, such as object shapes or color patterns. Similarly, implied volatility surfaces also contain features like the implied volatility smile. These features can be learned by the generative model to generate synthetic surfaces based on historical surfaces, ensuring the resulting surfaces have financial relevance. For instance, when a generative model is applied to images, it might learn the attributes of the digit ’1’ by training on numerous images containing this number, subsequently producing a new image of the digit '1'. When applied to implied volatility surfaces, the model can learn features of surfaces with a volatility smile, subsequently crafting a new surface with a smile.

Practicality and Direction of Application Generative models can not only be used to generate new surfaces with learned features but can also be used to repair or interpolate missing values or outliers in existing surfaces to enhance financial analysis. and decision-making accuracy. This is particularly important in aspects such as financial engineering and risk management, especially in situations where it is difficult to obtain an accurate volatility surface under certain market conditions.

4.2. Training Methodology

The general steps for training the two generative models, VAE and DDPM, are similar. They all need to take the complete implied volatility surface as input x, and after obtaining the latent variable z, calculate the loss function and perform a gradient descent step in the process of reconstructing z to x.

More specifically, the training process of VAE involves feeding the complete implied volatility surfaces as input to the encoder, represented by the latent variables. This encoded representation captures the essential features of the implied volatility. The decoder then attempts to reconstruct the original implied volatility surfaces from latent variables. During this reconstruction process, the loss function (3.18) is computed. The model's parameters are optimized via gradient descent to minimize this loss, iteratively refining the encoder and decoder to better represent and reconstruct the implied volatility surfaces, respectively. The whole process is described as Algorithm 1 below.

Algorithm 1 VAE Training Algorithm

```
1: repeat
2:     $(\mu, \sigma) \leftarrow \text{Encoder}(x)$
3:     $z \sim \mathcal{N}(\mu, \sigma)$
4:     $x_{\text{reconstructed}} \leftarrow \text{Decoder}(z)$
5:     Take gradient descent on $\Delta_\theta \left[\mathbb{E}_{q(z|x)}[\log p(x|z)] - \text{KL}(q(z|x) \parallel p(z))\right]$
6: until converged
```

For DDPM, the original implied volatility surface x is gradually added with noise (iid $\mathcal{N}(0,1)$) over 500 timesteps, until it becomes a pure noise latent variable in the final step. Subsequently, the neural network learns the inverse of this noise-adding process to reverse it. More specifically, in each of the 500 timesteps, a neural network fits the noise using the loss function (3.14) until the pure noise is restored to the original surface. The whole process is described as Algorithm 2 below.

Algorithm 2 DDPM Training Algorithm

```
1: repeat
2:     $t \sim \text{Uniform}(1, ..., T)$
3:     $\epsilon \sim \mathcal{N}(0, I)$
4:     Take gradient descent on $\Delta_\theta \left[L_{\text{simple}}\right]$
5: until converged
```

From this, we can see that, unlike VAE, DDPM requires training a neural network at each timestep of the reverse process to ensure that each ϵ_θ can fit ϵ. This means we need to train 500 neural networks throughout the entire process. So we chose to employ UNet for learning distributions principally due to its adeptness at retaining information. For the reverse process in DDPM, even the slightest degradation of information during a single step can blur the outcome. Over hundreds or thousands of steps, this could be detrimental. UNet, with its nearly lossless data transition capabilities, thus emerges as the aptest choice for the task.

4.3. APPLICATION ON IMPLIED VOLATILITY SURFACE

4.3.1. RECONSTRUCTION

We applied the reconstruction of implied volatility surfaces only to the VAE model. This is because the essence of the VAE model lies in capturing essential low-dimensional information within the surface as latent variables. Operations based on these latent variables are then carried out to achieve our subsequent objectives. Therefore, verifying whether the latent variables truly encapsulate sufficient information to represent the original surface is an indispensable step before real-world application. On the other hand, the latent variables of DDPM have the same dimensionality as the original surface, meaning they are not reduced in dimension, eliminating concerns in this regard.

The process of surface reconstruction is straightforward. Once the VAE model is fully trained, we feed the implied volatility surface from the test set into the VAE's encoder. Upon obtaining the latent variables, they are input into the decoder, resulting in a reconstructed surface that closely mirrors the original. The detailed algorithm is presented below.

Algorithm 3 Surface Reconstruction using VAE

```
1: Train VAE until convergence using the training set.
2: For a given implied volatility surface $x_{\text{test}}$:
3:     $(\mu, \sigma) \leftarrow \text{Encoder}(x_{\text{test}})$
4:     $z_{\text{test}} \sim \mathcal{N}(\mu, \sigma^2)$
5:     $\tilde{x}_{\text{test}} \leftarrow \text{Decoder}(z_{\text{test}})$
6: Output: the reconstructed surface $\tilde{x}_{\text{test}}$.
```

4.3.2. GENERATING NEW SURFACES

In this subsection, we will introduce how to use generating models to generate implied volatility surfaces which are synthetic but based on historical data. This is the most common application for generating models.

In general, once the generative model is trained, one can obtain latent variables through sampling from specific distributions. Then by transforming these variables back to the original space, newly generated data can be achieved.

For VAE, after the VAE model has been trained, new surfaces can be generated by sampling latent variables from a standard Gaussian distribution and feeding them into the VAE's decoder. This process decodes the latent variables into newly generated surfaces. The complete algorithm is shown in the following algorithm.

Algorithm 4 Surfaces Generating using VAE

```
1: Train VAE until convergence using the training set.
2: Determine the number of surfaces to generate, $N$.
3: for $i = 1$ to $N$ do
4:     Sample latent variable $z \sim \mathcal{N}(0,1)$.
5:     Generate implied volatility surface $x_{\text{generated}} \leftarrow \text{Decoder}(z)$.
6: end for
7: Output: Set of $N$ generated surfaces.
```

For DDPM, we return to the end of Section 3.3.2 to further explain the reverse process of DDPM, that is, how to sample and obtain the synthesized new surface.

During the training process, we have predicted the noise ϵ through the UNet neural network, so we can calculate the mean ϵ of the normal distribution using (3.9). Here, the variance is assumed to be independent of the noise and can be directly calculated from β. At this point, we have obtained the distribution of x_{t-1}. Given x_t, x_{t-1} can be obtained through reparameterization

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) + \beta_t \boldsymbol{\epsilon}. \tag{4.1}$$

The initial x_t here is pure Gaussian noise, which means that by inputting pure noise, a new synthetic surface can be generated step by step through the noise addition process. The algorithm is shown below.

Algorithm 5 Generating Surfaces with DDPM

```
1: Input: An already trained NN $\boldsymbol{\epsilon}_\theta$
2: Determine the number of timesteps to generate a surface, $T$.
3: Sample initial state $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.
4: for $t = T, \dots, 1$ do
5:     $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) + \beta_t \boldsymbol{\epsilon}$
6: end for
7: Output: A synthetic surface.
```

4.3.3. COMPLETION

In our research, the process of surface completion involves restoring incomplete surfaces to form a cohesive and smooth entity. To simulate surfaces with missing data, we intentionally remove parts from the corners of the implied volatility surfaces in our test set. Figure 4.1 shows a sample of an implied volatility surface with its corner parts missing. This specific absence indicates that the associated implied volatility, particularly for exercise prices near the time to maturity, is either missing or cannot be observed. This simulation approach reflects the reality that data incompleteness and absence are more prevalent as the time to maturity approaches.

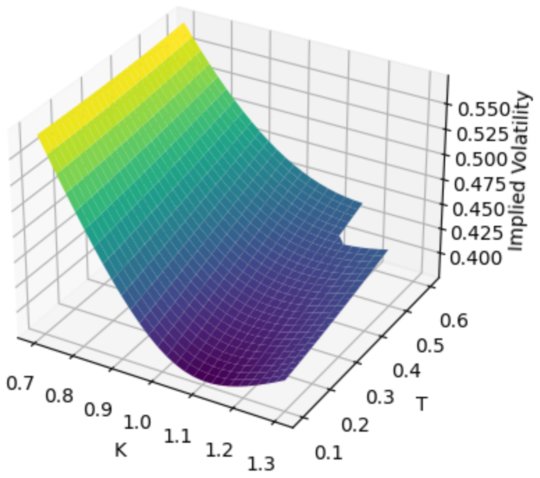


Figure 4.1: One sample incomplete implied volatility surface.

COMPLETION WITH OPTIMIZATION FOR VAE

For surface completion using VAE, we believe the most optimal scenario would be to directly employ the model trained on complete implied volatility surfaces, input an incomplete surface, and eventually obtain a fully reconstructed and completed surface. So the whole process is similar to the reconstruction process. The only difference is that in the process of surface completion, the input is the incomplete surface instead of the complete surface.

In our approach, we employed models trained on complete surfaces rather than incomplete ones. The reason behind this choice is that even though we currently simulate incomplete surfaces by removing parts from the complete ones, in real market conditions, we cannot access the completed form of these incomplete surfaces. As a result, it becomes unfeasible to compute the Mean Squared Error as the reconstruction error in the objective function.

When we employ the VAE to directly complete the surface as described in the initial approach, the outcome is less than ideal. Even though we do generate a surface that's both comprehensive and seamlessly smooth, the morphological attributes of this newly produced surface diverge significantly from the original one. This observation points towards an inherent need for an additional layer of refinement—specifically, an optimization step applied to the latent space created by the encoder. Such an optimization would be instrumental in guaranteeing that the surface regenerated from the decoder mirrors the original one as closely as feasible.

Our primary objective centers around the notion that, once the latent variable is introduced to the decoder, it should yield a surface that remains faithful to the known segments of the original one. To quantify this faithfulness, we adopt the Mean Squared Error (MSE) between the surface stemming from z and its original counterpart as our benchmark metric. This metric subsequently forms the basis upon which we calibrate and optimize z.

To achieve this optimization, we turn to the Levenberg-Marquardt (LM) algorithm, known for its prowess in parameter refinement. The LM algorithm shines particularly in scenarios where the parameter count remains on the lower side. In our context, given that our latent variables are relatively few, the LM algorithm not only offers an effective solution but also ensures computational efficiency. The methodology involves iteratively fine-tuning the latent variable values, a process driven by the difference between actual observations and the model's predictions. Through this iterative method, the LM algorithm facilitates the enhancement of latent portrayals, ensuring they aptly encapsulate the innate dynamics and interdependencies within the dataset.

Algorithm 6 Surface Incompletion with latent variable optimization using VAE

```
1: $z_{\text{origin}} \leftarrow \text{Encoder}(x_{\text{original}})$
2: Initialize $z_{\text{optimized}}$ with $z$
3: while not converged do
4:     Decode $z_{\text{optimized}}$ to obtain reconstructed surface: $\tilde{x} \leftarrow \text{Decoder}(z_{\text{optimized}})$
5:     Compute MSE between $\tilde{x}$ and $x_{\text{original}}$: $E = \text{MSE}(\tilde{x}, x_{\text{original}})$
6:     Update $z_{\text{optimized}}$ using the LM algorithm to minimize $E$
7: end while
8: return Optimized surface $\tilde{x}$
```

COMPLETION FOR DDPM

For DDPM, the method of conditioning becomes a core part of its completion strategy. When dealing with incomplete surfaces, DDPM can utilize the available information to guide the sampling process.

Instead of directly feeding the incomplete surface to the model, in the context of DDPM, this incomplete data serves as a condition or guiding constraint. The conditioning can be thought of as a way to "inform" the sampling process about which parts of the data space are known or available, thus refining the generated output to be consistent with these known portions.

For instance, suppose an incomplete implied volatility surface is missing data in a certain region. By conditioning the DDPM on this incomplete surface, the model can be guided to fill in the missing values, while being consistent with the known values. This conditioning effectively acts as a "soft constraint" that biases the sampling process towards regions of the data space that are likely to be consistent with the known values.

In general, this completion process utilizes a pre-trained unconditional DDPM as the generative prior. To regulate the generation process, the completion process only applies the reverse diffusion iterations by sampling from the missing areas of the given surface. As this method does not alter or adjust the original neural network itself, the model can generate high-quality and diverse outputs for any form of restoration. Furthermore, it is convenient since this approach does not require retraining a new DDPM model, especially considering that training DDPM consumes significant time and computational resources.

Initially, we are faced with an implied volatility surface that contains missing parts, which means it includes both x_0^{known} and x_0^{unknown}. For the known sections represented by x_0^{known}, according to the forward process (3.2) of the DDPM, one can directly introduce noise to derive the state at any given time t, symbolized as x_t^{known}.

During the completion procedure, there is a notable difference compared to the sampling process. Instead of indiscriminately introducing pure noise across the entire surface, the approach focuses on adding noise exclusively to the unknown segments. This strategy ensures that the completion is grounded in the known data while simultaneously allowing for the exploration of possible states for the unknown regions. This is mathematically represented as

$$x_{t-1} = \mathbf{m} \odot x_{t-1}^{\text{known}} + (1 - \mathbf{m}) \odot x_{t-1}^{\text{unknown}}.$$

In this equation, the parameter $\mathbf{m}$ stands for a mask whose value is 1 for the known regions and 0 for the unknown regions. The mask hence plays a pivotal role in guiding the noise addition process. Thus, $\mathbf{m} \odot x_{t-1}^{\text{known}}$ corresponds to the portion of the surface where we have known data and do not introduce noise. On the other hand, $(1 - \mathbf{m}) \odot x_{t-1}^{\text{unknown}}$ represents the section where the data is unknown, and noise is added to aid the completion process. The incorporation of noise in this targeted manner facilitates a more informed and efficient mechanism for data reconstruction, potentially leading to more accurate and reliable outcomes.

Algorithm 7 Completing IVS with DDPM

```
1: Input: An already trained NN $\boldsymbol{\epsilon}_\theta$ and incomplete IVS $\hat{\mathbf{x}}_0$.
2: Sample initial state $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.
3: for $t = T, \dots, 1$ do
4:     $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5:     $\mathbf{x}_{t-1}^{unknown} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) + \beta_t \mathbf{z}$
6:     $\mathbf{x}_{t-1}^{known} = \sqrt{\alpha_t}\hat{\mathbf{x}}_0 + (1 - \alpha_t)\boldsymbol{\epsilon}$
7:     $\mathbf{x}_{t-1} = \mathbf{m} \odot \mathbf{x}_{t-1}^{known} + (\mathbb{1} - \mathbf{m}) \odot \mathbf{x}_{t-1}^{unknown}$
8: end for
9: Output: A completed IVS $\mathbf{x}_0$.
```

4.4. EVALUATION METRIC

In the following section, we will dive deep into the evaluation criteria, providing a comprehensive overview of the metrics, methodologies, and the underlying rationale that guide our assessment.

4.4.1. TRAINING

Assessing the training process is relatively straightforward. Our evaluation is based on comparing the training time of different models and analyzing their training loss plots. These metrics offer insights into the model's complexity and the extent of its training. By combining these elements, we can measure both the efficiency and effectiveness of various models in the context of our specific objectives.

4.4.2. RECONSTRUCTION

For implied volatility surface reconstruction, the primary criterion for evaluating a model lies in the similarity between the output surface and the original surface. This similarity determines the accuracy and usability of the model. If the output surface closely mirrors the original one, it indicates that the model can effectively reconstruct the entire volatility surface. Conversely, if there is a significant deviation between the two, it might necessitate reconsideration of the model's approach or parameter choices. We chose to employ four distinct metrics: Mean Squared Error (MSE), Mean Absolute Error (MAE), Max distance, and Hausdorff Distance.

Given that surfaces can be represented as a series of points in a multi-dimensional space, the MSE serves as a fundamental measure. It quantifies the average squared deviations between the reconstructed surface's coordinates and those of the original. The formula for MSE is given by:

$$\mathrm{MSE}(X,Y) = \frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2$$

Larger discrepancies between surfaces will yield higher MSE values, making it a useful indicator of overall model accuracy. Similarly, MAE calculates the average absolute differences between predicted and true surface points, given by:

$$\mathrm{MAE}(X,Y) = \frac{1}{n}\sum_{i=1}^{n}|x_i - y_i|$$

Max distance, on the other hand, helps understand the most significant singular deviation and is crucial for surface data. It is calculated as:

$$\text{Max Distance}(X,Y) = \max_i d(x_i, y_i)$$

Lastly, the Hausdorff Distance is pivotal for surface reconstruction. It evaluates the extent to which each point on one surface is close to some point on another surface:

$$\mathrm{Hausdorff}(X,Y) = \max\left(\sup_{x\in X}\inf_{y\in Y} d(x,y), \sup_{y\in Y}\inf_{x\in X} d(x,y)\right)$$

4.4.3. COMPLETION

For the completion of implied volatility surfaces, we also focus on ensuring the similarity between the completed and original surfaces. This ensures the model can effectively fill in data gaps. The metrics used for completion are the same as those used for reconstruction, as they provide a comprehensive evaluation of the model's performance in terms of accuracy, typical deviations, largest discrepancies, and topological consistency.

4.4.4. Generation

When it comes to generating new surfaces, our evaluation criteria become somewhat hard to define. The quality of generated surfaces is challenging to define using conventional benchmarks.

The FID (Frechet Inception Distance) score stands out as a gold standard for generic image generation, predominantly within the realms of GANs and DDPM frameworks. This score quantifies both the variation and sharpness of the images produced. Its methodology involves a combination of the statistics of features extracted from authentic versus synthetic images. In mathematical terms, the FID score computes the Frechet distance between two multivariate Gaussians that are fitted to feature vectors mined from the Inception network, comparing real and produced images as

$$\mathrm{FID}(x,g) = |\mu_x - \mu_g|_2^2 + \mathrm{Tr}(\Sigma_x + \Sigma_g - 2(\Sigma_x \Sigma_g)^{0.5}),$$

where the pairs (μ_x, Σ_x) and (μ_g, Σ_g) represent the mean and covariance of the feature vectors for the authentic and synthetic images, respectively.

However, this metric is not suitable for our cases. One obvious problem is the inherent design of Inception networks for 3-channel images (usually RGB). However, our implied volatility surface has only one channel and cannot be input into the Inception network, so the traditional FID score is not applicable.

To synthesize IV surfaces that are synthetic but based on historical patterns, we introduce a metric to evaluate the precision of our methodology named Wasserstein distance. Wasserstein distance is used to express the similarity of two distributions. It measures the minimum average distance required to move data from one distribution to another distribution. Suppose G symbolizes the probabilistic distribution of implied volatility surfaces drawn from our model, while F embodies the original distribution. The formal expression of Wasserstein distance is as follows,

$$W(G,F) = \inf_{\gamma \sim \prod(G,F)} \mathbb{E}_{x,y\sim\gamma}[\|x-y\|]$$

Among them, $\prod(G,F)$ represents the set of all possible joint distributions combining the distributions G and F. For each possible joint distribution γ, you can sample $(x,y) \sim \gamma$ to get a sample x and y and calculate the pair The distance of the sample $\|x-y\|$. Therefore, the expected value of the sample distance $\mathbb{E}_{x,y\sim\gamma}[\|x-y\|]$ under the joint distribution γ can be calculated. The lower bound that can be obtained on this expected value among all possible joint distributions is the Wasserstein distance.

Given the suitability of the Wasserstein distance in the domain of probability distribution spaces, we have opted for the 1-Wasserstein distance to ascertain the discrepancy between our model's output and the genuine distribution. This metric can be approximated by analyzing the 1-Wasserstein distance amid the multivariate distribution of IVs at 784 grid points, comparing the Heston dataset against our model's generation. We label this as the Wasserstein metric.

5

NUMERICAL RESULTS ANALYSIS AND DISCUSSION

5.1. TRAINING

In this section, we delve into the training, which is a key stage in the construction of an implied volatility surface. The process is exhaustive, from parameter selection to surface generation and training dynamics. We begin by defining the parameters of the Heston model.

SETTINGS FOR HESTON MODEL

At the outset of constructing the implied volatility surface, the initial setup of crucial parameters becomes our primary point. In our research, we employ a 28×28 implied volatility surface. This implies that both T and K have 28 distinct values, culminating in a complete surface with 784 points. The parameters we set are depicted in Table 5.1 below.

Parameter	**Setting**
S_0	1
K	[0.7, 1.3]
T	[0.1, 0.6]
r	0
ρ	[-0.9, -0.1]
$\bar{v}$	[0.1, 0.3]
κ	[1.0, 2.0]
γ	[0.1, 0.9]

Table 5.1: Main Parameter Settings for the Implied Volatility Surface

Transitioning from parameter selection, our spotlight narrows down to the parameter γ. It is particularly noteworthy due to its robust association with the curvature variances observed in the implied volatility curve. As illustrated in Figure 5.1, an increase in the value of γ manifests a more pronounced implied volatility smile. Consequently, a higher volatility parameter increases the curvature of the implied volatility. In our research, we aim for the implied volatility surfaces used for training to exhibit a significant range in curvature. This would help in showcasing both the implied volatility smile and the implied volatility skew. Consequently, for γ, we have chosen a range from 0.1 to 0.9.

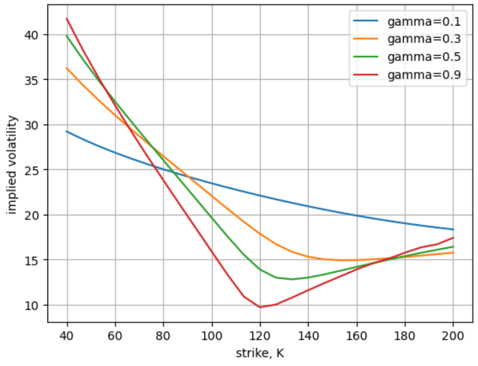


Figure 5.1: Impact of variation of the Heston vol-vol parameter γ.

To create a detailed implied volatility surface, we apply Latin hypercube sampling to get a variety of parameter combinations. This makes sure our implied volatility surfaces represent a wide range of scenarios. Then for the option pricing calculation, we use the COS method and the Heston model parameters from our sampling, then calculate the prices for European options for each parameter combination. At last, we need to reverse the calculation to calculate implied volatility. We start with an initial implied volatility, like 0.2. Then, we calculate the implied volatility in steps using the Black-Scholes formula and the market's option prices. After each step, we check if the error is small enough, like below 0.0001, to continue.

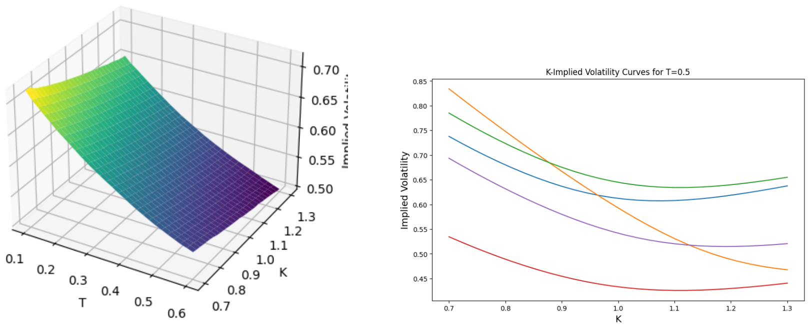


Figure 5.2: A sample of implied volatility surface generated by COS method (left) and some samples of implied volatility curves (right).

Ultimately, we generated 10,000 implied volatility surfaces for subsequent training data. As illustrated in Figure 5.2, the left panel shows a sample implied volatility surface generated using the COS method and adhering to the Heston model, capturing the characteristics of implied volatility surfaces observed in the market. The right panel presents several $K-\sigma$ slices at $T=0.5$, where σ represents the implied volatility. It demonstrates that the diversity in our generated implied volatility curves, with some exhibiting implied volatility smiles and others displaying skews.

Having delineated the processes and methodologies related to implied volatility surfaces, we now pivot our focus to the settings of the VAE and DDPM architectures.

TRAINING SETTINGS FOR VAE

In the realm of VAE, the onus of training predominantly rests on a singular neural network architecture, subject to repetitive cycles. In our research, to further compare the effects of different layers and different dimensions of the latent space on the experimental results, we used different architectures shown in Table 5.2. Considering that the Heston model is determined by five parameters, our choice for the dimensionality of the latent space in VAE was influenced by multiples of five. This decision was made to ensure a more intuitive alignment between the model parameters and the latent representations.

Architecture	**Encoder**	**Bottleneck**	**Decoder**
FC_dim10	3-layer fully connected (FC)	dim = 10	3-layer fc
FC_dim15	3-layer fc	dim = 15	3-layer fc
CNN_dim10	3-layer CNN + 1-layer fc	dim = 10	1-layer fc + 3-layer CNN
CNN_dim15	3-layer CNN + 1-layer fc	dim = 15	1-layer fc + 3-layer CNN

Table 5.2: Different architectures for VAE used in the research

As for the optimizer and learning rate in the training process, to refine our model, the Adam optimizer is deployed with a learning rate set at 5e-4. We choose the Adam optimizer because it is suitable for models with many parameters and it can automatically adjust the learning rate. The choice of the initial value of the learning rate could strike a balance between rapid convergence and stability, ensuring the optimal performance of the model throughout its training phase.

Furthermore, in the loss function of VAE, we need to balance the reconstruction error and the KL divergence. The reconstruction error ensures the VAE's decoder reconstructs the original data accurately, while the KL divergence encourages the encoder's distributions to approximate a prior distribution, a standard normal distribution. To explore the impact of the balance between these components, we experimented with different values of the weight parameter 'β'. Specifically, we employed 'β values of 0.05, 0.005, and 0.0005'. A smaller 'β' value emphasizes the reconstruction error more, while a larger value leans towards prioritizing the KL divergence.

TRAINING SETTINGS FOR DDPM

For DDPM, the UNet model is initialized with an input channel and output channel set for single-channel data processing. The core architecture comprises layers that successively deepen in nodes, starting with a foundational channel number of 96 nodes. It then doubles to 192 nodes, and maintains this depth for subsequent layers, handling the data's inherent complexities. The timestep is configured to 500, implying that evolving from pure noise to the final generated surface needs 500 sampling steps. The detail is shown in Table 5.3. One thing that needs to be mentioned is that the latent space in the table is not pure noise in the DDPM model. Other settings like the learning rate and optimization function are the same as VAE.

Architecture	**Encoder**	**Bottleneck**	**Decoder**
UNet	3-layer CNN	shape = (7×7)	3-layer CNN

Table 5.3: The architectures used for DDPM in the research

In addition to the above, the DDPM model also involves unique parameter settings. During the noise-adding and denoising processes in DDPM, we also need to provide a series of hyperparameters for the variances of Gaussian distributions, denoted as $\{\beta_t \in (0,1)\}$. The value of β should increase with the increase of t. Common approaches include linear interpolation, cosine interpolation, etc. Here, we choose the first method, linearly interpolating from 0.0001 to 0.02.

Training duration analysis

The computational efficiency and speed of training play crucial roles, especially when dealing with intricate architectures and large datasets. The following table details the training duration for each of the architectures we employed in our research.

Model Architecture	**Training Duration (hours)**
VAE (FC_dim10)	0.12
VAE (FC_dim15)	0.15
VAE (CNN_dim10)	2.96
VAE (CNN_dim15)	3.21
DDPM	9.3

Table 5.4: Training duration for various architectures. We used the DelftBlue supercomputer to train the DDPM model [44].

Looking at the training times, it is clear that different architectures have different demands of training. The VAE models using fully connected layers, like FC_dim10 and FC_dim15, trained faster, taking only 0.12 and 0.15 hours respectively. This is much quicker than the VAE models with convolutional layers. The jump in training time from FC_dim10 to FC_dim15 shows that adding more to the latent space can make training take longer. The DDPM model took the longest at 9.3 hours, showing that its design is complex and needs more time to train properly. These times remind us that choosing a model is often a balance: some models might give better results but take longer to train.

5.2. VAE

Reconstruction

As we discussed before, only in the context of the VAE, the ability to faithfully reconstruct the original input from the encoded latent space is essential. This is especially significant since the latent space needs to include all the necessary information, to ensure the model's efficacySo in this section, we aim to determine the best choice for the β value in the VAE's loss function with respect to reconstruction quality and the optimal architecture. By employing a set of metrics—MSE, MAE, Maximum Distance, and Hausdorff Distance—we can systematically evaluate and select the conditions under which the VAE offers the most precise reconstruction of the implied volatility surface.

Following an assessment, we ascertained the most appropriate architectural choices and β values for the reconstruction quality of the implied volatility surface. The detailed comparison can be found in the appendices. After running numerous trials and scrutinizing the outcomes under the lens of the chosen metrics—MSE, MAE, Maximum Distance, and Hausdorff Distance—we converged on an optimal configuration for the VAE. The CNN architecture with 10 dimensions of latent space surfaced as the superior choice. Within this framework, the $beta$ value in the loss function that produced the most accurate reconstructions was found to be 0.005. This setting effectively balanced the trade-off between the fidelity of reconstructions and the smoothness or regularity of the encoded latent space.

A sample of the implied volatility surface reconstructed using the trained VAE is depicted in Figure 5.4. The surface captures the shifts inherent to the original data. To further elucidate the exactitude of our reconstruction, we present slices of the surface along the $K-\sigma$ dimension. These slices serve to highlight specific cross-sections. By this, we can get a more detailed and focused view of how closely our reconstructed surface adheres to the original.

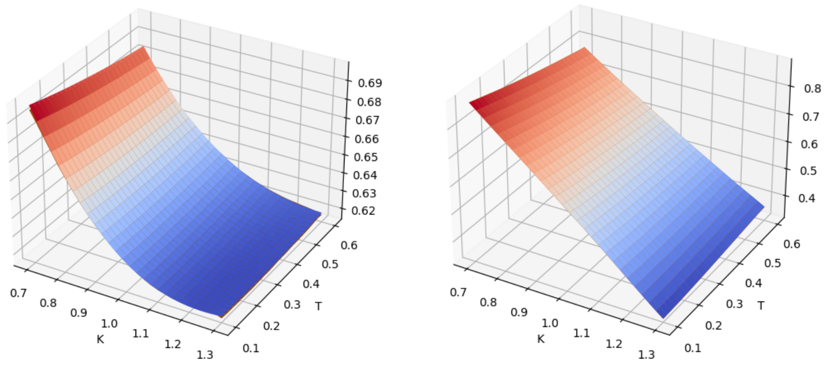


Figure 5.3: Examples of the implied volatility surfaces reconstructed by the VAE.

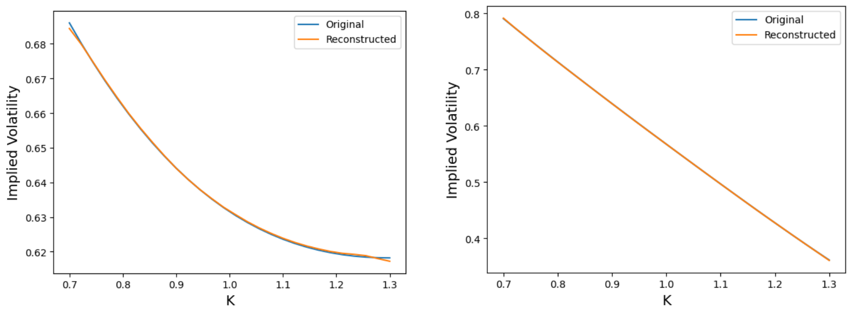


Figure 5.4: Slices of the reconstructed surface along $K-\sigma$.

The numerical metrics are tabulated as Table 5.5. The metrics show our VAE's strong ability to reconstruct. Both MSE and MAE are low, meaning the original and reconstructed surfaces are very close. The Hausdorff Distance confirms our model keeps the shape of the original data well. And even the Max Distance, which shows the biggest difference, is still pretty low. This means good quality throughout the surface.

It should be noted that the table below, as well as the tables appearing later in this chapter, are all based on the average values obtained from a random selection of 640 results.

Metric	**Value**
MSE	0.0274
MAE	0.1373
Max Distance	0.0481
Hausdorff Distance	0.3507

Table 5.5: Quantitative metrics for the reconstructed surface.

In conclusion, we have identified an optimal setup for the VAE to achieve a high-quality reconstruction of the implied volatility surface. This setup serves as a basis for future VAE-based volatility surface modeling.

GENERATION

Then we turn to see the result of using VAE to generate synthetic implied volatility surfaces. The use of VAE for generating new surfaces has been a topic of interest in prior works, so our exploration here will be succinct, focusing on essential outcomes and observations and leaving a detailed analysis for our subsequent discussions on the DDPM.

Having trained our VAE on the training dataset, we were able to generate nearly a thousand implied volatility surfaces. For a visual representation, Figure 5.5 shows one of the generated surfaces, emblematic of the quality and features exhibited by our model.

A pivotal metric in our evaluation is the 1-Wasserstein distance, a measure denoting the difference between our generated and original distributions. The VAE model achieves a 1-Wasserstein distance score of 7.57. This score is a quantitative testament to the VAE's effectiveness, indicating the model's proficiency in generating surfaces that closely resemble the original data distribution. While it indicates a small distinction, in the multifaceted landscape of financial markets, such a result is notable. It shows the VAE's efficacy in approximating the original volatility structure while introducing nuanced variations, which is the hallmark of generative models. Of course, interpreting the score in isolation can be challenging, but its relevance will become clearer when contrasted with the Wasserstein score from the DDPM model in the subsequent sections.

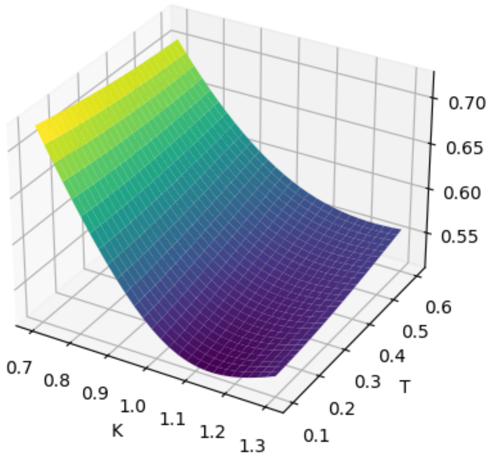


Figure 5.5: An example of an implied volatility surface generated by the VAE.

For our experiment, we procured test set surfaces and deliberately removed a central portion to simulate surfaces with missing data. This created an environment that mimics real-world scenarios where parts of surface data might be unavailable or corrupted. Once these 'incomplete' surfaces were generated, we subjected them to the VAE for completion.

Completion

VAEs are not just for reconstruction. They can also fix or complete incomplete implied volatility surfaces. Calibrating the latent space makes this repair work better. We compared the results before and after calibrating the latent space. The results are clear in the table and figure below.

Metric	**Before optimization**	**After optimization**
MSE	0.1762	1.1836e-06
MAE	0.3085	0.0008
Max Distance	0.4729	0.0051
Hausdorff Distance	0.9674	0.0305

Table 5.6: Comparison of repair metrics before and after latent space optimization.

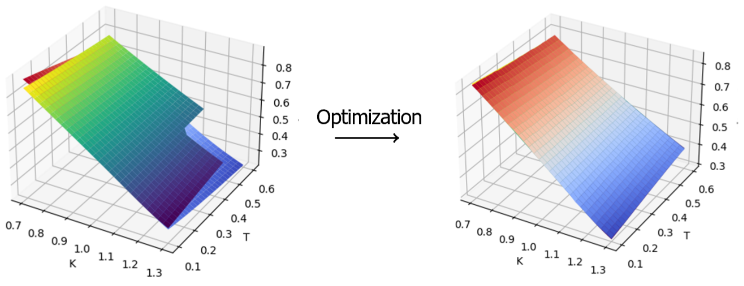


Figure 5.6: Comparison of surface completion before (left) and after (right) latent space optimization.

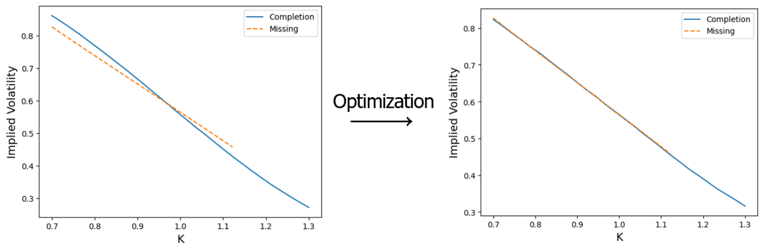


Figure 5.7: Comparison of the slice of surface completion before (left) and after (right) latent space optimization.

The metrics present compelling evidence of the advantages of optimization. The sharp decline in MSE highlights how closely the repaired surface matches the original after optimization. Similarly, the significant drop in MAE indicates fewer average differences across the entire surface. Furthermore, the changes in Max Distance and Hausdorff Distance are noteworthy. While the Max Distance provides insights into the maximum deviation between the original and repaired surfaces, the Hausdorff Distance measures shape preservation. The improvements in both these metrics underscore the VAE's enhanced ability to maintain consistency and shape fidelity post-optimization.

But beyond the numbers, the visual representation in the figure offers a tangible perspective. The completion and smoothness of the surface after optimization are evident, demonstrating the practical benefits of the process.

5.3. DDPM

GENERATION

Next, we turn to the DDPM method. In Section 3.3, we visually present a step-by-step progression of how the DDPM method is employed to generate a sample of IVs. This illustration clearly represents the transformation from an initial state to the eventual outcome. As shown in Figure 5.8, the starting point represents noise following a standard Gaussian distribution. This noise serves as the foundational layer upon which subsequent transformations are built. As we advance through the diffusion model, each timestep progressively applies a denoising procedure to this base. This denoising, extended over 500 timesteps. By the end of this process, we eventually obtain the synthetic but based-on-historical surface. In this experiment, we generated 960 synthetic surfaces by training 8000 original surfaces, which will be evaluated and analyzed in this section.

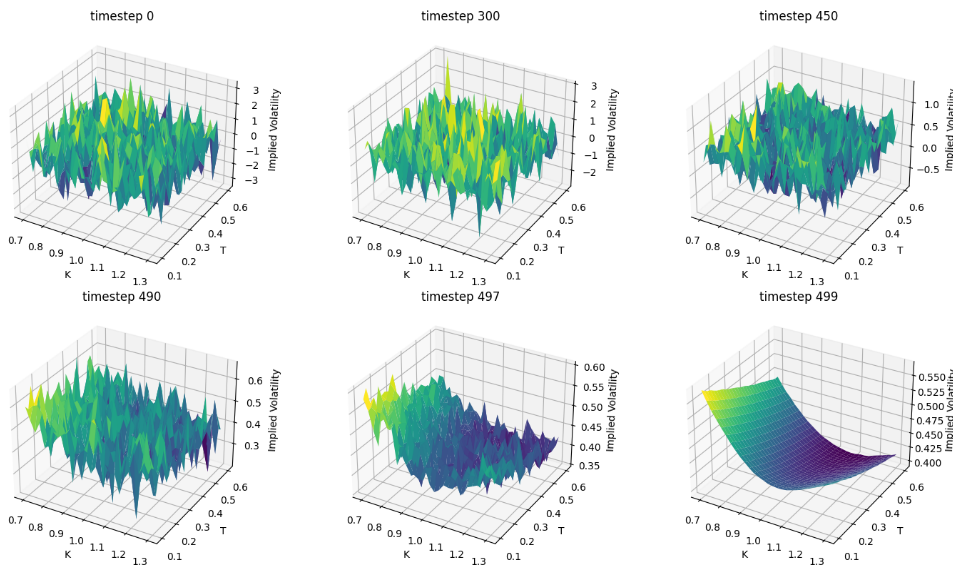


Figure 5.8: 500 Timesteps to Implied Volatility Surface.

Before we go any further with the analysis, it is noticeable that the emergence of the implied volatility surface shape is mainly concentrated in the last half timesteps. This means that the reverse process of DDPM manifests significant features in the final steps. This phenomenon can be explained to some extent by the DDPM sampling formula. As shown in Figure 5.9, with the increase of t, β_t grows linearly, reflecting the proportion of noise added during the forward diffusion at each timestep. On the other hand, the growth of $\sqrt{1-\bar{\alpha}_t}$ is nonlinear due to the cumulative multiplication effect, capturing the accumulated proportion of noise up to the timestep t. As a result, in the reverse denoising process, the earlier timesteps primarily address the removal of a smaller portion of noise. As we approach the end of the process, the timesteps begin to remove larger and more significant portions of the noise, which correspond to the main features of the original surface. This culminates in the significant emergence of the primary characteristics of the surface, notably in the final steps.

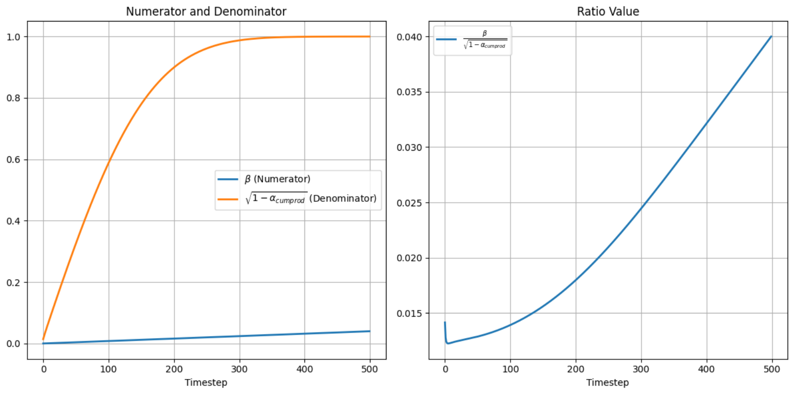


Figure 5.9: Ratio value.

Next, we will analyze the generated surfaces. An example of the generated surface is showcased in Figure 5.8. The subfigure at timestep=499 represents the implied volatility surface produced by the DDPM at its final stage. Inspecting multiple surfaces can be challenging to the naked eye, so we start by looking at cross-sectional curve plots to observe more features of the generated samples. This approach also provides insight into the behavior of options in the market. Therefore, to further assess the quality of the generated surfaces and to align with common analytical practices, we chose to examine a $K-\sigma$ slice, capturing the relationship between the implied volatility and the strike price at $T = 0.5$.

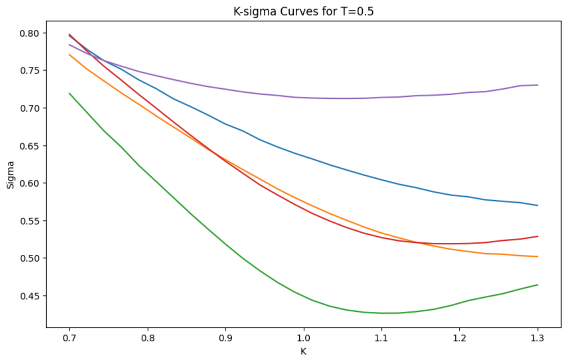


Figure 5.10: Generated curves post-training.

Visually, the generated curves display a diverse range in slope and value range. More importantly, they accurately capture the distinctive features of implied volatility, highlighting both the skew and the smile curve patterns. This underscores the ability of DDPM to grasp and reproduce the complex behaviors observed in market data and also demonstrates that the model can distinguish and emulate these pivotal attributes inside the implied volatility.

Figure 5.11 provides a visualization of the relationship between the distribution of the original implied volatility surface and the distribution synthesized through the DDPM method. We can see the similarity between the two sets of medians, which suggests that the DDPM method effectively reflects the central tendency of the original surface. Upon closer inspection, it is clear that the original surface has a slightly wider interquartile range (IQR). This suggests that the middle range of the original dataset is slightly more diverse compared to the DDPM-generated surfaces. However, the latter's more compact IQR may also indicate its efficiency in capturing the most frequent and representative patterns from the original dataset. It is worth noting that outliers are present in both datasets. Although the raw surface has a wider range, it is commendable that the DDPM method is still able to model some of these extreme cases.

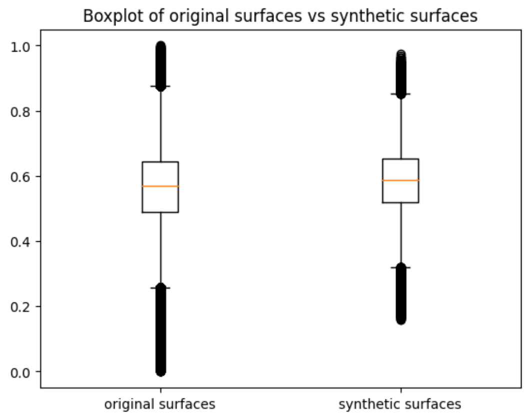


Figure 5.11: The box plot of original surfaces v.s. synthetic surfaces.

Analyzing further using the Wasserstein metric, Table 5.7 shows that the between the generated and historical surfaces progressively reduces as the denoising process continues. This reduction indicates the model's capability to converge towards the historical data distribution. Furthermore, it implies that as denoising refines the samples, they become closer in distribution to the true data. Consistent with the surface generation visualized earlier, the most significant reduction in distance happens in the last few steps. This emphasizes that the primary characteristics of the surface become evident towards the end of the denoising process.

Timestep	**1-Wasserstein Distance**
0	611.23
300	461.37
450	75.52
490	7.09
497	4.04
499	3.68

Table 5.7: 1-Wasserstein distance between generated and historical surfaces at different timesteps.

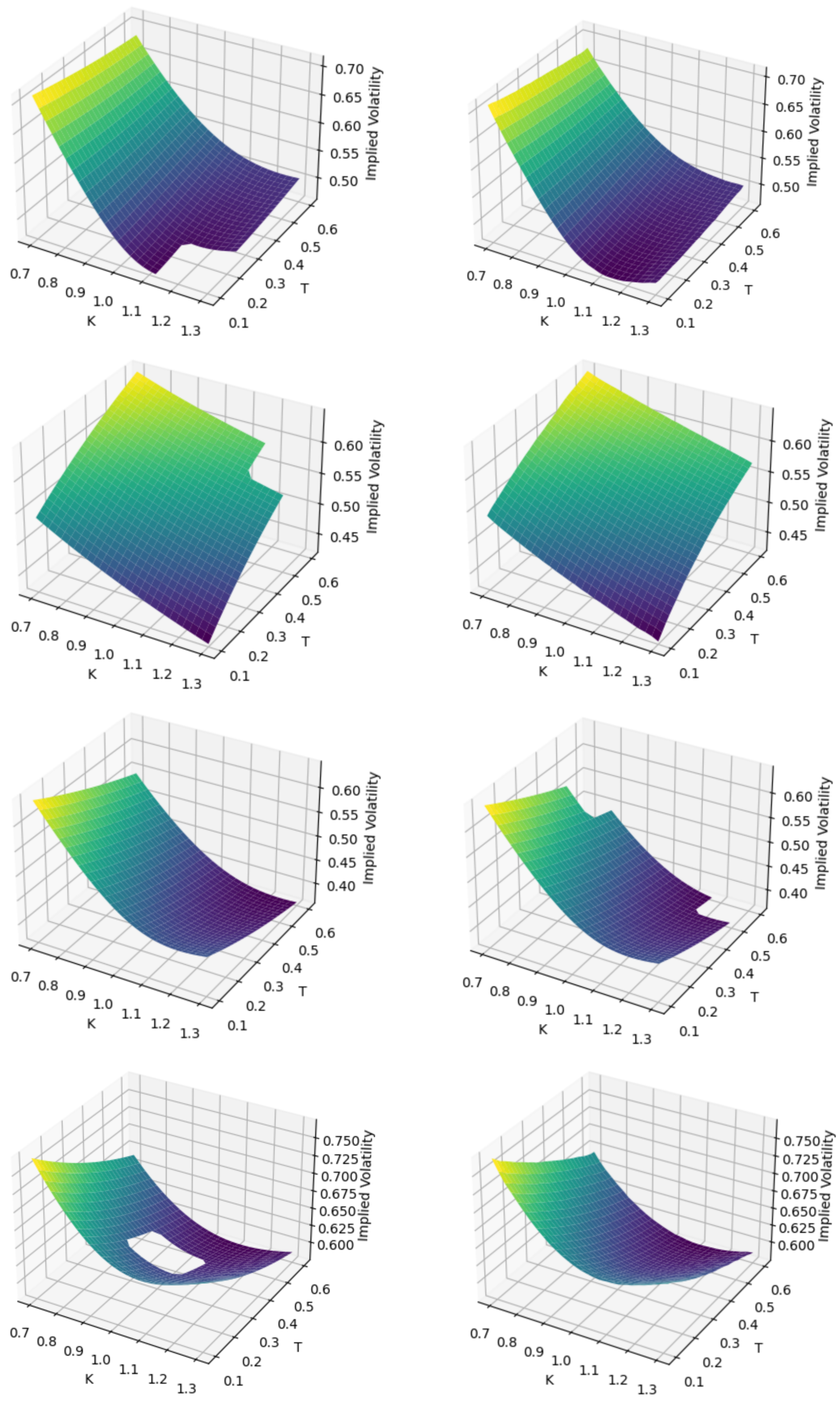


Figure 5.12: An illustration of the transition from the incomplete surfaces (left) to their completions (right).

COMPLETION

For the DDPM method, upon obtaining the completed surfaces from it, we also provide the relevant metrics and comparison images before and after completion.

From the comparison images in Figure 5.12, it is clear that the DDPM method effectively completes the surface. The restored areas integrate smoothly with the original sections, eliminating any evident discrepancies. This results in a uniform and seamless surface, demonstrating the strength of the DDPM approach. The ability to maintain the surface's integrity, even in areas that were previously incomplete, speaks volumes about its precision and reliability. The visual outcome ensures the method's robustness and its adeptness at ensuring continuity.

The provided metrics in Table 5.8 show the proficiency of the DDPM model in its predictions. With an extremely low MSE and MAE, the model demonstrates remarkable accuracy, suggesting that its predictions closely align with actual observations. Even in the worst-case scenarios, as indicated by the Max Distance, the deviation of DDPM remains minimal. While the Hausdorff Distance is relatively larger, it still underscores the capability of DDPM to produce results that are close to the actual data. Overall, these metrics collectively highlight the DDPM's efficacy and precision in its application.

Metric	**Value**
MSE	1.598×10^{-6}
MAE	0.001
Max Distance	0.004
Hausdorff Distance	0.035

Table 5.8: Numerical results for completion of DDPM.

To provide a more tangible representation of our results, we have obtained the $K - \sigma$ slices of implied volatility surfaces to highlight the ability of DDPM in the completion task. Each $K - \sigma$ slice embodies specific features commonly observed in implied volatility data, ensuring a comprehensive assessment of the adaptability and robustness of DDPM.

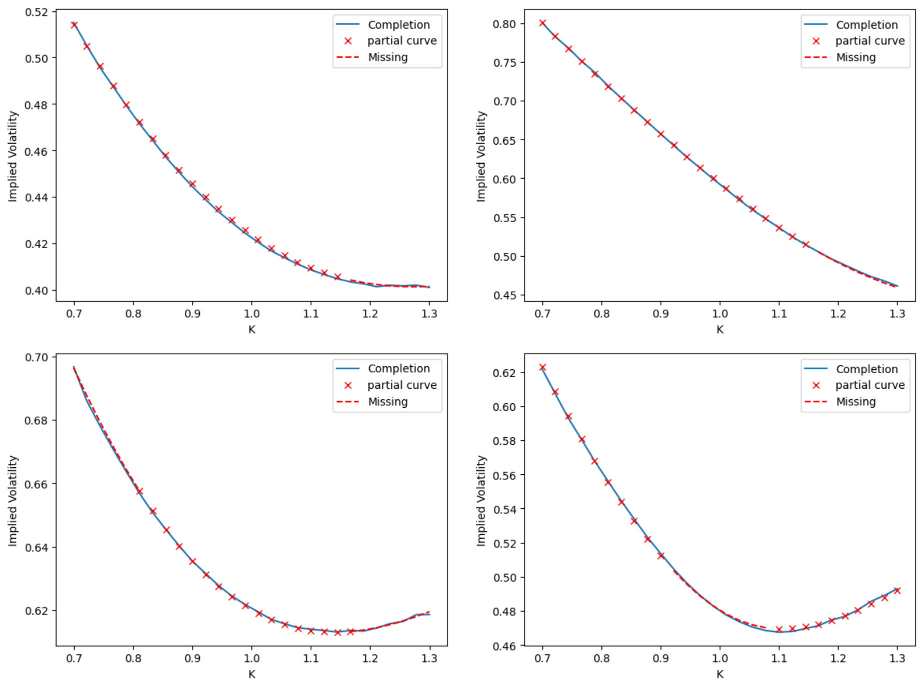


Figure 5.13: The $K-\sigma$ curves of completion of implied volatility surfaces.

All surfaces were effectively and accurately completed by the DDPM, demonstrating its capacity to understand and reproduce intricate patterns. The consistency observed across both surfaces emphasizes the capability to handle diverse data scenarios, even when faced with specific complexities.

5.4. SUMMARIZATION

As seen from our experiments, DDPM stands out for its accuracy in generating implied volatility surfaces. DDPM performs well in tasks related to surface generation with a significantly lower 1-Wasserstein distance (only 3.68), which contrasts with the higher values found in the VAE model. This low distance highlights the excellent accuracy of DDPM in modeling the original data distribution, but also its fast convergence rate. Furthermore, DDPM's lower score points to its enhanced ability to consistently produce outputs that are more aligned with historical data. It implies a higher degree of reliability and robustness, especially when dealing with complex patterns and outliers. Therefore, when solely considering the 1-Wasserstein distance, DDPM stands out as the more favorable model for generating implied volatility surfaces, offering greater precision, consistency, and adaptability compared to the VAE.

Furthermore, DDPM's capabilities are not limited to surface generation but extend to surface completion tasks. Its performance in these tasks demonstrates superior adaptability and accuracy. The inherent functionality of the model allows for impressive results without the need for additional optimization processes. The DDPM is also capable of producing satisfactory completions when the complexity of volatile surfaces is high.

The DDPM has showcased itself as a more reliable approach for generating implied volatility surfaces, given its superior accuracy and consistency. However, it's crucial to underscore that its advantages come into play when the costs associated with its deployment are feasible. On the other hand, while VAE might exhibit a slight trade-off in precision, it stands as a valuable alternative in scenarios constrained by time and budget. The decision between the two, thus, should be made based on a balance of desired accuracy and available resources.

6 Conclusions and Future Research

6.1. Conclusions

Implied volatility, which can not be straightforwardly computed, remains a pivotal concept in financial markets. There are mainly two methodologies for determining it: through mathematical model fitting and through data-driven computations. Our work leans toward the latter, aiming to apply image generating models to implied volatility surfaces in the financial domain.

We started our work from the VAE model, the subject of extensive prior research. During training, VAE extracts pivotal information from original images, compressing them into a lower-dimensional latent space. This latent representation is then decoded back to produce the original images. Our investigations on VAE revolved around three key dimensions. Firstly, we ensured that through surface reconstruction, our latent space captured a rich set of information. Secondly, when generating new surfaces, the 1-Wasserstein distance between our newly formed surface set and the training set stood at 7.57. Lastly, we employed optimization on the latent space to fill in missing surfaces, resulting in a MSE of about 10^{-6} between the completing and original surfaces.

Conversely, the DDPM employs a different philosophy. It continually adds noise to the original images until a point of pure noise is reached. The model then learns the denoising process to restore the noise-infused images to their original state during the training process. our extensive training and evaluation of the DDPM model yielded promising results. The surfaces generated under DDPM not only displayed remarkable clarity but also showcased diverse characteristics, vividly representing implied volatility smiles and skews. An impressive aspect of DDPM's capability was its 1-Wasserstein distance, which measured at a mere 3.68. More strikingly, when it came to completing surfaces, the DDPM effortlessly achieved an MSE of roughly 10^{-6} without needing extra optimization.

In comparing the VAE and DDPM models, each has its unique benefits. The VAE model only needs to train one neural network to get reconstructed surfaces. This makes VAE suitable for situations where quick results with less computational power are desired. On the other hand, DDPM stands out for its precision and quality. The crispness and diversity of surfaces generated by DDPM, as well as its ability to capture features distinctly sets it apart. In a word, while VAE had its advantages, DDPM showcased superior performance in applications.

However, it is worth noting that our current study relies on data from the Heston model only. Further research and testing are needed to assess their potential in real-world financial markets.

6.2. FUTURE WORK

For future work, it would be beneficial to incorporate financial constraints, such as no-arbitrage conditions, into the DDPM framework. This could be accomplished through conditional DDPM techniques such as the Classifier Guidance Diffusion Model in [45] or the Semantic Guidance Diffusion in [46]. First is the Classifier Guidance Diffusion Model, this method does not need to train the diffusion model additionally. We can use an external classifier to guide the generation of arbitrage-free surfaces on top of the original trained diffusion model. The only place that needs to be changed is the mean value of the Gaussian sampling in the sampling process. During the sampling process, it is expected that the closer the sampling center of the noisy surface is to the arbitrage-free condition guided by the discriminator, the better. We can also consider Semantic Guidance Diffusion, which can be implemented to guide the denoising process of the diffusion model by replacing the no-arbitrage condition with other more complex discriminators instead of classifiers. This enhancement could ensure that generated volatility surfaces are not only statistically precise but also in compliance with fundamental financial principles, potentially improving the model's fidelity to the finance domain.

In addition, applying DDPM to real-world financial market data is worth investigating. The performance of our current model in the Heston dataset is promising, but real-world financial markets present a more complex and volatile landscape. Testing the model against real-time market conditions would provide a clearer assessment of its utility and robustness.

Finally, the potential application of Latent Diffusion Models to larger and more complex datasets presents a promising area for exploration. It is particularly adept at navigating the intricacies of big datasets and could offer significant improvements in the accuracy and computational efficiency of generating implied volatility surfaces.

Bibliography

[1] S. L. Heston, “A closed-form solution for options with stochastic volatility with applications to bond and currency options,” *The Review of Financial Studies*, vol. 6, no. 2, pp. 327–343, 1993.

[2] D. Duffie, J. Pan, and K. Singleton, “Transform analysis and asset pricing for affine jump-diffusions,” *Econometrica*, vol. 68, no. 6, pp. 1343–1376, 2000.

[3] M. R. Fengler, “Arbitrage-free smoothing of the implied volatility surface,” *Quantitative Finance*, vol. 9, no. 4, pp. 417–428, 2009.

[4] V. Piterbarg, “A fast strong approximation monte carlo scheme for stochastic volatility models,” *Quantitative Finance*, vol. 6, no. 2, pp. 135–140, 2005.

[5] R. Cont and J. da Fonseca, “Dynamics of implied volatility surfaces,” *Quantitative Finance*, vol. 2, no. 1, pp. 45–60, 2002.

[6] C. Alexander and D. Korovilas, “Orthogonal methods for generating large positive semi-definite matrices,” *Working Paper*, 2001.

[7] B. Ning, S. Jaimungal, X. Zhang, and M. Bergeron, “Arbitrage-free implied volatility surface generation with variational autoencoders,” *arXiv preprint arXiv:2108.04941*, 2021.

[8] M. Bergeron, N. Fung, J. Hull, Z. Poulos, and A. Veneris, “Variational autoencoders: A hands-off approach to volatility,” *The Journal of Financial Data Science*, 2022.

[9] T. Dierckx, J. Davis, and W. Schoutens, “Towards data-driven volatility modeling with variational autoencoders,” in *Machine Learning and Principles and Practice of Knowledge Discovery in Databases: International Workshops of ECML PKDD 2022, Grenoble, France, September 19–23, 2022, Proceedings, Part II.* Springer, 2023, pp. 97–111.

[10] S. Borovkova and M. van den Oever, “Variational autoencoders with student-t distribution for large portfolios,” *Available at SSRN 4264976*, 2022.

[11] A. Na, M. Zhang, and J. Wan, “Learning volatility surfaces using generative adversarial networks,” *arXiv preprint arXiv:2304.13128*, 2023.

[12] F. Black and M. Scholes, “The pricing of options and corporate liabilities,” *Journal of Political Economy*, vol. 81, no. 3, pp. 637–654, 1973.

[13] R. C. Merton, “Theory of rational option pricing,” *The Bell Journal of Economics and Management Science*, vol. 4, no. 1, pp. 141–183, 1973.

[14] J. Gatheral, *The Volatility Surface: A Practitioner's Guide*. John Wiley & Sons, 2006.

[15] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*. Springer, 2015, pp. 234–241.

[16] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[17] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[18] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[19] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.

[20] A. van den Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves, and K. Kavukcuoglu, "Conditional image generation with pixelcnn decoders," in *Advances in Neural Information Processing Systems*, 2016, pp. 4790–4798.

[21] R. Gómez-Bombarelli, J. N. Wei, D. Duvenaud, J. M. Hernández-Lobato, B. Sánchez-Lengeling, D. Sheberla, J. Aguilera-Iparraguirre, T. D. Hirzel, R. P. Adams, and A. Aspuru-Guzik, "Automatic chemical design using a data-driven continuous representation of molecules," *ACS central science*, vol. 4, no. 2, pp. 268–276, 2018.

[22] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerritt *et al.*, "Natural tts synthesis by conditioning wavenet on mel spectrogram predictions," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 4779–4783.

[23] N. Justesen, P. Bontrager, J. Togelius, and S. Risi, "Illuminating generalization in deep reinforcement learning through procedural level generation," *arXiv preprint arXiv:1806.10729*, 2018.

[24] L. Theis, A. van den Oord, and M. Bethge, "A note on the evaluation of generative models," in *International Conference on Learning Representations*, 2015.

[25] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," *arXiv preprint arXiv:1401.4082*, 2014.

[26] C. Doersch, "Tutorial on variational autoencoders," *arXiv preprint arXiv:1606.05908*, 2016.

[27] J. An and S. Cho, “Variational autoencoder based anomaly detection using reconstruction probability,” 2015.

[28] C. P. Burgess, I. Higgins, A. Pal, L. Matthey, N. Watters, G. Desjardins, and A. Lerchner, “Understanding disentangling in β-vae,” *NeurIPS*, 2018.

[29] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling, “Semi-supervised learning with deep generative models,” *NeurIPS*, 2014.

[30] R. Gómez-Bombarelli, J. N. Wei, D. Duvenaud, J. M. Hernández-Lobato, B. Sánchez-Lengeling, D. Sheberla, J. Aguilera-Iparraguirre, T. D. Hirzel, R. P. Adams, and A. Aspuru-Guzik, “Automatic chemical design using a data-driven continuous representation of molecules,” *ACS Central Science*, 2018.

[31] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther, “Autoencoding beyond pixels using a learned similarity metric,” *arXiv preprint arXiv:1512.09300*, 2015.

[32] R. T. Q. Chen, X. Li, R. Grosse, and D. Duvenaud, “Isolating sources of disentanglement in variational autoencoders,” *NeurIPS*, 2018.

[33] J. Engel, C. Resnick, A. Roberts, S. Dieleman, M. Norouzi, D. Eck, and K. Simonyan, “Neural audio synthesis of musical notes with wavenet autoencoders,” *arXiv preprint arXiv:1704.01279*, 2017.

[34] D. Liang, R. G. Krishnan, M. D. Hoffman, and T. Jebara, “Variational autoencoders for collaborative filtering,” *The Web Conference*, 2018.

[35] W. Feller, “On the theory of stochastic processes, with particular reference to applications,” 1949.

[36] J. Sohl-Dickstein, E. A. Weiss, N. Maheswaranathan, and S. Ganguli, “Deep unsupervised learning using nonequilibrium thermodynamics,” in *International Conference on Machine Learning*. PMLR, 2015, pp. 2256–2265.

[37] J. Song, C. Meng, and S. Ermon, “Denoising diffusion implicit models,” *arXiv preprint arXiv:2010.02502*, 2020.

[38] A. Nichol *et al.*, “Glide: Towards photorealistic image generation and editing with text-guided diffusion models,” *arXiv preprint arXiv:2112.10741*, 2021.

[39] A. Ramesh *et al.*, “Hierarchical text-conditional image generation with clip latents,” *arXiv preprint arXiv:2204.06125*, 2022.

[40] C. Saharia *et al.*, “Photorealistic text-to-image diffusion models with deep language understanding,” *arXiv preprint arXiv:2205.11487*, 2022.

[41] R. Rombach *et al.*, “High-resolution image synthesis with latent diffusion models,” *arXiv preprint arXiv:2112.10752*, 2021.

[42] W. Feller, “Retracted chapter: On the theory of stochastic processes, with particular reference to applications,” in *Selected Papers I*. Springer, 2015, pp. 769–798.

[43] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, “High-resolution image synthesis with latent diffusion models,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 10 684–10 695.

[44] Delft High Performance Computing Centre (DHPC), *DelftBlue Supercomputer (Phase 1)*, 2022, https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase1.

[45] P. Dhariwal and A. Nichol, “Diffusion models beat gans on image synthesis,” *Advances in neural information processing systems*, vol. 34, pp. 8780–8794, 2021.

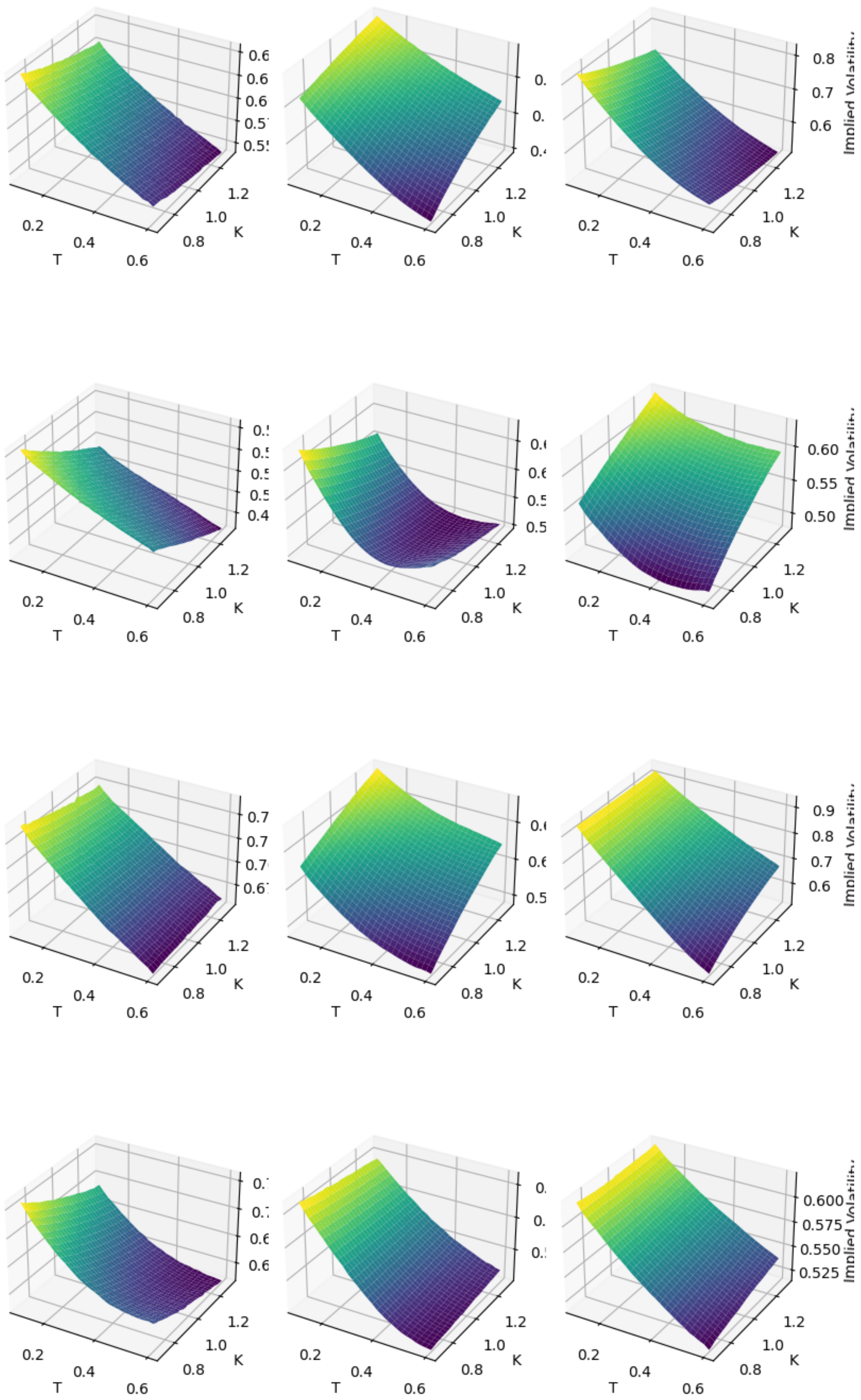
[46] X. Liu, D. H. Park, S. Azadi, G. Zhang, A. Chopikyan, Y. Hu, H. Shi, A. Rohrbach, and T. Darrell, “More control for free! image synthesis with semantic diffusion guidance,” in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2023, pp. 289–299.

7

AppendixA

DDPM generates implied volatility surfaces

We provide a collection of ten implied volatility surface examples generated by the DDPM.

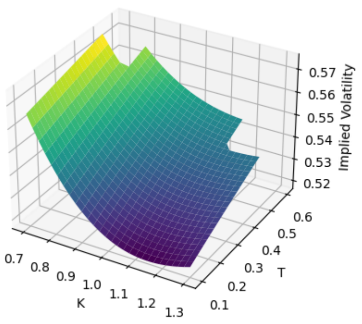

T
K
Implied Volatility
0.2
0.4
0.6
0.8
1.0
1.2

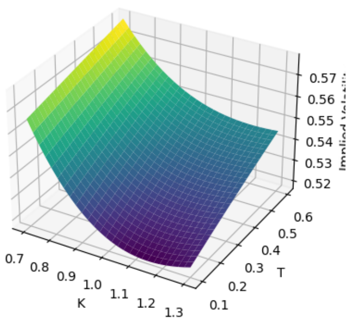
8

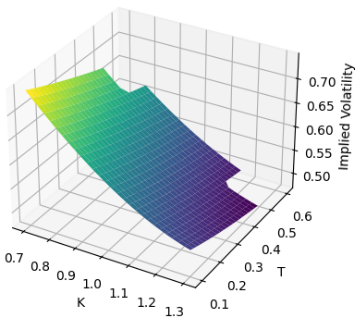
APPENDIX B

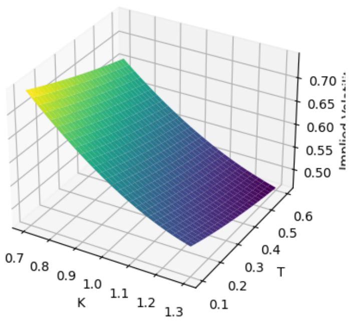
DDPM COMPLETES MISSING IMPLIED VOLATILITY SURFACES

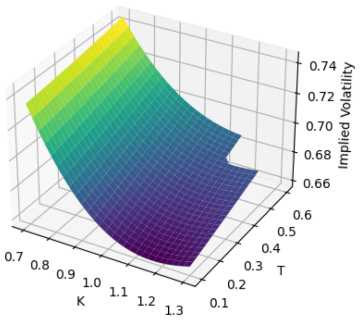
We provide a collection of eight implied volatility surfaces with missing parts completed by the DDPM.

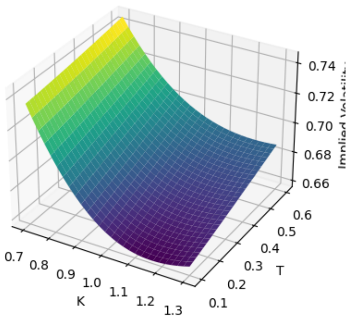

0.57
0.56
0.55
0.54
0.53
0.52
Implied Volatility
0.7 0.8 0.9 1.0 1.1 1.2 1.3
K
0.1 0.2 0.3 0.4 0.5 0.6
T

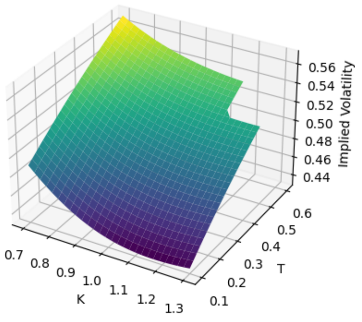

0.57
0.56
0.55
0.54
0.53
0.52
0.7 0.8 0.9 1.0 1.1 1.2 1.3
K
0.1 0.2 0.3 0.4 0.5 0.6
T

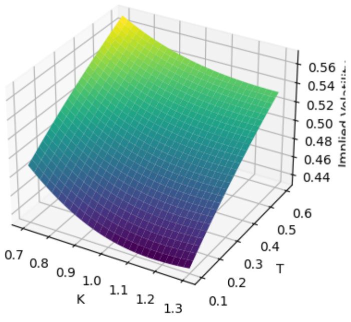

0.70
0.65
0.60
0.55
0.50
Implied Volatility
0.7 0.8 0.9 1.0 1.1 1.2 1.3
K
0.1 0.2 0.3 0.4 0.5 0.6
T

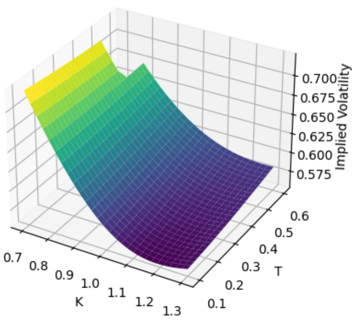

0.70
0.65
0.60
0.55
0.50
0.7 0.8 0.9 1.0 1.1 1.2 1.3
K
0.1 0.2 0.3 0.4 0.5 0.6
T

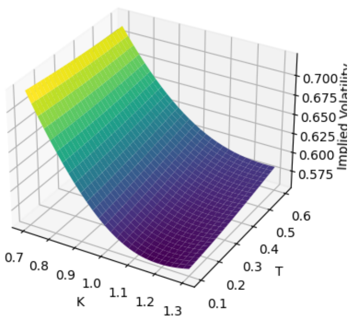

0.74
0.72
0.70
0.68
0.66
Implied Volatility
0.7 0.8 0.9 1.0 1.1 1.2 1.3
K
0.1 0.2 0.3 0.4 0.5 0.6
T

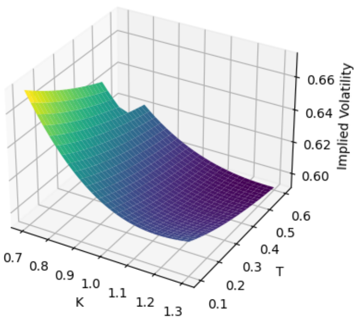

0.74
0.72
0.70
0.68
0.66
0.7 0.8 0.9 1.0 1.1 1.2 1.3
K
0.1 0.2 0.3 0.4 0.5 0.6
T

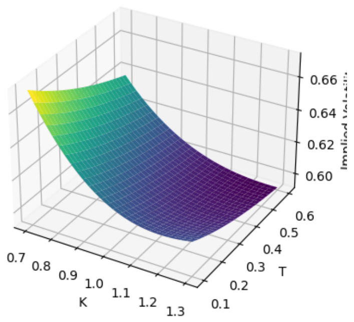

0.56
0.54
0.52
0.50
0.48
0.46
0.44
Implied Volatility
0.7 0.8 0.9 1.0 1.1 1.2 1.3
K
0.1 0.2 0.3 0.4 0.5 0.6
T

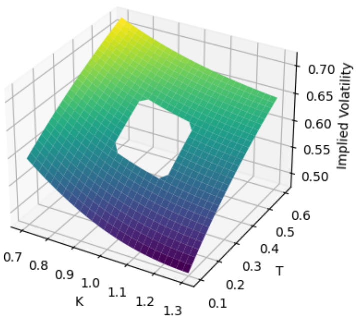

0.56
0.54
0.52
0.50
0.48
0.46
0.44
0.7 0.8 0.9 1.0 1.1 1.2 1.3
K
0.1 0.2 0.3 0.4 0.5 0.6
T

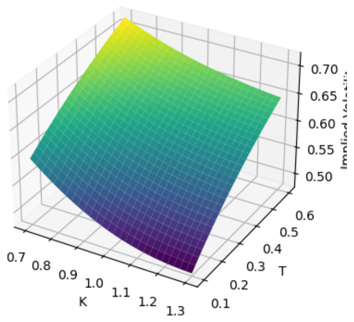

0.700
0.675
0.650
0.625
0.600
0.575
Implied Volatility
K
0.7 0.8 0.9 1.0 1.1 1.2 1.3
T
0.1 0.2 0.3 0.4 0.5 0.6

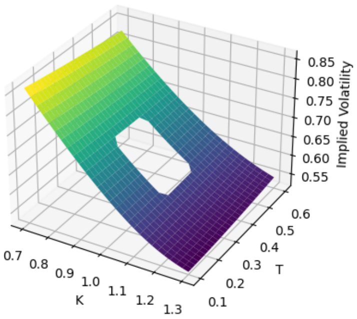

0.700
0.675
0.650
0.625
0.600
0.575
K
0.7 0.8 0.9 1.0 1.1 1.2 1.3
T
0.1 0.2 0.3 0.4 0.5 0.6

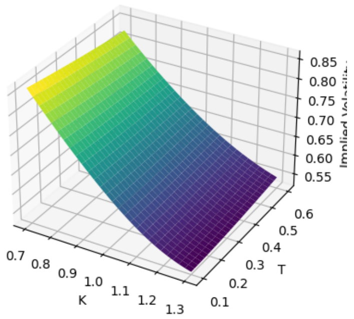

0.66
0.64
0.62
0.60
Implied Volatility
K
0.7 0.8 0.9 1.0 1.1 1.2 1.3
T
0.1 0.2 0.3 0.4 0.5 0.6


0.66
0.64
0.62
0.60
K
0.7 0.8 0.9 1.0 1.1 1.2 1.3
T
0.1 0.2 0.3 0.4 0.5 0.6


0.70
0.65
0.60
0.55
0.50
Implied Volatility
K
0.7 0.8 0.9 1.0 1.1 1.2 1.3
T
0.1 0.2 0.3 0.4 0.5 0.6


0.70
0.65
0.60
0.55
0.50
K
0.7 0.8 0.9 1.0 1.1 1.2 1.3
T
0.1 0.2 0.3 0.4 0.5 0.6


0.85
0.80
0.75
0.70
0.65
0.60
0.55
Implied Volatility
K
0.7 0.8 0.9 1.0 1.1 1.2 1.3
T
0.1 0.2 0.3 0.4 0.5 0.6


0.85
0.80
0.75
0.70
0.65
0.60
0.55
K
0.7 0.8 0.9 1.0 1.1 1.2 1.3
T
0.1 0.2 0.3 0.4 0.5 0.6