Autoencoder Option Pricing Models*

Gustavo Freire† **Evgenii Vladimirov**‡

January 26, 2024

Preliminary draft: Please do not circulate

Abstract

We propose a new framework allowing to estimate non-parametric affine and non-affine option pricing models. Our method applies autoencoder neural networks to the log-characteristic function implied by observed option prices. Since the logarithm of the characteristic function is linear in the state factors under the affine assumption, we obtain a data-driven affine model by specifying a linear mapping in the autoencoder architecture. Alternatively, we let the data speak about any needed non-linearities to estimate a non-affine model. Using an extensive panel of S&P 500 option data, our approach reveals that the non-affine class of models significantly outperforms the affine class in pricing options out-of-sample.

JEL classification: C14, C58, G13

Keywords: Option pricing, Characteristic function, Affine jump-diffusion models, Machine learning, Autoencoders

*We thank conference participants at the FinEML Conference 2023 (Rotterdam) for useful comments.

†E-mail: freire@ese.eur.nl, Erasmus University Rotterdam, Tinbergen Institute and ERIM.

‡E-mail: vladimirov@ese.eur.nl, Erasmus University Rotterdam and Tinbergen Institute.

1 Introduction

Options written on the S&P 500 index are widely popular financial instruments offering the opportunity to gain exposure to or hedge against various types of risk associated with market returns. As such, observed option prices contain valuable information about the time variation of these risks and the compensation required by investors to bear them. Starting with the seminal work of Black and Scholes (1973), researchers have developed a plethora of models to extract this information, mostly relying on parametric assumptions about the dynamics of the underlying asset. Prominent examples include models incorporating stochastic volatility (Heston, 1993), jumps in the underlying asset price (Pan, 2002), jumps in volatility (Eraker, Johannes, & Polson, 2003; Broadie, Chernov, & Johannes, 2007), a second volatility factor (Bates, 2000) and a separate left tail jump factor (Andersen, Fusari, & Todorov, 2015a, 2015c).

The vast majority of parametric models employed to study equity and option markets, including those described above, belongs to the affine jump-diffusion (AJD) class of Duffie, Pan, and Singleton (2000). This class provides tractability as it allows for a quasi-closed form option pricing formula, which greatly simplifies option evaluation and model estimation. Whether the affine assumption is justifiable empirically, however, is an open question. Andersen, Fusari, and Todorov (2015b) note that incorporating time variation in the shape of jump tails might improve their model fit, but at the cost of going beyond the AJD class. Investigating the validity of the affine assumption directly, Ait-Sahalia and Kimmel (2007) and Christoffersen, Jacobs, and Mimouni (2010) show that non-affine stochastic volatility models outperform the affine Heston (1993) model under different criteria. More recently, Li and Zhang (2013) find that affinity of variance-swap prices, which holds under the AJD class, is rejected by the option data.

In this paper, we propose a new framework allowing to estimate non-parametric affine and non-affine option pricing models. Under a general dynamics of the underlying asset price, our approach extracts the latent state factors driving this dynamics and builds the option pricing function based on the extracted latent factors. We use our method to empirically investigate whether non-affine models provide a better fit for the panel of

options than affine models. Our evidence, which does not depend on specific parametric assumptions, can shed light on the fundamental differences between the two types of model and the features of the option panel that each model is able to reproduce.

Our framework makes use of *autoencoders*, a machine learning tool that employs neural networks to non-parametrically obtain a low-dimensional factor representation of the data. The key feature of our method is that we use as input the log-characteristic function of the underlying return implied by observed option prices.[1] The characteristic function is the most important ingredient of option pricing models, summarizing not only all probabilistic information that matters for pricing, but also relevant economic modeling restrictions. More precisely, under the AJD class, the log-characteristic function is a linear function of the state vector driving the underlying dynamics. This is such that we can estimate a data-driven affine model by specifying a linear mapping between the input and the latent factors in the autoencoder architecture. Alternatively, we let the data speak about any needed non-linearities to estimate a non-affine model.

Empirically, we use daily data on an extensive panel of S&P 500 options from January 2011 to February 2023. We first extract, for each day, the log-characteristic function implied by the cross-section of option prices across different maturities. Then, using our method, we estimate affine and non-affine option pricing models with different numbers of state factors by fitting the panel of option-implied characteristic functions. We compare their pricing accuracy out-of-sample and use the Diebold and Mariano (2002) test to assess the statistical significance of differences in performance.

We find that the non-affine specification significantly outperforms the affine one, regardless of the number of factors and the maturity of the options. In particular, the affine model usually requires twice the number of factors to yield comparable performance to the non-affine model. We also show that there is little empirical support for models with a low-dimensional set of state factors, where increasing the number of factors always leads to improvements in out-of-sample performance, albeit at a decreasing rate. Finally,

[1]The log-characteristic function can be obtained from observed option prices in a model-free manner using the payoff spanning results of Carr and Madan (2001). The option-implied characteristic function is also used, e.g, by Todorov (2019) to non-parametrically estimate spot volatility and Boswijk, Laeven, and Vladimirov (2022) to efficiently estimate parametric option pricing models.

we compare popular one-factor AJD parametric models in the literature (Heston, 1993; Bates, 1996; Pan, 2002) with our one-factor non-parametric models. The non-parametric affine model outperforms the best parametric model, suggesting that the considered parametric models do not capture all relevant information within the AJD class. However, the main improvement in performance still comes from allowing for non-linearities in the log-characteristic function with a non-affine model.

The remainder of the paper is organized as follows. After a brief review of the related literature, Section 2 provides our theoretical framework. Section 3 introduces the autoencoder methodology we propose to estimate non-parametric affine and non-affine option pricing models. Section 4 describes our data, while Section 5 contains the empirical analysis. Finally, Section 6 concludes the paper.

1.1 Related literature

Our paper mainly relates to three strands of the literature. The first strand proposes option pricing models based on parametric assumptions about the underlying asset price dynamics. The majority of extant models fall within the affine jump-diffusion class of Duffie et al. (2000). This includes the seminal Black and Scholes (1973) model, the Heston (1993) stochastic volatility model, the jump-diffusion model of Bates (2000) and Pan (2002), the double-jump stochastic volatility model of Eraker et al. (2003) and Broadie et al. (2007), and the three-factor model of Andersen et al. (2015a, 2015c). As an important advantage, these models afford analytical tractability. Examples of non-affine parametric models, on the other hand, can be found in Jones (2003), Ait-Sahalia and Kimmel (2007) and Christoffersen et al. (2010). We contribute by showing how to estimate non-parametric affine and non-affine models using autoencoders and the option-implied log-characteristic function of returns. We empirically analyze the option pricing performance of these models out-of-sample.

The second strand investigates the validity of the affine jump-diffusion assumption, which is predominant in the option pricing literature. Jones (2003) and Ait-Sahalia and Kimmel (2007) find that a non-affine stochastic volatility model in the constant

elasticity of variance (CEV) class generates more realistic return behavior than the Heston (1993) model. Christoffersen et al. (2010) show that non-affine stochastic volatility models outperform the Heston (1993) model in fitting realized volatilities, underlying returns and option prices. The evidence in these papers depends on the specific parametric affine and non-affine models under consideration. In contrast, our evidence is non-parametric and based on optimally chosen option pricing models within the affine and non-affine class. More recently, Li and Zhang (2013) exploit the fact that variance-swap prices are affine functions of the state variables under AJD models to test for this assumption. They find that the option data rejects the affine property. There are two main differences between their approach and ours. First, they test whether the underlying dynamics is affine under both the physical and risk-neutral measures, while we assess the weaker property of affinity under the risk-neutral measure only (which is the necessary one for obtaining a closed-form option pricing formula). Second, we are able to directly assess how affine and non-affine models compare in fitting the option panel for different numbers of factors.

Finally, our paper contributes to a growing literature adopting machine learning methods to analyze financial markets. Gu, Kelly, and Xiu (2020) compare different machine learning tools for predicting stock returns. Kelly, Pruitt, and Su (2019) propose a latent factor model for stock returns that allows factor exposures to linearly depend on asset characteristics. More closely related to our work, Gu, Kelly, and Xiu (2021) generalize the latter approach by modeling exposures as a flexible non-linear function of covariates using autoencoders. While these papers focus on reduced-form asset pricing models, we combine autoencoders with economic restrictions to extract latent factors driving the dynamics of the underlying asset and price options. In the option pricing literature, Hutchinson, Lo, and Poggio (1994), Garcia and Gençay (2000) and Dugas, Bengio, Bélisle, Nadeau, and Garcia (2009) approximate the option pricing function using neural networks under different specifications. Neural networks are also employed by Liu, Oosterlee, and Bohte (2019), Ruf and Wang (2022) and Almeida, Fan, Freire, and Tang (2023) to numerically solve financial models, hedge options and correct parametric option pricing models, respectively. In contrast to the black box nature of these methods,

our autoencoder network maps to a structural option pricing model. In fact, our goal is to learn about the dynamics of the underlying asset price, rather than achieving the most accurate approximation of option prices possible.

2 Theoretical framework

In this section, we provide the theoretical framework for the approach adopted in this paper. We start by formulating the general option pricing model and extracting the option-implied conditional characteristic function (CCF). Then, we discuss the implications of the affine assumption on the option-implied CCF.

2.1 The option pricing framework

Let us denote by F_t the futures price at time t written on an asset under consideration. We assume the absence of arbitrage, which implies the existence of a risk-neutral probability measure $\mathbb{Q}$, locally equivalent to the physical probability measure $\mathbb{P}$. Since we are interested in extracting information from options, we formulate the option pricing model under the risk-neutral measure. In particular, we assume the following dynamics for the log futures price $y_t := \log F_t$ under $\mathbb{Q}$:

$$\mathrm{d}y_t = \mu_y(X_t;\theta)\mathrm{d}t + \sigma_y(X_t;\theta)\mathrm{d}W_t + Z_t^y \mathrm{d}N_t, \tag{1}$$

$$\mathrm{d}X_t = \mu_X(X_t;\theta)\mathrm{d}t + \sigma_X(X_t;\theta)\mathrm{d}W_t + Z_t^X \mathrm{d}N_t, \tag{2}$$

where X_t is a d-dimensional latent state vector driving the dynamics of y_t; W_t is a standard d_W-dimensional Brownian motion; N_t is a pure jump process with intensity $\lambda(X_t;\theta)$; Z_t^y and Z_t^X are jump sizes with generic but fixed conditional distribution ν; and θ is a vector of model parameters that govern the model for y_t and X_t. The functional form of the drifts $\mu_y(x)$ and $\mu_X(x)$, diffusive variances $\sigma_y(x)\sigma_y(x)'$ and $\sigma_X(x)\sigma_X(x)'$, and jump intensity $\lambda(x)$ are left unspecified, but assumed to satisfy the regularity conditions that guarantee a unique solution to the stochastic differential equations (1) and (2).

Let us further denote by $O_t(\tau, K)$ the out-of-the-money (OTM) European option price observed at time t written on the same underlying asset with a fixed time-to-maturity $\tau > 0$ and strike price K. Under the no-arbitrage assumption, the OTM option prices are determined by the conditional expectation, given the information $\mathcal{F}_t$ at time t, of the corresponding payoff functions under the risk-neutral measure $\mathbb{Q}$:

$$O_t(\tau, K) = \begin{cases} e^{-r\tau}\mathbb{E}^{\mathbb{Q}}[(F_{t+\tau} - K)^+|\mathcal{F}_t], & \text{if } K > F_t, \\ e^{-r\tau}\mathbb{E}^{\mathbb{Q}}[(K - F_{t+\tau})^+|\mathcal{F}_t], & \text{if } K \leq F_t, \end{cases}$$

where we assume a constant risk-free rate r.

Once the functional form and parameters of the stochastic differential equations in (1) and (2) are known, we can compute the conditional distribution of the future asset price and, consequently, the option prices. However, this information is unknown a priori. Instead, what we observe in the market are the option prices for different sets of strikes and maturities, which can be utilized to infer information about the distribution of future asset prices. In particular, using the payoff spanning result of Carr and Madan (2001), the CCF of the log return, $\phi_t(u, \tau) := \mathbb{E}^{\mathbb{Q}}[e^{\mathrm{i}u \log(F_{t+\tau}/F_t)}|\mathcal{F}_t]$, can be spanned as a portfolio of OTM options:

$$\phi_t(u, \tau) = 1 - (u^2 + \mathrm{i}u)\frac{1}{F_t}\int_{\mathbb{R}} e^{r\tau + (\mathrm{i}u - 2)k} \cdot O_t(\tau, k)\mathrm{d}k, \quad u \in \mathbb{R}, \tag{3}$$

where $k = K/F_t$ represents the moneyness of an option with strike price K. The spanning result in (3) is exact and completely model-free, i.e., it is independent of the functional forms in (1) and (2). The option-implied CCF is also used, e.g, by Todorov (2019) for nonparametric estimation of spot volatility from options and Boswijk et al. (2022) for the estimation of parametric option pricing models.

The option replication result for the CCF (3) is practically infeasible since it requires option prices for a continuum of strikes. Nevertheless, we can approximate this CCF in practice using a Riemann sum approximation of the integral in (3) and a finite number of options. In the empirical application, we follow the procedure outlined in Boswijk et

al. (2022) to obtain a computationally feasible counterpart of the option-implied CCF, which we denote as $\widehat{\phi}_t(u,\tau)$.

2.2 Affine and non-affine frameworks

The option pricing literature is primarily dominated by affine jump-diffusion (AJD) models due to availability in a semi-closed form of their Laplace transforms. The AJD class is often defined as a class in which the drift terms $\mu_y(x)$ and $\mu_X(x)$, diffusive variances $\sigma_y(x)\sigma_y(x)'$ and $\sigma_X(x)\sigma_X(x)'$, and jump intensity $\lambda(x)$ are functionally affine in the state vector X_t (see Duffie et al., 2000). The affine dependence of these functions implies that the joint CCF of the log return and the state vector X_t has an exponential-affine form. Duffie, Filipović, and Schachermayer (2003) show that this property fully characterizes the AJD class under some mild regularity conditions. Therefore, we will exploit this exponentially-affine structure of the CCF to investigate if relaxing the affine assumption can further improve the option pricing performance.

In particular, our analysis is based on the property that under the AJD assumptions, the logarithm of the (option-implied) CCF of log returns, $\widehat{\psi}_t(u,\tau) := \log \widehat{\phi}_t(u,\tau)$, is affine in the state vector, i.e.,

$$\widehat{\psi}_t(u,\tau) = \alpha(u,\tau;\theta) + \beta(u,\tau;\theta)\cdot X_t + \xi_t(u,\tau), \qquad u \in \mathbb{R}, \tag{4}$$

where $\alpha(u,\tau;\theta)$ and $\beta(u,\tau;\theta)$ are model-dependent deterministic coefficients and $\xi_t(u,\tau)$ collects the measurement errors in the CCF approximation.[2] While many parametric option pricing models belong to the AJD class, which implies that equation (4) can be used as a linear measurement equation, there is (implicit) evidence that the AJD assumption can be restrictive. When the AJD condition is not satisfied, the logarithm of the CCF is no longer an affine function in the state vector X_t. We aim to exploit this property for an empirical test of the AJD assumption.

To operationalize the functional complex-valued linear relation (4), we consider a

[2]This functional linear equation, including the measurement error term, is discussed in more detail in Boswijk et al. (2022).

finite number of CCF arguments $u \in \mathcal{U}$ from a set $\mathcal{U} \subseteq \mathbb{R}$ with cardinality q, and p different maturities $\tau_i,\ i = 1, \ldots, p$. For each maturity, we stack the option-implied log CCFs along the chosen arguments u:

$$\widehat{\psi}_{t,i} := \left(\widehat{\psi}_t(u_1, \tau_i), \ldots, \widehat{\psi}_t(u_q, \tau_i)\right)', \quad i = 1, \ldots, p,$$

which are further stacked over their real and imaginary parts, as well as p maturities:

$$\widehat{\Psi}_t := \left(\Re(\widehat{\psi}_{t,1}), \Im(\widehat{\psi}_{t,1}), \ldots, \Re(\widehat{\psi}_{t,p}), \Im(\widehat{\psi}_{t,p})\right).$$

The stacked real-valued n-dimensional vector $\widehat{\Psi}_t$ with $n := 2pq$ is still linearly loaded on the state vector:

$$\widehat{\Psi}_t = w_\alpha + w_\beta X_t + \varepsilon_t, \tag{5}$$

where ε_t and w_α are n-dimensional vectors and w_β is a $n \times d$ matrix of the stacked outputs of the functions $\xi_t(u, \tau),\ \alpha(u, \tau; \theta)$, and $\beta(u, \tau; \theta)$, respectively. Our main interest lies in examining the affine property of the log CCF, rather than analyzing the specific values of these elements themselves.

In particular, under the AJD class, the state vector X_t can be obtained as a linear combination of the elements of the option-implied log CCF vector $\widehat{\Psi}_t$. For instance, one can employ principal component analysis (PCA) to extract the orthogonal factors from the matrix $\widehat{\Psi} := \left(\widehat{\Psi}_1, \ldots, \widehat{\Psi}_T\right)$, where T is the time dimension. This approach works well when the AJD assumption is satisfied, and the log CCF maintains a linear relationship with the state vector (Boswijk, Laeven, Marijnen, & Vladimirov, 2023). However, when the AJD assumption is violated, applying PCA on $\widehat{\Psi}$ can yield a spurious number of factors since the log CCF is no longer linear in the state vector. Therefore, in this situation the state vector needs to be extracted from the option-implied CCF using a non-linear approach.

We utilize machine learning tools to extract the factors under both the affine and

non-affine assumptions. By employing autoencoder neural network techniques, we can effectively extract the relevant factors that are essential for accurate option pricing, even when the AJD assumption does not hold. This allows us to explore the implications of non-linearity in the log CCF for option pricing models and potentially improve their performance compared to traditional affine models.

3 Methodology

In this section, we describe the methodology we exploit to test the affine property of option pricing models empirically. We start by briefly reviewing standard autoencoders as a special case of neural networks. Then, we describe how autoencoder methodology can be cast into the affine and non-affine frameworks discussed in the previous section. Finally, we discuss the training procedure and estimation of parametric models aligned with the non-parametric approaches.

3.1 Autoencoders

An autoencoder (AE) is a special type of neural network that aims to learn an efficient representation of the input data. The input layer is compressed (or *encoded*) into a low-dimensional intermediate hidden layer (the *coder*), which is then unpacked (or *decoded*) to reconstruct the output layer. Typically, the output data is the same as the input, such that an AE is considered an unsupervised learning technique that is often used for dimension reduction. In particular, it can be seen as a generalization of PCA, allowing for dimension reduction via non-linear transformations of the input data.

More specifically, consider an AE with $l = 1, ..., L$ hidden layers, each containing d_l output units (or *neurons*) stacked in the vector $z^{(l)}$. Let z denote the n-dimensional vector of input data, such that $z^{(0)} = z$. The hidden layers of the AE can be represented via the following recursive formula:

$$\underset{d_l \times 1}{z^{(l)}} = \overset{\circ}{h}_l \left(\underset{d_l \times d_{l-1}}{A^{(l-1)}} \; \underset{d_{l-1} \times 1}{z^{(l-1)}} + \underset{d_l \times 1}{b^{(l-1)}} \right), \quad \text{for } \; l = 1, ..., L, \tag{6}$$

where $A^{(l-1)}$ is a matrix of weight parameters, $b^{(l-1)}$ is a vector of so-called *bias* parameters and $\overset{\circ}{h}_l : \mathbb{R}^{d_l} \to \mathbb{R}^{d_l}$ applies a fixed scalar *activation* function $h_l(\cdot)$ to each element of the vector $A^{(l-1)}z^{(l-1)} + b^{(l-1)}$. The output layer is then given by:

$$\underset{n\times 1}{f(z, b, A)} = \overset{\circ}{h}_{L+1}\left(\underset{n\times d_L}{A^{(L)}}\ \underset{d_L\times 1}{z^{(L)}} + \underset{n\times 1}{b^{(L)}}\right). \tag{7}$$

The weight and bias parameters are estimated by minimizing a given loss function (usually mean squared error) measuring how well the prediction $\widehat{z} := f(z, b, A)$ approximates z.

There are two main features that distinguish the AE from a standard feedforward neural network. First, as already mentioned, AEs are usually employed as an unsupervised learning technique, where the output $f(z, b, A)$ is meant to approximate the input. Second, the main interest with an AE lies in identifying a low-dimensional representation of the data, i.e., in learning a specific hidden layer l^*, the coder layer, with $d_{l^*} \ll n$ factors that best characterizes the input. Hidden layers such that $l \leq l^*$ constitute the encoder function $E(z, \eta^e) : \mathbb{R}^n \to \mathbb{R}^{d_{l^*}}$ that compresses information in z into the factors, where η^e collects the weight and bias parameters of these layers. Analogously, the decoder function $D(z^{l^*}, \eta^d) : \mathbb{R}^{d_{l^*}} \to \mathbb{R}^n$ recreates the input and is composed of layers $l > l^*$ with parameters η^d. To illustrate, Figure 1 depicts the graphical representation of the architecture of an AE with $d_{l^*} = 1$ factor.

It is easy to see why AEs encompass PCA as a particular case. Consider an AE with one hidden layer. The encoder maps the input z to the neurons in the hidden layer as $z^{(1)} = \overset{\circ}{h}_1(A^{(0)}z^{(0)} + b^{(0)})$. The d_1-dimensional vector $z^{(1)}$ represents the low-dimensional factors. Then, the decoder maps the factors into an estimate of the input: $\widehat{z} = \overset{\circ}{h}_2(A^{(1)}z^{(1)} + b^{(1)})$. If one sets the activation functions to be linear, a linear factor model with d_1 factors is obtained: $\widehat{z} = A^{(1)}(A^{(0)}z^{(0)} + b^{(0)}) + b^{(1)}$. Therefore, estimating the AE by minimizing mean squared error leads to factor loadings and factors that are equivalent (up to a linear transformation) to those obtained by applying PCA to the covariance matrix of z.[3] Incorporating non-linear activation functions together with multiple hidden layers makes the AE more general and flexible in the sense that factors can

[3]For a formal proof of this equivalence, see Proposition 1 in Gu et al. (2021).

be non-linear functions of z and the approximation $f(z, b, A)$ can be a non-linear function of the factors.

In the context of option pricing models, we use the option-implied log CCF as the input and output data, i.e., we set $z = \widehat{\Psi}_t$. As discussed in the previous section, our motivation stems from the fact that the CCF is the most important ingredient of option pricing models, not only summarizing all probabilistic information that matters for pricing, but also containing all relevant economic modeling restrictions. Importantly, under the AJD class, the log CCF is a linear function of the state vector. Therefore, to obtain the latent states under an affine non-parametric option pricing model, we can use the AE with one hidden layer and linear activation function, or, equivalently, the standard PCA. On the other hand, under a more general non-affine class of option pricing models, the log CCFs are no longer linear functions of the latent states. Thus, an AE with multiple hidden layers and nonlinear activation functions can be used to approximate these non-linearities.

Therefore, by comparing the fit from applying PCA and AE to the option-implied log CCF, we are able to compare two classes of option pricing models. We emphasize that, since both PCA and AE are non-parametric approaches, we compare the affine and non-affine classes, rather than specific parametric option pricing models as typically done in the related literature.

3.2 Implementation details

In this subsection, we discuss in detail the estimation procedure of the AE models, including the choice of hyper-parameters and the regularization techniques we employ to avoid overfitting. First, we split our data sample into three different subsamples: training, validation, and testing. The training subsample is used to estimate the model for a given set of hyper-parameters values. The hyper-parameters control the complexity of the AE models and are selected from a grid of potential values to minimize the loss function in the validation subsample. More specifically, for each set of hyper-parameters, we predict the log CCF in the validation sample based on the model estimated in the training sample, and then choose the hyper-parameters that deliver better performance

in the validation subsample. Finally, a previously unused testing subsample is used to evaluate the truly out-of-sample performance of the estimated models.

There are plenty of choices to make regarding the AEs' architecture, including the number of layers, the number of neurons in each layer, activation functions, regularization techniques, etc. Therefore, one has to balance the model complexity, computational costs, and model overfitting. Fortunately, the 'universal approximation' result suggests that multilayer feed-forward neural network architectures with enough hidden layers and neurons can represent a wide variety of smooth functions (Hornick, Stinchcombe, & White, 1989). Hence, to keep model estimation feasible while retaining flexibility, we fix some of the hyper-parameters and hyper-tune others based on the validation subsample.

In particular, we use the same Gaussian error linear unit (GELU) activation function for all hidden layers, except for the coder and the output layers for which we use the simple linear activation to avoid restrictions. The GELU often performs similarly to the popular ReLU activation function. However, it allows for a small, non-zero gradient when the input in the neuron is negative, circumventing the 'dying ReLU' problem, which, in turn, results in a potentially more robust optimization process (Maas, Hannun, Ng, et al., 2013). The results are generally robust to the choice of activation functions.

To allow for flexibility and complexity in the models, we consider architectures with up to seven hidden layers (excluding the coder part) in the encoder and decoder of AE models. The number of neurons in each of the hidden layers is considered according to the geometric pyramid rule (Masters, 1993). In particular, for an AE with $d_{l^*} = d$ neurons in the coder (which corresponds to the number of factors) and m dimensional output vector, the number of neurons d_l in the decoder is chosen according to the following rule:

$$d_l = \frac{m}{s_d^{(L-l^*-(l-l^*-1))}} = \frac{m}{s_d^{L-l+1}}, \quad l > l^*,$$

where $L - l^*$ is the number of hidden layers in the decoder and s_d is the decoder's scaling that we allow taking values of 1.2, 1.4, 1.6, 2, and 3. For the encoder part, we apply a mirrored rule for the number of neurons:

$$d_l = \frac{n}{s_e^l}, \quad l < l^*,$$

where s_e is the encoder's scaling from the same set of possible values as s_d. The number of hidden layers and the scaling values are hyper-tuned using the random search hyper-parameter optimization technique.

Due to the large number of parameters and non-linearities in the neural network architectures, the estimation of these models becomes a non-trivial task. We adopt the widely used Adam algorithm, introduced by Kingma and Ba (2014). The Adam algorithm is an extension of the stochastic gradient descent algorithm for the learning of neural networks that adaptively adjusts the learning rate based on the first and second moments of the gradients. To prevent overfitting, we also employ the 'early stopping' algorithm as one of the regularization tools and adopt a learning rate reduction once the validation metric starts deteriorating.

Finally, for selected architectures, we adopt an ensemble averaging approach to mitigate the stochastic nature of the optimization algorithms. Specifically, we retrain the selected architectures ten times using different random seeds to initialize the AEs. The final predictions are then obtained as the average of the individual predictions from these ten models.

3.3 Estimating parametric option pricing models

As discussed earlier, applying the PCA and AE methodologies to the panel of option-implied log CCFs allows us to compare affine and non-affine classes of option pricing models in a non-parametric fashion. On the other hand, it can also be of interest to quantify the differences our non-parametric approaches and popular parametric models. In particular, we can analyze how well a parametric AJD option pricing model explains the data compared to the non-parametric affine model given by PCA. Naturally, for a fair comparison, the estimation of the parametric models also needs to be conducted on

the same log CCF space. For that, one possibility would be to use the quasi-maximum likelihood estimation procedure of Boswijk et al. (2022) for AJD models based on the Kalman filter. However, since neither PCA nor AE directly exploit time-series dependence (while the Kalman filter does), the comparable procedure would be to simply calibrate the parametric model on the log CCF space by minimizing the same type of mean squared errors as PCA and AE.

More specifically, for a certain parametric option pricing model within the AJD class, the coefficients in equation (5) have a structural form as functions of the model parameter vector θ, i.e.,

$$\widehat{\Psi}_t = w_\alpha(\theta) + w_\beta(\theta) X_t + \varepsilon_t, \tag{8}$$

where the structural coefficients $w_\alpha(\theta)$ and $w_\beta(\theta)$ are obtained by stacking the model-dependent coefficients $\alpha(u, \tau; \theta)$ and $\beta(u, \tau; \theta)$ in a similar manner as the log CCF. To estimate the parameters of the model, we can minimize the difference between the model-dependent log CCF and model-free option-implied log CCF:

$$\min_{\theta, \{X_t\}_{t=1,\ldots,T}} \sum_{t=1}^{T} ||\widehat{\Psi}_t - w_\alpha(\theta) - w_\beta(\theta) X_t||^2, \tag{9}$$

where $||x||$ is the Euclidean norm of vector x. This is akin to minimizing the difference between model-implied and market-observed option prices (often quoted in terms of their implied volatilities), which is commonly done in the literature (see, e.g., Bakshi, Cao, & Chen, 1997 and Broadie et al., 2007). However, a significant advantage of minimizing differences between the log CCFs is that this difference is linear in the latent state vector X_t. Therefore, for each time t, we can easily concentrate the unobserved state out by solving the ordinary least squares (OLS) problem. That is, the minimization problem results in:

$$\min_{\theta} \sum_{t=1}^{T} ||\widehat{\Psi}_t - w_\alpha(\theta) - w_\beta(\theta)\hat{X}_t||^2, \tag{10}$$

where $\hat{X}_t = (w_\beta(\theta)' w_\beta(\theta))^{-1} w_\beta(\theta)'(\widehat{\Psi}_t - w_\alpha(\theta))$ is the OLS estimate at time t for a given parameter set θ. The minimization problem (10) is computationally less expensive than estimation using option prices since it does not require estimation of the latent parameter for each time point and does not involve the pricing of options given the model parameters.

We conclude by noting that both PCA and AE models are trained by minimizing the mean squared error (MSE), which is equivalent to minimizing the criterion function in (9) for parametric models. When evaluating the different non-parametric and parametric models, we will assess their MSE for predicting the option-implied log CCF in the testing subsample. That is, the estimation and evaluation criteria are completely aligned.

4 Data

We use daily options data on the S&P 500 stock market index obtained from OptionMetrics. These options are the most actively traded European contracts and have been widely used in the academic literature. We consider the period of more than 12 years from January 2011 to February 2023, covering 3046 trading days, including the turbulent period during the Covid-19 pandemic.

To construct our options data samples, we first apply a few data filters. In particular, we keep those option observations that meet the following criteria: *(i)* positive bid price; *(ii)* maturity is larger than or equal to 1 calendar day; *(iii)* it is not an early-closure day. These rather generous criteria filter out illiquid observations, ultra-short maturities, and shortened trading sessions. We then use mid-quotes data as option observations and determine the moneyness level based on the implied forward price. The latter is calculated as the median of the five forward prices implied from the closest to at-the-money option pairs based on the put-call parity. The risk-free rates are calculated by interpolating the LIBOR rates to each maturity.

To ensure the construction of a reliable option-implied CCF approximation, it is essential to have option observations that span a wide range of strike prices. Therefore, for each specific maturity, we employ an interpolation-extrapolation technique to reduce approximation errors following Boswijk et al. (2022). Specifically, we use cubic splines with carefully selected knot sequence to interpolate option prices. This reduces the discretization errors in the construction of the option-implied CCFs. We also extrapolate beyond the observable range of strikes to mitigate the impact of truncation errors. For that we use a parametrization that satisfies the arbitrage-free conditions and the asymptotic result of Lee (2004). By applying the interpolation-extrapolation scheme, we are able to incorporate a sufficiently broad spectrum of strike prices.

To construct a standardized and balanced panel of input and output option data, we consider five fixed tenors with maturities of 10, 30, 60, 120, and 180 days and interpolate the available options data to these selected maturities. In particular, for each of the five tenors, we identify two option slices with tenors just below and above the chosen maturity. After having applied the interpolation-extrapolation technique for these two volatility slices, we linearly interpolate between these slices in total Black-Scholes implied variance, $\omega(\tau, k) := \sigma^2_{BS}(\tau, k)\tau$, to τ for each strike price. The interpolation to the selected maturities using the total implied variance is similar to the construction of the VIX based on near-term and next-term options (CBOE, 2015).

The option-implied CCF for each day and each selected maturity are calculated based on the Riemann sum approximation applied to the result of the interpolation in moneyness and maturity dimensions. We use the CCF arguments of the following form:

$$u_j = \frac{jL}{\sqrt{\tau}}, \quad j = 1, \ldots, q,$$

where q is the number of CCF arguments and $L > 0$ is some constant that defines the grid of the arguments. In what follows, we set $L = 0.5$ and consider $q = 10$ CCF arguments, resulting in the cross-sectional input dimension $n = 100$. We also vary q for the robustness check. The division by $\sqrt{\tau}$ homogenizes different maturities, where a similar scaling is used by Todorov (2019). Figure 2 plots the time series dynamics of the at-the-money

(ATM) implied volatilities (IVs) and the log of option-implied CCF for the five considered tenors. The variables reflect similar dynamics, spiking during periods of market distress, such as the Covid-19 pandemic. However, it can be seen that co-movements of different tenors of the log CCFs are closer to linear than for IVs.

5 Empirical analysis

5.1 Model evaluation

Having constructed the option-implied log CCF data, we now proceed by estimating the affine (PCA) and non-affine (AE) non-parametric option pricing models with different numbers of latent state components d. For the training and assessment of each of the models, we first split the data into training, validation, and testing subsamples. Since our goal is to understand whether non-affine models can improve the explanatory power for options data, rather than forecast into the future, we consider a non-chronological split of the data. We adopt the following scheme: for each week, we randomly select one day for the validation subsample, one day for the testing, and the remaining days for the training subsample. This ensures that all three subsamples cover the same time period and approximately have the same distribution.[4]

To measure the predictive improvement of the AE relative to the PCA with the same number of factors, we consider the partial R^2 defined as:

$$R^2_{AE/PCA} := 1 - \frac{MSE_{AE}}{MSE_{PCA}},$$

where MSE_{AE} and MSE_{PCA} are the mean squared errors for AE and PCA, respectively. To assess the statistical significance of differences between the AE and PCA models, we apply the Diebold and Mariano (2002) test. Following Gu et al. (2020), we define the

[4]This is not necessarily the case with the chronological split. For instance, including the high volatility regime during the Covid-19 pandemic in the test but not in the training or validation subsamples, would likely lead to different distributions of the subsamples. This, in turn, would not help us compare two models in terms of their ability to explain the option data, but rather would focus on their ability to forecast into the high volatility regimes.

forecast loss differential as:

$$d_{12,t} = \frac{1}{m}\sum_{i=1}^{m}\left(\widehat{\varepsilon}_{i,t}^{(PCA)}\right)^2 - \left(\widehat{\varepsilon}_{i,t}^{(AE)}\right)^2,$$

where m is the dimension of the cross-section of output log CCF, and $\widehat{\varepsilon}_{i,t}^{(AE)}$ and $\widehat{\varepsilon}_{i,t}^{(PCA)}$ denote the out-of-sample prediction errors for the AE and PCA models, respectively. The Diebold-Mariano test statistic is then applied to this differential, i.e., $DM_{12} = \bar{d}_{12}/\widehat{\sigma}_{\bar{d}_{12}}$ with $\bar{d}_{12}$ being the sample mean of $d_{12,t}$ and $\widehat{\sigma}_{\bar{d}_{12}}$ the Newey-West standard deviation of $\bar{d}_{12}$. Under the null hypothesis of no differences between the two models, the Diebold-Mariano test statistic is standard normally distributed.

Finally, to compare multiple models on the same basis, we consider the explained variance (EV) metric. This is similar to the PCA literature, but applied to the test sample only, that is:

$$EV = 1 - \frac{MSE}{TV},$$

where TV is the total variation of the test sample and MSE is the mean squared error for a particular model. Note that the EV is simply the partial R^2 with respect to the naive sample mean prediction.

5.2 Empirical results

Panel (a) of Table 1 reports the partial R^2 of AE relative to PCA for different numbers of factors d. Focusing first on the results for the entire testing subsample, the R^2's are always positive and at least 50%, regardless of the number of factors. The largest improvement coming from the non-affine class relative to the affine class is for two-factor models. Breaking down the performance to specific maturity categories, the AE again outperforms the PCA for any number of factors, regardless of the category. To assess the statistical significance of these differences, Panel (b) provides the corresponding Diebold-Mariano test statistics. For the entire testing sample, the null hypothesis of equal predictive

accuracy is rejected at the 5% level for any number of factors, indicating that the observed outperformance of non-affine models is statistically significant. The evidence is the same across the different maturities, with the exception of the 30-day maturity category, for which the test statistics are smaller and, in particular, insignificant for $d = 2, 3, 4$. This suggests that most of the improvement of allowing for non-linearities with the AE comes from fitting the log CCF for relatively shorter and longer maturities.

Given that non-affine option pricing models outperform affine ones for the same number of latent factors, a natural question that arises is how many additional factors the affine model requires to be able to match the performance of the non-affine model with d factors. To address this question, Panel (a) of Table 2 displays the pairwise partial $R^2_{AE/PCA}$ results for all AE-PCA pairs with different number of factors.[5] As can be seen, to outperform a one-factor non-affine model, the affine model requires at least two factors. Similarly, as soon as the AE includes two (three) factors, PCA requires at least four (six) factors to beat the non-affine model. In other words, affine option pricing models require in general twice the number of factors to fit the option panel as well as the non-affine model. Panel (b) reports the corresponding Diebold-Mariano test statistics, which confirm the main takeaway from the table: to significantly outperform the AE with one, two and three factors, PCA requires three, four and seven factors, respectively.

We next investigate whether there is a clear preferable number of factors for each class of models. Panel (a) of Table 3 reports the partial R^2 of the AE with d factors relative to the AE with $d-1$ factors. The greatest relative improvement comes from adding the second factor to the model. While including additional factors always improves performance, the R^2 is decreasing in d. Patterns are similar across maturity categories. Including up to four factors improves the out-of-sample fit in similar magnitude across all maturities, while additional factors seem to be more relevant to specific categories. Panel (b) provides the corresponding Diebold-Mariano test statistics. Most of the observed improvements are statistically significant, with the exception of the introduction of the fifth factor. In sum, one needs at least four factors in the non-affine option pricing model

[5]The diagonal of this table corresponds to the results for the total testing subsample in Table 1.

for an unambiguously better fit of the option panel relative to lower dimensional models.

Table 4 contains the analogous results for the PCA. Panel (a) reveals that, similarly to the AE, adding factors to the PCA model always improves performance, albeit at a decreasing rate. However, relative to the non-affine model, the affine model generally benefits more from additional factors, as seen from higher R^2's. Panel (b) shows that most improvements are statistically significant, except for the sixth factor. This is such that a five-factor model seems to be preferable for the affine class.

Finally, we also compare the out-of-sample performance of the non-parametric models we estimate to that of popular affine parametric option pricing models with one state factor. More specifically, we consider the models by Heston (1993), Bates (1996), and Pan (2002). To compare the models on the same basis, we compute their EV in the testing subsample, which is reported in Figure 3. As would be expected, the simpler Heston (1993) model has the worst performance. Adding jumps in returns with Bates (1996) model improves performance by 8%, while further incorporating stochastic jump intensity with Pan (2002) model yields an additional 3% improvement. Even so, the non-parametric affine model given by PCA still outperforms the best parametric model, with a 7% higher EV. This indicates that the considered parametric models do not capture all relevant information within the affine class. However, the main improvement in performance still comes from allowing for non-linearities in the log CCF with a non-affine model, where the AE explains 20% more variation in the option panel than the best parametric model.

6 Conclusion

In this paper, we provide a new framework allowing to estimate non-parametric affine and non-affine option pricing models. Our method combines autoencoder neural networks with the underlying return log-characteristic function implied by observed option prices. We are able to obtain a data-driven affine model by incorporating linearity in the autoencoder architecture between the log-characteristic function and the state factors driving

the underlying asset price dynamics. Alternatively, we estimate a non-affine model by letting the data speak about any needed non-linearities to extract the latent factors.

Using an extensive panel of S&P 500 option data, our approach reveals that non-affine models significantly outperform affine models in predicting the option panel out-of-sample. In particular, affine models generally require twice the number of factors to fit the option panel as well as non-affine models. Importantly, these findings do not depend on a specific parametrization, but rather rely on comparing the affine and non-affine classes in a non-parametric fashion. We also show that there is little empirical support for models with a low-dimensional set of state factors: increasing the number of factors, at least up until five, unambiguously yields better out-of-sample performance for both affine and non-affine models. Finally, we show that existing affine parametric option pricing models are outperformed by our affine non-parametric specification.

References

Ait-Sahalia, Y., & Kimmel, R. (2007). Maximum likelihood estimation of stochastic volatility models. *Journal of Financial Economics*, *83*(2), 413-452.

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

Andersen, T. G., Fusari, N., & Todorov, V. (2015a). Parametric inference and dynamic state recovery from option panels. *Econometrica*, *83*(3), 1081–1145.

Andersen, T. G., Fusari, N., & Todorov, V. (2015b). *The pricing of short-term market risk: Evidence from weekly options* (Tech. Rep.). National Bureau of Economic Research.

Andersen, T. G., Fusari, N., & Todorov, V. (2015c). The risk premia embedded in index options. *Journal of Financial Economics*, *117*(3), 558-584.

Bakshi, G., Cao, C., & Chen, Z. (1997). Empirical performance of alternative option pricing models. *The Journal of finance*, *52*(5), 2003–2049.

Bates, D. S. (1996). Jumps and stochastic volatility: Exchange rate processes implicit in deutsche mark options. *The Review of Financial Studies*, *9*(1), 69–107.

Bates, D. S. (2000). Post-'87 crash fears in the S&P 500 futures option market. *Journal of Econometrics*, *94*(1-2), 181-238.

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

Boswijk, H. P., Laeven, R. J., Marijnen, N., & Vladimirov, E. (2023). Characteristic function-based factor modelling of affine jump diffusions using options. *Working Paper*.

Boswijk, H. P., Laeven, R. J., & Vladimirov, E. (2022). Estimating option pricing models using a characteristic function-based linear state space representation. *Tinbergen Institute Discussion Paper 2022-075/III*.

Broadie, M., Chernov, M., & Johannes, M. (2007). Model specification and risk premia: Evidence from futures options. *The Journal of Finance*, *62*(3), 1453–1490.

Carr, P., & Madan, D. (2001). Optimal positioning in derivative securities. *Quantitative Finance*, *1*(1), 19–37.

CBOE. (2015). *VIX white paper.* `https://cdn.cboe.com/resources/vix/vixwhite.pdf`.

Christoffersen, P., Jacobs, K., & Mimouni, K. (2010). Volatility dynamics for the S&P 500: Evidence from realized volatility, daily returns and option prices. *The Review of Financial Studies*, *23*(8), 3141–3189.

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

Duffie, D., Filipović, D., & Schachermayer, W. (2003). Affine processes and applications in finance. *The Annals of Applied Probability*, *13*(3), 984–1053.

Duffie, D., Pan, J., & Singleton, K. (2000). Transform analysis and asset pricing for affine jump-diffusions. *Econometrica*, *68*(6), 1343–1376.

Dugas, C., Bengio, Y., Bélisle, F., Nadeau, C., & Garcia, R. (2009). Incorporating functional knowledge in neural networks. *Journal of Machine Learning Research*, *10*, 1239–1262.

Eraker, B., Johannes, M., & Polson, N. (2003). The impact of jumps in volatility and returns. *The Journal of Finance*, *58*(3), 1269-1300.

Garcia, R., & Gençay, R. (2000). Pricing and hedging derivative securities with neural networks and a homogeneity hint. *Journal of Econometrics*, *94*(1-2), 93–115.

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.

Heston, S. L. (1993). A closed-form solution for options with stochastic volatility with applications to bond and currency options. *The Review of Financial Studies*, *6*(2), 327–343.

Hornick, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, *2*(5), 359–366.

Hutchinson, J. M., Lo, A. W., & Poggio, T. (1994). A nonparametric approach to pricing and hedging derivative securities via learning networks. *The Journal of Finance*, *49*(3), 851–889.

Jones, C. S. (2003). The dynamics of stochastic volatility: Evidence from underlying and options markets. *Journal of Econometrics*, *116*(1-2), 181-224.

Kelly, B., 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*.

Lee, R. W. (2004). The moment formula for implied volatility at extreme strikes. *Mathematical Finance: An International Journal of Mathematics, Statistics and Financial Economics*, *14*(3), 469–480.

Li, G., & Zhang, C. (2013). Diagnosing affine models of options pricing: Evidence from VIX. *Journal of Financial Economics*, *107*(1), 199-219.

Liu, S., Oosterlee, C. W., & Bohte, S. M. (2019). Pricing options and computing implied volatilities using neural networks. *Risks*, *7*(1), 1–22.

Maas, A. L., Hannun, A. Y., Ng, A. Y., et al. (2013). Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml* (Vol. 30, p. 3).

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

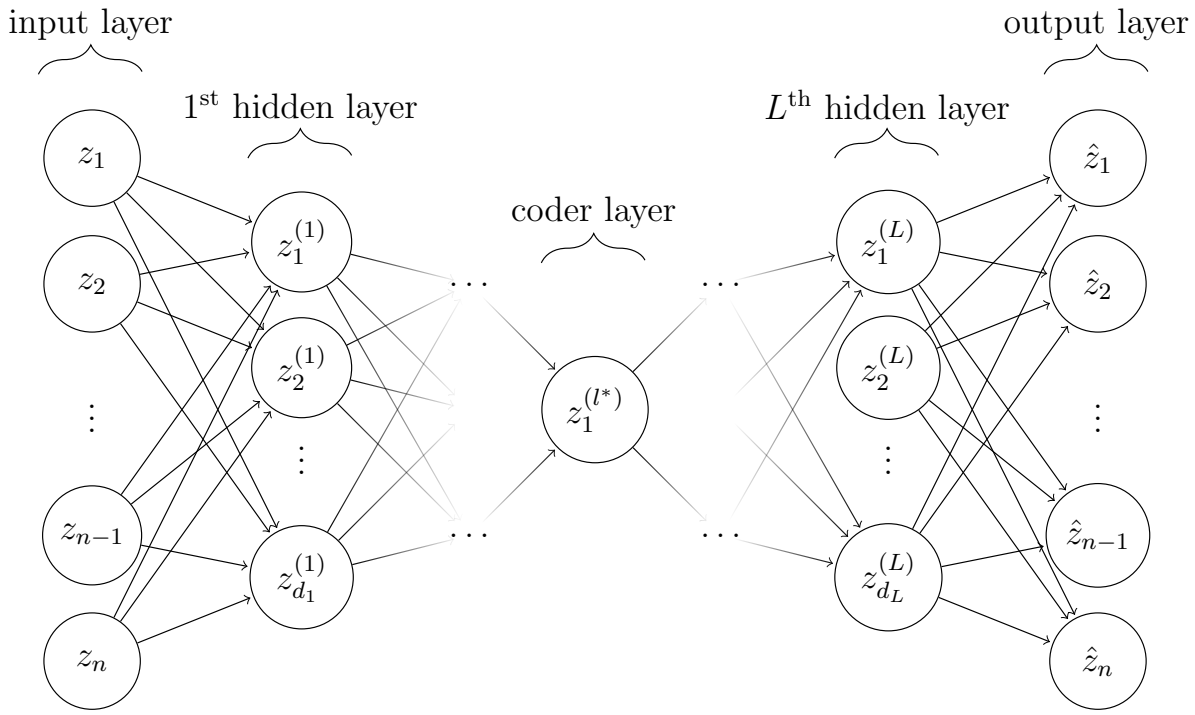
Pan, J. (2002). The jump-risk premia implicit in options: Evidence from an integrated time-series study. *Journal of Financial Economics*, *63*(1), 3–50.

Ruf, J., & Wang, W. (2022). Hedging with linear regressions and neural networks. *Journal of Business and Economic Statistics*, *40*(4), 1442–1454.

Todorov, V. (2019). Nonparametric spot volatility from options. *The Annals of Applied Probability*, *29*(6), 3590–3636.

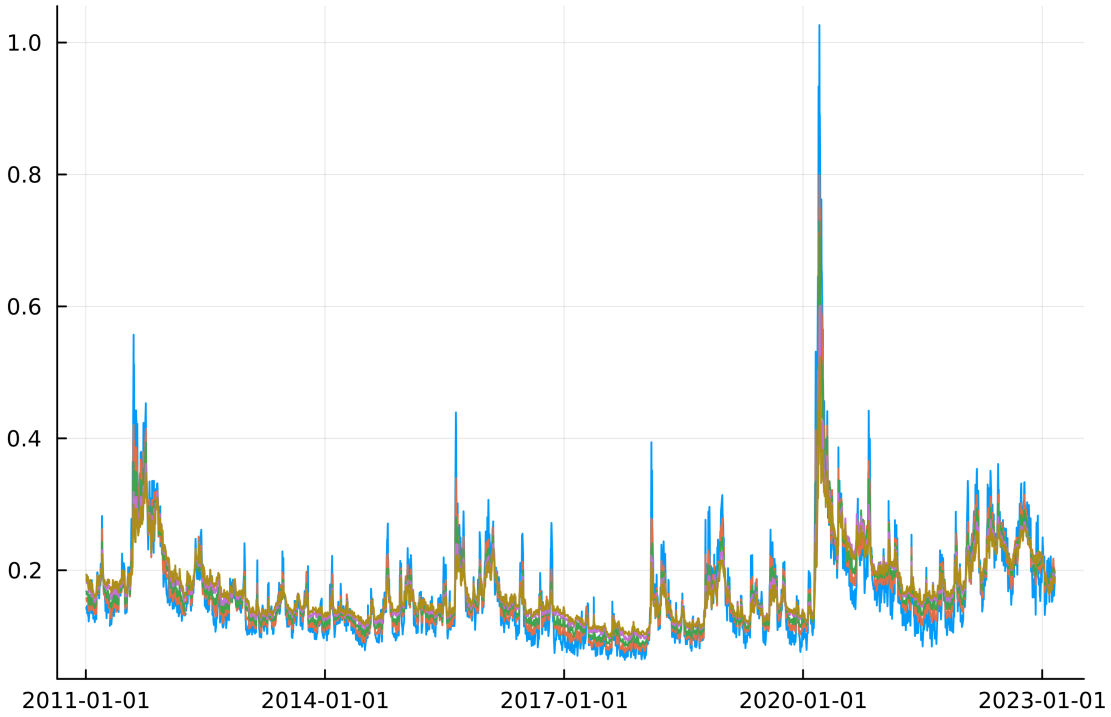
Appendix A Figures and Tables

Figure 1: Standard autoencoder with one factor

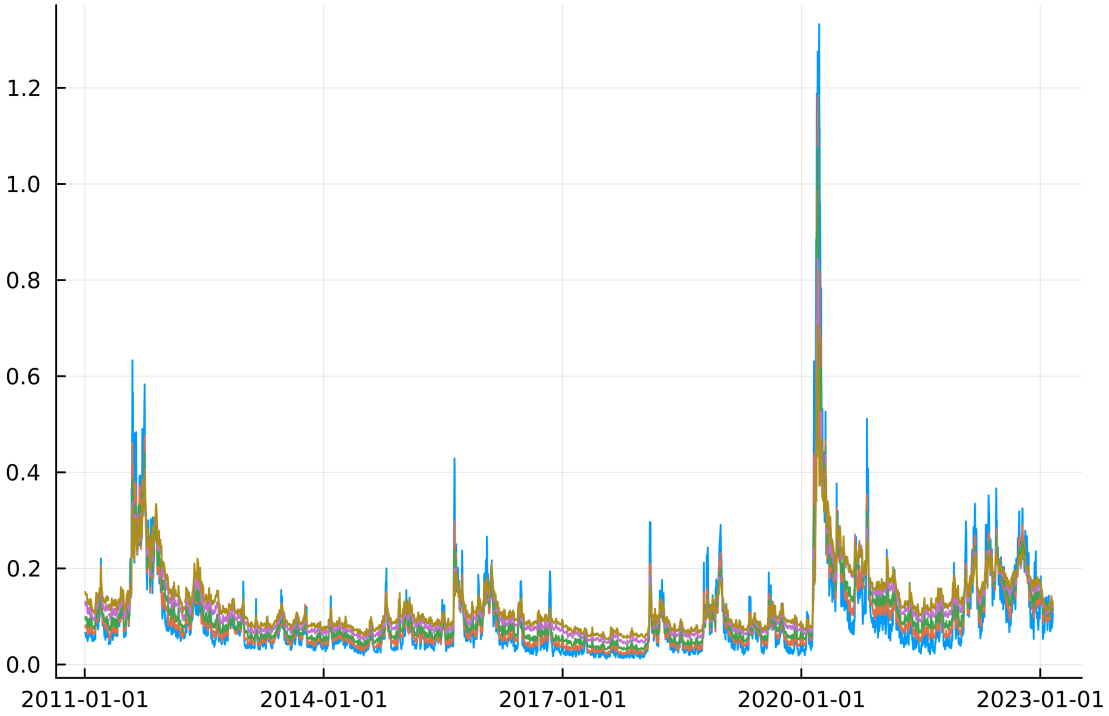


Note: This figure depicts a representation of the standard autoencoder neural network with one latent factor and an arbitrary number of hidden layers (L) and neurons in each layer (d_l).

Figure 2: Time series of IV and log CCF



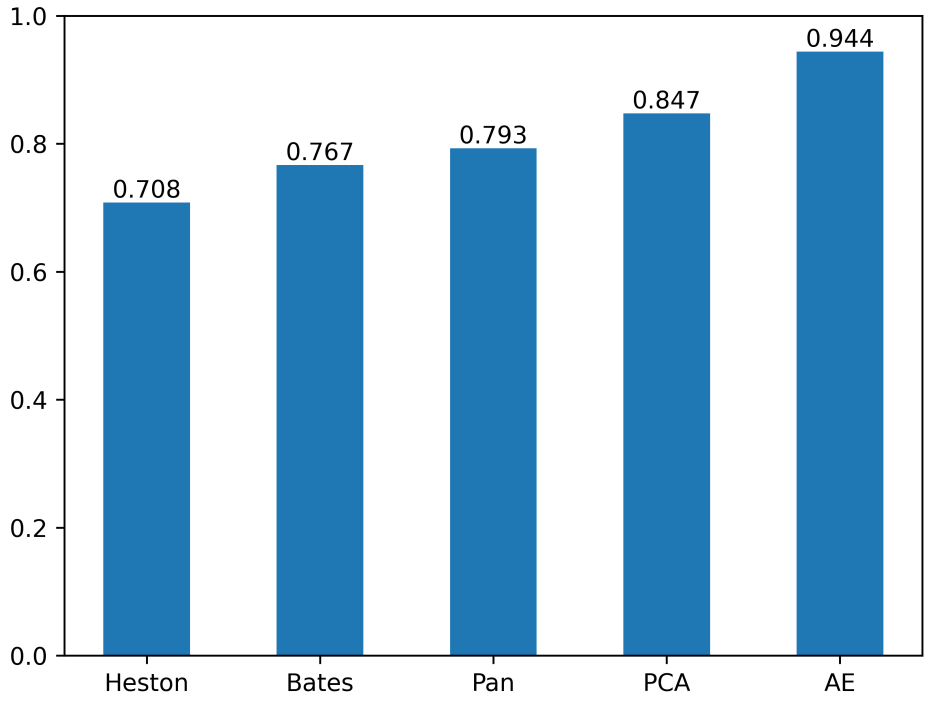
(a) ATM IV



(b) log CCF, $\Re\left(\widehat{\psi}_t(u_5)\right)$

Note: This figure plots the time series dynamics of the at-the-money IVs and the log of option-implied CCF for the five fixed tenors with 10, 30, 60, 120, and 180 days. For the log CCF, we plot only the real parts, and argument $u_5 = 2.5/\sqrt{\tau}$.

Figure 3: Out-of-sample Explained Variance for 1-factor models



Note: This figure plots the Explained Variance in the testing subsample, as defined in Section 5.1, associated with the one-factor parametric AJD option pricing models (Heston, 1993; Bates, 1996; Pan, 2002) and one-factor non-parametric models (PCA and AE).

Table 1: Test sample results for AE vs. PCA for same number of factors

(a) $R^2_{AE/PCA}(\%)$

$\tau \backslash d$	1	2	3	4	5	6	7
total	63.29	66.92	66.09	61.83	56.94	55.11	49.85
10	61.09	63.31	73.69	61.84	39.24	53.67	53.21
30	64.34	24.80	37.43	49.56	69.62	58.33	40.02
60	65.77	46.12	57.70	66.62	62.76	60.22	43.45
120	63.27	79.50	71.30	64.53	37.11	37.58	49.86
180	64.05	82.81	72.82	71.39	62.64	58.69	58.33

(b) Diebold-Mariano test statistics

$\tau \backslash d$	1	2	3	4	5	6	7
total	**2.43**	**4.87**	**3.49**	**4.15**	**3.25**	**5.85**	**6.69**
10	1.76	**4.20**	**2.03**	**2.83**	**3.01**	**4.26**	**2.35**
30	**2.23**	1.77	1.53	1.93	**2.11**	**4.15**	**4.35**
60	**2.70**	**2.89**	**3.73**	**5.74**	**4.21**	**5.61**	**4.74**
120	**3.46**	**3.99**	**3.68**	**4.29**	**2.19**	**5.00**	**6.10**
180	**3.79**	**5.73**	**5.31**	**4.50**	**3.59**	**5.52**	**6.73**

This table provides the partial $R^2_{AE/PCA}(\%)$ (Panel a) and the Diebold-Mariano test statistics (Panel b) based on the test sample for the different number of factors d. The row 'Total' reports the results for the whole test sample, while the other rows are for the subsamples with different maturitiy levels. Bold font in Panel (b) indicates the significance at 5% level.

Table 2: Pairwise comparison of AE vs. PCA for different numbers of factors

(a) Pairwise $R^2_{AE/PCA}(\%)$

AE\PCA	1	2	3	4	5	6	7
1	63.29	-9.25	-130.47	-330.87	-541.41	-817.73	-1157.16
2	88.88	66.92	30.22	-30.46	-94.21	-177.88	-280.66
3	94.60	83.92	66.09	36.60	5.62	-35.05	-84.99
4	96.75	90.32	79.58	61.83	43.18	18.70	-11.37
5	97.54	92.67	84.53	71.08	56.94	38.39	15.61
6	98.20	94.66	88.73	78.93	68.63	55.11	38.51
7	98.54	95.64	90.81	82.81	74.42	63.39	49.85

(b) Pairwise Diebold-Mariano test statistics

AE\PCA	1	2	3	4	5	6	7
1	**2.43**	-0.65	**-3.33**	**-3.40**	**-3.34**	**-3.33**	**-3.38**
2	**2.65**	**4.87**	**2.32**	**-2.42**	**-2.99**	**-3.27**	**-3.51**
3	**2.73**	**5.14**	**3.49**	**2.94**	0.39	-1.51	**-2.30**
4	**2.77**	**5.28**	**3.87**	**4.15**	**3.13**	1.21	-0.45
5	**2.75**	**4.96**	**3.49**	**3.41**	**3.25**	**4.99**	**2.25**
6	**2.75**	**4.94**	**3.52**	**3.50**	**3.59**	**5.85**	**6.18**
7	**2.76**	**4.94**	**3.55**	**3.56**	**3.77**	**6.05**	**6.69**

This table provides the pairwise partial $R^2_{AE/PCA}(\%)$ (Panel a) and the pairwise Diebold-Mariano test statistics (Panel b) based on the full test sample for the different number of factors d in AE and PCA models. Bold font in Panel (b) indicates the significance at 5% level.

Table 3: Test sample results for AE(d) vs. AE($d-1$)

(a) $R^2_{AE(d)/AE(d-1)}(\%)$

$\tau \backslash d$	2	3	4	5	6	7
total	69.72	51.40	39.80	24.22	27.14	18.45
10	66.54	61.72	30.02	28.01	26.97	26.12
30	64.97	42.73	34.23	40.59	36.94	11.77
60	69.27	47.06	46.60	11.77	21.87	3.64
120	77.69	44.10	47.62	-9.85	28.82	25.52
180	74.68	47.73	53.95	16.26	16.07	19.14

(b) Diebold-Mariano test statistics

$\tau \backslash d$	2	3	4	5	6	7
total	**2.63**	**3.29**	**3.59**	1.07	**3.09**	**3.38**
10	**2.38**	**2.35**	**3.46**	**1.98**	**2.10**	1.15
30	**2.42**	**3.07**	**3.13**	1.09	**3.05**	**3.38**
60	**3.04**	**3.27**	**2.40**	0.73	**4.37**	0.22
120	**4.32**	**4.39**	**2.74**	-0.64	**2.14**	**3.23**
180	**5.51**	**6.90**	**4.70**	1.09	**2.62**	1.03

This table provides the partial $R^2_{AE(d)/AE(d-1)}(\%)$ (Panel a) and the Diebold-Mariano test statistics (Panel b) for AE(d) against AE($d-1$) based on the test sample. The row 'Total' reports the results for the whole test sample, while the other rows are for the subsamples with different maturitiy levels. Bold font in Panel (b) indicates the significance at 5% level.

Table 4: Test sample results for PCA(d) vs. PCA($d-1$)

(a) $R^2_{PCA(d)/PCA(d-1)}(\%)$

	2	3	4	5	6	7
total	66.40	52.60	46.51	32.82	30.11	27.00
10	64.52	46.62	51.75	54.78	4.22	26.84
30	83.39	31.16	18.43	1.34	54.03	38.71
60	80.47	32.56	32.34	20.90	26.86	32.21
120	60.03	60.06	57.62	38.05	28.28	7.29
180	47.07	66.93	56.25	35.87	24.10	19.83

(b) Diebold-Mariano test statistics

	2	3	4	5	6	7
total	**2.08**	**4.47**	**2.78**	**2.24**	1.92	**3.66**
10	1.59	**6.14**	**1.96**	**1.96**	0.55	**5.56**
30	**2.19**	**7.43**	**3.05**	0.30	1.83	**3.80**
60	**2.73**	**2.54**	**2.21**	**4.11**	**3.14**	**4.01**
120	**3.35**	**4.21**	**4.06**	**3.63**	**2.31**	**2.02**
180	**3.15**	**5.54**	**6.31**	**3.29**	**2.14**	**5.07**

This table provides the partial $R^2_{PCA(d)/PCA(d-1)}(\%)$ (Panel a) and the Diebold-Mariano test statistics (Panel b) for PCA(d) against PCA($d-1$) based on the test sample. The row 'Total' reports the results for the whole test sample, while the other rows are for the subsamples with different maturitiy levels.