Quantitative Finance and Economics

http://www.aimspress.com/journal/QFE

QFE, 7(2): 207–228.
DOI: 10.3934/QFE.2023011
Received: 01 March 2023
Revised: 20 April 2023
Accepted: 23 April 2023
Published: 04 May 2023

Research article

Accelerated American option pricing with deep neural networks

David Anderson[1] **and Urban Ulrych**[1,2,3,*]

[1] Department of Banking and Finance, University of Zurich, Zurich, Switzerland
[2] College of Management of Technology, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland
[3] Swiss Finance Institute, Geneva, Switzerland

* **Correspondence:** Email: urban.ulrych@epfl.ch; Tel: +41216932466.

Abstract: Given the competitiveness of a market-making environment, the ability to speedily quote option prices consistent with an ever-changing market environment is essential. Thus, the smallest acceleration or improvement over traditional pricing methods is crucial to avoid arbitrage. We propose a method for accelerating the pricing of American options to near-instantaneous using a feed-forward neural network. This neural network is trained over the chosen (e.g., Heston) stochastic volatility specification. Such an approach facilitates parameter interpretability, as generally required by the regulators, and establishes our method in the area of eXplainable Artificial Intelligence (XAI) for finance. We show that the proposed deep explainable pricer induces a speed-accuracy trade-off compared to the typical Monte Carlo or Partial Differential Equation-based pricing methods. Moreover, the proposed approach allows for pricing derivatives with path-dependent and more complex payoffs and is, given the sufficient accuracy of computation and its tractable nature, applicable in a market-making environment.

Keywords: American option pricing; deep neural networks; explainable artificial intelligence; parametric option pricing; speed-accuracy trade-off; market making

JEL Codes: C45, C63, G13

1. Introduction

Equity derivative market-making environments are known for their competitive pricing requirements in terms of both speed and accuracy. Large fluctuations of the underlying security price, together with other pricing parameters, can occur fast and without warning and are influenced by company news, as well as macroeconomic, financial, and political factors. If market makers cannot adjust the corresponding option prices instantaneously, arbitrage opportunities can occur.

There are many reasons why a market maker could be slow in updating the price of a product. Firstly, high overall system latency can potentially cause a delay in the arrival of market data to the trading system. Secondly, the calibration process of an implied volatility surface (e.g., using a chosen parametric volatility model such as the Heston stochastic volatility model) based on new market price arrivals can represent another potential source of latency. The third step, however, is typically the most prevalent source of computational latency. Here, current market data and the corresponding fitted parametric volatility model are inputted into an algorithm to evaluate the price of a derivative. Depending on the option type, the computational performance of a pricer is affected by its underlying algorithm and the complexities in the payoff structure of the option.

This paper proposes a method that accelerates the American-style option pricing mechanism. We do not investigate the closely connected data acquisition, calibration, and price publishing parts. We assume that a trading system gathers market data and other parameters during the data acquisition and calibration phases and provides them to the pricing engine. We focus on the pricing algorithm that calculates what is known as the fair price of the option. This is the price at which no theoretical arbitrage exists – the probabilistic expectation of a profit on both sides of a transaction is equal to zero. In practice, a bid-ask spread is then generated around the computed fair price, and both bid and ask prices are subsequently published on the market platform.*

Since the main objective of the proposed method is its adoption in market making, a key responsibility of the proposed algorithm lies in its compliance with regulatory guidelines. Even though we use neural networks to accelerate American option pricing, we ensure that our algorithm is not a black box. This is achieved by generating a training set using a standard well-known pricing method (e.g., the PDE-based pricer under the Heston stochastic volatility model) instead of alternative neural network-based methods employing market data in the training set. Subsequently, we utilize the generated training set for training a neural network over the assumed (e.g., Heston-based) parameter space. By training the neural network over a set of all possible parameter space configurations given a chosen parameter range offline, such an approach allows for: i) decreasing the latency in the pricing of American options to near-instantaneous and ii) enhancing the transparency and explainability of the American option pricing algorithm. Furthermore, we show that the prices resulting from the proposed deep explainable pricer are sufficiently smooth with respect to the input parameters–one of the algorithmic criteria in eXplainable Artificial Intelligence.

In pricing path-dependent options, conventional methods rely extensively on computationally expensive numerical techniques. The most common pricing approaches are binomial models, numerical partial differential equation (PDE) methods, and Monte Carlo (MC) methods. Since the difficulties in pricing American options lie in the question of how to evaluate optimal exercise, the key differences between the methods arise from how they approach the optimal exercise decision. Cox et al. (1979) proposed the initial binomial tree method, which is fast and reasonably accurate. However, it does not work in the case of stochastic volatility. Furthermore, a trinomial tree that converges faster than the binomial tree was developed by Boyle (1986). Tree models rely on constant volatility and determine the optimal exercise by comparing the immediate exercise value and the discounted probability-weighted cashflows at each node.

* Depending on the current inventory (i.e., risk position) of the market-making trading desk that is pricing the option, the trading system can allocate aggressiveness to either the bid or the ask side by adjusting the spread around the fair price. Aggressiveness settings allow a trading desk to reduce its risk without hedging by attracting a specific direction of transactions.

To generalize to the non-constant volatility models, researchers developed methods based on numerical solutions to the Black-Scholes-Merton PDE, which holds in general for both European and American option styles. PDE methods for pricing American options admit the representation of the optimal exercise decision as a PDE boundary condition. The PDE with its boundary condition is solved numerically using various numerical solvers such as modified finite difference, finite element, and boundary projection methods. One of the pioneers in this area were Brennan and Schwartz (1977), who priced American put options using a boundary projection method. Later, Ikonen and Toivanen (2004) proposed an approach using the Crank-Nicolson method (i.e., a type of finite difference method, see Crank and Nicolson (1947)), which was found to be more efficient than the boundary projection method. Nevertheless, PDE methods are generally too slow for a market-making environment, especially in the presence of stochastic interest rates, stochastic volatility models, jump processes, multi-dimensional options, or derivatives with other complicated features.

Furthermore, researchers analyzed the use of statistical simulation methods known as Monte Carlo (MC) methods for option pricing. These methods are slower than the other methods mentioned above. However, they proved to be more flexible in terms of their ability to price derivatives under different volatility dynamics, more complex path-dependent payoff structures, and further departures from the basic Black et al. (1973) framework. Initial MC methods for American option pricing suffered under the curse of dimensionality since the decision process of optimal exercise relied on performing an additional MC simulation for each in-the-money path as proposed by Boyle et al. (1997). Thus, the number of paths increased exponentially. Famously, Longstaff and Schwartz (2001) proposed an MC method where the exercise decision relies upon comparing the current exercise payoff and a least-squares approximation of the option continuation value. This method circumvents the dimensionality problem of the nested MC approach and is thus highly effective. However, the Longstaff-Schwartz (LS) method is too slow for a market-making environment. Recently, the area of rough volatility models has evolved, see Bayer et al. (2016). Due to the non-Markovian dynamics of rough volatility models, pricing with Monte Carlo methods is required.

The use of artificial neural networks (ANNs) for option pricing started with Malliaris and Salchenberger (1993) and Hutchinson et al. (1994). The authors show that the Black-Scholes-based European-style option valuation can be successfully recovered with a neural network. These works serve as a valuable reference point that spurred a wide range of future research involving machine learning methods applied to option pricing and hedging.[†] Lajbcygier and Connor (1997) study the difference between hybrid neural network-based option valuation and observed market option prices and show that the pricing bias can be reduced with bootstrap methods. Anders et al. (1998) show that applying statistical inference techniques to build neural network pricing models leads to superior out-of-sample pricing performance compared to the Black-Scholes model. Garcia and Gençay (2000) incorporate financial domain knowledge in the form of a homogeneous option pricing function directly into the construction of an ANN pricer and show that such an approach reduces the out-of-sample mean squared prediction error. On a similar note, Dugas et al. (2009) show that the generalization performance of a learning algorithm can be improved by incorporating prior knowledge into its architecture. The authors propose new types of function classes that incorporate the a priori constraints based on the knowledge that the function to be learned is non-decreasing in its two arguments and convex in one of them. Pricing of S&P-500 European call options with a non-parametric modular neural network model is investigated in Gradojevic et al. (2009).

[†] An extensive literature review of the applications of neural networks to option pricing is provided in Ruf and Wang (2020).

Neural networks have also been studied in the context of option hedging as well as implied volatility smile and its calibration. The European-style option hedging is investigated in Chen and Sutcliffe (2012) and Shin and Ryu (2012). More recently, Buehler et al. (2019) present a deep hedging-based framework for hedging a portfolio of derivatives in the presence of market frictions. The implied volatility smile is, in a GARCH-neural network approach, studied in Meissner and Kawano (2001). Computation of implied volatilities is, among others, also studied in Andreou et al. (2010) and Liu et al. (2019b). More recently, Horvath et al. (2021) propose an application of neural networks in a two-step process where firstly, the European option pricing map is learned, and secondly, the European option implied volatility surfaces are calibrated to various parametric models. Deep learning calibration is studied also in Hernandez (2016), Ferguson and Green (2018), Liu et al. (2019a), and Itkin (2019).

The literature presented up to now studies an application of ANNs to the pricing of European-style options. Additionally, applying machine learning methods to solving the American option pricing and hedging problem has already been explored.[‡] Kelly (1994) presents the first study of pricing and hedging American put options using neural networks. A similar analysis is presented also in Pires and Marwala (2004) and Morelli et al. (2004). More recently, Gaspar et al. (2020) use neural networks to price American put options on four large U.S. companies. Generally, the American option pricing approaches fall into the categories of i) regression-enhancement for basket options as in the case of Kohler et al. (2010), ii) solving the optimal stopping problem as in the case of Becker et al. (2020), or iii) a mix of both as in Chen and Wan (2021). Moreover, machine learning methods can be used to approximate value functions emerging in dynamic programming, for example, arising in the American option pricing problem, see Ye and Zhang (2019). Similarly, Fecamp et al. (2019) apply ANNs to solve the pricing and hedging problem under market frictions such as transaction costs.

This paper contributes to the literature on option pricing by introducing a direct application of deep neural networks to price American-style options, which is relevant as the majority of the single stock options traded on the market are of the American style. In contrast to existing literature that focuses on European-style options, such as Horvath et al. (2021), we extend our methodology to American-style options by learning not only asset price dynamics but also the dynamics of a chosen volatility model, as well as the American option optimal exercise strategy, which has not been explored in the existing literature. One of the key advantages of our approach is that it learns the direct map between model parameters and price, providing a more interpretable and transparent approach than previous methods that rely on an implied volatility surface.

In addition to these contributions, our approach can be seen as an explainable method for pricing American-style options. By explicitly modeling the dynamics of the assumed volatility model and the corresponding optimal exercise strategy, our model provides a transparent and intuitive way to understand how American option prices are determined. Our results demonstrate that the proposed neural network pricer considerably improves upon the widely-used American-style option pricing models, such as PDE or Monte Carlo methods, in terms of both parameter interpretability and speed-accuracy trade-off.[§] Overall, our model offers a more interpretable, transparent, and computationally efficient approach for pricing

[‡] It should be noted that while our paper focuses on the pricing of American options, other types of options, such as exchange options, have been thoroughly explored in recent works (Pasricha and He, 2023; He and Lin, 2023a).

[§] Speed-accuracy trade-off in terms of derivative pricing, hedging, and fitting has also been confirmed by Spiegeleer et al. (2018). Moreover, Glau et al. (2020) extend the tensorized Chebyshev interpolation of conditional expectations in the parameter and state space and show that, at comparable accuracy, the interpolation in American option prices provides a promising gain in computational efficiency, leading to a speed-accuracy trade-off.

American-style options, which has the potential to be widely accepted by regulators and applicable in a market-making environment. In addition to its practical implications, our approach also opens up new avenues for future research in the field of Artificial Intelligence (AI) in finance.

In the last few years, methods from AI have been successfully applied in the field of finance. As more and more black-box approaches are being adopted, the demand for transparency from regulators, investors, developers, and managers has been increasing. This paper presents a contribution by proposing a method for pricing American-style options that falls within the field of eXplainable AI (XAI).[¶] The neural network, often considered a black-box algorithm, is, in our approach, applied solely as a function approximator that learns from a training set generated by a chosen (e.g., PDE-based) well-known pricing model. We show that by doing this, our pricer behaves in an interpretable, tractable, and trustworthy fashion. As such, our approach can be understood as a machine learning method that produces an accelerated and explainable option pricing algorithm while maintaining a high level of pricing accuracy. Model explainability is one of the crucial aspects that would ensure investors and regulators with understanding and trust to use the proposed machine learning pricing model in a market-making environment.

The remainder of the paper is organized as follows: Section 2 introduces the computational model and its underlying framework. In Section 3, the numerical results are presented and the speed-accuracy trade-off is analyzed. Finally, we conclude in Section 4.

2. Model

We propose a method where American-style option prices are generated using a standard pricing method, and the price data is subsequently utilized for training a neural network over the underlying parameter space. Since the option prices are constructed with a well-known pricing method, the neural network learns the pricing map by utilizing the generated training set that is by construction free of arbitrage. Essentially, this neural network provides the capability to learn the behavior of the American option price to changes within its parameter space. By generating our own training data and training the neural network offline (i.e., not during trading hours), the latency in the pricing of American options during trading hours can be reduced to near-instantaneous. Furthermore, the same neural network could be used for other purposes such as calculating implied volatilities or calculating the Greeks (i.e., a set of derivatives of an option with respect to various parameters), see Larguinho et al. (2022), at near-instantaneous speeds.

2.1. Deep Explainable Pricing Algorithm

We consider a typical parameter set prevalent in a derivative pricing framework $\Psi := (r, q, T - t, M, \psi_\sigma^{Model})$, where r denotes the interest rate, q the continuous dividend yield, $T - t$ the time-to-maturity, M the moneyness, and ψ_σ^{Model} the set of variables parameterizing a chosen volatility specification. A standard Black-Scholes-Merton setting represents a trivial case of constant volatility with $\psi_\sigma^{BS} := \sigma_0$, whereas under the Heston stochastic volatility model, the parameterization is denoted by $\psi_\sigma^{Heston} := \{v_0, \sigma, \kappa, \theta, \rho\}$, where the parameter definitions are provided in the model description below.

For the purpose of this study, the Heston stochastic volatility model was chosen as the basis for the proposed approach to pricing American-style options. This choice of a stochastic volatility model is motivated by the need to capture the dynamic nature of implied volatility, which is crucial for

[¶] An extensive review of the literature around XAI is provided in Arrieta et al. (2020).

pricing options with path-dependent payoffs. Although there are many stochastic volatility models in the literature, the Heston model is widely studied and well-understood, making it a popular choice in practice. Additionally, the Heston model is capable of reproducing some of the stylized facts present in observed financial returns, such as volatility clustering and mean reversion, see Cont (2001). However, the Heston model does have some limitations in fitting the short-end of the implied volatility surface, which can be addressed by including jumps (Xie et al., 2021) or a local volatility component (Farkas et al., 2022). Nonetheless, the Heston model remains a good choice for theoretical tractability and comparability with other studies in the literature. It is important to note that our proposed method can be extended to any parametric volatility model, including Lévy-type volatility models (Schoutens, 2003; Glasserman and Kim, 2011; Barndorff-Nielsen and Stelzer, 2013; Ascione et al., 2023; He and Lin, 2023b; Lin and He, 2023), as long as the volatility model is parametric. Therefore, the choice of the Heston model is primarily driven by its popularity and theoretical tractability, and our proposed method can be applied to a broad range of volatility models.

By computing American option prices using, for example, a PDE method on a dense grid across the parameter set Ψ, we generate a training set over which a neural network pricing functional is trained. As standard methods are used offline to generate the training data for the neural network, a major component of this paper is the accurate generation of the training data. Due to various universal approximation theorems (discussed below), it can be assumed that the neural network is able to approximate the option pricing dynamics to, at best, the precision of the underlying standard model-generated data. Thus, the error of the standard model data serves as a lower bound to the neural network error, illustrating the requirement for accurate standard model input data.

Definition 2.1 (Heston Stochastic Volatility Model)**.** Consider a probability space $\{\Omega, \mathcal{F}, \mathbb{Q}\}$ adapted to a filtration $\mathcal{F}_t$. Let S_t and v_t denote the asset price and instantaneous variance at time t. Then, according to the Heston model, as presented in Heston (1993), the asset price and volatility processes are, under the pricing measure $\mathbb{Q}$, defined by a system of stochastic differential equations (SDEs)

$$\begin{aligned} dS_t &= (r-q)S_t dt + \sqrt{v_t} S_t dW_t, \\ dv_t &= \kappa[\theta - v_t]dt + \sigma\sqrt{v_t} dZ_t, \\ dZ_t dW_t &= \rho dt, \end{aligned} \tag{1}$$

where $\kappa > 0$ denotes the speed of mean reversion, $\theta > 0$ the long term variance of the price process, $\sigma > 0$ the volatility of volatility, $-1 \leq \rho \leq 1$ the correlation coefficient between the geometric Brownian motion processes Z_t and W_t, and $S_0, v_0 > 0$ are the starting values of the stochastic processes S_t and v_t, respectively. In order that the instantaneous variance $v_t > 0$ is strictly positive, the parameters need to obey the Feller condition $2\kappa\theta > \sigma^2$.

The value of an American call option with maturity T and strike K, is, at time $t = 0$, given by

$$V(S_0, 0) = \sup_{\tau \in \mathcal{T}(T)} \mathbb{E}_{\mathbb{Q}} \left[e^{-r\tau} (S_\tau - K)^+ \right], \tag{2}$$

where $\mathcal{T}(T)$ denotes the set of stopping times τ with values in $[0, T]$. Assuming the system of SDEs from Equation (1), one can derive the solution to Equation (2) in terms of a PDE. The resulting Heston American call option PDE (expressed in terms of the option price) is given by

$$\frac{\partial V}{\partial t} + \frac{1}{2} v S^2 \frac{\partial^2 V}{\partial S^2} + \rho\sigma v S \frac{\partial^2 V}{\partial v \partial S} + \frac{1}{2}\sigma^2 v \frac{\partial^2 V}{\partial v^2} - rV + (r-q)S\frac{\partial V}{\partial S} + [\kappa(\theta - v) - \lambda(S, v, t)]\frac{\partial V}{\partial v} = 0, \tag{3}$$

in the region $\mathcal{D} = \{0 \leq t \leq T,\ 0 \leq S_t \leq b(v,t),\ 0 \leq v < \infty\}$, subject to the boundary conditions

$$V(S,v,T) = (S-K)^+,$$
$$V(b(v,t),v,t) = b(v,t) - K,$$
$$\lim_{S \to b(v,t)} \frac{\partial V}{\partial S} = 1,$$
$$\lim_{S \to b(v,t)} \frac{\partial V}{\partial v} = 0,$$

where $b(v,t)$ represents the early exercise boundary at time t for the Heston volatility v, and $\lambda(S,v,t)$ denotes the corresponding market price of volatility risk. See Heston (1993) and AitSahlia et al. (2010) for a detailed proof and more information.

Algorithm 1 Neural network-based American option pricer under stochastic volatility

Input: Chosen parameter ranges for r, q, v_0, σ, κ, θ, ρ, M, and $T-t$.
Output: Current price of an American-style option per underlying applicable in a market-making environment.

(A) Training data generation: Evaluation of American option prices using a chosen standard pricing method across an entire parameter space Ψ.
(B) Learning: Offline training of a neural network over a parameter-price data set.
(C) Pricing: Application of a neural network pricer in a market-making environment.

In Algorithm 1, the proposed American option pricing methodology is summarized in a three-step process that generates a direct explainable neural network pricer. In the case when the training data is generated with a standard PDE method, Step (A) reduces to solving the Heston PDE from Equation (3) using a finite difference method. The parameter ranges over which the learning takes place need to be chosen in this step. In our case, these are the Heston model parameters whereas the same algorithm works for any other parametric specification. We detail the chosen parameter ranges (assuring that the Feller condition is satisfied) in Section 3.1. Note that the chosen parameter ranges largely depend on the available computational capabilities. In Step (B), offline training takes place. One needs to choose the structure of the neural network and the corresponding hyperparameters in this step of the algorithm. We detail the architecture of our chosen neural network in Section 2.2 and provide the analysis of the training process in Section 3.1. The deep neural network here is utilized merely as a tool to learn the pricing map between the chosen standard (e.g., PDE-based) pricing method in Step (A) and the corresponding American option prices. Such an approach enhances the trust and transparency of our algorithm. Even though a neural network is applied, the algorithm itself is not a black box. As shown in the next section, our algorithm simply accelerates the computation of American option prices obtained by a well-understood pricing approach chosen in Step (A) of the algorithm. Finally, after the offline learning process from Step (B) is completed, the deep explainable (i.e., Heston-based) neural network pricer can be employed in a market-making environment in Step (C) of the algorithm. Given our aim for computational explainability, the proposed algorithm has a large potential of being understandable and trusted by humans, and as such accepted by regulators and investors.

Since this neural network is targeted for market making, a key responsibility of any algorithm in a trading environment is its compliance with regulation such as the Markets in Financial Instruments

Directive (MiFID) or Securities and Exchange Commission (SEC) guidelines. For example, MiFID specifically states that banks employing algorithms in trading should be able to "reconstruct efficiently and evaluate the strategies that algorithmic traders employ", see Directive-65/EU (2014). This implies a requirement that black-box pricing algorithms such as neural networks need to have an explainable element. As such, we find it important to stress that our XAI-based pricing method indeed fulfills this requirement since we can show, see Section 3.2, that the error is bounded on the underlying (i.e., Heston) parameter space.

Note that one can use any underlying data-generating process as well as any American option pricing method for the training data generation since the calibration of the deep neural network works independently of the training price generation. As such, we provide a possible extension of the algorithm proposed above to pricing under the rough Bergomi volatility specification in Appendix A.[‖]

To improve upon the computationally slower standard American option pricing algorithms, we apply neural networks to serve as the functional evaluators–essentially substituting the underlying numerical pricing algorithm. This is primarily due to the ability of neural networks to model complex non-linear dynamics, differentiability, fast execution, and the ability to be evaluated using very basic low-level functionals. Furthermore, given the universal approximation properties of neural networks, there exists a neural network (trained upon a chosen basic pricing method) that is able to price American-style options to any desired accuracy up to the pricing error imposed by the generated training data.

We aim to improve the process of American option pricing with respect to the speed-accuracy trade-off compared to the standard American option pricers. The motivation for improving the speed-accuracy trade-off lies in the ability of the neural network to learn the American option pricing map offline. Because of the simplicity of the neural network representation, their evaluation speed is unparalleled to the speed of other option pricing models. The boost in speed of the computation and the algorithm interpretability are the main arguments granting our approach a suitable candidate improving upon the currently used market-making pricing systems.

2.2. Deep Neural Network Structure

We exploit a feed-forward neural network with a rectified linear unit (ReLU) activation function merely as a tool to map from a given parameter space to the American option prices. The near-instantaneous evaluation of neural networks and their ability to approximate continuous functions provide for the above-mentioned speed advantages over standard pricing methods. Neural networks can evaluate option prices in less than milliseconds, whereas PDE methods, binomial methods, or MC methods have significantly slower evaluation speeds. The deep neural network we employ is structured as a generic multi-layer perceptron as described in Caterini and Chang (2018).

Definition 2.2 (Multi-Layer Perceptron)**.** Given a data set with an input dimension of k and an output dimension of m, a multi-layer perceptron is defined as the neural network with an input layer of dimension $n_0 := k$ and an output layer of dimension $n_{L+1} := m$, where L specifies the number of hidden layers between the input and the output layer. Analogously, each layer between layers 0 and $L + 1$ is of dimension n_i. For each layer i, we denote the weight and bias matrices by $W_i \in \mathbb{R}^{n_{i+1} \times n_i}$ and $b_i \in \mathbb{R}^{n_{i+1}}$, respectively. Per layer, the following function is applied to provide the input to the subsequent layer:

[‖] Rough Bergomi model improves upon the classical stochastic volatility counterparts by reflecting the implied volatility surface dynamics more accurately (e.g., especially for short maturities). Under the rough Bergomi model, derivative pricing must be performed via Monte Carlo simulations. Thus, the computation time could potentially be greatly reduced using our proposed accelerated pricing method.

$g_i(x_i; W_i, b_i) = \gamma_i(W_i \cdot x_i + b_i)$, where $x_i \in \mathbb{R}^{n_{i+1}}$ and γ_i denotes the corresponding activation function of the layer.

Some of the fundamental theorems of deep neural networks postulate that under a specific set of criteria, neural networks can approximate continuous functions to any degree of accuracy. These theorems are known as universal approximation theorems. Some of them focus on the depth (i.e., a number of hidden layers) of a neural network, and others on the width (i.e., the number of nodes per hidden layer). A widely known universal approximation theorem from Hornik et al. (1989) specifies that a neural network can approximate any continuous real-valued function on a compact set to an arbitrary accuracy $\epsilon > 0$ provided certain criteria for activation functions and depth hold. This is known as the depth-bounded universal approximation theorem. Extensions of this universal approximation theorem for ReLU activation functions are provided in Stinchcombe and White (1989). Moreover, Lu et al. (2017) propose a specific universal approximation function that provides an upper bound to the width of a neural network with ReLU activation functions. This result is relevant for this paper since we use the ReLU activation functions on the neural network's hidden layers. The corresponding theorem is stated below.

Theorem 2.1 (Width-Bounded Universal Approximation Theorem for ReLU Networks)**.** *For any Lebesgue-integrable function f:* $\mathbb{R}^n \to \mathbb{R}$ *and any* $\epsilon \geq 0$*, there exists a fully-connected ReLU network G with width* $d_L \leq n + 4$*, such that the neural network function* F_G *approximating* f *satisfies:*

$$\int_{R^n} |f(x) - F_G| \, dx < \epsilon. \tag{4}$$

Our choice of the ReLU activation function relies on some of its special properties. ReLU activation functions are special since for $x < 0$, the activation function does not fire[**]—causing a sparser neural network. Furthermore, as x becomes large, the neurons do not become saturated[††] as in popular sigmoid activation functions. Lastly, ReLU functions are known to train faster compared to other activation functions, as shown by Krizhevsky et al. (2017). One problem with ReLU activation functions noted by Caterini and Chang (2018) is that neurons can easily die, becoming what are known as "dead neurons" or neurons that always give a zero weight—a downside of allowing a sparse structure. This problem can be fixed by using a leaky ReLU activation function defined by

$$\gamma(x) \coloneqq \left\{ \begin{array}{ll} x, & x > 0 \\ 0.01x, & x \leq 0 \end{array} \right\}, \tag{5}$$

which returns a small negative value for $x < 0$, allowing the neurons to recover.

Following standard literature, we utilize the ADAM optimizer[‡‡] for the training with the Mean Absolute Error (MAE) cost function:

$$\text{MAE} \coloneqq \frac{1}{n} \sum_{\Psi} |V_{NN}(\psi_k) - V_{TS}(\psi_k)| \, , \tag{6}$$

[**] The so-called 'firing' of a node in a neural network is etymologically derived from the biological definition of neurons firing, i.e., sending a non-null impulse when an input is received.

[††] Saturated means that for a very large positive or negative number, the gradient of the sigmoid activation function is near zero.

[‡‡] ADAM optimizer offers many advantages compared to the typical stochastic gradient descent, see Kingma and Ba (2014) for more details.

where n is the number of price samples, $\psi_k \in \Psi$ is a particular parameter specification, $V_{NN}(\psi_k)$ is the output of the neural network at each training iteration for a respective parameter combination ψ_k, and $V_{TS}(\psi_k)$ is the training data set for a respective parameter combination ψ_k. We decided to train with the MAE cost function because using an alternative cost function, for example, the widely-used mean squared error (MSE), would exaggerate the contribution induced by outliers by squaring their error in the cost function. This is relevant in our case since prices can vary widely within the parameter ranges (e.g., from 0.5 USD to 30 USD) and, consequently, the error varies as well.[§§]

3. Numerical experiments

3.1. Training process

The training of the neural network consists of the determination of the hyperparameters such as the learning rate, exponential learning rate adjustment (ELRA), batch size, epochs, and depth and width of the hidden layers. The final neural network structure is set to four hidden layers with 40 nodes per layer. We follow Horvath et al. (2021) and Liu et al. (2019b) in that we decide to use a neural network with four hidden layers ensuring that the depth requirement for the universal approximation theorems is reached.

We generate the training sample (TS) parameter set Ψ_{TS} by choosing 12 equally spaced points for the parameter ranges $M \in [90, 110]$, $T - t \in [1, 255]$, $\kappa \in (0, 5]$, $\rho \in [-1, 0]$, and $\sigma \in (0, 1]$, and 12 logarithmically spaced points for $\theta \in (0, 0.41]$ and $v_0 \in (0, 0.41]$, since the long-term variance and the initial variance levels prevalently undertake lower (i.e., less than 50%) values.[¶¶] Using this parameter set Ψ_{TS} as an input to the finite difference PDE method, we generate the training set on a very fine mesh of 400 points in time, 400 points in price space, and 100 points in volatility space (i.e., $400 \times 400 \times 100$). This results in 12^7 price samples prior to the application of the Feller condition. Following the application of the Feller condition, we are left with a training set consisting of 24,634,368 parameter-price pairs. Moreover, our training set parameter ranges are in line with the ones studied in the existing literature, see Spiegeleer et al. (2018).

To prepare the data for training, we standardize the input parameter data Ψ_{TS} as is a typically required practice in the training of neural networks since different input data have different absolute scales. We apply the same standardization (i.e., same mean and variance) to the parameter sets for all validation and test input data. Then, by applying the ADAM optimizer on the MAE cost function with the leaky ReLU activation function as described above, we train the neural network using an exponential learning rate adjustment. The training stops when the validation error does not improve for another 1000 iterations after reaching a running minimum. Naturally, during training, the difference between the validation and the training error is closely observed in order to prevent over-fitting to the input data.

Due to instabilities in the ADAM optimizer, one typically applies an exponential learning rate adjustment that exponentially decreases the learning rate during the training session. This allows the optimizer to initially explore the cost function surface for the minima, and subsequently, by slowly reducing the learning rate, the likelihood that the optimizer jumps out of the minimum is reduced. In Figure 1, the effect of this adjustment for a sample training session is illustrated.

[§§] We also performed training with a Vega-Weighted Absolute Error (VWAE) cost function as presented in Hamida and Cont (2005). Thereby, the vega weight "converts" the errors in price into errors in implied volatility. The results, however, did not improve compared to the use of the conventional MAE cost function.

[¶¶] Note that r and q have been fixed to be 1% and 3%, respectively. We set r and q to be constant since they are parameters that should not change on an hour-by-hour basis. If necessary, the neural network can be retrained when necessary.

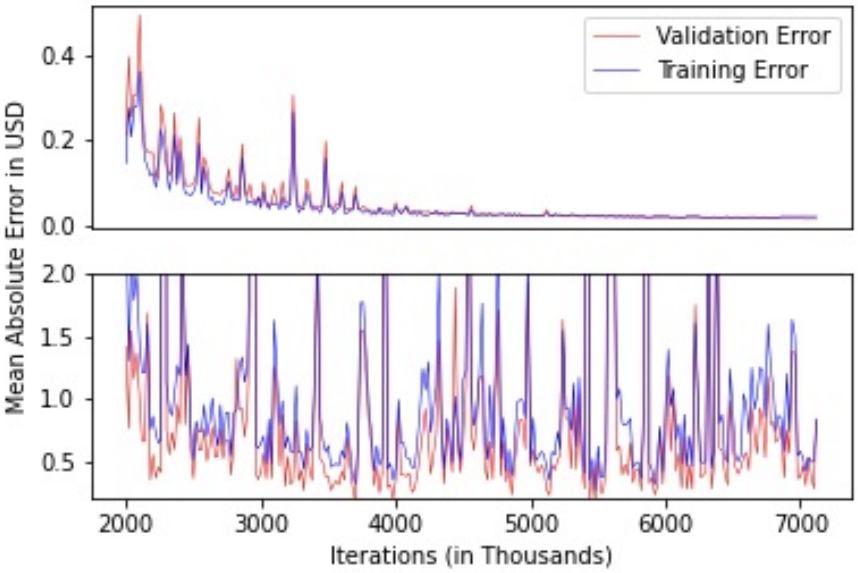

Figure 1. Training (in blue) and validation error (in red) per ADAM iteration with (upper subplot) and without (lower subplot) the exponential learning rate adjustment. The same pattern continues also after the 7000th iteration.

After 23,000 iterations over the entire training set, the stopping rule was satisfied and the training was stopped. The applied neural network structure is presented in Table 1.

3.2. Pricing results

Having trained the neural network offline, we test its out-of-sample performance in this section. We generate the out-of-sample (OOS) parameter set Ψ_{OOS} by using the finite difference PDE pricer with the same mesh size as the one used for the training set, $400 \times 400 \times 100$. Note that the OOS parameter set Ψ_{OOS} is independent of the training sample parameter set Ψ_{TS} so that the error bias is avoided. Using Ψ_{OOS}, we evaluate the neural network-based prices V_{NN} and the corresponding near-exact (i.e., PDE-based) V_{PDE} prices. By "exact" we imply that a highly fine-meshed finite difference method

Table 1. This table presents the chosen neural network hyperparameters.

	Neural Network Structure:	
	n	24,634,368
	Batch Size	1,231,719
	Initial Learning Rate	0.5
	ELRA	0.997
	Hidden Layers (Depth)	4
	Nodes per Layer (Width)	40
	Parameters to Train	5281
	Epoch Size	23,000
	Training Time	41 hours

was applied to construct the option prices. In Table 2, the neural network prices are compared to the near-exact PDE prices in terms of mean absolute error (MAE), mean absolute percentage error (MAPE), and absolute percentage error (APE).

Table 2. Model Summary under the Heston Stochastic Volatility Model.

Heston Stochastic Volatility Case:			
OOS n	14336	FD Grid Density	$400 \times 400 \times 100$
NN OOS MAE	0.02 USD	NN OOS MAPE	0.26%
NN OOS SD of Error	0.014 USD	Evaluation Time for OOS n	0.000026 sec.
NN OOS Min APE	0.0000236%	NN OOS Max APE	3.26%

Now that we have an overview of the average performance of the neural network pricer, we take a closer look at how it performs for each area of the parameter space Ψ. Consider a regular pricing environment with parameter values of $v_0 = 5\%$, $\theta = 15\%$, $\sigma = 0.3$, $\kappa = 0.8$, $\rho = -0.5$, $T - t = 165$ (i.e., in days), and $M = 100\%$. In Figure 2, we vary one parameter at a time over its respective parameter space and plot the resulting neural network price in comparison to the near-exact price as computed by the PDE method. Such an analysis demonstrates the explainability nature of our pricing algorithm.

Notice that the neural network is able to learn the pricing function under the Heston stochastic volatility specification. Moreover, the well-studied Heston parametric structure is inherited by the deep explainable pricer with overall low relative pricing errors. Being able to provide such parametric interpretations positions our model in the field of XAI. Nevertheless, the neural network underperforms in some specific areas of the parameter space. Notice that, for example, $\rho > -0.4$, $M < 95\%$, $T - t < 20$, $v_0 < 0.05$, and σ all exhibit above-average errors. One can tackle this issue by simulating more parameter-price pairs, especially in the under-performing parameter regions, and consequently improve the accuracy of the NN-based pricer. Moreover, one can also experiment with different cost functions or even train multiple neural networks for different parameter regions (e.g., one NN in the short time to maturity region and other for the rest). Given the stochastic nature of the learning algorithm, a certain level of pricing error is inevitable. However, as shown in Figure 2, the NN-based pricer learns the American option pricing map under the Heston stochastic volatility model with a low relative error and, above all, performs pricing in an explainable and interpretable fashion. Given the increasing demand for transparency from investors and regulators, this is an influential finding facilitating the potential compliance of the proposed pricing algorithm with regulatory guidelines.

Figure 3 shows the implied volatility surface as computed by the deep explainable pricer and its PDE-based benchmark. One can observe that the deep explainable pricer reproduces the shape of the implied volatility surface by training from the data generated by the PDE-based pricer. The right subplot presents the error calculated as the value of the implied volatility as computed by the deep explainable pricer minus the value computed by the PDE-based pricer. Note that the errors are generally very low. The only area with larger deviations are the points close to maturity.

3.3. Computational Speed Comparison

The main incentive for applying neural networks in pricing American options are the benefits in terms of pricing speed. Next, we perform an analysis comparing the speed of evaluation of the proposed neural

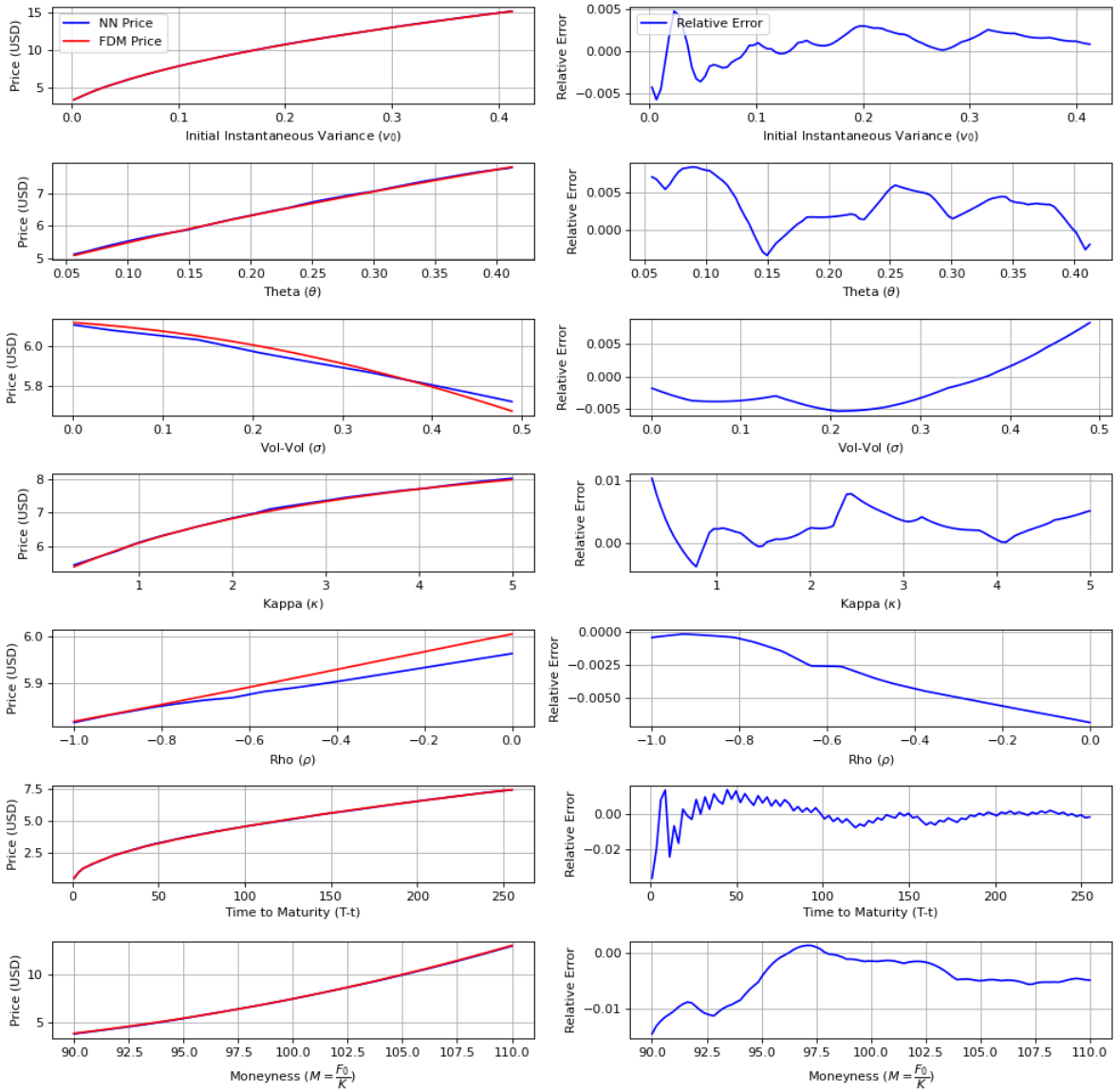

Figure 2. The left part of the figure depicts a comparison between the prices computed by the neural network (NN) under the Heston stochastic volatility—in blue—and the training sample computed by the finite difference method (FDM)—in red—in dependence of different model parameters. The right part of the figure equivalently shows the relative error of the NN approach in comparison to the FDM.

network pricer (over a set of 14,366 input parameter sets) with the speeds of standard methods such as the finite difference method, Monte Carlo simulation of Longstaff and Schwartz (2001), and binomial method of Cox et al. (1979) in Table 3. The computations were performed on an Intel Xeon Gold 6126 2.60GHz CPU.

The PDE method is implemented as proposed by Ikonen and Toivanen (2007) for an option of $T = 0.25$. Naturally, the methods' speed differs given different maturities, step sizes, iterations, and a number of paths. Nevertheless, no other method is able to evaluate an American option price nearly as fast as the neural network pricer.

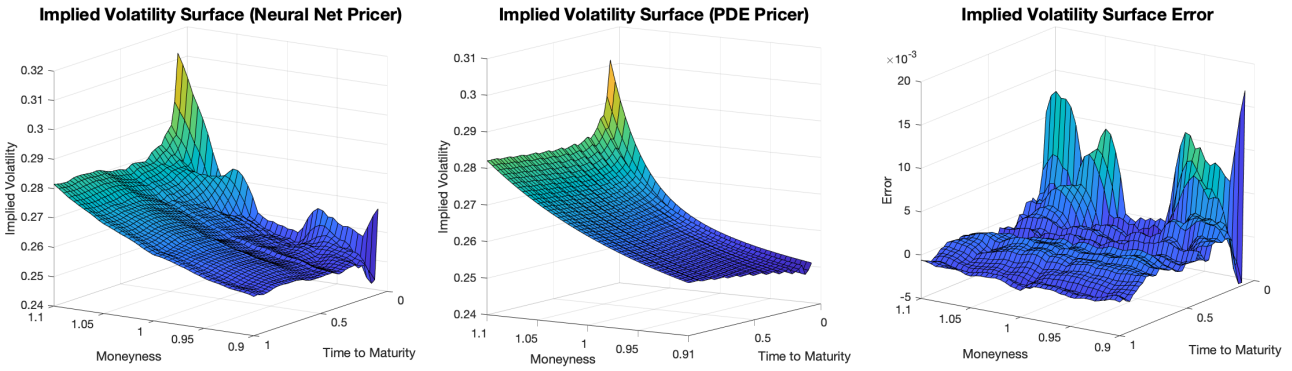

Figure 3. The left subplot depicts the implied volatility surface as computed by the deep explainable pricer. The middle subplot shows the implied volatility surface as computed by the benchmark PDE-based pricer. The right subplot presents the error in the implied volatility surface of the neural network pricer computed as the difference between the deep explainable and PDE-based implied volatility surfaces.

Table 3. Evaluation speed of the neural network pricer in comparison to other pricing methods for Heston stochastic volatility.

Pricing Method	Average Evaluation Speed (secs)	Compared to Neural Network Speed
Neural Network Method	0.000026	—
Finite Difference PDE Method	0.61	23,461 times slower
Binomial Method (Constant Volatility)	7.3	280,769 times slower
Longstaff-Schwartz Method	139.2	53,538,461 times slower

To illustrate the speed-accuracy trade-off, we choose the FDM method (i.e., the fastest competitor of the NN) and reduce the fineness of its grid such that, on average, the mean absolute error of the neural network is matched with the FDM-based error. By comparing the two methods at the same error level, we analyze the practicality of both pricing approaches in a market-making environment where both speed and accuracy are of utmost importance. Table 4 shows that at the same level of accuracy, the neural network pricer exhibits a considerable speed improvement over the FDM-based pricing method. In particular, the FDM method is, at the same level of accuracy, 58 times slower compared to the deep explainable American option pricer proposed in this paper.

Table 4. Speed-accuracy trade-off of the neural network pricer in comparison to the FDM pricing method for Heston stochastic volatility.

Model:	Error	Average Speed(secs)
Neural Network	26bps	0.000026
FDM (26×26)	26bps	0.0015

Figure 4 depicts the overall speed-accuracy spectrum for constant volatility (CV) and Heston stochastic volatility specifications compared to the deep explainable pricer. Speed is measured in terms of the number of pricings per second and accuracy is given in terms of relative error expressed in

basis points against a "true" value defined as the PDE method (for CV or Heston) with a very fine mesh grid. One can observe the outperformance of the neural network with respect to the alternative pricing methods. The deep explainable pricer not only significantly outperforms the PDE Heston benchmark, but also outperforms PDE, Longstaff-Schwartz, and binomial methods under constant volatility. Moreover, with an enhanced training set and longer training times, further improvements in error could be achieved–shifting the neural network point far to the left since the evaluation speed of a neural network is approximately constant. These results show that the proposed deep explainable pricer achieves a reasonable accuracy of computation while greatly reducing the speed of the computation. As such, it serves as a speedy alternative to more classical American option pricing methods currently used in a market-making environment.

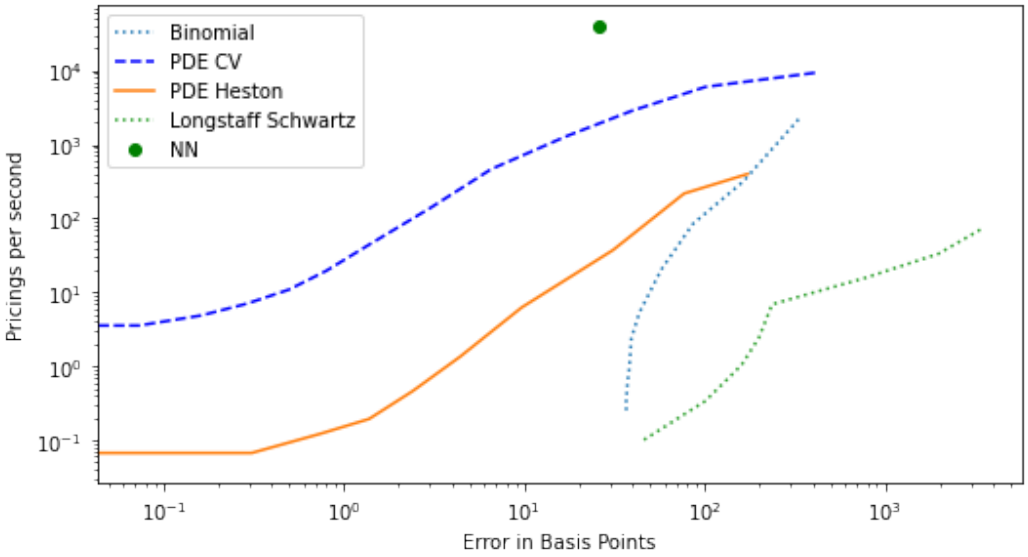

Figure 4. Speed-accuracy trade-off for various American option pricing methods in both constant volatility (CV) and Heston stochastic volatility environments compared to our neural network (NN) pricer.

4. Conclusions

Market-making in the equity derivatives markets requires high-precision and fast pricing algorithms. If the prices are not adjusted accurately and instantaneously, arbitrage opportunities occur. This paper presents an American-style option pricing method based on a deep neural network that accelerates the pricing while maintaining sufficient accuracy for market-making applicability.

The proposed model is based on generating training data by using a standard American option pricing method. We utilize a PDE-based method and price under the Heston stochastic volatility model in order to construct the training data. The obtained training data is subsequently utilized to train a deep neural network over the underlying parameter space offline. Consequently, the underlying (i.e., Heston) parameter interpretability is inherited by the neural network pricer. Such interpretability improves the trust and transparency of the proposed deep explainable pricer, significantly increases the speed of American-style option pricing, and associates our algorithm with the area of XAI for finance.

The simulation analysis shows that our approach provides a substantial improvement with respect to the speed-accuracy trade-off over standard American option pricers. The improvement primarily arises due to the ability of neural networks to learn the American option pricing map offline. Moreover, given the simplicity of neural network representations, their evaluation speed is considerably higher compared to the speed of standard option pricing models. This speed increase is naturally more pronounced for Monte Carlo methods than for PDE methods due to the complexity of simulation methods. Also, we show that this increase in speed occurs without a significant loss in accuracy when methods are compared at the same precision level. Therefore, our model exhibits the speed-accuracy and explainability features required in a challenging market-making environment.

Furthermore, potentially increased computing resources and the universal approximation properties of neural networks imply that the level of accuracy presented in this paper could be improved by i) increasing the number of simulated parameter-price data points over which the training is performed, ii) experimenting with alternative neural network architectures, iii) further optimizing the neural network hyperparameters, iv) experimenting with alternative cost functions, or v) training multiple neural networks at different parameter regions.

While these steps could potentially improve the accuracy of the neural network pricer, other interesting areas of future research could expand the presented deep explainable pricing framework. Particularly, since the speed improvements are greatest when compared to Monte Carlo methods, any pricing model that relies purely on MC methods would be a perfect candidate for neural network acceleration. One could, for example, expand the model to utilize large-deviation MC techniques for out-of-the-money and short-maturity options. Moreover, one could also consider alternative dynamics for the underlying asset and volatility processes, such as the rough volatility, possibly including cash dividends and more complex payoffs that are priceable only by the MC techniques.

Acknowledgments

The authors are grateful to Jérôme Detemple, Walter Farkas, Damir Filipović, Isabella Kooij, Markus Leippold, Álvaro Leitao Rodríguez, Dan Lupu, Ana Mão de Ferro, Leander Gayda, Pit Götz, Ruggero Jappelli, Alexis Marchal, Ludovic Mathys, Michael Mi, Angelina Steffens, Nikola Vasiljević, Rudi Zagst, Žan Žurič, and the participants of the SFI Research Days 2021, the 7th International Young Finance Scholars Conference (2021), the Society of Financial Econometrics Summer School 2021, the 34th Australasian Finance and Banking Conference (2021), the XIX International Conference on Finance and Banking (2022), the 4th International Conference on Computational Finance 2022, and the International Risk Management Conference 2022 for helpful comments and suggestions.

Conflict of interest

The authors declare no conflict of interest.

References

AitSahlia F, Goswami M, Guha S (2010) American option pricing under stochastic volatility: an efficient numerical approach. *Comput Manag Sci* 7: 171–187. https://doi.org/10.1007/s10287-008-0082-3

Anders U, Korn O, Schmitt C (1998) Improving the pricing of options: a neural network approach. *J Forecasting* 17: 369–388. https://doi.org/10.1002/(SICI)1099-131X(1998090)17:5/6<369::AID-FOR702>3.0.CO;2-S

Andersen L, Lake M, Offengenden D (2016) High performance American option pricing. *J Comput Financ* 20: 39–87. https://doi.org/10.21314/JCF.2016.312

Andreou PC, Charalambous C, Martzoukos SH (2010) Generalized parameter functions for option pricing. *J Bank Financ* 34: 633–646. https://doi.org/10.1016/j.jbankfin.2009.08.027

Arrieta AB, Díaz-Rodríguez N, Ser JD, et al. (2020) Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Inf Fusion* 58: 82–115. https://doi.org/10.1016/j.inffus.2019.12.012

Ascione G, Mehrdoust F, Orlando G, et al. (2023) Foreign exchange options on Heston-CIR model under Lévy process framework. *Appl Math Comput* 446: 127851. https://doi.org/10.1016/j.amc.2023.127851

Barndorff-Nielsen OE, Stelzer R (2013) The multivariate supOU stochastic volatility model. *Math Financ* 23: 275–296. https://doi.org/10.1111/j.1467-9965.2011.00494.x

Bayer C, Friz P, Gatheral J (2016) Pricing under rough volatility. *Quant Financ* 16: 887–904. https://doi.org/10.1080/14697688.2015.1099717

Bayer C, Tempone R, Wolfers S (2020) Pricing American options by exercise rate optimization. *Quant Financ* 20: 1749–1760. https://doi.org/10.1080/14697688.2020.1750678

Becker S, Cheridito P, Jentzen A (2020) Pricing and hedging American-style options with deep learning. *J Risk Financ Manag* 13: 158. https://doi.org/10.3390/jrfm13070158

Black F, Scholes M, Merton R (1973) The pricing of options and corporate liabilities. *J Polit Econ* 81: 637–654. https://doi.org/10.1086/260062

Boyle P (1986) Option valuation using a three-jump process. *Int Options J* 3: 7–12.

Boyle P, Broadie M, Glasserman P (1997) Monte Carlo methods for security pricing. *J Econ Dyn Control* 21: 1267–1321. https://doi.org/10.1016/S0165-1889(97)00028-6

Brennan M, Schwartz E (1977) The valuation of American put options. *J Financ* 32: 449–462. https://doi.org/10.2307/2326779

Buehler H, Gonon L, Teichmann J, et al. (2019) Deep hedging. *Quant Financ* 19: 1271–1291. https://doi.org/10.1080/14697688.2019.1571683

Caterini A, Chang DE (2018) *Deep Neural Networks in a Mathematical Framework*. Springer. https://doi.org/10.1007/978-3-319-75304-1

Chen F, Sutcliffe C (2012) Pricing and hedging short sterling options using neural networks. *Intell Syst Account Financ Manag* 19: 128–149. https://doi.org/10.1002/isaf.336

Chen Y, Wan JWL (2021) Deep neural network framework based on backward stochastic differential equations for pricing and hedging American options in high dimensions. *Quant Financ* 21: 45–67. https://doi.org/10.1080/14697688.2020.1788219

Cont R (2001) Empirical properties of asset returns: stylized facts and statistical issues. *Quant Financ* 1: 223–236. https://doi.org/10.1080/713665670

Cox JC, Ross SA, Rubinstein M (1979) Option pricing: A simplified approach. *J Financ Econ* 7: 229–263. https://doi.org/10.1016/0304-405X(79)90015-1

Crank J, Nicolson P (1947) A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type. *Math Proc Cambridge* 43: 50–67. https://doi.org/10.1017/S0305004100023197

Directive-65/EU (2014) Directive 2014/65/eu of the european parliament and of the council of 15 may 2014 on markets in financial instruments and amending directive 2002/92/ec and directive 2011/61/eu. *Official Journal of the European Union*.

Dugas C, Bengio Y, Bélisle F, et al. (2009) Incorporating functional knowledge in neural networks. *J Mach Learn Res* 10: 1239–1262.

Farkas W, Ferrari F, Ulrych U (2022) Pricing autocallables under local-stochastic volatility. *Front Math Financ* 1: 575–610. https://doi.org/10.3934/fmf.2022008

Fecamp S, Mikael J, Warin X (2019) Risk management with machine-learning-based algorithms. *Working paper*.

Ferguson R, Green A (2018) Deeply learning derivatives. *Available at SSRN*.

Garcia R, Gençay R (2000) Pricing and hedging derivative securities with neural networks and a homogeneity hint. *J Econometrics* 94: 93–115. https://doi.org/10.1016/S0304-4076(99)00018-4

Gaspar RM, Lopes SD, Sequeira B (2020) Neural network pricing of American put options. *Risks* 8: 73. https://doi.org/10.3390/risks8030073

Gatheral J, Jaisson T, Rosenbaum M (2018) Volatility is rough. *Quant Financ* 18: 933–949. https://doi.org/10.1080/14697688.2017.1393551

Glasserman P, Kim KK (2011) Gamma expansion of the Heston stochastic volatility model. *Financ Stoch* 15: 267–296. https://doi.org/10.1007/s00780-009-0115-y

Glau K, Kressner D, Statti F (2020) Low-rank tensor approximation for Chebyshev interpolation in parametric option pricing. *SIAM J Financ Math* 11: 897–927. https://doi.org/10.1137/19M1244172

Gradojevic N, Gençay R, Kukolj D (2009) Option pricing with modular neural networks. *IEEE T Neural Networ* 20: 626–637. https://doi.org/10.1109/TNN.2008.2011130

Hamida SB, Cont R (2005) Recovering volatility from option prices by evolutionary optimization. *J Comput Financ* 8: 43–76. https://doi.org/10.21314/JCF.2005.130

He XJ, Lin S (2023a) Analytically pricing exchange options with stochastic liquidity and regime switching. *J Futures Markets* 43: 662–676. https://doi.org/10.1002/fut.22403

He XJ, Lin S (2023b) A closed-form pricing formula for European options under a new three-factor stochastic volatility model with regime switching. *Jpn J Ind Appl Math*40: 525–536. https://doi.org/10.1007/s13160-022-00538-7

Hernandez A (2016) Model calibration with neural networks. *Available at SSRN*. https://doi.org/10.2139/ssrn.2812140

Heston SL (1993) A closed-form solution for options with stochastic volatility with applications to bond and currency options. *Rev Financ Stud* 6: 327–343. https://doi.org/10.1093/rfs/6.2.327

Hornik K, Stinchcombe M, White H (1989) Multilayer feed-forward networks are universal approximators. *Neural Networks*. https://doi.org/10.1016/0893-6080(89)90020-8

Horvath B, Muguruza A, Tomas M (2021) Deep learning volatility: a deep neural network perspective on pricing and calibration in (rough) volatility models. *Quant Financ* 21: 11–27. https://doi.org/10.1080/14697688.2020.1817974

Hutchinson JM, Lo AW, Poggio T (1994) A nonparametric approach to pricing and hedging derivative securities via learning networks. *J Financ* 49: 851–889. https://doi.org/10.1111/j.1540-6261.1994.tb00081.x

Ikonen S, Toivanen J (2004) Operator splitting methods for American option pricing. *Appl Math Lett* 17: 104–126. https://doi.org/10.1016/j.aml.2004.06.010

Ikonen S, Toivanen J (2007) Pricing American options using lu decomposition. *Numer Meth Part D E* 24: 104–126. https://doi.org/10.1016/j.aml.2004.06.010

Itkin A (2019) Deep learning calibration of option pricing models: some pitfalls and solutions. *Working paper*.

Kelly DL (1994) Valuing and hedging American put options using neural networks. Available from: http://moya.bus.miami.edu/ dkelly/papers/options.pdf

Kingma DP, Ba J (2014) Adam: A method for stochastic optimization. *Third International Conference For Learning Representations, San Diego*.

Kohler M, Krżyzak A, Todorovic N (2010) Pricing of high-dimensional American options by neural networks. *Math Financ* 20: 383–410. https://doi.org/10.1111/j.1467-9965.2010.00404.x

Krizhevsky A, Sutskever I, Hinton GE (2017) ImageNet classification with deep convolutional neural networks. *Adv Neural Inf Processing Syst* 25. https://doi.org/10.1145/3065386

Lajbcygier PR, Connor JT (1997) Improved option pricing using artificial neural networks and bootstrap methods. *Int J Neural Syst* 8: 457—471. https://doi.org/10.1142/S0129065797000446

Larguinho M, Dias JC, Braumann CA (2022) Pricing and hedging bond options and sinking-fund bonds under the CIR model. *Quant Financ Econ* 6: 1–34. https://doi.org/10.3934/QFE.2022001

Lin S, He XJ (2023) Analytically pricing variance and volatility swaps with stochastic volatility, stochastic equilibrium level and regime switching. *Expert Syst Appl* 217: 119592. https://doi.org/10.1016/j.eswa.2023.119592

Liu S, Borovykh A, Grzelak LA, et al. (2019a) A neural network-based framework for financial model calibration. *J Math Ind* 9. https://doi.org/10.1186/s13362-019-0066-7

Liu S, Oosterlee C, Bohte S (2019b) Pricing options and computing implied volatilities using neural networks. *Risks* 7: 1–22. https://doi.org/10.3390/risks7010016

Longstaff FA, Schwartz ES (2001) Valuing American options by simulation: A simple least-squares approach. *Rev Financ Stud* 14: 113–147. https://doi.org/10.1093/rfs/14.1.113

Lu Z, Pu H, Wang F, et al. (2017) The expressive power of neural networks: A view from the width. *NIPS 2017*.

Malliaris M, Salchenberger LM (1993) A neural network model for estimating option prices. *Appl Intell* 3: 193–206. https://doi.org/10.1007/BF00871937

Maruyama G (1955) Continuous Markov processes and stochastic equations. *Rendiconti del Circolo Matematico di Palermo* 4: 276–292. https://doi.org/10.1007/BF02846028

Meissner G, Kawano N (2001) Capturing the volatility smile of options on high-tech stocks—a combined GARCH-neural network approach. *J Econ Financ* 25: 276–292. https://doi.org/10.1007/BF02745889

Morelli MJ, Montagna G, Nicrosini O, et al. (2004) Pricing financial derivatives with neural networks. *Phys A* 338: 160–165. https://doi.org/10.1016/j.physa.2004.02.038

Pasricha P, He XJ (2023) Exchange options with stochastic liquidity risk. *Expert Syst Appl* 223: 119915. https://doi.org/10.1016/j.eswa.2023.119915

Pires MM, Marwala T (2004) American option pricing using multi-layer perceptron and support vector machine. *In 2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No.04CH37583)*, 2: 1279–1285.

Ruf J, and Wang W (2020) Neural networks for option pricing and hedging: A literature review. *J Comput Financ* 24: 1–45. https://doi.org/10.21314/JCF.2020.390

Schoutens W (2003). *Lévy processes in finance: pricing financial derivatives*. Wiley. https://doi.org/10.1002/0470870230

Shin HJ, Ryu J (2012) A dynamic hedging strategy for option transaction using artificial neural networks. *Int J Software Eng Appl* 6: 111–116.

Spiegeleer JD, Madan DB, Reyners S, et al. (2018) Machine learning for quantitative finance: fast derivative pricing, hedging and fitting. *Quant Financ* 18: 1635–1643. https://doi.org/10.1080/14697688.2018.1495335

Stinchcombe M, White H (1989) Universal approximation using feedforward networks with non-sigmoid hidden layer activation functions. *Proceedings of the International Joint Conference on Neural Networks*, 613–618.

Xie B, Li W, Liang N (2021) Pricing S&P 500 index options with Lévy jumps. *arXiv preprint*.

Ye T, Zhang L (2019) Derivatives pricing via machine learning. *J Math Financ* 9: 561–589. https://doi.org/10.2139/ssrn.3352688

A. Appendix: Deep Explainable Pricing Algorithm - Rough Volatility Extension

Definition A.1 (Rough Bergomi Volatility Model)**.** Let S_t and v_t denote the asset price and instantaneous variance at time t. The rough Bergomi model, introduced by Gatheral et al. (2018), is, under a pricing measure, defined by a system of stochastic differential equations

$$\begin{aligned}\frac{dS_t}{S_t} &= (r-q)dt + \sqrt{v_t}(\rho\, dW_t + \sqrt{1-\rho^2}\, dW_t^{\perp}),\\ v_t &= \xi_0(t)\mathcal{E}(\eta W^{\alpha})_t,\end{aligned}$$

where r is the interest rate, q the is continuous dividend yield, W_t and $W_t^{\perp}$ are independent Brownian motions, $-1 \le \rho \le 1$, and the instantaneous variance v_t is a product of a forward variance curve $t \mapsto \xi_0(t)$, known at time 0, and the Wick exponential $\mathcal{E}(\eta W^{\alpha})_t = \exp(\eta W_t^{\alpha} - \frac{1}{2}\mathrm{Var}(\eta W_t^{\alpha}))$ for $\eta > 0$ and a Gaussian random variable W_t^{α}. The random variable W_t^{α} is given by the Gaussian Riemann–Liouville process

$$W_t^{\alpha} = \sqrt{2\alpha+1}\int_0^t (t-s)^{\alpha} dW_s, \quad t \ge 0,$$

where the parameter $-\frac{1}{2} < \alpha < 0$ controls the roughness of paths. The corresponding paths have a Hölder regularity of $\alpha + \frac{1}{2}$ and locally behave like the paths of a fractional Brownian motion with $H = \alpha + \frac{1}{2}$. The three time-homogeneous parameters, η, ρ, and α, can be interpreted as the smile, the skew, and the near-maturity explosion of the smile and the skew, expressed in terms of the implied volatility surface.

In the case of rough Bergomi volatility, a Monte Carlo simulation is required to generate the initial training data needed for the neural network calibration. This means that the gains in speed due to training a neural network price evaluator are potentially greater than in the case of the Heston stochastic volatility model.

In Algorithm 2, a neural network pricer for the case of Monte Carlo generated training data set is presented. In Step (A), the Euler-Maruyama method, see Maruyama (1955), is applied to simulate paths of the volatility and price processes for a fine mesh across the parameter set Ψ. On each path, optimal exercise opportunities are evaluated, and a corresponding payoff for each path is determined.

For the evaluation of optimal exercise opportunities in Step (B), the methodology presented in Andersen et al. (2016) can be utilized. Their method of approximating the early exercise boundary allows for the computation of an early exercise boundary value for every point along the MC paths. The optimal exercise strategy is calculated by comparing the current simulated spot price level with the Andersen boundary value. By discounting and averaging over the payoffs of the simulated paths for each parameter combination, a Monte Carlo estimator of the American option price is obtained. Finally, by repeating this procedure for all combinations of parameters across Ψ, one arrives at a set of American

Algorithm 2 Neural network-based American option pricer under rough volatility

Input: ξ_0, η, α, ρ, M, $T - t$.
Output: Current price of an American-style option per underlying applicable in a market-making environment.

(A) SIMULATION: Simulation of the volatility and price processes.
(B) OPTIMAL EXERCISE: Evaluation of optimal exercise strategies for each simulated path.
(C) TRAINING DATA GENERATION: Evaluation of American option prices across an entire parameter space Ψ.
(D) LEARNING: Offline training of a neural network over a parameter-price data set.
(E) PRICING: Application of a neural network pricer in a market-making environment.

option prices across the entire parameter space – Step (C). In Step (D), the offline training occurs, and the deep neural network rough Bergomi-based pricer can potentially be employed in a market-making environment in Step (E).

As mentioned in Bayer et al. (2020), under rough volatility models, the optimal exercise strategy should depend on past observations of the spot price and not only the current price. However, the authors concluded that there is no significant difference in the computed price due to including past spot prices in the exercise decision. Based on this reasoning, one could assume that the current spot price contains sufficient information to decide upon the American option exercise. However, further investigation is necessary.

AIMS Press