Erasmus University Rotterdam
Erasmus School of Economics
Master Thesis Quantitative Finance

Reconstructing and Completing Single Equity IVS Using Auto-Encoders

Lucas van Opstal (619219)

Supervisor:	Dr. E. Vladimirov
Second assessor:	Dr. M. Grith
Date final version:	5th February 2024

Abstract

This research explores the performance of auto-encoder models in reconstructing and completing implied volatility surfaces for single equity options, with a focus on comparing them to index options of the S&P 500. The study leverages grid-wise auto-encoder models and investigates the added value of calibrating such models. Models are trained on data from 2010 through 2018, and tested on the highly volatile years 2020 and 2021. The analysis reveals that, in terms of Mean Squared Error, none of the single equity options outperforms the S&P 500 options in reconstructing surfaces. However, one of the single equity options does outperform the index when completing the implied volatility surface. The research emphasizes the ability of auto-encoders to capture market volatility dynamics, evidenced by similarities between latent values and the VIX index. It is concluded that the general volatility level, slope, and term-structure are encoded in some of the latent dimensions. Furthermore, the results demonstrate the practical utility of these models for completing implied volatility surfaces, showcasing the importance of calibrating the latent vector in this process. Overall, this research adds to the literature understanding the potential applications and dynamics of auto-encoders in the context of implied volatility surfaces for single equity options.

Acknowledgments

This thesis concludes my academic journey spanning well over seven years. The programme Quantitative Finance has given me the basis to deploy machine learning methods, and mathematics in the world of finance and financial markets. The intersection of these three fields were the basis of the subject of this thesis. Before this thesis I had never used or modeled a Neural Network, which seemed a daunting task at first. I could not have written this research without the extensive help of Dr. Evgenii Vladimirov, which has proposed some very insightful suggestions that have helped the quality of the work tremendously. I want to thank everyone who has helped me throughout these last five months in finalizing this research.

Contents

1 Introduction

Options are complex financial instruments commonly used by financial institutions to hedge their positions. These products are listed on financial markets and are characterized by three main components: The underlying asset, the time to maturity (tenor), and the strike price. To ensure the options are priced in a way that arbitrage is not possible, Black and Scholes (1973) came up with their famous Black-Scholes Option pricing model. The only uncertain variable in their model is the volatility, and hence the option price in the market *implies* a volatility based on the BS model. Traders and market participants who are interested in comparing options with different strikes and tenors use the implied volatilities of each option to asses if they want to buy or sell this option. By plotting the implied volatility of several options against the tenor and strike price, one can obtain the Implied Volatility Surface (IVS).

The IVS is a commonly used metric for stock market participants when comparing options with several tenors and strike prices. It is, therefore, vital for market participants to accurately predict or infer the complete IVS when managing positions involving options. This is especially the case when a market participant wants to understand the right price for an option not listed in the market yet. In recent years, machine learning techniques, and in particular neural networks, have taken flight in current research on predicting, interpolating, and extrapolating the IVS (Ackerer et al., 2020; Almeida et al., 2023; Bergeron et al., 2022; Zhang et al., 2023).

The Artificial Neural Networks (ANN) were first introduced by McCulloch and Pitts (1943). By representing the neurons in a brain through mathematical relationships, the researchers have laid the basis for Neural Network (NN) machine learning applications in various fields. As described in Almeida et al. (2023), some well-known parametric models (e.g., Black-Scholes, Heston, Carr-Wu) incur prediction errors when predicting the IVS. Their research uses a non-parametric approach by leveraging the power of feedforward neural networks. They conclude that using non-parametric models to correct the pricing error of parametric models, always outperform their respective original model across various S&P 500 index options.

A special kind of NN is an auto-encoder, essentially a nonlinear alternative to Principal Component Analysis (PCA) (Kramer, 1991). These auto-encoders have been used in an asset pricing context in the research of Gu et al. (2021). Auto-encoders aim to approximate the input variables through a particular neural network. The main difference between a standard feedforward neural network and auto-encoders is that the first tries to predict a target vector y from input vector x. While auto-encoders aim to predict x from x.

Since Gu et al. (2021) have done their research in an asset pricing context, it is interesting to look at research that has been done related to the IVS of options. Examples are Bergeron et al. (2022) and, more recently , Zhang et al. (2023). In their papers, they apply variational auto-encoders (VAEs) to estimate (missing points on) the IVS. VAEs are a variation on a standard auto-encoder and were first written about by Kingma and Welling (2013). An interesting finding from Bergeron et al. (2022) is that VAEs performed quite well at interpolating the volatility surfaces in environments with limited data. Their research and the research of Zhang et al. (2023) consider that the predictions should be made so that static arbitrage is not possible. Both papers did not look at the IVS of single equity options but rather researched the IVS dynamics of forex exchange (FX) pairs and index options.

Other papers also showed that reconstructing the IVS is possible under no-arbitrage restrictions with the help of a NN. However, a fallacy of NNs is that the hidden layers are challenging to interpret economically. Gong et al. (2022) propose the PCA VAE model to represent descriptors into a latent space of three dimensions. Their model yields three different latent dimensions within the VAE that have an economic interpretation and will, therefore, aid in the decision-making of option traders. Another paper aims to add interpretability to understanding implied volatility movements by adding the VIX as a feature to test whether the behavior of the volatility surface in high volatility environments is different from that in low volatility environments (Cao et al., 2020).

What all the papers mentioned earlier have in common is that they show the power of NNs, specifically AEs and VAEs, in reconstructing or completing the IVS for certain types of assets and derivatives. However, none of these papers have looked into the performance of these models in the case of single equity options. Therefore, this thesis will focus on the performance of auto-encoders in reconstructing and completing the IVS for single equity options. This will be done by considering the interpretability of the proposed machine learning method. To do so, the methodology of Bergeron et al. (2022) will be closely followed to asses the forecasting performance as well as the economic interpretation of the latent space of the models.

This thesis aims to contribute in two ways. Firstly, by understanding how machine learning can be leveraged and assess its capabilities in reconstructing and completing the IVS of single equity options. Secondly, this paper aims to add to the interpretability of auto-encoders in reconstructing the IVS. All this will provide new insights into understanding the dynamics of the IVS for single equity options. This has led to the following research question:

How do auto-encoder models perform in completing the implied volatility surface of single equity options?

To help answer this research question, the following sub-questions are proposed:

- What is the relative performance of a standard auto-encoder between an index option and a single equity option?
- Does the auto-encoder yield economically interpretable features?
- Does a trade-off exists between predictive performance and interpretability?

2 Literature

This section will elaborate on the most relevant financial mathematics and machine learning literature.

Using machine learning to predict or understand asset pricing dynamics has taken flight in the literature during the past few years. A distinction between two fields of interest can be made in the literature. The first category seems to mainly focus on assessing the predictive power of machine learning methods in the case of reconstructing or completing the implied volatility surface. The second research category sheds more light on the interpretability of these machine-learning models.

Ackerer et al. (2020) aims to answer how to interpolate and extrapolate the IVS in an arbitrage-free way properly. They propose using an NN on top of the standard BS or the SSVI model of Gatheral and Jacquier (2014). Combining a multi-layer NN significantly improves the correct construction of the IVS regarding the root-mean-square error (RMSE) and mean absolute percentage error (MAPE). As explained in the introduction, the VAEs have been increasingly popular in the fields mentioned earlier in this thesis. Bergeron et al. (2022) researched how VAEs can generate synthetic yet realistic volatility surfaces when the VAE is trained on multiple assets. They show this by empirically testing its performance on different foreign exchange rates. Hence, the authors do not predict the IVS but solely interpolate the missing points on the observable IVS at a certain point in time.

The research of both Cao et al. (2020) and Zheng et al. (2021) aim to better understand implied volatility movements. Although these papers try to give some form of interpretability to the machine learning outcomes, this is still quite limited compared to research that considers characteristics of the underlying assets or option contracts, which will be discussed in the next paragraph. Zheng et al. (2021) use an NN, which they compare to the SVI or the SSVI model as the benchmarks. The authors use a combination of single networks, which predict the implied volatility separately, and a weighting network, which calculates so-called 'votings' of the predicted implied volatilities of each single network and how important they are towards the final volatility prediction. Their model (combining single networks and a weighting network) outperforms the benchmarks and a standard vanilla NN (one hidden layer). They note that incorporating no-arbitrage principles does not result in significant underperformance due to the added restrictions. The authors state that it is important to consider these principles since they are a way to incorporate prior financial knowledge.

Cao et al. (2020) developed an NN approach to understand implied volatility movements. They construct a 3-feature neural network with features like the S&P 500 daily change, time-to-maturity, and the practitioner Black-Scholes delta. With their target, the implied volatility change. Later, they added the VIX index as a feature to understand the behavior of the volatility surface in high-volatility environments and how it differs from that in low-

volatility environments. They note that when volatilities are low, and the index return is particularly high, there is a tendency for volatilities to increase. Furthermore, they see that the model incorporating the VIX as a feature has a 62% higher gain than the three-feature model.

In the literature, there exists research that focuses on what kind of characteristics can help predict volatility surfaces or even returns of options themselves. Kelly et al. (2019) propose the instrumented PCA (IPCA) to allow for latent factors and time-varying loadings. They do this by introducing observable characteristics of an asset that instrument for the observable dynamic loadings. They explain their empirical findings in an asset pricing context. Later, in the paper of Gu et al. (2020), it is argued that ML models that incorporate dimension reduction and non-linearities can yield an increase of 27 percentage points in the Sharpe ratio for asset investors. Their predictor set includes 94 characteristics for each stock, interactions of each characteristic with eight aggregate time-series variables, and 74 dummy variables representing the industry sectors.

Bali et al. (2023) have conducted a comprehensive study of the predictability of option returns with machine learning. They test linear, nonlinear, and ensemble models (L-En or NL-En) of those methods to predict option returns. They use 273 variables, of which 80 are option-based characteristics (e.g., illiquidity and tenors) and 193 stock-based characteristics (e.g., book-to-market ratio). Furthermore, they consider trading costs when forming portfolios based on their models, such that their results have an economical and practical merit. The authors found that the ensemble methods L-En and N-En outperform their corresponding single methods when predicting option returns. The N-En models are deemed as the best ones found. Furthermore, they state that the option contract-based characteristics are the most important predictors for future option returns. "So knowing where an option lies on the underlying's IVS and where that IVS lies relative to the market is essential when making option return forecasts." The nonlinear methods that the researchers used are tree-based methods and a simple feedforward NN. Hence, the research does not examine the performance of (variational) auto-encoders.

3 Data

The raw data is obtained from OptionMetrics from the Wharton research database. The database contains a vast array of mostly American-listed options, which are also generally American style. Due to this fact, finding an option's implied volatility is more complex because the Black-Scholes formula can not be inverted for American-styled options. However, OptionMetrics uses a proprietary pricing algorithm based on the Cox-Ross-Rubinstein (CRR) binomial tree model (Cox et al., 1979; OptionMerics, 2023). Therefore, the data available on OptionMetrics can be used, after some data preprocessing, as an immediate input for the models. For every stock and one index, data is obtained starting from 2010-01-04 up until 2021-12-31. The training set contains all data until the end of 2018, covering almost eight years of training data. The validation set is 2019 and will solely be used for hypertuning. Finally, the test set is the COVID-19 pandemic year 2020 and 2021. The reasoning is that it is interesting to infer if the models can perform in turbulent years when they have not seen any data of these highly volatile years. The global financial crisis of 2008 is excluded, and the years 2020 and 2021 are used for testing.

Then, following the framework in Bergeron et al. (2022), I will work with a prespecified grid of 40 points: eight times to maturity (one month, two months, three months, six months, nine months, one year, one and a half years, and two years) and five different deltas (0.1, 0.25, 0.5, 0.75, and 0.9). OptionMetrics provides option contract data for all these points and interpolates if this point does not exist on a certain day.

To ensure enough data is available for a specific option, only stocks listed on the S&P 500 are considered. These include stocks with high liquidity and volumes and imply higher liquidity levels for their derivatives. The index tracking the S&P 500 (ticker: SPX) is included as a benchmark. Then, three different stocks are chosen, part of several industries, like tech, automotive, and oil. The stocks are Microsoft (ticker: MSFT), Tesla (ticker: TSLA), and ExxonMobil (ticker: XOM).

Table 1 Descriptive statistics for the used tickers

Ticker	Statistic					
	Mean vol	Std vol	Min vol	Max vol	Mean volume	Mean Open Interest
MSFT	0.2576	0.0638	0.1212	1.2829	120189	1484548
TSLA	0.5050	0.1435	0.1702	2.1429	149556	604026
XOM	0.2347	0.0995	0.0934	1.4132	42774	579183
SPX	0.1906	0.0730	0.0537	0.9275	381702	4750096

Note: Sample size running from 2010-01-04 through 2021-12-31. For TSLA, the IPO took place on 2010-06-29, and the data starts on that date. The first four columns show statistics related to the volatility of all the options for the specific ticker. For those four columns, each specific option with a different delta and tenor has an implied volatility. The last two columns show the mean volume and open interest. The volume and open interest on a specific day are calculated over all the options in the market for that specific ticker on a specific day.

The stocks are all constituents of the S&P 500 index. Where, as of 22-01-2024, *MSFT* constitutes 7.29% of the index, *TSLA* constitutes 1.46%, and *XOM* constitutes 0.97% of the index. The reasoning behind choosing these specific stocks is that those are the biggest constituents in the index within their industry. For example, the index's first eight stocks are all considered tech stocks of which *MSFT* is the largest. Table 1 shows that the stocks' derivatives are more volatile throughout the dataset than the index *SPX*. The ticker *TSLA* is an interesting addition to the mix since it shows high levels of volatility as well as an extremely high volatility of 214.29% for a specific day and option contract. The single equity options generally show a high daily volume and open interest, where *XOM* is traded less daily than the other single equity options.

4 Methodology

In this section the proposed methodology will be elaborated upon. First in section 4.1 the mathematics behind neural networks will be explained. Then, in section 4.2 the auto-encoder model used in this thesis will be introduced, in combination with the loss function used during training. Then, in section 4.3 the application of the model will be explained. Section 4.4 will delve deeper in the model architecture and hypertuning. The following section 4.5 briefly explains how the models will be trained and the machine learning work flow. Section 4.6 will explain how the models can be calibrated in a practical way. Finally, section 4.7 explains how the IVS will be reconstructed and completed and how the models performance is assessed.

4.1 Neural Networks

This section will briefly introduce how a neural network functions and how it can learn nonlinear relations based on the input it is given. This study uses a feed-forward neural network as the basis for an auto-encoder. The network takes as input the implied volatilities of 40 different option contracts. That is, for 40 different combinations of deltas and maturities. These are stored in a vector called $\mathbf{x}_i$ for a specific ticker i. To keep notation consistent for every layer, it is customary to refer to the values in the nodes of layer l as $\mathbf{z}^{(l)}$, and hence, $\mathbf{z}^{(0)} = \mathbf{x}_i$. The values from one layer undergo a nonlinear transformation to find the values of the next layer. This transformation is done by multiplying the previous layer with a weight matrix and adding a bias. Then, over this formula, an activation function is applied to end up with the values of the next layer. This yields:

$$\mathbf{z}^{(l)} = \sigma(W^{(l-1)}\mathbf{z}^{(l-1)} + \mathbf{b}^{(l-1)}), \text{ for } l = 1, \ldots, L, \tag{1}$$

where σ is the activation function of choice, W is the weight matrix, $\mathbf{b}$ is the bias vector, and L is the number of hidden layers. The activation function is the practitioner's choice and depends on the use case. It can also be considered a hyperparameter. Later, in section 4.4, it will be determined that the Hyperbolic Tangent activation function (Tanh) yields the best performance and is therefore used in all the network layers. Tanh is defined as follows:

$$Tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}. \tag{2}$$

The activation function is plotted in Figure 1.

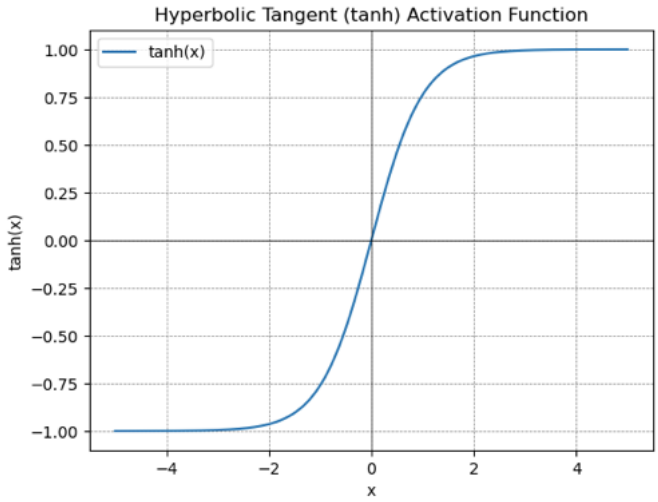


Figure 1 Hyperbolic Tangent activation function used within layers of the system

The basis for the auto-encoder that will be used is a combination of two standard feed-forward neural networks. The nodes get activated by equation 1 with the Tanh as the activation function. The number of layers and each layer's size is called the model's architecture. This architecture is a hyperparameter and will be discussed in section 4.4. For this study, the only architectures considered used the geometric pyramid rule of Masters (1993). The rule states that the number of hidden units will halve when adding a new layer. The only exception to this rule is the size of the latent vector since this will also be a hyperparameter and will have a varying size between one and four.

4.2 The model

An auto-encoder is a particular type of neural network in which the input is encoded by an encoder to a lower-dimensional vector. The input for the used auto-encoder has a total of 40 points. The implied volatility surface of 40 points gets flattened into a vector of size 40, which will serve as the input. These 40 points will be shrunk to a smaller size in the next layer(s). From now on, the auto-encoder shrinks the 40-dimensional input to a latent layer, referred to as z. Then, the latent layer will be expanded back to a 40-dimensional output, which can be compared to the input. This is what the *decoder* is used for. The architecture of the decoder is the mirror image of the encoder. The architecture of a standard auto-encoder can be seen in Figure 2. In the figure, a five-dimensional input gets shrunk to a two-dimensional latent layer called z. In this case, the top node in z would be referred to as z_1 and the bottom node as z_2. Note that in this picture, the geometric pyramid rule is not used. The two z-values will be fed to the decoder $D(z)$, which will construct, through weights and biases, an output that is again five-dimensional.

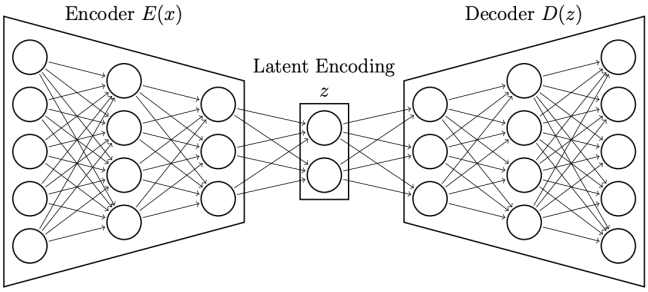


Figure 2 The architecture of a standard auto-encoder (Bergeron et al., 2022)

The parameters of the weights and biases of an auto-encoder are chosen so that the difference between the input and the output is minimized. This is called the reconstruction error (RE) and is defined as:

$$RE = \frac{1}{M}\sum_{i=1}^{M}(X_i - Y_i)^2, \tag{3}$$

where M is the dimensionality for the input and output, X_i the ith input value, and Y_i is the ith output value from the decoder $D(z)$. The network training entails optimizing weights and biases to minimize equation 3. This optimization process is facilitated by an iterative algorithm called gradient descent. The algorithm systematically updates the weights and biases by moving in the direction that minimizes the reconstruction error. By moving in the direction for which the gradients are negative, the algorithm aims to enhance the model's ability to reconstruct the input data accurately.

In addition to gradient descent, the choice of the activation function σ from equation 1 plays an important role in the network. The activation functions allow the neural network to introduce non-linearities to the model, making it able to capture more complex patterns, which is more difficult for a linear method like PCA. An auto-encoder and PCA are related since they are both, in general, unsupervised dimensionality reduction methods. The main difference, however, is that an auto-encoder can capture nonlinear relationships due to the dynamics of a neural network that it leverages.

4.3 Application of the model

Now that the initial idea of the auto-encoder model has been sketched, it is important to discuss how the model will be applied in the context of this thesis. To do so, it is important to distinguish between two terms used throughout the paper's next sections. These are: *reconstructing* and *completing* the implied volatility surface. The first term, reconstructing, refers to using the whole auto-encoder model to reconstruct an output based on the input.

For this research, this means that 40 points are given to the encoder as an input, and the output will be an implied volatility surface also consisting of 40 points. The input and the output can be compared to see how well the auto-encoder *reconstructs* the volatility surface.

However, from a trader's point of view, solely reconstructing does not yield any practical uses of the auto-encoder. As described in Bergeron et al. (2022), it might be the case that on a given trading day, only a few points on the implied volatility surface can be observed. For a trader or market maker, it might be interesting to understand the implied volatilities of options that are not observed in the market yet. Here comes the term *completing* the volatility surface in play. For a trader, it is interesting to see how an auto-encoder can help complete the implied volatility surface based on only a few observed real-time implied volatilities. How this can be done precisely will be discussed in section 4.6 on calibration.

In Figure 3, the encoding architecture of the auto-encoder is illustrated.

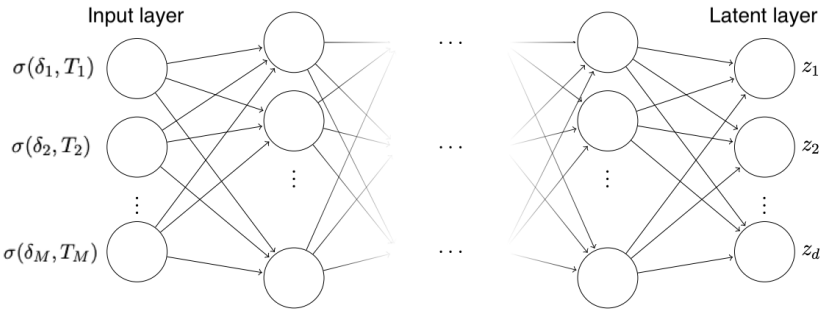


Figure 3 Architecture of the *encoder* of a standard auto-encoder. The latent layer will vary in size and is considered to be a hyperparameter (Bergeron et al., 2022)

In this case, the volatilities of the 40 different contracts (i.e., different delta and maturities) are fed to the encoder as a 40-dimensional vector. The encoder shrinks these 40 values to a latent vector for which the size is a hyperparameter. After that, the latent layer is used as an input to the decoder. This is called the grid-wise approach for auto-encoders. The decoder structure for the model is given in Figure 4.

The hidden latent layer z gets expanded back to a whole surface throughout the network's architecture. This vector can be converted back to an implied volatility surface, and that can be compared to the input. Another way to construct the auto-encoder is called the point-wise approach. In Bayer et al. (2019), the advantages of a point-wise approach are compared to that of a grid-wise approach. In the former approach, the latent vector gets extended by the delta values and tenor values of a specific option contract. Then, throughout the network, this latent layer gets expanded to the second-last layer, which collapses back down to a single volatility forecast. Figure 5 gives the architecture of the decoder of a point-wise architecture.

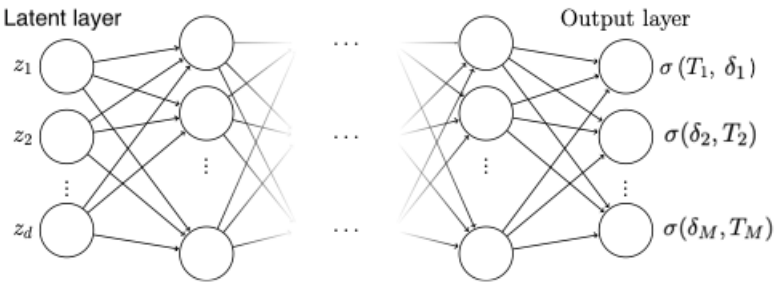


Figure 4 Grid-wise architecture of the *decoder*. The output is a 40-dimensional vector corresponding to the volatility surface (Bergeron et al., 2022)

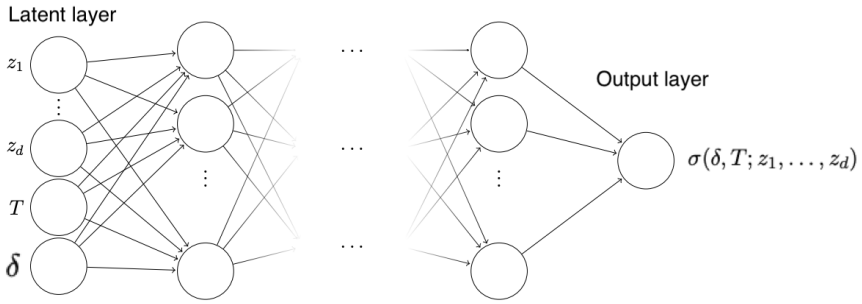


Figure 5 Point-wise architecture of the *decoder*. The output is a single point vector corresponding to the volatility of one specific option (Bergeron et al., 2022)

For this thesis, only the grid-wise approach will be considered. This is due to time constraints and the computational complexity when using a point-wise architecture. For more extensive research on completing the implied volatility surfaces using a point-wise approach, please see Bergeron et al. (2022).

4.4 Model architecture and hyperparameter search

When designing the architecture of an auto-encoder, the practitioner has to deal with many choices regarding hyperparameters. The most straightforward ones are the width and depth of the network. How many hidden layers do the encoder and decoder have, and how many nodes will each layer have? The first choice made regarding the network's architecture is that it should be symmetrical. This means that the encoder and decoder have the same amount of layers and are each exact mirror image, just as in Figure 2. A hyperparameter grid-search approach is used to search over a prespecified grid of network architectures. This grid search is done over a validation data set from 2019-01-03 through 2019-12-30. The best-performing architecture was one where the encoder and decoder (latent layer thus excluded) consisted of three layers. In which the input of 40 went to 32, which went to 16,

which went to 8. Then, these eight nodes were shrunk to the latent size and finally expanded back to 32 nodes in the last hidden layer of the decoder. These 32 nodes were used to yield an output of 40 specific points, as shown in Figure 4. The other parameters needed to be hypertuned are the number of epochs used to train the network, the learning rate, and the weight decay. The model's performance did not improve much after four epochs and performed the best with a learning rate of 0.01 and a weight decay of 0.001. Finally, the Tanh activation function was chosen between each of the hidden layers. During training, it was found that the model was subject to vanishing gradients, resulting in inconsistent output. Therefore, batch normalization is applied during training between each layer. Later, batch normalization is not applied during calibration to ensure the output is consistent. This yields the following hyperparameters that were used during training, depicted in Table 2.

Table 2 Hyperparameter settings used for training the model

Architecture	Epochs	Learning Rate	Weight Decay	Activation function
[40, 32, 16, 8, z, 8, 16, 32, 40]	4	0.01	0.001	Tanh

4.5 Model training and reconstructing

The models are trained for each ticker separately on data from 2010-01-04 through 2018-12-30. With the given hyperparameters as inputs, a trained network is created for four different latent sizes (i.e., the dimensionality of z). The latent dimension takes on integers between 1 and 4. Using the grid-wise approach, a grid of 40 different points is being output by the decoder. These 40 points together form a new reconstructed surface. Then, by using equation 3, the actual surface gets compared to the reconstructed surface, and a loss will be calculated. The optimizer used in regards to the loss function is the ADAM optimizer proposed by Kingma and Ba (2014). For computational reasons, a batch size of 8 is chosen, resulting in shorter training time. After four epochs, training is stopped, and the model with its weights and biases is saved for calibration. This results in four models per ticker, each with the same architecture, except for the latent dimension.

With this trained network, the model can be loaded and put to the test to *reconstruct* volatility surfaces. By feeding each test set sample to the trained model, the model will encode the 40 points given to the appropriate latent size. Then, it will decode these latent values back to a grid of 40 points, yielding a reconstructed surface. By comparing this reconstructed surface with the actual input, the mean-squared error (MSE) can be calculated and compared across the several models. Since we are dealing mostly with values lower than one, the MSEs will be relatively small. To make sure that the performance of the models is statistically different from each other, the Diebold-Mariano (DM) test statistic is used to compare the model's performance (Diebold & Mariano, 2002).

The DM test between two models is constructed as follows

$$DM_{A,B} = \frac{\bar{d}_{AB}}{\hat{\sigma}_{\bar{d}_{AB}}} \xrightarrow{\text{d}} N(0,1), \tag{4}$$

where $\bar{d}_{AB}$ is the sample mean of the squared loss differential between the error surface of model A and model B, and $\hat{\sigma}_{\bar{d}_{AB}}$ is the standard error of the squared loss differential of the error surface of model A and model B. An error surface of one specific date is just the surface obtained when the loss differential between the ground truth on a specific day and the surface output of a model on that same day are being squared.

4.6 Calibration

Once the model is trained, it may be calibrated to ensure that it accurately captures the dynamics of the IVS. In this context, model calibration involves keeping the weights and biases fixed and finding the latent variables (i.e., the inputs for the decoder), which maximize the fit to market data. Essentially, the encoder $E(x)$ can be 'thrown' away, and only the latent dimension z and the decoder $D(z)$ should be considered during calibration. So, a trader would only twist and tweak the values in the latent vector z such that the decoded surface will fit the available market data. A more visual explanation of this calibration step will be given in the next section 4.7. In practice, the most used calibration techniques are gradient-based optimizers. These include the popular BFGS (Nocedal & Wright, 1999) or L-BFGS (Zhu et al., 1997). However, due to the simplicity of the optimization, the Nelder-Mead optimization Nelder and Mead (1965) was leveraged. During calibration, optimization will happen over the z variable. This results in the following optimization equation,

$$z^* = \arg\min_z \frac{1}{M} \sum_{i=1}^{M} (X_i - D(\delta_i, T_i, z))^2, \tag{5}$$

where $D(\cdot)$ is referring to the decoder of the network.

4.7 Completing the IVS

To understand how the models will perform, the performance is tested by randomly sampling points from the 'observed' 40 points on a given day. What will happen next is that the weights and biases of the whole *trained* network will be fixed. Then, by solving equation 5, the latent encodings that generate the best fit for the surface at those randomly sampled locations are chosen. In other words, an optimization search will be conducted, in which the latent variables in the vector z get tweaked so that the decoder will generate the best fit of the IVS on the randomly chosen 'known' points. This is visualized in Figure 6. The figure shows the first three steps in optimizing the three-dimensional vector z in the bottom left. Each blue dot in the 3-D graph of the latent space is another step in the optimization procedure of solving equations 5 starting from the origin point $(0, 0, 0)$. Since the weights and biases in the decoder $D(z)$ are fixed after training, the specific choice of z corresponds

with a decoded volatility surface, depicted in the bottom middle. The ten black dots are the randomly sampled points. The optimization of the z vector is finished after 100 steps, and the model will be fixed. This means all the weights and biases within the trained network are fixed, including the found z values. Eventually, these inferred latent values in z will be used to complete the missing parts of the volatility surface, which is depicted in the right part of Figure 6.

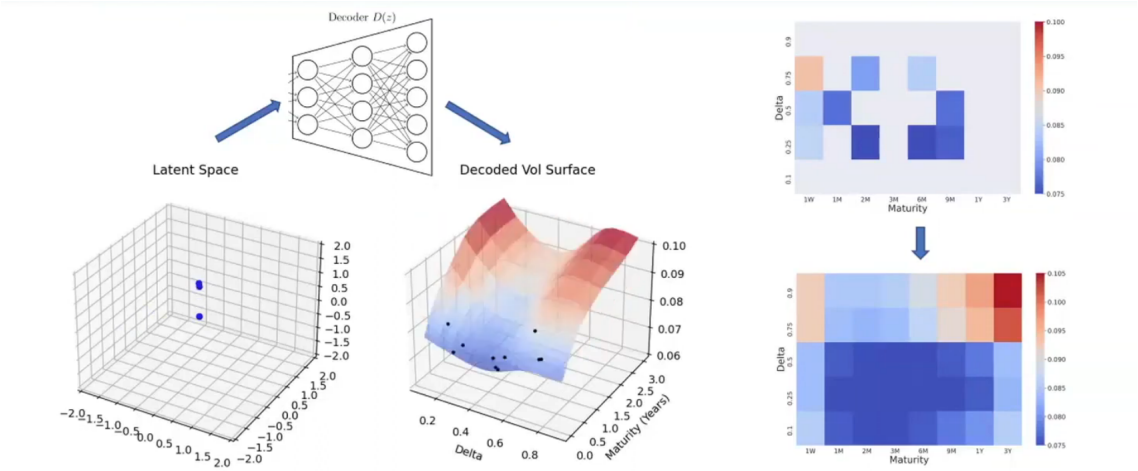


Figure 6 Completing the IVS, adapted from a presentation of Bergeron (2021)

5 Results

In this section, the most important findings will be discussed. First, in section 5.1, several plots of the reconstructed implied volatility surfaces will be shown and compared. Then, in section 5.2, the performance of the autoencoder for each ticker will be discussed. Next, in section 5.3, the behavior of the latent dimension will be explored. Some interesting things can be noted by comparing the dynamics of the latent dimension to the dynamics of the VIX. Finally, in section 5.4, the results of the calibration exercise and the completion of the implied volatility surface will be discussed.

5.1 General model outcome

This section commences by showing a visualization of the auto-encoder's output. Due to the structure of an auto-encoder, it is interesting to compare the implied volatility surface the model gives as an output to the implied volatility surface the model has gotten as an input. Since each model is trained for a specific ticker, and per ticker four different latent sizes were used, this yields a total of $5 \cdot 4 = 20$ models. Since the test set contains the years 2020 through 2021, it is of interest to see how capable the auto-encoders are in reconstructing implied volatility surfaces during normal trading days (i.e., before the COVID-19 crisis occurred) versus the reconstruction capabilities of the auto-encoders during high volatility trading days (i.e., mid-March 2020).

In Figure 7, twelve implied volatility surfaces are depicted in a heat plot. Higher volatility values are shaded red, while lower volatility values are shaded blue. In the left column, the reconstructed implied volatility surface is depicted for a certain ticker on a specific date. In Figure 7, a normal trading day at the beginning of the data set is chosen. The notion of 'normal' comes from the fact that the impact of the COVID-19 crisis did not have a visible impact on the volatility in the financial markets. The actual observed implied volatility surface for that specific day is given in the middle column. This implied volatility surface is also the auto-encoder input, eventually leading to the reconstructed surface found left to it. Bear in mind that the z-axis is scaled differently across the figures. The reason for not using a fixed scale is that it makes it easier to spot the curvature in the surfaces. If the scale of the z-axis grows, some surfaces with a smaller range will look more flat and less curved. The squared error differential between the surfaces is given in the rightmost column. The reason for using a squared error measure is the fact that the differential between the reconstructed surface and the actual surface take values that are both positive and negative.

An initial visual inspection of the differences between the reconstructed implied volatility surface and the actual implied volatility surface shows that options with a longer time to maturity especially have a smaller reconstruction error. This can be seen by looking at the slices for longer tenors. When moving along this axis, it can be noted that the model has difficulty reconstructing options with a shorter time to maturity. This is especially true for the ticker *MSFT* and *TSLA*. What can be noted about these two tickers on this specific day

is that the volatility has high levels in general, especially the derivatives related to Tesla stocks. Furthermore, it can be seen that *XOM* shows the worst reconstruction capabilities for options with a high delta. The reconstructed implied volatility surface for the ticker *SPX* seems to be visually similar to the actual implied volatility surface of the SPX options on 2020-01-16. While the squared error differential is relatively tiny, option contracts with a low delta seem to perform the worst on this specific day.

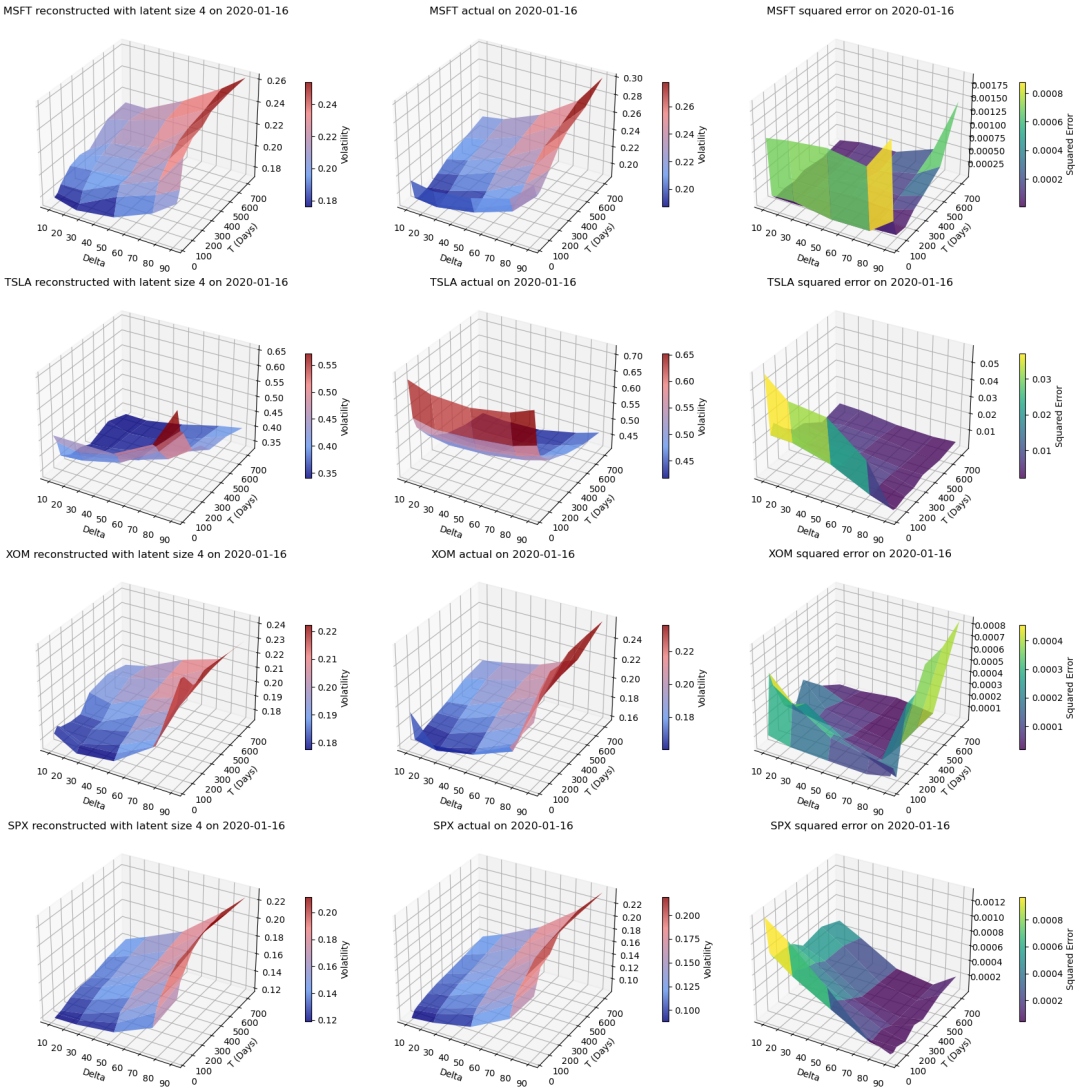


Figure 7 Forecasted, reconstructed and error surfaces on 2020-01-16, 'normal trading day'
Note: The left column depicts the reconstructed IVS when the auto-encoder inputs the actual IVS. The auto-encoder used has a latent dimension of four. The middle column shows the actual values. The right column shows the squared error differential between the two surfaces. Note that the z-axis may differ between columns

Next, in Figure 8, a similar collection of implied volatility and error surfaces are plotted. This time, however, a different date has been chosen. More specifically, the 20th of March 2020, which is characterized by high volatility. These high levels of volatility can be attributed to the impact of the COVID-19 crisis unfolding in the markets, and investors' uncertainty was reflected in the options. When comparing each of the tickers' reconstructed output with its actual output, it can be seen that the reconstructed volatility levels are far off than those of the actual levels. Where some options for *XOM* found volatility levels reaching around 100%, the auto-encoder found only values ranging between 30% and 60%. Furthermore, a visual inspection of all the reconstructed surfaces indicates that the auto-encoder has difficulty getting the general volatility levels right. Especially options near expiry performed the worst, as inferred from the error surfaces. Note that the auto-encoders were trained on data that contained no extreme trading days like the global financial crisis or the COVID-19 crisis.

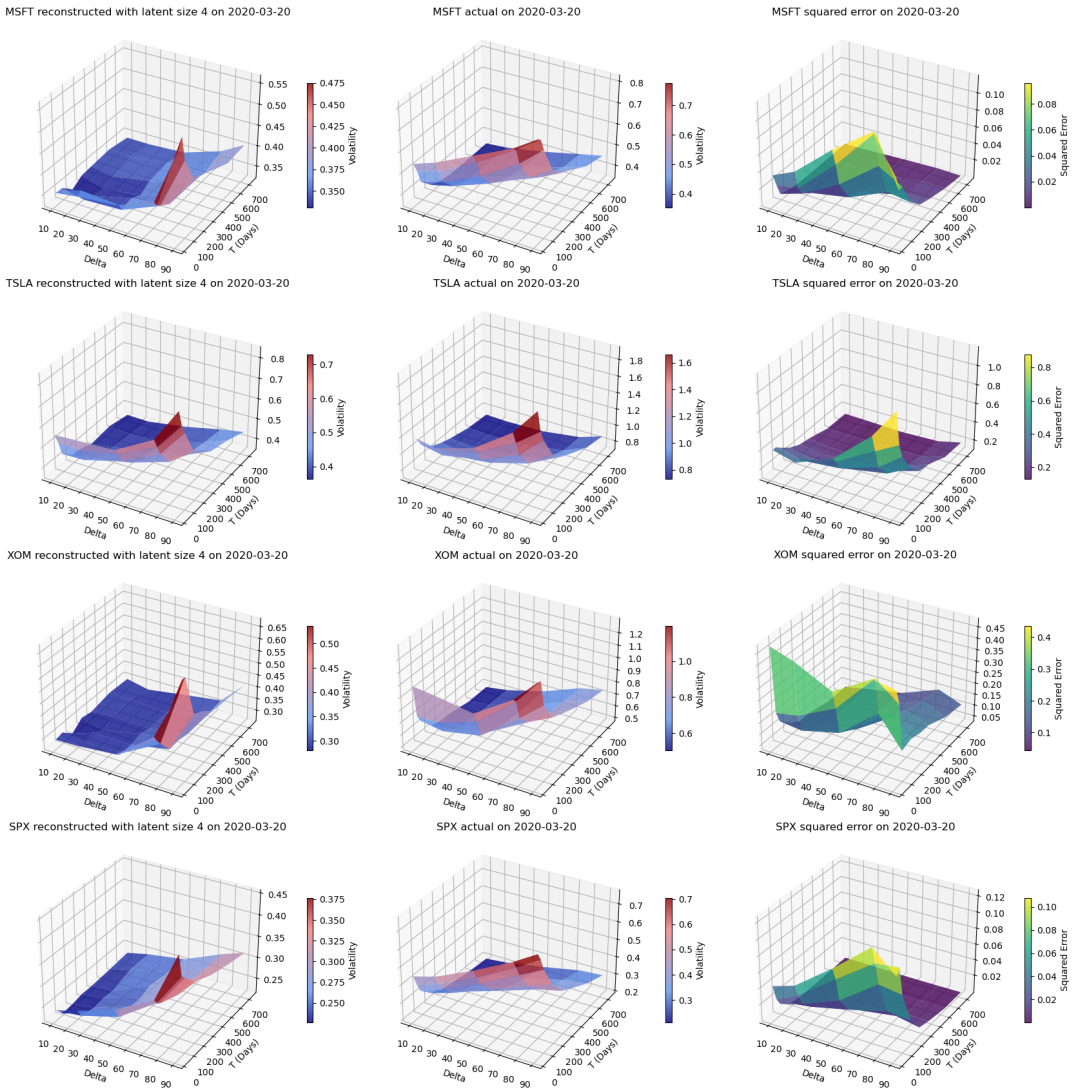


Figure 8 Forecasted, reconstructed and error surfaces on 2020-03-20, 'hectic trading day'
Note: The left column depicts the reconstructed IVS when the auto-encoder inputs the actual IVS. The auto-encoder used has a latent dimension of four. The middle column shows the actual values. The right column shows the squared error differential between the two surfaces. Note that the z-axis may differ between columns

5.2 Model performance

After the preceding rudimentary visual inspection of the model's output, it is interesting to discuss the average reconstruction loss the auto-encoder yields over the whole test sample. As a reminder, the test sample runs from 2020-01-02 through 2021-12-31 for all the tickers. As the methodology describes, every ticker is trained on its historical data. Furthermore, for every ticker, a total of four different models are trained and used, which differ in the dimension of the latent vector z. Table 3 gives the MSEs for the different models. What can be noted from the table is the fact that for equity tickers (i.e., *MSFT, TSLA* and *XOM*), the models do not show a preference for a specific latent size. Surprisingly, the model with a latent size of three performs best for *MSFT* while it performs the worst for the ticker *TSLA*. The asterisk symbol is denoted when a model statistically differs from the other models for a specific ticker at the 1%

Table 3 MSE loss of the AE for each ticker and different latent dimensions

Latent size	Ticker			
	MSFT	TSLA	XOM	SPX
1	0.00493*	0.06561*	0.03021*	0.01675*
2	0.00389*	**0.04775***	0.03093*	0.00300*
3	**0.00343***	0.09300*	0.01941*	**0.00167***
4	0.00382*	0.07120*	**0.01622***	0.00334*

*Note: The MSE loss for fully reconstructing the IVS for four different latent dimension models. An * denotes that the performance is statistically different from the other three models with different latent size for the same ticker as per the Diebold Mariano test.*

The preceding Table 3 shows the MSE when the fully reconstructed IVS is compared to the real observed IVS. All 40 points on the grid are reconstructed and compared to the values for those 40 specific tenors and deltas. However, it is interesting to observe the model's performance for specific subsets of the 40 points. Especially options with a short time to maturity and at the money are expected to have a higher implied volatility. This notion would mean that the reconstruction error around these options would be higher since the auto-encoder would have difficulty correctly reconstructing higher volatility points. The MSE losses for specific slices of the volatility surface for each ticker are given in the Tables 4 through 7. Six different slices are made: three to isolate certain options with specific tenors and three that isolate in-the-money, at-the-money, or out-of-the-money options. The tenors are divided into short time to maturity (i.e., 1 or 2 months), medium time to maturity (i.e., longer than three months but less than one year), and finally, options that will expire between 18 months or 24 months from now.

Table 4 MSE loss of the AE for *MSFT* for different subset slices and latent dimensions

Latent size	Slice					
	(1/2 M)	(3/6/9/12 M)	(18/24 M)	(10/25 delta)	(50 delta)	(75/90 delta)
1	0.0090	0.0043	0.0021	0.0037	0.0047	0.0063
2	0.0081	0.0031	**0.0008**	**0.0014**	**0.0022**	0.0070
3	0.0077	**0.0025**	0.0009	0.0022	0.0029	**0.0050**
4	**0.0072**	0.0033	0.0015	0.0021	0.0029	0.0060

Note: The MSE loss for ticker MSFT for different deltas and tenors. Months are abbreviated to M.

In Table 4, it can be seen that across all the six slices, no specific latent size consistently outperforms the others for the ticker *MSFT*. Considering the slices that isolate specific tenors, it can be seen that the MSE drops for options with a longer time to expiry. The increase in performance when reconstructing a shorter time to maturity (1 or 2 months) versus a longer time to maturity (18 or 24 months) equals about 84% on average for all four latent sizes. When considering the MSE loss for IVS slices that isolate specific delta's, it can be seen that the loss increases when the delta also increases. This is true across all four used *MSFT* models. For the model with latent size two, it can be seen that the difference in MSE loss between at-the-money options and out-of-the-money options is lower than when the same at-the-money options are compared to in-the-money options.

Table 5 MSE loss of the AE for *TSLA* for different subset slices and latent dimensions

Latent size	Slice					
	(1/2 M)	(3/6/9/12 M)	(18/24 M)	(10/25 delta)	(50 delta)	(75/90 delta)
1	**0.0729**	0.0642	0.0611	0.0812	0.0620	**0.0518**
2	0.0730	**0.0432**	**0.0316**	**0.0437**	**0.0351**	0.0582
3	0.1246	0.0880	0.0713	0.1038	0.0824	0.0875
4	0.0782	0.0714	0.0638	0.0861	0.0648	0.0595

Note: The MSE loss for ticker TSLA for different deltas and tenors. Months are abbreviated to M.

Table 5 depicts the MSE loss for the ticker *TSLA* for the different subset slices. As for the full grid, the model with latent size two performs the best compared to the other models for four of the six slices. From the table, it can be inferred that of all the tenor slices, the ones that are far from expiry perform the best. For the best-performing model with a latent size of two, the reconstruction of high-tenor options has a 23% lower MSE than the reconstructed options that expire within one or two months. No clear pattern can be inferred concerning the delta slices *TSLA*. The model with a latent size of two performs best in predicting at-the-money options compared to the other delta slices.

Table 6 MSE loss of the AE for *XOM* for different subset slices and latent dimensions

Latent size	Slice					
	(1/2 M)	(3/6/9/12 M)	(18/24 M)	(10/25 delta)	(50 delta)	(75/90 delta)
1	0.0362	0.0278	0.0290	0.0301	0.0313	0.0297
2	0.0367	0.0290	0.0290	0.0307	0.0317	0.0308
3	0.0239	0.0178	0.0182	0.0190	0.0199	0.0195
4	**0.0217**	**0.0146**	**0.0141**	**0.0144**	**0.0158**	**0.0183**

Note: The MSE loss for ticker XOM for different deltas and tenors. Months are abbreviated to M.

For the ticker *XOM*, an auto-encoder with a latent dimension of four performs best across all six slices. The results are shown in Table 6. This model also performed best when considering the whole grid. All the models with different latent sizes show that *XOM* options that are close to expiry are more difficult to reconstruct. Furthermore, the model with a latent size of four performed best in reconstructing out-of-the-money options.

Lastly, the MSE loss for the six different slices for the index tracker *SPX* are given in Table 7. As before, the model with the latent size that performs the best for the whole grid also shows the best performance for each separate subset slice. The index tracker *SPX* performs better for options with high tenors. This was also the case for *MSFT*. What is interesting to note for the S&P 500 tracker is that the auto-encoder for three out of the four models showed better performance for low delta options than high delta options. This relation is reversed when the latent size of one is considered.

Table 7 MSE loss of the AE for *SPX* for different subset slices and latent dimensions

Latent size	Slice					
	(1/2 M)	(3/6/9/12 M)	(18/24 M)	(10/25 delta)	(50 delta)	(75/90 delta)
1	0.0176	0.0155	0.0180	0.0233	0.0182	0.0092
2	0.0047	0.0027	0.0018	0.0009	0.0023	0.0054
3	**0.0024**	**0.0013**	**0.0014**	**0.0006**	**0.0012**	**0.0027**
4	0.0051	0.0031	0.0020	0.0008	0.0022	0.0064

Note: The MSE loss for ticker SPX for different deltas and tenors. Months are abbreviated to M.

5.3 Exploring the latent dimensions

This section will explore the behavior of the latent dimension and its impact on the output, as well as the relation to the generation of implied volatility surfaces.

In this section, the VIX plays a central role. For this reason, the historical graph for the VIX is shown in Figure 9. The graph shows the value of the VIX from 2020-01-04 through 2021-12-31, which is the same as the test-sample time frame. From the figure, it can be seen that March 2020 was a highly volatile month. Apart from this spike in the VIX, four other noteworthy spikes are around 2020-06, 2020-10, 2021-02, and 2021-12. These months are characterized by high volatility in the financial markets due to investors' uncertainty concerning the COVID-19 crisis.

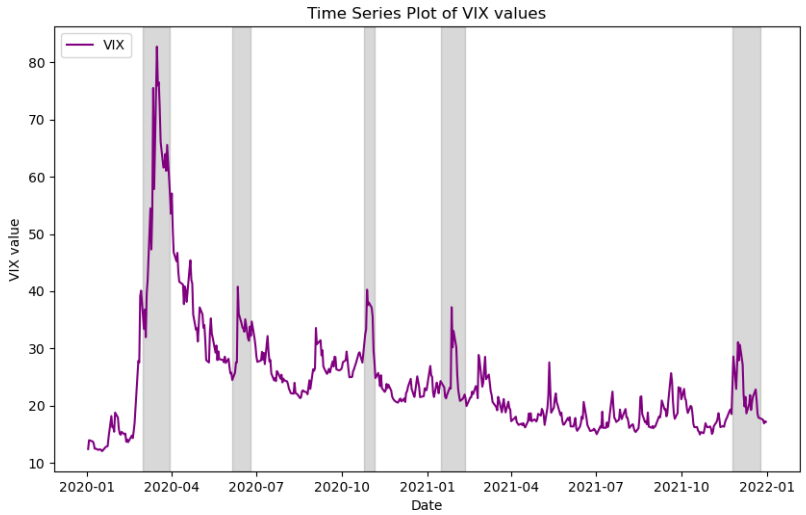
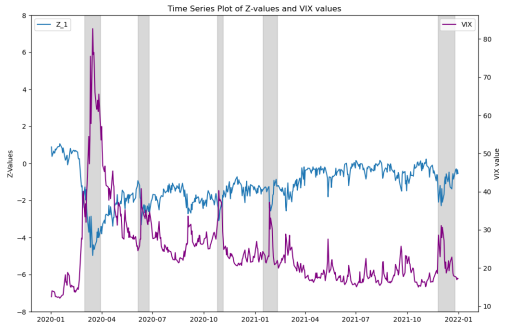
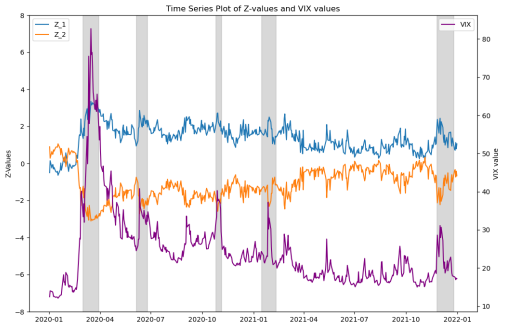


Figure 9 Historical VIX graph from 2020-01-04 through 2021-12-31
Note: The shaded areas correspond to periods of time where the VIX showed a notable spike. The shaded areas correspond to the following range of dates: 2020-03-01 through 2020-03-30, 2020-06-05 through 2020-06-25, 2020-10-25 through 2020-11-05, 2021-01-15 through 2021-02-10, 2021-11-25 through 2021-12-25.

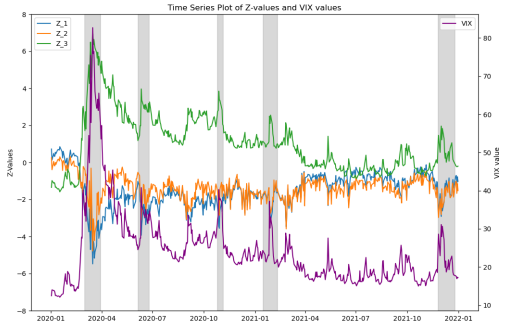
In Figure 10, four different time series plots for the four different auto-encoder *SPX* models are shown in combination with the VIX in purple. It shows the general dynamics of the z-values encoded during out-of-sample testing. What can be deduced from the four subplots is that in all cases, the weights in the latent vector tend to spike upwards or downwards in the grey boxes. In some cases, the latent values could be multiplied by -1 to obtain a figure similar to the VIX. Therefore, the dynamics of the latent dimension for the *SPX* options show a strong resemblance with the VIX dynamics. Similar plots for the single equity options are found in Appendix A.



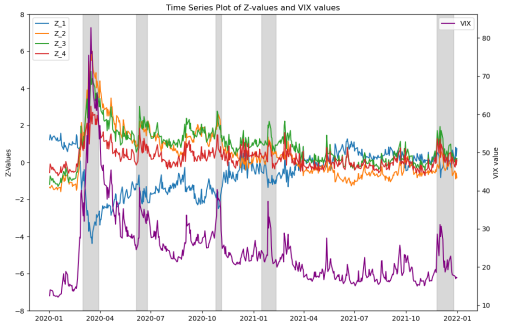
(a) SPX model with latent size 1



(b) SPX model with latent size 2



(c) SPX model with latent size 3



(d) SPX model with latent size 4

Figure 10 Time series plot of the latent vector and the VIX for *SPX*

Note: The shaded areas correspond to periods of time where the VIX showed a notable spike. The shaded areas correspond to the following range of dates: 2020-03-01 through 2020-03-30, 2020-06-05 through 2020-06-25, 2020-10-25 through 2020-11-05, 2021-01-15 through 2021-02-10, and 2021-11-25 through 2021-12-25.

To further understand how the latent dimension influences the reconstructed surfaces, an extra analysis is performed. For this analysis, the model with a latent size of three is considered for the *SPX* ticker since that was the best-performing model for the index options. Some notions about the influence of the latent dimensions can be made by fixing two of three dimensions at a specific value and changing the third value. The reconstructed surfaces can be found in Figure 11.

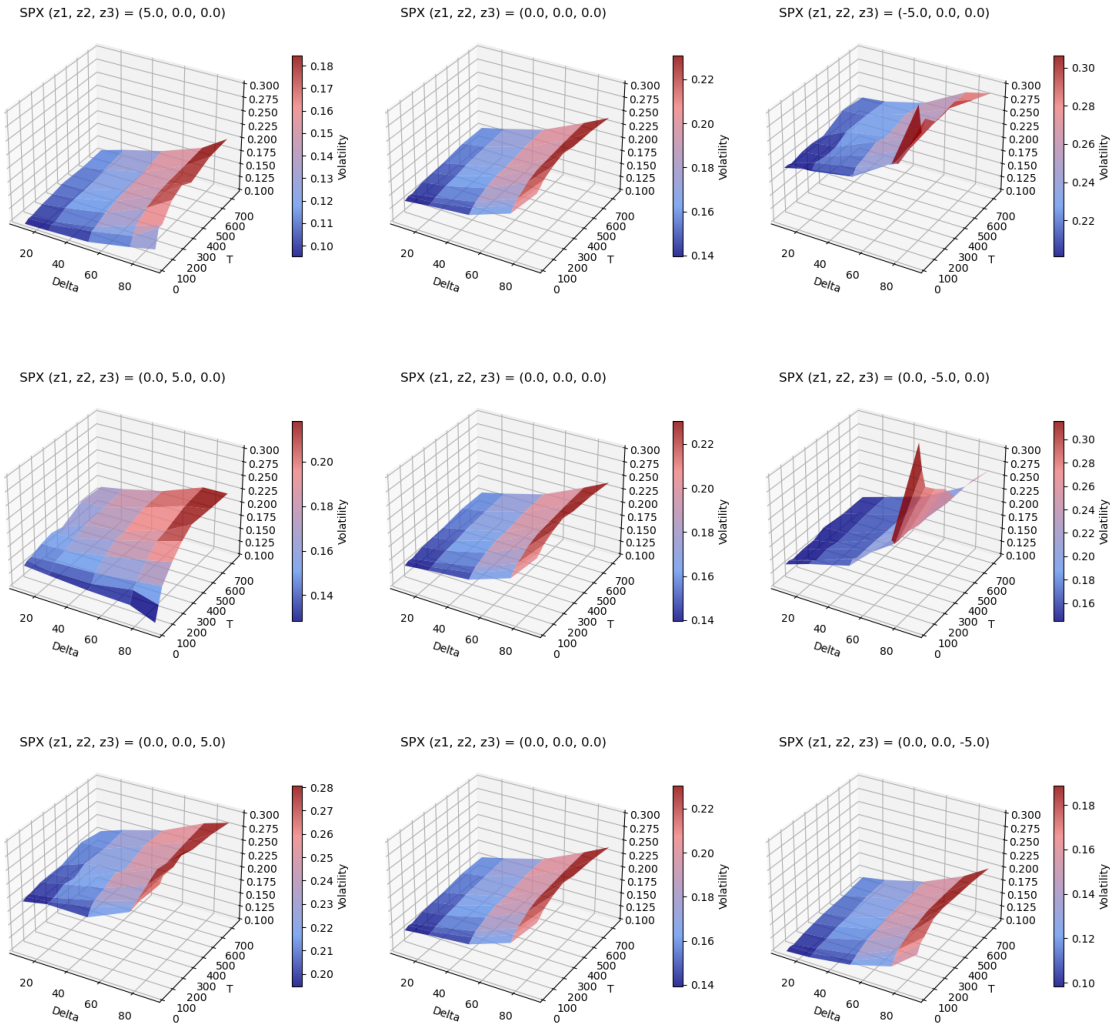


Figure 11 Nine surfaces while changing one of the three latent variables for *SPX*

In every row of the figure, three volatility surfaces are plotted. In each row, two latent variables are fixed at 0.0 while the others range between -5.0 and 5.0. The first row indicates that the first latent dimension impacts the general volatility levels since decreasing the value for $z1$ results in a higher volatility surface. Furthermore, it can be seen that as $z1$ decreases, the-term structure is also changing. For negative values of $z1$, we see higher volatility for options close to maturity, whereas, for a high value of $z1$, the volatility is skewed upwards for options far from expiry. This indicates that $z1$ has an effect on the term-structure.

In the second row of Figure 11, the general level of volatility remains at similar levels when the $z2$ variable is changing. However, what can be seen is that the slope gets heavily influenced when $z2$ increases or decreases. Even more so than the changes seen when controlling for $z1$.

The last latent variable for a three-dimensional latent auto-encoder can be seen in the bottom row of Figure 11. Here, a decreasing third latent dimension corresponds with a decrease in the general volatility level. The shape and skewness of the surface remain relatively similar across the three surfaces. It is interesting to note that for the ticker *SPX*, the case that $z1$ and $z3$ have an opposite impact on the direction in which the surface moves.

Due to the fact that the three latent variables seem to contain different information about the implied volatility surface it is of interest to discuss a time-series plot of the three latent variables and three non-parametric measures of the level, slope and term-structure. These measures are obtained as follows:

$$M_{level_t} = IV_{(50,30)_t} \tag{6}$$

$$M_{slope_t} = IV_{(10,30)_t} - IV_{(90,30)_t} \tag{7}$$

$$M_{term_t} = IV_{(50,30)_t} - IV_{(50,730)_t}, \tag{8}$$

where in equations 6 through 8 the variable $IV_{\delta,T}$ refers to the implied volatility of an option with a delta equal to δ and days to maturity equal to T.

The time-series plot of the six variables can be seen in Figure 12. From the figure it can be seen that the level and term-structure show similar behavior, but in different magnitudes. Recall that the $z1$ variable moved in opposite direction as the general volatility level and the term-structure measure M_{term_t} from equation 8. This behavior can also be seen in the time-series plot of Figure 12. The values for $z2$ move in the same direction, but at a lower magnitude, as the slope time-series. This coincides with the dynamics seen in the nine surface plots of Figure 11. This strengthens the argument that the dynamics of the slope of the implied volatility surface are encoded in the latent variable $z2$. Finally, the values of $z3$ show similar behavior as the level measure. The spikes seen in the in $z3$ also coincide with spikes seen in the time-series of the level measure.

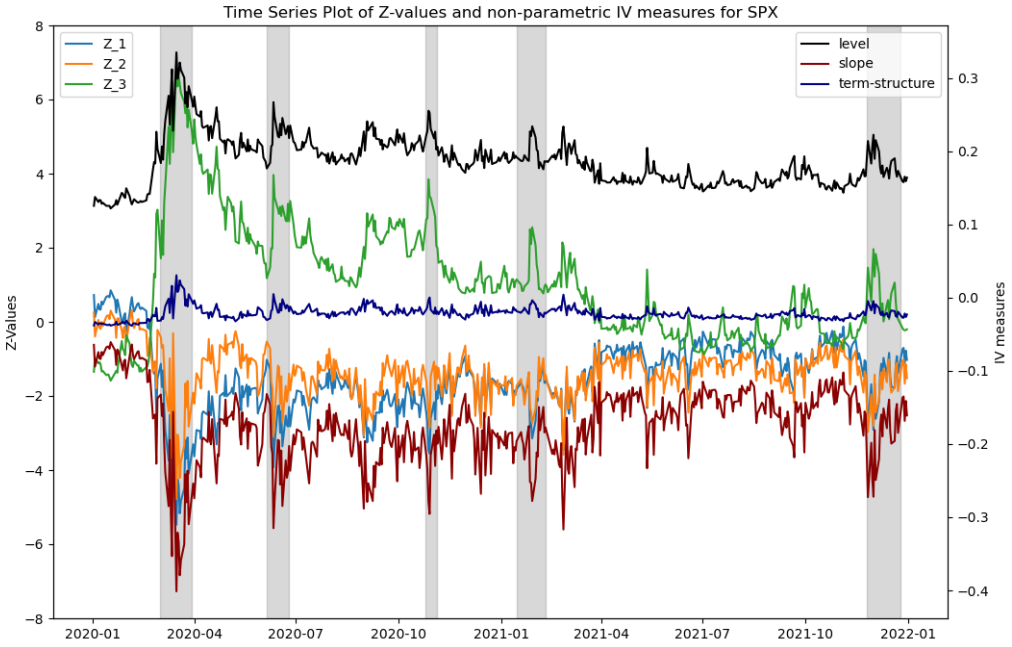


Figure 12 Time series plot of the latent vector and three implied volatility measures for *SPX*

Note: The shaded areas correspond to periods of time where the VIX showed a notable spike. The shaded areas correspond to the following range of dates: 2020-03-01 through 2020-03-30, 2020-06-05 through 2020-06-25, 2020-10-25 through 2020-11-05, 2021-01-15 through 2021-02-10, and 2021-11-25 through 2021-12-25.

5.4 Calibration results

The results for the calibration procedure described in section 4.6 will be discussed in this section. For each ticker, the MSE loss is given when the model has the opportunity to calibrate its latent vector based on an assumed number of subset points. Recall that these points were always randomly subsampled. Table 8 shows the results for *MSFT*. For reference, the MSE loss for the uncalibrated model from Table 3 is added in the second-to-last column. As was the case for the uncalibrated model, the auto-encoder with a latent size of three shows the best performance. It is noteworthy that in all cases, the models calibrated on just a subsample of five points performed better than those not calibrated at all. Furthermore, a clear advantage for increasing the subset size can be seen in the figure since the MSE values decrease when the subset size increases. The last column shows the % performance increase between the uncalibrated model and the calibrated model. In case of the best-performing model of a latent size of three, an increase of almost 76% can be attained by solely calibrating the latent vector.

Table 8 MSE loss for calibrating *MSFT* for different subset sizes and latent dimensions

Latent	Number of points used for calibration							
	5	10	20	30	40	avg. cal.	uncal.	perf.
1	0.00426	0.00401	0.00393	0.00389	0.00387	0.00399	0.00493	19.0%
2	0.00141	0.00088	0.00077	0.00074	0.00075	0.00091	0.00389	76.6%
3	0.00148	0.00076	0.00059	0.00051	**0.00050**	0.00083	0.00343	75.7%
4	0.00166	0.00124	0.00126	0.00101	0.00107	0.00125	0.00382	67.3%

Note: The MSE loss for ticker MSFT when the model is calibrated. The first five columns depicts the loss when a specific model with latent size is calibrated on a (sub)set of visible points. The average calibration MSE for a certain latent size is given under ***avg. cal.*** *and the* ***uncal.*** *column shows when calibration does not happen, and the full surface gets fed to the encoder and gets decoded. Finally the* ***perf*** *column shows the increase in performance between calibrating and not calibrating.*

The results for calibrating the *TSLA* model are given in Table 9. For this ticker, it is also the case that calibrating the model based on some randomly observed options in the market performs better than not being able to calibrate. This is true for all the latent sizes. For *TSLA*, the uncalibrated model performed best when it had a latent size of two. This is also true for the calibrated models, where a latent size of two outperforms the other models. Increasing the number of points used for calibration also increases the reconstruction performance. The average performance increase attained by calibrating the latent vector of two is 74.2%.

Table 9 MSE loss for calibrating *TSLA* for different subset sizes and latent dimensions

Latent	Number of points used for calibration							
	5	10	20	30	40	avg. cal.	uncal.	perf. %
1	0.06021	0.05816	0.05627	0.05563	0.05431	0.05692	0.06561	13.3%
2	0.01354	0.01246	0.01198	0.01186	**0.01181**	0.01233	0.04775	74.2%
3	0.02181	0.01944	0.01886	0.01892	0.01855	0.01952	0.09300	79.0%
4	0.02750	0.02537	0.02180	0.02142	0.02000	0.02322	0.07120	67.4%

Note: The MSE loss for ticker MSFT when the model is calibrated. The first five columns depicts the loss when a specific model with latent size is calibrated on a (sub)set of visible points. The average calibration MSE for a certain latent size is given under ***avg. cal.*** *and the* ***uncal.*** *column shows when calibration does not happen, and the full surface gets fed to the encoder and gets decoded. Finally the* ***perf*** *column shows the increase in performance between calibrating and not calibrating.*

Next, in Table 10 the MSE losses for the ticker *XOM* are given. For this particular ticker, the calibrated models also yield a lower MSE when compared to the uncalibrated models. This is even the case for the models only calibrated on five points. Also, for this ticker, the best-performing calibrated model is the same as the one that was not calibrated, namely the model with a four-dimensional latent size. The auto-encoder calibrated on the full 40 points showed the best performance, 0.00256, regarding MSE loss. The average calibration MSE loss is 0.00389, which is a 76.0% performance increase compared to the uncalibrated model.

Table 10 MSE loss for calibrating *XOM* for different subset sizes and latent dimensions

Latent	Number of points used for calibration							
	5	10	20	30	40	avg. cal.	uncal.	perf. %
1	0.02577	0.02437	0.02336	0.02279	0.02266	0.02379	0.03021	21.3%
2	0.00685	0.00659	0.00430	0.00464	0.00277	0.00503	0.03093	83.7%
3	0.00654	0.00434	0.00345	0.00363	0.00375	0.00434	0.01941	77.6%
4	0.00565	0.00415	0.00342	0.00366	**0.00256**	0.00389	0.01622	76.0%

Note: The MSE loss for ticker MSFT when the model is calibrated. The first five columns depicts the loss when a specific model with latent size is calibrated on a (sub)set of visible points. The average calibration MSE for a certain latent size is given under ***avg. cal.*** *and the* ***uncal.*** *column shows when calibration does not happen, and the full surface gets fed to the encoder and gets decoded. Finally the* ***perf*** *column shows the increase in performance between calibrating and not calibrating.*

The results after calibration for the index ticker *SPX* can be seen in Table 11. The model with the lowest MSE had a latent size of three and a subset of 40 points available to calibrate. It can be seen that each of the calibrated models performs better than the uncalibrated ones, with one exception. The model with a latent size of three that only had five points to calibrate performed worse than the uncalibrated model. It is noteworthy that calibrating the best performing model of a latent size of three only yields a 7.4% increase in MSE terms. For the single equity options this is more than 10 times larger.

Table 11 MSE loss for calibrating *SPX* for different subset sizes and latent dimensions

Latent	Number of points used for calibration							
	5	10	20	30	40	avg. cal.	uncal.	perf. %
1	0.00206	0.00188	0.00182	0.00178	0.00177	0.00186	0.01675	88.8%
2	0.00168	0.00141	0.00126	0.00115	0.00110	0.00132	0.00300	55.9%
3	0.00209	0.00162	0.00129	0.00115	**0.00108**	0.00155	0.00167	7.4%
4	0.00209	0.00159	0.00130	0.00129	0.00133	0.00119	0.00334	64.4%

Note: The MSE loss for ticker MSFT when the model is calibrated. The first five columns depicts the loss when a specific model with latent size is calibrated on a (sub)set of visible points. The average calibration MSE for a certain latent size is given under ***avg. cal.*** *and the* ***uncal.*** *column shows when calibration does not happen, and the full surface gets fed to the encoder and gets decoded. Finally the* ***perf*** *column shows the increase in performance between calibrating and not calibrating.*

Finally, the z-values found during calibration of the *SPX* model with a latent size of three are given in Figure 13. The dark lines show the dynamics of the calibrated z-values for the model when it is able to calibrate on 40 points. This is the best performing model overall in MSE terms. Note that the y-axes are scaled differently due to the fact that the calibrated z-values show huge spikes, especially during March 2020. From the figure it can be seen that the found z-values during calibration differ quite in magnitude during the first highly volatile period of March 2020. During that period the variable $z1$ reached values higher than -50 for the calibrated version, while for the uncalibrated model this value was around -5. However, the general behavior of the calibrated z-values seem to coincide with

the dynamics seen in the uncalibrated values. The main difference is the magnitude of the of the movements.

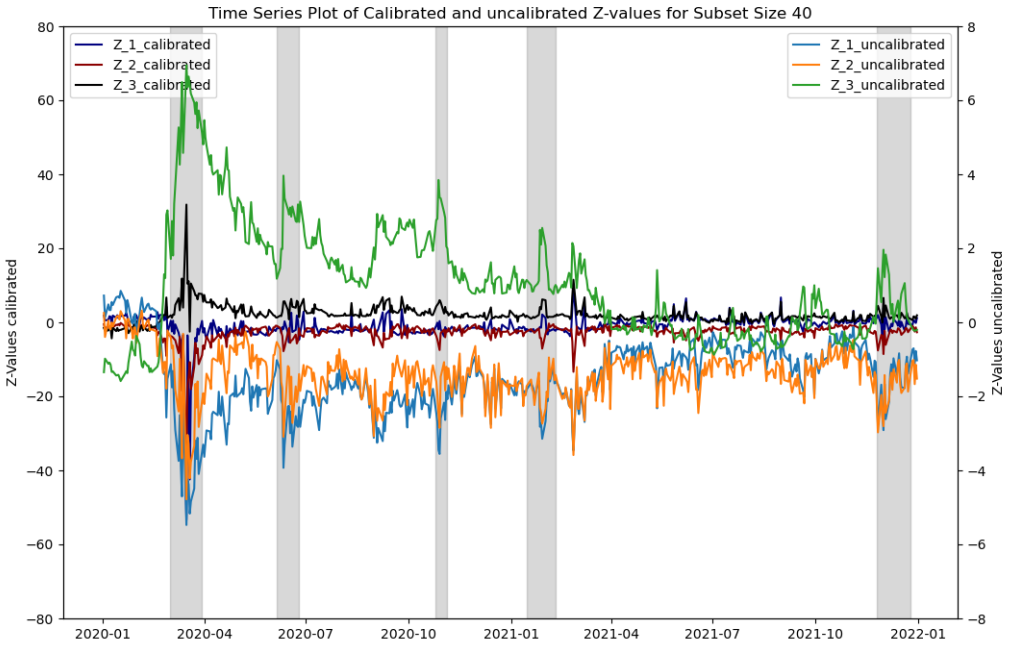


Figure 13 Time series plot of calibrated and uncalibrated z-values for *SPX*
Note: The darker colored lines correspond to the found z-values when the model was calibrated on 40 points. Due to the difference in magnitude both the y-axes have different scalings. The shaded areas correspond to periods of time where the VIX showed a notable spike. The shaded areas correspond to the following range of dates: 2020-03-01 through 2020-03-30, 2020-06-05 through 2020-06-25, 2020-10-25 through 2020-11-05, 2021-01-15 through 2021-02-10, and 2021-11-25 through 2021-12-25.

For clarity a similar plot is made but only for the 2021, where the spikes in volatility were smaller compared to 2020. These are depicted in Figure 14. The figure shows that the calibrated values remain to show higher spikes than the uncalibrated values. Take as an example the black line for the calibrated $z3$ and the green line for the uncalibrated $z3$. The black line shows some notable high spikes when the green line stays relatively flat. This is especially visible around 2021-09. Furthermore, it is noteworthy that the calibrated $z1$ value in navy color moves in the opposite direction as the uncalibrated $z1$ value in blue during the peak in the first grey shaded box. Therefore, a certain trade-off between higher performance and interpretability can be noted.

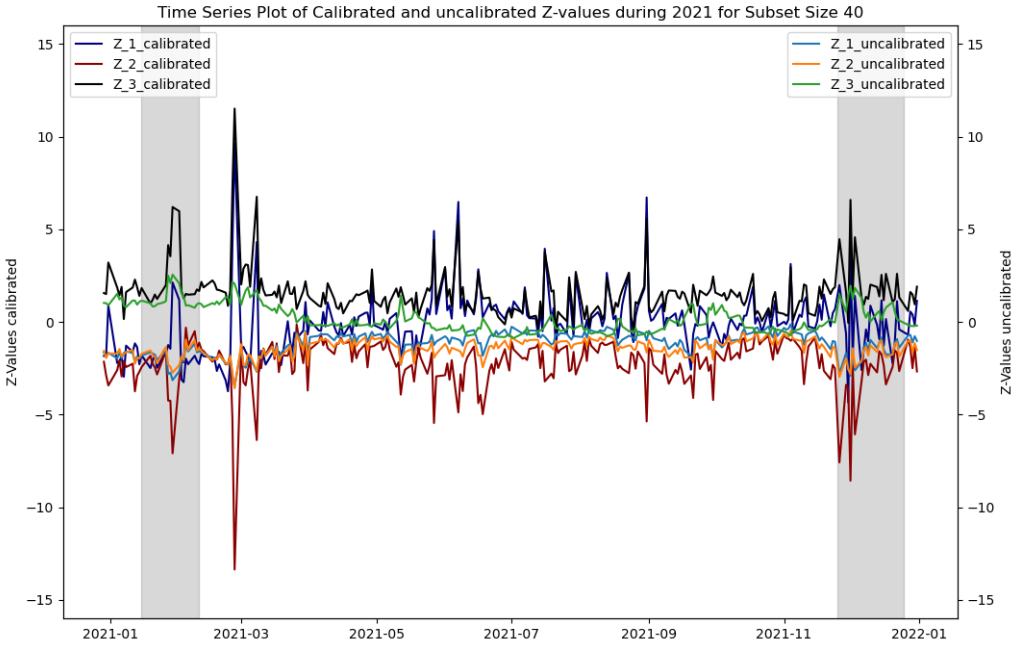


Figure 14 Time series plot of calibrated and uncalibrated z-values for *SPX* in 2021

Note: The darker colored lines correspond to the found z-values when the model was calibrated on 40 points. The shaded areas correspond to periods of time where the VIX showed a notable spike. The shaded areas correspond to the following range of dates: 2021-01-15 through 2021-02-10, and 2021-11-25 through 2021-12-25.

6 Discussion

This section discusses and interprets the results from the previous section in depth. The results' significance and practical implications will be discussed in the first subsection 6.1. Then, in section 6.2, some limitations of the thesis and model will be discussed.

6.1 Implications of the results

One of the subquestions is related to whether a standard auto-encoder performs better for single equity options or an index tracker like the *SPX*. The results show that none of the single equity options outperforms the *SPX* in reconstructing volatility surfaces. The best-performing model for the index yields a loss of 0.00167, while for *MSFT* this is 0.00343. The option contracts for Tesla and Exxon Mobil perform in the best-case scenarios 96% and 90% worse than the S&P 500 index options. A possible explanation for the fact that *MSFT* shows close performance to the *SPX* is the fact that Microsoft constitutes about 7.29% of the index. The combined three single equity options make up about 9.72%. Furthermore, the general volatility levels of the index are on average lower than that of the single equity options during the training and testing period, which further explains the lower MSE for *SPX*. However, when the models are calibrated, the ticker *MSFT* is able to outperform the *SPX* ticker, where the former obtains an MSE of 0.00050 and the latter an MSE of 0.00108. Furthermore, the performance increase due to calibration is almost ten times better for Microsoft options than it is for S&P 500 options. For *TSLA* and *XOM* the performance increase due to calibration is on average 74.2% and 76.0% respectively. This shows that for single equity options calibrating the latent vector has a larger effect than calibrating the latent vector for index options.

When considering auto-encoders in a general sense and their capabilities in reconstructing implied volatility surfaces, there does not exist a clear preference for the size of the latent dimension. For the ticker *XOM*, the auto-encoder prefers a latent vector consisting of four values. While for *MSFT* and *TSLA*, they perform better with a latent size of three and two, respectively. When slices of the implied volatility surface are considered, one general dynamic could be seen across the four tickers. This is the fact that implied volatilities for options with a longer time to maturity are better reconstructed than options that are close to maturity. This underlines the fact that for the four tickers, the term structure is upward-sloping, indicating that traders expect the underlying to become more volatile over time. Furthermore, for three out of the four tickers, it is the case that options with a higher delta are better reconstructed than options out of the money or at the money. Only *TSLA* shows behavior in which no clear pattern could be recognized when the delta slices are considered. The fact that *TSLA* showed unstable results might be related to the extreme high volatilities the stock was subject to during 2020.

To better understand the auto-encoder's latent dimension, the time-series plot of the z-values were discussed for *SPX*. The found z-values during reconstruction show similarities

with the VIX index. When the VIX shows high spikes during volatile times, the latent z-values show similar spikes either upward or downward. This means that the models could capture the complexity of market volatility that manifests in the VIX. The model with a latent size of three performs best for the *SPX* and als captures similar dynamics as the VIX. Furthermore, it can be concluded that the three latent values contain information about the implied volatility surface. Where the level, slope and term-structure are all encoded in one of these three variables, or a combination of them. The time-series plot of these three implied volatility measures also show a strong resemblance with the dynamics of the latent variables throughout the testing period.

The practical implications of the models come into play when considering the model's ability to *complete* implied volatility surfaces rather than reconstruct them. It is interesting for a practitioner or trader to see how a trained model performs when only a subset of a specific volatility surface is present. By only considering the latent vector in combination with the decoder, a trader can tweak the values in the latent vector such that the surface decoded from these values best fits the observed points in the market. The results show that with even as few as five observed points, the models can complete the rest of the surface with a lower reconstruction error compared to the case where the model is not calibrated. In all cases, the calibration of the latent vector and decoding of those values preforms better than feeding the actual, implied volatility surface to the encoder and then obtaining a surface. This fact underlines the importance of the latent vector and the ability of the decoder to obtain well-formed surfaces. Furthermore, the calibration yielded at least a 75% performance increase compared to not calibrating for all the single equity tickers. In contrast, this performance increase is only 7.4% for the index options.

6.2 Limitations

One of the implications of this study is that during training, the COVID-19 crisis is left out on purpose. This choice is made to understand what the performance of the auto-encoder would be during highly volatile years. However, from a trader's point of view, it would make sense to use an expanding window and retrain the model daily. This way, when the COVID-19 crisis was unfolding, the auto-encoder could learn some of the dynamics of the implied volatility surfaces during extremely volatile days. This would decrease the probability that the auto-encoder collapses when it sees extreme implied volatility surfaces, which it finds difficult to reconstruct properly.

The models used to generate the results are only trained on the data of the specific ticker. This means that each auto-encoder has never seen data from other equity options or the index tracker. To be able to better understand if auto-encoders are viable options for single equity options in general, it is wise to see what the performance would be when an auto-encoder is trained on multiple different equity options. However, the fact that each equity option preferred a different latent size might indicate that it is hard to generalize the

implied volatility surfaces of equity options that operate in different industries.

Finally, during calibration, a random subsample is chosen to calibrate on. For example, in the case of calibrating on five points, these five points could be scattered to the edges of the implied volatility surface, resulting in very different deltas and tenors. It would be more realistic not to subsample these points randomly but to choose five points that are relatively close to one another. This is relevant since, in the real world, it is more likely that options that expire two years from now with a high delta are not visible, and that options that are at-the-money and close to expiry are visible. This way, the calibration exercise would mimic real-world scenarios better.

7 Conclusion

This section will bring the thesis to a close. The thesis's purpose and main research question will be discussed and reflected upon. The most important findings will be reiterated, and the section will conclude with some final recommendations.

To reiterate, the main research question of this thesis is as follows:

How do auto-encoder models perform in completing the implied volatility surface of single equity options?

To answer this question, several subquestions are proposed related to the relative performance between index and equity options and the trade-off between performance and interpretability. It is concluded that the index option outperformed all the single equity options during *reconstruction*. However, calibrating the volatility surfaces and then *completing* results in a performance increase. This increase in performance is at least ten times higher for single equity options than it is for the index options. This show that the auto-encoders have the potential to be used practically for traders who can calibrate their models.

Furthermore, auto-encoders could capture the dynamics of market volatility. The best-performing model for the *SPX* showed similar behavior as the VIX. Each of the separate z-values in the latent vector is responsible for different behavior in the implied volatility surface of the ticker *SPX*. This adds to the interpretability of the latent dimension in this specific auto-encoder. This can help practitioners understand what kind of volatility surfaces will be generated when moving or calibrating the latent space.

To conclude, this research lays an initial basis for understanding the implications and dynamics behind auto-encoders for reconstructing and completing implied volatility surfaces for single equity options in particular. It is shown that for a small selection of well-known single equity options, no outperformance compared to index options could be realized when implied volatility surfaces were reconstructed. However, the added value of calibration has become clear for the three single equity options. This thesis shows the potential upside for traders to calibrate the latent size for single equity options. Some directions for further research would be to use an expanding window and retrain the models each time a new sample becomes available to train on. Furthermore, in past research, the potential of *variational* auto-encoders has been shown for forex options but not yet for single equity options. These considerations could help researchers and traders better understand the dynamics of implied volatility surfaces and how these can be completed or reconstructed when available data is sparse.

References

Ackerer, D., Tagasovska, N., & Vatter, T. (2020). Deep smoothing of the implied volatility surface. *Advances in Neural Information Processing Systems*, *33*, 11552–11563.

Almeida, C., Fan, J., Freire, G., & Tang, F. (2023). Can a machine correct option pricing models? *Journal of Business & Economic Statistics*, *41*(3), 995–1009.

Bali, T. G., Beckmeyer, H., Moerke, M., & Weigert, F. (2023). Option return predictability with machine learning and big data. *The Review of Financial Studies*, *36*(9), 3548–3602.

Bayer, C., Horvath, B., Muguruza, A., Stemper, B., & Tomas, M. (2019). On deep calibration of (rough) stochastic volatility models. *arXiv preprint arXiv:1908.08806*.

Bergeron, M. (2021). *Maxime bergeron - completing partial implied vol surfaces with variational autoencoders* [Online Lecture], Quants Hub & BTRM.

Bergeron, M., Fung, N., Hull, J., Poulos, Z., & Veneris, A. (2022). Variational autoencoders: A hands-off approach to volatility. *The Journal of Financial Data Science*.

Black, F., & Scholes, M. (1973). The pricing of options and corporate liabilities. *Journal of political economy*, *81*(3), 637–654.

Cao, J., Chen, J., & Hull, J. (2020). A neural network approach to understanding implied volatility movements. *Quantitative Finance*, *20*(9), 1405–1413.

Cox, J. C., Ross, S. A., & Rubinstein, M. (1979). Option pricing: A simplified approach. *Journal of financial Economics*, *7*(3), 229–263.

Diebold, F. X., & Mariano, R. S. (2002). Comparing predictive accuracy. *Journal of Business & economic statistics*, *20*(1), 134–144.

Gatheral, J., & Jacquier, A. (2014). Arbitrage-free svi volatility surfaces. *Quantitative Finance*, *14*(1), 59–71.

Gong, Z., Frys, W., Tiranti, R., Ventre, C., O'Hara, J., & Bai, Y. (2022). A new encoding of implied volatility surfaces for their synthetic generation. *arXiv preprint arXiv:2211.12892*.

Gu, S., Kelly, B., & Xiu, D. (2020). Empirical asset pricing via machine learning. *The Review of Financial Studies*, *33*(5), 2223–2273.

Gu, S., Kelly, B., & Xiu, D. (2021). Autoencoder asset pricing models. *Journal of Econometrics*, *222*(1), 429–450.

Kelly, B. T., Pruitt, S., & Su, Y. (2019). Characteristics are covariances: A unified model of risk and return. *Journal of Financial Economics*, *134*(3), 501–524.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kingma, D. P., & Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Kramer, M. A. (1991). Nonlinear principal component analysis using autoassociative neural networks. *AIChE journal*, *37*(2), 233–243.

Masters, T. (1993). *Practical neural network recipes in c++*. Morgan Kaufmann.

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, *5*, 115–133.

Nelder, J. A., & Mead, R. (1965). A simplex method for function minimization. *The computer journal*, *7*(4), 308–313.

Nocedal, J., & Wright, S. J. (1999). *Numerical optimization*. Springer.

OptionMerics. (2023). *Ivydb file and data reference manual*. Version 5.4. OptionMetrics.

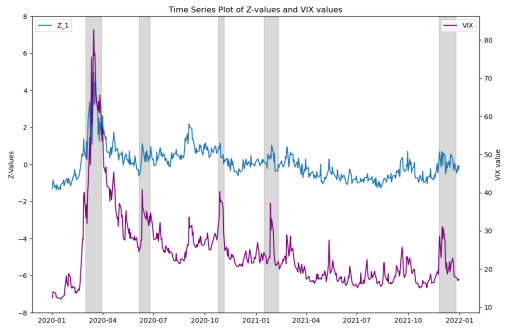
Zhang, W., Li, L., & Zhang, G. (2023). A two-step framework for arbitrage-free prediction of the implied volatility surface. *Quantitative Finance*, *23*(1), 21–34.

Zheng, Y., Yang, Y., & Chen, B. (2021). Incorporating prior financial domain knowledge into neural networks for implied volatility surface prediction. *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 3968–3975.

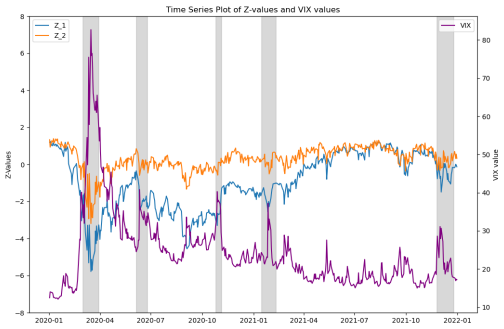
Zhu, C., Byrd, R. H., Lu, P., & Nocedal, J. (1997). Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on mathematical software (TOMS)*, *23*(4), 550–560.

A Additional Results

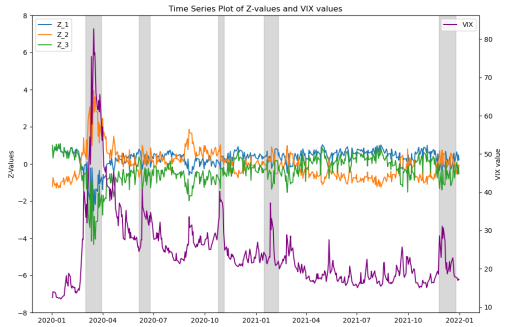
In this section additional graphs and results which are referenced in the main text will be shown.



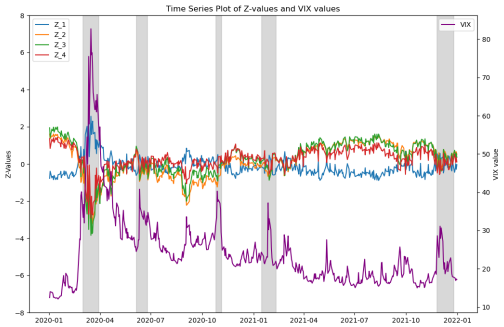
(a) MSFT model with latent size 1



(b) MSFT model with latent size 2



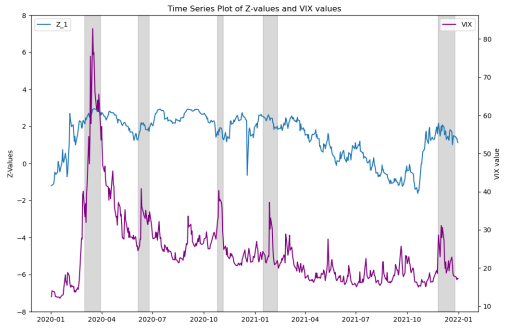
(c) MSFT model with latent size 3



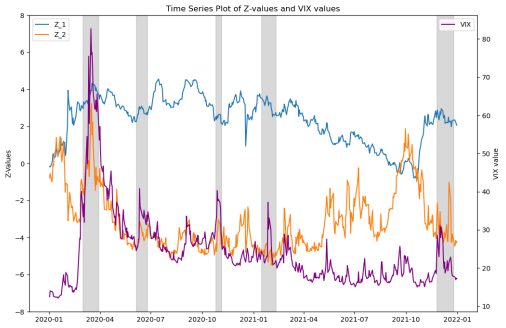
(d) MSFT model with latent size 4

Figure 15 Time series plot of the latent vector and the VIX for *MSFT*

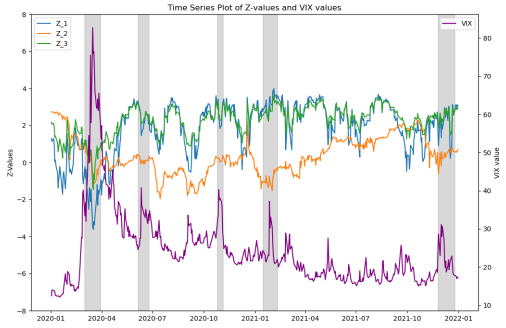
Note: The shaded areas correspond to periods of time where the VIX showed a notable spike. The shaded areas correspond to the following range of dates: 2020-03-01 through 2020-03-30, 2020-06-05 through 2020-06-25, 2020-10-25 through 2020-11-05, 2021-01-15 through 2021-02-10, and 2021-11-25 through 2021-12-25.



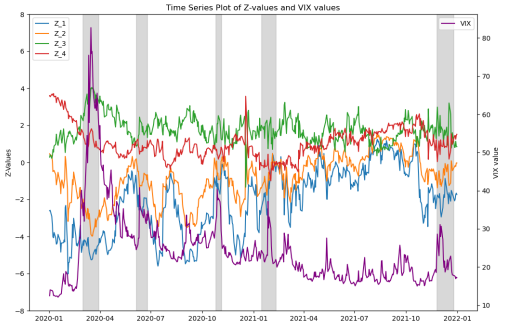
(a) TSLA model with latent size 1



(b) TSLA model with latent size 2



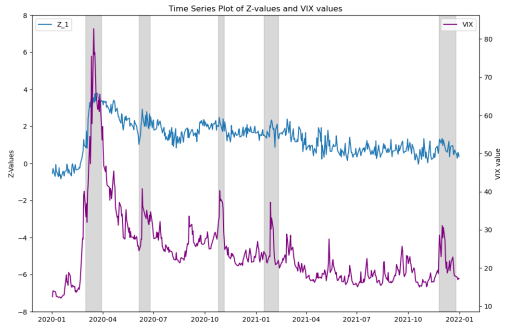
(c) TSLA model with latent size 3



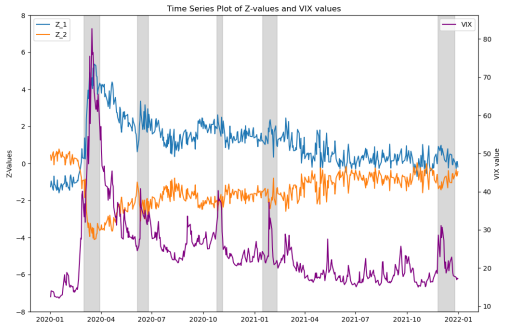
(d) TSLA model with latent size 4

Figure 16 Time series plot of the latent vector and the VIX for *TSLA*

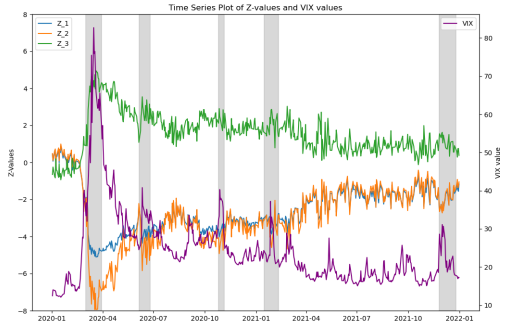
Note: The shaded areas correspond to periods of time where the VIX showed a notable spike. The shaded areas correspond to the following range of dates: 2020-03-01 through 2020-03-30, 2020-06-05 through 2020-06-25, 2020-10-25 through 2020-11-05, 2021-01-15 through 2021-02-10, and 2021-11-25 through 2021-12-25.



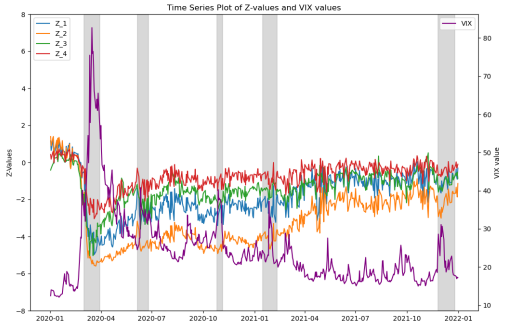
(a) XOM model with latent size 1



(b) XOM model with latent size 2



(c) XOM model with latent size 3



(d) XOM model with latent size 4

Figure 17 Time series plot of the latent vector and the VIX for *XOM*

Note: The shaded areas correspond to periods of time where the VIX showed a notable spike. The shaded areas correspond to the following range of dates: 2020-03-01 through 2020-03-30, 2020-06-05 through 2020-06-25, 2020-10-25 through 2020-11-05, 2021-01-15 through 2021-02-10, and 2021-11-25 through 2021-12-25.

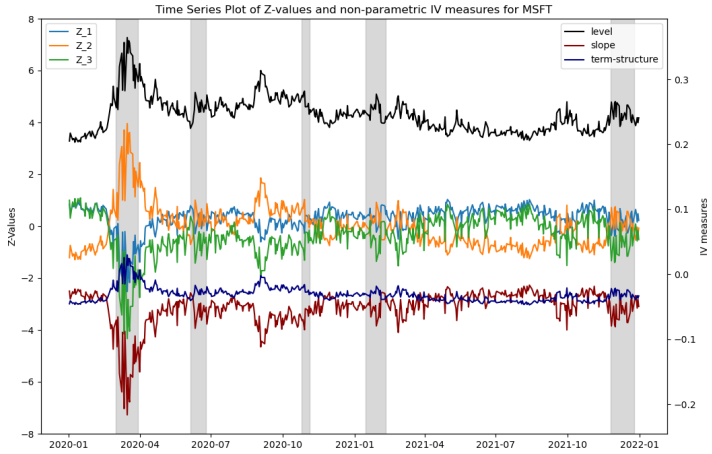


Figure 18 Time series plot of the latent vector and three implied volatility measures for *MSFT*

Note: The shaded areas correspond to periods of time where the VIX showed a notable spike. The shaded areas correspond to the following range of dates: 2020-03-01 through 2020-03-30, 2020-06-05 through 2020-06-25, 2020-10-25 through 2020-11-05, 2021-01-15 through 2021-02-10, and 2021-11-25 through 2021-12-25.

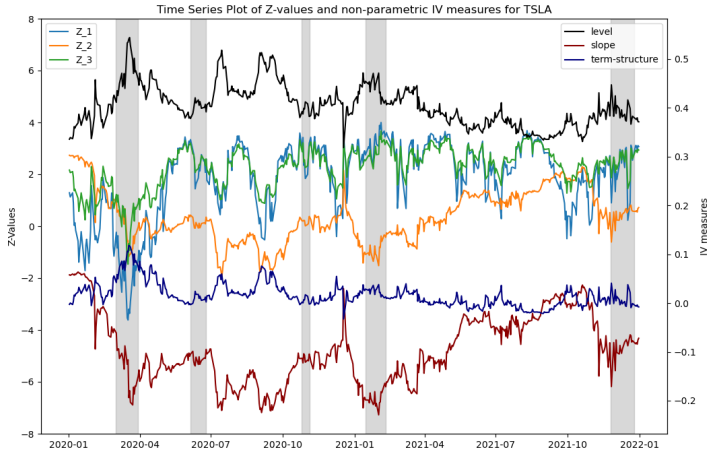


Figure 19 Time series plot of the latent vector and three implied volatility measures for *TSLA*

Note: The shaded areas correspond to periods of time where the VIX showed a notable spike. The shaded areas correspond to the following range of dates: 2020-03-01 through 2020-03-30, 2020-06-05 through 2020-06-25, 2020-10-25 through 2020-11-05, 2021-01-15 through 2021-02-10, and 2021-11-25 through 2021-12-25.

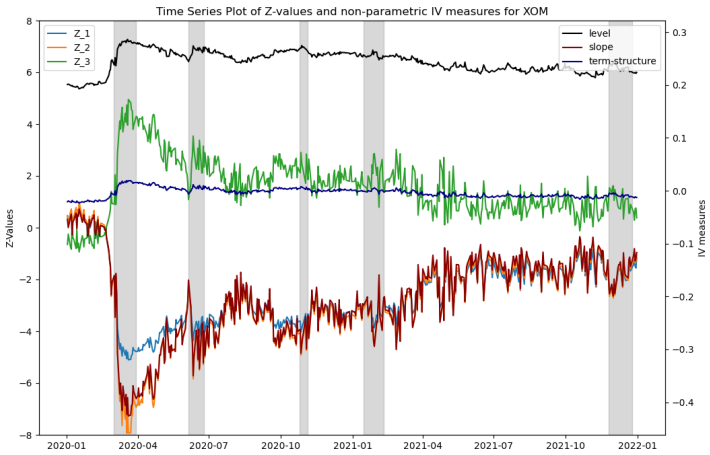


Figure 20 Time series plot of the latent vector and three implied volatility measures for *XOM*

Note: The shaded areas correspond to periods of time where the VIX showed a notable spike. The shaded areas correspond to the following range of dates: 2020-03-01 through 2020-03-30, 2020-06-05 through 2020-06-25, 2020-10-25 through 2020-11-05, 2021-01-15 through 2021-02-10, and 2021-11-25 through 2021-12-25.