Universidad de San Andrés
Escuela de Negocios

Licenciatura en Finanzas

Kolmogorov-Arnold Networks in Finance: A Comparative Analysis for Derivative Pricing Models

Autor:
Fernando Kricun

Legajo: 33639

Director: Gabriel Basaluzzo

Buenos Aires, 30 de septiembre de 2024

Kolmogorov-Arnold Networks in Finance: A Comparative Analysis for Derivative Pricing Models

Fernando Kricun (fkricun@udesa.edu.ar)

Universidad de San Andres

December 29, 2024

Abstract

Kolmogorov-Arnold Networks (KANs) have recently been introduced as an alternative to traditional Multilayer Perceptrons (MLPs) for neural network representation. In this study, we investigate the application of KANs in constructing physics-informed machine learning models to solve the Black-Scholes equation for pricing derivatives. We conduct a comparative analysis between KAN-based models and those using standard MLP architectures. Our findings reveal that KANs, despite having significantly fewer parameters, achieve comparable or superior accuracy to larger MLPs, with faster convergence across fewer training epochs. These results highlight the potential of KANs as a more efficient alternative to MLPs in the context of derivative pricing models.

Keywords: Kolmogorov-Arnold Networks, Multilayer Perceptrons, Derivative Pricing, Black-Scholes Equation, Physics-Informed Neural Networks, Financial Modeling

1 Introduction

Multilayer Perceptrons (MLPs) are a foundational class of feedforward artificial neural networks, composed of an input layer, one or more hidden layers, and an output layer (Haykin, 1998; Cybenko, 1989). According to the Universal Approximation Theorem (Hornik *et al.*, 1989), MLPs are capable of approximating complex, non-linear functions, making them central to many modern deep learning applications. However, MLPs exhibit several limitations and may not be the most efficient way to optimize univariate functions.

To address these challenges, Kolmogorov-Arnold Networks (KANs) have been proposed as an alternative neural network architecture (Liu *et al.*, 2024). Based on the Kolmogorov-Arnold representation theorem (Kolmogorov, 1961), KANs achieve their functionality by decomposing multivariate functions into sums of univariate functions, enabling greater flexibility in the choice of activation functions compared to MLPs. Although promising, KANs typically employ computationally expensive learnable B-splines as activation functions, which can significantly elevate training costs.

In the domain of financial modeling, particularly in derivative pricing, neural networks have shown promise in solving partial differential equations (PDEs) such as the Black-Scholes equation. The Black-Scholes model provides a crucial mathematical framework for estimating the fair value of financial derivatives (Black and Scholes, 1973; Hull and Basu, 2016). Recently, Physics-Informed Neural Networks (PINNs) have emerged as a powerful method for embedding the governing physical laws of systems, such as differential equations, into the training process of neural networks (Raissi *et al.*, 2019). By incorporating these physical constraints into the loss function, PINNs can more effectively solve PDEs like the Black-Scholes equation while ensuring the learned solutions respect both data and the underlying physics.

This paper investigates the application of KANs to the option pricing problem, comparing their performance with traditional MLP-based models in the framework of physics-informed machine learning. Specifically, we develop PINNs based on both KANs and MLPs to solve the Black-Scholes equation for single-asset and multi-asset options. By leveraging the advantages of KANs, we aim to assess whether they can provide more effective solutions for derivative pricing compared to standard MLP architectures.

To the best of our knowledge, this paper represents the first implementation of KANs for financial pricing. While previous works have combined MLPs with KANs (SS *et al.*, 2024; He *et* *al.*, 2024), they have not specifically used this approach to reduce dimensionality before applying the KAN architecture. Here, we demonstrate that such a hybrid strategy not only addresses scalability challenges but also significantly enhances performance in high-dimensional settings.

Our findings demonstrate that KANs significantly outperform MLPs in terms of convergence speed, accuracy, and computational efficiency. KANs achieve lower relative errors and exhibit greater stability during training due to their adaptive activation functions, which allow for a more precise approximation of complex functions. Notably, the incorporation of domain-specific insights, such as smoothing payoff functions, further accelerates convergence and enhances model performance. These results suggest that KANs offer a promising approach for solving PDEs in financial modeling, providing both computational efficiency and accuracy.

The rest of the paper is organized as follows: Section 2 describes the theoretical framework, detailing the neural network architectures and the Black-Scholes model. Section 3 outlines the methodology, including training procedures and evaluation metrics. Section 4 presents the results, highlighting the comparative performance of MLP and KAN architectures. Finally, Section 5 discusses the implications of our findings and future directions for research.

2 Theoretical Framework

2.1 Multilayer Perceptron (MLP)

Multilayer Perceptrons (MLPs) consist of multiple layers of nodes, where each layer is fully connected to the next. Each node in an MLP, commonly referred to as a neuron, computes a weighted sum of its inputs, applies a non-linear activation function, and passes the result to the next layer. MLPs have proven to be highly versatile in their ability to approximate complex functions, making them a cornerstone in modern machine learning.

Each layer is composed of neurons, which process the data by applying an activation function σ to a weighted sum of the input. Mathematically, for the i-th neuron in layer l, the output can be expressed as:

$$z_i^{(l)} = \sigma\left(\sum_j w_{ij}^{(l-1)} x_j^{(l-1)} + b_i^{(l)}\right), \tag{1}$$

where $x_j^{(l-1)}$ represents the input from the previous layer, $w_{ij}^{(l-1)}$ are the weights between neurons in consecutive layers, and $b_i^{(l)}$ is the bias term for the i-th neuron. The non-linearity introduced by σ, commonly implemented as the sigmoid, ReLU, or tanh function, allows MLPs to approximate complex functions. This structure, combined with a sufficient number of neurons and an appropriate choice of activation function, allows MLPs to approximate virtually any continuous function on compact subsets of $\mathbb{R}^n$, as established by the Universal Approximation Theorem (Hornik *et al.*, 1989). This theorem underpins the ability of neural networks to model complex, non-linear relationships.

2.2 Kolmogorov-Arnold Network (KAN)

Kolmogorov-Arnold Networks (KANs) are a type of neural network that builds upon the Kolmogorov-Arnold representation theorem, which provides a method to decompose multivariate continuous functions into compositions of univariate functions (Liu *et al.*, 2024). This allows for a structured breakdown of complex functions into simpler components, facilitating interpretability and flexibility in modeling.

The Kolmogorov-Arnold representation theorem (Kolmogorov, 1961) states that any multivariate continuous function on a bounded domain can be represented as a sum of compositions of univariate functions. In formal terms, for a continuous function f with n variables, this can be expressed as:

$$f(x_1, \ldots, x_n) = \sum_{q=1}^{2n+1} \Phi_q\left(\sum_{p=1}^{n} \varphi_{q,p}(x_p)\right),$$

where $\varphi_{q,p}$ are univariate functions of the inputs x_p, and Φ_q are continuous outer functions. This decomposition enables KANs to model complex, high-dimensional relationships by breaking them down into simpler, lower-dimensional components.

KANs make use of the Kolmogorov-Arnold theorem by replacing conventional linear weights with univariate functions parameterized by splines. Unlike standard Multilayer Perceptrons (MLPs), which use fixed activation functions, KANs implement adaptive activation functions that are learned during training. These functions, modeled as B-splines, are applied between nodes and are dynamically adjusted to better fit the data. Formally, a KAN layer can be represented as $\Phi = \{\varphi_{q,p}\}$, where $\varphi_{q,p}$ are the learnable univariate functions.

The architecture of deeper KANs is formed by stacking multiple KAN layers, which enhances their ability to capture more complex patterns. The structure of a deep KAN is defined as:

$$\mathrm{KAN}(x) = (\Phi_{L-1} \circ \Phi_{L-2} \circ \ldots \circ \Phi_0)(x),$$

where each Φ_l represents a layer in the network, and L is the total number of layers. By using multiple layers, KANs can represent intricate nonlinear relationships in data, offering a flexible and powerful approach to function approximation and modeling.

2.3 Physics-Informed Neural Network (PINN)

Physics-Informed Neural Networks (PINNs) are a class of neural networks designed to incorporate physical laws, expressed through partial differential equations (PDEs), into the learning process (Raissi *et al.*, 2019). By embedding these governing equations into the loss function, PINNs enable the model to approximate solutions to PDEs while adhering to the underlying physics, making them highly effective for scientific computing and engineering applications.

In a PINN, the objective is not only to minimize the discrepancy between the neural network's predictions and the observed data (in our case, the boundary conditions) but also to ensure that the network's outputs satisfy the governing PDEs of the problem. Suppose we are solving a PDE of the form:

$$\mathcal{N}(u(x)) = 0, \quad x \in \Omega,$$

where $\mathcal{N}$ is a differential operator acting on the solution $u(x)$ in the domain Ω. The solution is approximated by a neural network $u_\theta(x)$ with learnable parameters θ, and the loss function is modified to include the residuals of the PDE. The total loss $\mathcal{L}$ is thus a combination of two terms:

$$\mathcal{L} = \mathcal{L}_{\text{boundary}} + \mathcal{L}_{\text{PDE}},$$

where $\mathcal{L}_{\text{boundary}}$ represents the error between the network's predictions and the boundary conditions, and $\mathcal{L}_{\text{PDE}}$ penalizes the deviation from the governing physical laws, ensuring that the network respects the underlying physics.

PINNs offer a powerful approach for solving PDEs, particularly when traditional numerical methods are computationally expensive. By incorporating the physics directly into the network, PINNs can generalize better than purely data-driven models and often require less training data to achieve high accuracy. Furthermore, PINNs are well-suited for solving high-dimensional PDEs, where traditional grid-based methods struggle due to the curse of dimensionality (Hu *et al.*, 2024).

PINNs have been successfully applied to a wide range of problems, including fluid dynamics, heat transfer, electromagnetism, and structural mechanics. Their ability to integrate domain knowledge with deep learning enables them to provide accurate, physically consistent solutions even in complex systems with limited or noisy data.

2.4 Option Pricing and the Black-Scholes PDE

Option pricing plays a fundamental role in financial markets, where derivatives such as options are evaluated using mathematical models. One of the most prominent models for option pricing is the Black-Scholes partial differential equation (PDE), which provides a theoretical framework for pricing various types of options.

The Black-Scholes model, introduced by Fischer Black and Myron Scholes (Black and Scholes, 1973), assumes that the underlying asset follows a geometric Brownian motion with constant volatility and drift. The price of a derivative, $V(S,t)$, at time t with underlying asset price S, satisfies the Black-Scholes PDE:

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

where σ is the volatility of the asset, r is the risk-free interest rate, and S is the asset price.

For European options, closed-form solutions to the Black-Scholes PDE exist. These solutions are expressed as the Black-Scholes formula for call and put options. The value of a European call option $C(S,t)$ and a European put option $P(S,t)$ at time t, with strike price K and maturity T, are given by:

$$C(S,t) = S\Phi(d_1) - Ke^{-r(T-t)}\Phi(d_2),$$
$$P(S,t) = Ke^{-r(T-t)}\Phi(-d_2) - S\Phi(-d_1),$$

where

$$d_1 = \frac{\ln(S/K) + (r + \sigma^2/2)(T-t)}{\sigma\sqrt{T-t}}, \quad d_2 = d_1 - \sigma\sqrt{T-t},$$

and $\Phi(\cdot)$ is the cumulative distribution function of the standard normal distribution. These formulas account for the time value of money, asset volatility, and the probability of the option expiring in-the-money (Black and Scholes, 1973; Hull and Basu, 2016).

A geometric basket option is a derivative whose payoff depends on the geometric mean of several underlying assets. Consider a basket of n assets, where the geometric average of their prices is defined as:

$$S_{\text{geom}} = \prod_{i=1}^{n} S_i^{w_i}.$$

The Black-Scholes PDE for a geometric basket option can be derived by assuming that the underlying assets $S_1, S_2, \ldots, S_n$ follow a correlated geometric Brownian motion. The PDE for the price of a geometric basket option, V, is given by:

$$\frac{\partial V}{\partial t} + \sum_{i=1}^{n} rS_i \frac{\partial V}{\partial S_i} + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \sigma_i\sigma_j\rho_{ij}S_iS_j \frac{\partial^2 V}{\partial S_i \partial S_j} - rV = 0,$$

Since the geometric mean of the assets follows a geometric Brownian motion, the basket can be treated as a single asset with effective volatility σ_{eff} given by:

$$\sigma_{\text{eff}} = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n} w_iw_j\sigma_i\sigma_j\rho_{ij}}, \tag{2}$$

The price of a European geometric basket call option is then given by the Black-Scholes formula for a single asset with the initial value $S_{0\text{geom}}$, effective volatility σ_{eff}, and risk-free rate r. The closed-form solution is (Kemna and Vorst, 1990):

$$V_{\text{call}} = e^{-r(T-t)}\left(S_{0\text{geom}}e^{(r-\frac{1}{2}\sigma_{\text{eff}}^2)(T-t)}\Phi(d_1) - K\Phi(d_2)\right), \tag{3}$$

where K is the strike price, $\Phi(\cdot)$ is the cumulative distribution function (CDF) of the standard normal distribution, and:

$$d_1 = \frac{\ln\left(\frac{S_{0\text{geom}}}{K}\right) + \left(r + \frac{\sigma_{\text{eff}}^2}{2}\right)(T-t)}{\sigma_{\text{eff}}\sqrt{T-t}},$$

$$d_2 = d_1 - \sigma_{\text{eff}}\sqrt{T-t}.$$

The Black-Scholes PDE is fundamental in the pricing of financial derivatives. Closed-form solutions like the Black-Scholes formula provide efficient pricing methods for simple derivatives. However, for complex instruments such as American-style options, numerical methods such as finite difference methods, Monte Carlo simulations, and more recently, neural networks like Physics-Informed Neural Networks (PINNs), are often required to solve these PDEs effectively (Raissi *et al.*, 2019).

3 Methodology

This study aims to compare the convergence and accuracy of two neural network architectures—Multilayer Perceptron (MLP) and Kolmogorov-Arnold Network (KAN)—for solving the Black-Scholes partial differential equation (PDE) in the context of option pricing. The derivatives being priced include a European call and put, and a geometric European basket call with 100 underlying assets. The performance of these models will be evaluated using the Physics-Informed Neural Network (PINN) loss function, along with the relative L1 and L2 errors compared to exact solutions. For the calculation of the L1 and L2 relative errors, 20,000 points will be randomly sampled.

3.1 Neural Network Architectures

Two different neural network architectures are implemented in this study:

3.1.1 Multilayer Perceptron (MLP)

The MLP architecture consists of an input layer, two hidden layers, and an output layer. Each hidden layer has 256 neurons, with the hyperbolic tangent (tanh) activation function. The input to the MLP consists of randomly sampled collocation points representing stock value and time, and the output layer provides the option price at these points. During each epoch, 100 collocation points are sampled randomly.

3.1.2 Kolmogorov-Arnold Network (KAN)

The KAN architecture leverages the Kolmogorov-Arnold representation theorem, which decomposes multivariate functions into sums of univariate functions. For one-asset options, this architecture includes two hidden layers, each with 10 learnable activation functions. These activation functions are parameterized as B-splines with a grid size of 5 and a spline order of 3. For geometric basket options, the KAN model is extended to three hidden layers, containing 5, 10, and 10 learnable B-spline activation functions, respectively. Additionally, before the hidden layers, a linear activation function followed by a *tanh* transformation is applied to reduce the input dimension to 5. These configurations were chosen to ensure that the training speed in terms of epochs per second is comparable to the MLP architecture. The model uses collocation points as input and predicts the option prices, capturing the nonlinearities present in pricing models with its adaptive activation functions.

3.2 Training and Collocation Points

For both architectures, the input layer consists of randomly sampled collocation points, representing stock value and time. In each epoch, 100 points are sampled. The networks are trained using Adam optimizer (Kingma, 2014), with the PINN loss function guiding the learning process. This loss function ensures that the model respects both the PDE and boundary conditions of the option pricing problem.

3.3 Output Layer Configurations

To explore how different configurations affect model performance, the following variations will be applied to the output layer during training:

- **Unmodified output layer**: Initially, the output layer will remain unmodified, predicting the option price directly based on the collocation points. This will serve as a baseline configuration.
- **Output layer with payoff added**: The second configuration involves adding the payoff function to the output layer before training. Since the shape of the option price solution is closely related to the payoff, this approach may help guide the network toward faster convergence.
- **Output layer with smoothed payoff**: In the third configuration, a smoothed version of the payoff function, generated using the Softplus function with $\beta = 1$, will be added to the output layer. This smooth approximation of the payoff could further improve convergence by making the learning process more stable.

3.4 Derivatives and Evaluation Metrics

The models are applied to price a variety of financial derivatives: European call, European put, and a geometric European basket call option with 100 underlying assets. The key evaluation metrics include:

- **PINN loss**: This measures how well the neural network approximates the underlying PDE and its boundary conditions.
- **L1 and L2 relative errors**: These are computed with respect to the exact solution. L1 error represents the mean absolute error, while L2 measures the root-mean-square error. For the calculation of these errors, 20,000 points will be sampled.

3.5 Loss Function

The loss function employed in this study is designed to optimize the Physics-Informed Neural Network (PINN) model with respect to both boundary conditions and the underlying partial differential equation (PDE). The overall loss, $\mathcal{L}$, is composed of two main components: the boundary condition loss, $\mathcal{L}_{\text{boundary}}$, and the PDE loss, $\mathcal{L}_{\text{PDE}}$. Mathematically, it can be expressed as:

$$\mathcal{L} = \mathcal{L}_{\text{boundary}} + \mathcal{L}_{\text{PDE}}$$

3.5.1 Boundary Loss

The boundary loss, $\mathcal{L}_{\text{boundary}}$, measures the discrepancy between the predicted solution $u_\theta(S_i, t_i)$ provided by the neural network and the true solution $u(S_i, t_i)$ at the boundary points. This is computed as a mean squared error over a set of n collocation points (in our case, $n = 100$):

$$\mathcal{L}_{\text{boundary}} = \frac{1}{n}\sum_{i=1}^{n} \left(u_\theta(S_i, t_i) - u(S_i, t_i)\right)^2$$

3.5.2 PDE Loss for Single-Asset Options

For the single-asset option pricing, the PDE loss, $\mathcal{L}_{\text{PDE}}$, is based on the discrepancy between the predicted neural network solution and the true dynamics governed by the Black-Scholes PDE. It is defined as:

$$\mathcal{L}_{\text{PDE}} = \frac{1}{n}\sum_{i=1}^{n} \left(\frac{\partial u_\theta(S_i, t_i)}{\partial t} + \frac{1}{2}\sigma^2 S_i^2 \frac{\partial^2 u_\theta(S_i, t_i)}{\partial S^2} + rS_i \frac{\partial u_\theta(S_i, t_i)}{\partial S} - ru_\theta(S_i, t_i)\right)^2$$

3.5.3 PDE Loss for Geometric Basket Options

For geometric basket options, we utilize an unbiased lower-dimensional estimator inspired by (Hu *et al.*, 2024). The PDE loss for the basket option pricing is given by:

$$\mathcal{L}_{\text{PDE}} = \frac{1}{n}\sum_{i=1}^{n} \left(\frac{\partial u_\theta(S_i, t_i)}{\partial t} + \sum_{j,k}\left(rS_j \frac{\partial u_\theta(S_i, t_i)}{\partial S_j} + \frac{1}{2} m\sigma_j\sigma_k\rho_{jk} S_j S_k \frac{\partial^2 u_\theta(S_i, t_i)}{\partial S_j \partial S_k}\right) - ru_\theta(S_i, t_i)\right)^2$$

where m is the number of assets (in our case, $m = 100$), and j and k represent a set of indices randomly sampled from 1 to m. This approach scales linearly with the number of assets, as opposed to exponentially, and if effective, could provide a computationally efficient alternative for pricing derivatives in high-dimensional spaces.

3.6 Convergence Analysis

The convergence behavior of the MLP and KAN models will be assessed by observing the rate at which the L1 and L2 errors decrease over time during training. Faster convergence indicates better performance in approximating the pricing model. The architectures were selected in order to match the training speed (epochs per second), ensuring a fair comparison.

The comparison of these architectures will provide insights into their effectiveness in solving complex option pricing PDEs, with particular focus on how well each model handles nonlinearity and high-dimensionality.

4 Results

4.1 Comparison of Output Layer Configurations for KAN

In this section, we compare the performance of three variations of the Kolmogorov-Arnold Network (KAN) with different configurations of the output layer:

- **KAN1**: Unmodified output layer (blue line).
- **KAN2**: Output layer with the payoff function added (green line).

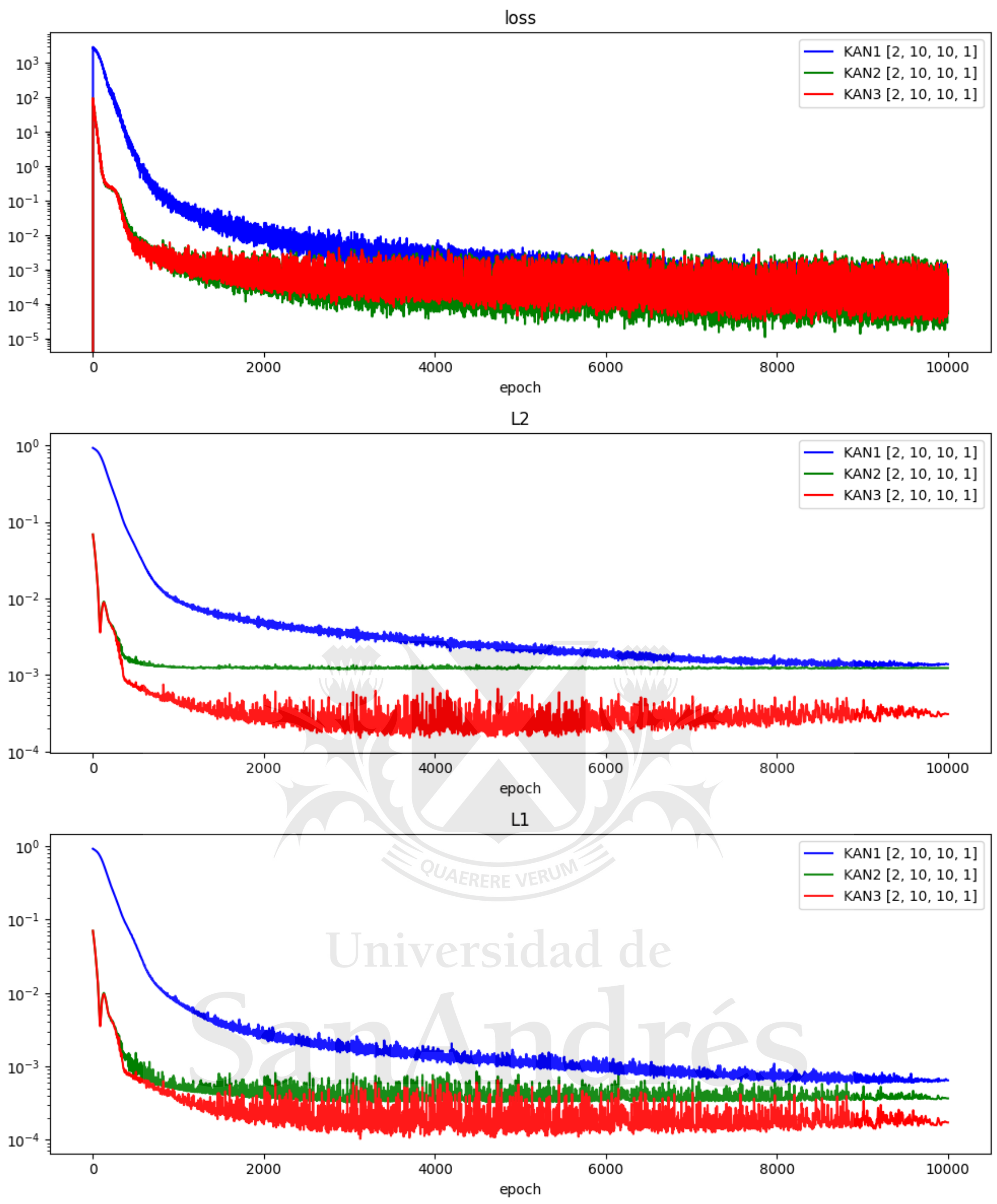

Figure 1: Comparison of training performance for KAN with different output layer configurations: KAN1 (Unmodified, blue), KAN2 (Payoff Added, green), and KAN3 (Smoothed Payoff, red). The graphs show (top) PINN loss, (middle) relative L_2 error, and (bottom) relative L_1 error over 10,000 epochs.

- **KAN3**: Output layer with a smoothed version of the payoff (Softplus function with $\beta = 1$, red line).

Figure 1 illustrates the training performance of the three models over 10,000 epochs for three metrics: PINN loss, relative L_2 error, and relative L_1 error of the pricing of an european call option with T=1.0, r=0.05, $\sigma = 0.2$ and K=10.

4.1.1 Loss Comparison

The PINN loss over the training period is depicted in the top graph of Figure 1. We observe that **KAN3** (red line) and **KAN2** (green line) achieve the fastest convergence, significantly outperforming **KAN1** (blue line) . The smoothed payoff in **KAN3** helps the network converge more rapidly to a lower loss, indicating that smoothing improves gradient stability during training.

In contrast, **KAN1**, which lacks any payoff adjustment, converges more slowly and reaches a higher loss plateau, indicating that the model has more difficulty approximating the target price surfac4. **KAN2**, which incorporates the unmodified payoff, also demonstrates better performance than **KAN1**.

4.1.2 L2 Relative Error

The second graph in Figure 1 shows the L_2 relative error for each KAN configuration. In this case, **KAN3** achieves the smallest L_2 error. By epoch 2000, **KAN3** stabilizes at a significantly lower L_2 error compared to **KAN1**, while **KAN2** performs comparably to **KAN1**, but does not reach the accuracy of **KAN3**.

4.1.3 L1 Relative Error

The third graph presents the L_1 relative error, which follows a pattern similar to that of the L_2 error. **KAN3** maintains the lowest L_1 error throughout training, while **KAN2** demonstrates a slightly better accuracy than **KAN1.**

4.1.4 Discussion

The results indicate that incorporating a smoothed payoff function in the output layer (**KAN3**) significantly improves the performance of the KAN model in approximating option prices. Both **KAN2** and **KAN3** outperform **KAN1** across all metrics, with **KAN3** providing the best overall convergence and error reduction. The smoother payoff in **KAN3** appears to facilitate faster convergence and more accurate predictions.

In summary, the addition of domain-specific knowledge, such as the payoff function, directly into the network architecture helps to improve the model's accuracy. The smoothened version of the payoff (**KAN3**) is particularly effective, achieving the lowest PINN loss, L_2, and L_1 relative errors.

Metric	**KAN1 (Unmodified)**	**KAN2 (Payoff Added)**	**KAN3 (Smoothed Payoff)**
Relative L_2 Error	0.13%	0.12%	**0.031%**
Relative L_1 Error	0.064%	0.036%	**0.017%**

Table I: Summary of final results for KAN configurations after 10,000 epochs.

4.2 Comparison of MLP and KAN Architectures

In this section, we present a comprehensive comparison between the Multi-Layer Perceptron (MLP) and Kolmogorov-Arnold Network (KAN) architectures in the context of solving various option pricing tasks. The tasks include the pricing of a European call option, a European put option, and a geometric basket call option with 100 assets. Each model's output layer utilizes a smoothed payoff function, and the models are trained to approximate the solution of the corresponding partial differential equations (PDEs). The comparisons are conducted over several metrics, including the Physics-Informed Neural Network (PINN) loss, relative L_2 error, and relative L_1 error.

Figures 2, 3, and 4 depict the performance of both architectures on the respective tasks. The MLP architecture is represented in orange, and the KAN architecture in blue.

4.2.1 Loss Comparison

The top panels of Figures 2, 3, and 4 illustrate the PINN loss comparison between the MLP and KAN architectures. For the European call option (Figure 2), the KAN architecture (red line) achieves consistently lower loss values compared to the MLP (orange line). The KAN model demonstrates faster convergence and reduced oscillations, suggesting more efficient learning. This behavior is likely attributed to KAN's adaptable activation functions, which enable more effective approximation of the solution space.

In the case of the European put option (Figure 3), the KAN model again converges faster and achieves a lower loss than the MLP. However, the convergence for both models takes longer than for the call option, likely due to the more complex boundary conditions inherent to put options. Nonetheless, KAN consistently outperforms MLP in terms of both speed and final loss value.

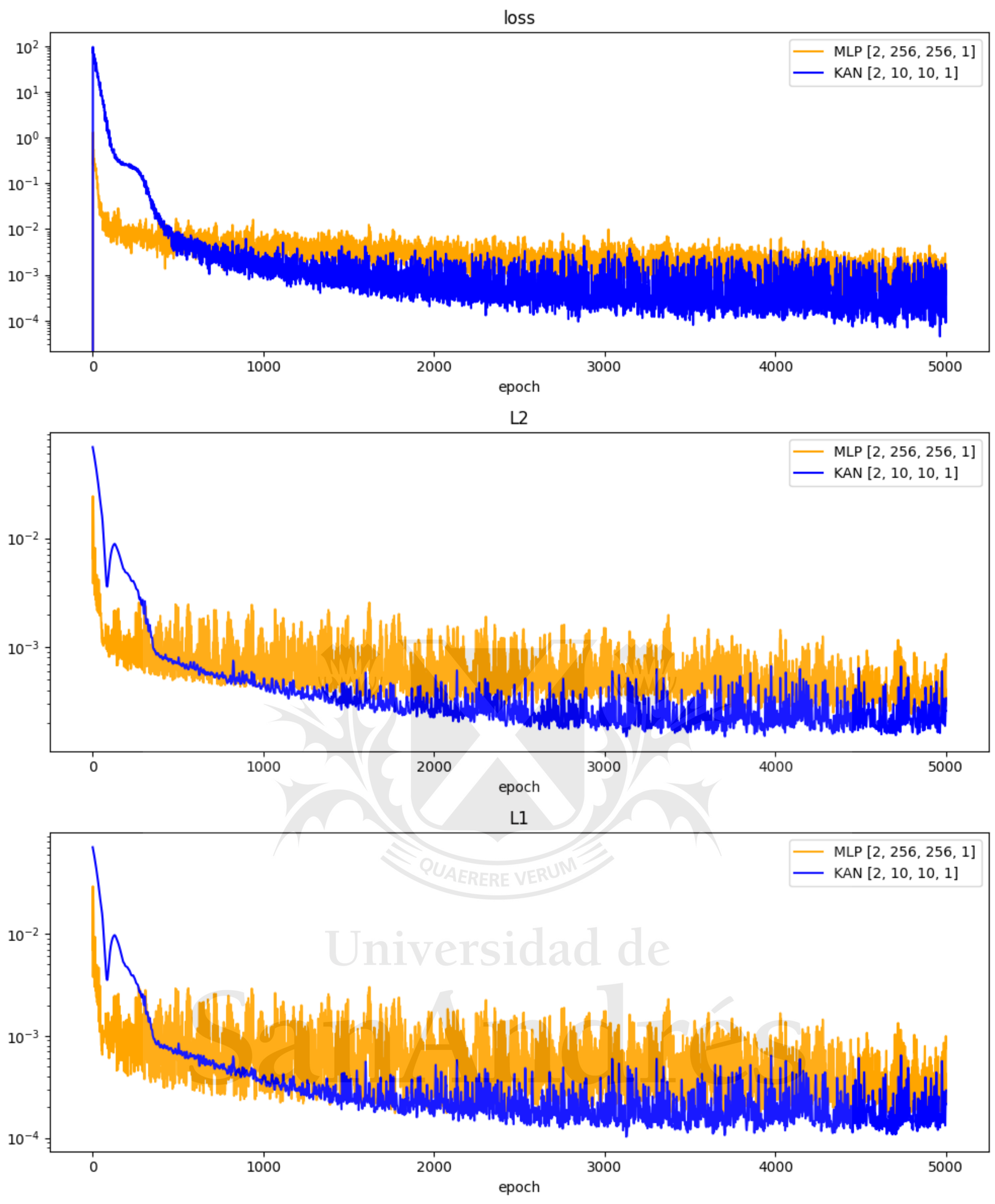

Figure 2: Comparison of MLP (orange) and KAN (blue) architectures for a European call option. The graphs show (top) PINN loss, (middle) relative L_2 error, and (bottom) relative L_1 error over 5000 epochs. T=1.0, r=0.05, $\sigma = 0.2$ and K=10

For the geometric basket option (Figure 4), the loss exhibits more pronounced oscillations, especially for the MLP. This behavior is expected, given the increased complexity of approximating the basket option's PDE with 100 assets. Although both models struggle with stability in loss reduction, KAN shows a slight edge in achieving a more stable convergence after 5000 epochs.

4.2.2 L2 Relative Error

The middle panels of Figures 2, 3, and 4 show the evolution of the L_2 relative error. For the European call option (Figure 2), the KAN model significantly outperforms the MLP, stabilizing at a lower L_2 error after approximately 2000 epochs. The MLP, on the other hand, exhibits higher variance and slower convergence. This result suggests that KAN captures the underlying dynamics of the European call option more effectively.

Similarly, for the European put option (Figure 3), the KAN architecture shows superior per-

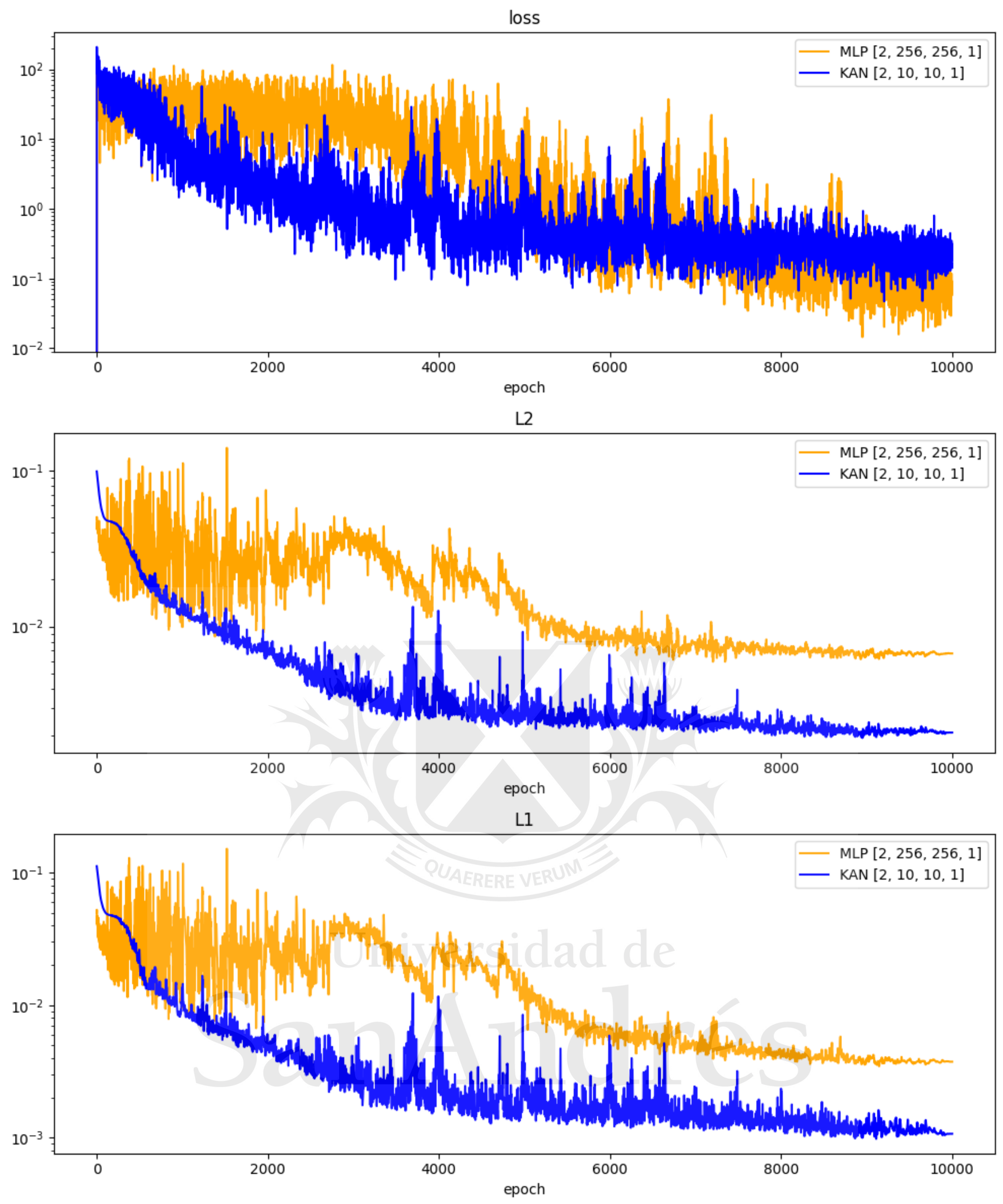

Figure 3: Comparison of MLP (orange) and KAN (blue) architectures for a European put option. The graphs show (top) PINN loss, (middle) relative L_2 error, and (bottom) relative L_1 error over 10000 epochs. T=1.0, r=0.05, σ = 0.2 and K=90

formance, reaching an L_2 error of almost 3.5 times lower than that of the MLP by epoch 10000. The MLP struggles with higher variance throughout training, indicating less robust convergence.

In the geometric basket option (Figure 4), the KAN model initially converges more slowly than the MLP. However, by the end of the training, KAN achieves a much lower L_2 error, demonstrating its capacity to handle high-dimensional problems more effectively than MLP.

4.2.3 L1 Relative Error

The bottom panels of Figures 2, 3, and 4 illustrate the L_1 relative error. As with the L_2 error, the KAN architecture consistently achieves lower L_1 errors across all tasks compared to the MLP.

For the European call option, KAN rapidly reduces the L_1 error, stabilizing at a lower value than MLP. In the case of the put option, the KAN model again shows a clear advantage, converging to a lower L_1 error after fewer epochs. For the basket option, although both models show slow

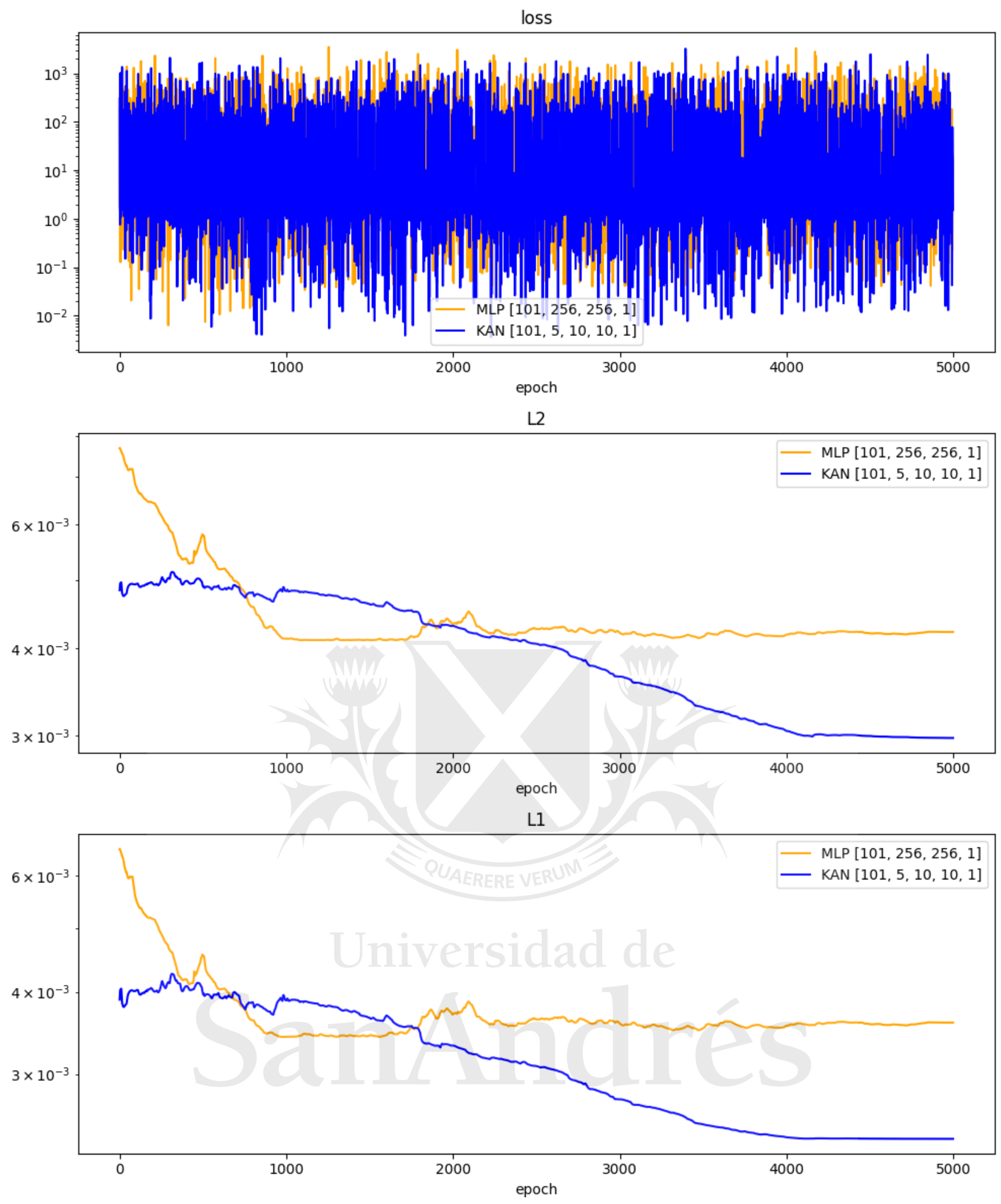

Figure 4: Comparison of MLP (orange) and KAN (blue) architectures for a geometric basket call option with 100 assets. The graphs show (top) PINN loss, (middle) relative L_2 error, and (bottom) relative L_1 error over 5000 epochs. T=1.0, r=0.05, $\sigma = 0.2$ and K=10

convergence, KAN achieves a lower final error, further demonstrating its ability to handle complex, high-dimensional tasks.

4.2.4 Discussion

The comparative analysis of MLP and KAN architectures across different option pricing tasks demonstrates the clear advantage of the KAN model. The learnable activation functions in KAN allow for greater flexibility and adaptability in approximating the solution of PDEs, particularly in higher-dimensional problems such as the geometric basket option. Across all metrics—PINN loss, L_2 relative error, and L_1 relative error—KAN consistently outperforms the MLP, exhibiting faster convergence, lower final errors, and greater stability. These results highlight the potential of KAN as a robust alternative to traditional neural network architectures for financial option pricing and other complex PDE-based applications.

Metric	MLP	KAN
European Call	0.068%	**0.034%**
European Put	0.67%	**0.20%**
Geometric Basket	0.42%	**0.30%**

Table II: Summary of relative L_2 errors for MLP and KAN architectures after convergence.

5 Conclusion

In this work, we compared the performance of the Multi-Layer Perceptron (MLP) and Kolmogorov-Arnold Network (KAN) architectures in solving various financial option pricing problems using Physics-Informed Neural Networks (PINNs). Specifically, we analyzed the pricing of European call and put options, as well as a geometric basket call option involving 100 underlying assets.

The KAN architecture consistently demonstrated superior performance across multiple metrics, including the PINN loss, L_2 relative error, and L_1 relative error. The ability of KAN to employ learnable activation functions contributed to faster convergence, lower final errors, and greater stability throughout training when compared to the MLP. This advantage was evident across all tasks, including the high-dimensional basket option, where the complexity of the problem caused larger oscillations and slower convergence for both models.

Our analysis reveals the following key insights:

- **PINN loss**: The KAN model achieved consistently lower PINN loss across all options, with faster convergence and reduced oscillations. This suggests that KAN's flexibility in representing complex functions allows for a more efficient approximation of the solution space.
- L_2 **relative error**: The KAN architecture significantly outperformed the MLP in reducing the L_2 error, particularly for the European put and geometric basket options.
- L_1 **relative error**: Similar to the L_2 error, the L_1 error consistently favored the KAN model, which demonstrated faster and more stable error reduction across all experiments.

Overall, the results indicate that the KAN architecture is more effective than MLP for solving partial differential equations (PDEs) in financial option pricing, particularly for complex, high-dimensional problems. The flexibility provided by learnable activation functions in the KAN model allows for more precise approximations of the solution, making it a promising candidate for future applications in computational finance and other domains requiring efficient solutions to PDEs.

Future work could involve extending the analysis of the KAN architecture by exploring alternative network configurations and evaluating its performance in frameworks beyond the Black-Scholes model. Moreover, the investigation could include the impact of training strategies, such as employing advanced optimization techniques or adaptive sampling methods, to enhance the efficiency and accuracy of these neural network models. Additionally, future research could deepen the analysis by varying other parameters, such as the dimensionality of the problem, to assess the scalability and robustness of the architecture in more complex scenarios.

References

Black, Fischer and Scholes, Myron (1973). "The pricing of options and corporate liabilities", *Journal of political economy*, Vol. 81 No. 3, pp. 637–654.

Cybenko, George (1989). "Approximation by superpositions of a sigmoidal function", *Mathematics of control, signals and systems*, Vol. 2 No. 4, pp. 303–314.

Haykin, Simon (1998). *Neural networks: a comprehensive foundation*, Prentice Hall PTR.

He, Yunhong *et al.*, (2024). "MLP-KAN: Unifying Deep Representation and Function Learning", *arXiv preprint arXiv:2410.03027*,

Hornik, Kurt, Stinchcombe, Maxwell, and White, Halbert (1989). "Multilayer feedforward networks are universal approximators", *Neural networks*, Vol. 2 No. 5, pp. 359–366.

Hu, Zheyuan *et al.*, (2024). "Tackling the curse of dimensionality with physics-informed neural networks", *Neural Networks*, Vol. 176, p. 106369.

Hull, John C and Basu, Sankarshan (2016). *Options, futures, and other derivatives*, Pearson Education India.

Kemna, Angelien Gertruda Zinnia and Vorst, Antonius Cornelis Franciscus (1990). "A pricing method for options based on average asset values", *Journal of Banking & Finance*, Vol. 14 No. 1, pp. 113–129.

Kingma, Diederik P (2014). "Adam: A method for stochastic optimization", *arXiv preprint arXiv:1412.6980*,

Kolmogorov, Andrei Nikolaevich (1961). *On the representation of continuous functions of several variables by superpositions of continuous functions of a smaller number of variables*, American Mathematical Society.

Liu, Ziming *et al.*, (2024). "KAN: Kolmogorov-Arnold Networks", *arXiv preprint arXiv:2404.19756*,

Raissi, Maziar, Perdikaris, Paris, and Karniadakis, George E (2019). "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations", *Journal of Computational physics*, Vol. 378, pp. 686–707.

SS, Sidharth, AR, Keerthana, KP, Anas, *et al.*, (2024). "Chebyshev polynomial-based kolmogorov-arnold networks: An efficient architecture for nonlinear function approximation", *arXiv preprint arXiv:2405.07200*,