Adv. Appl. Clifford Algebras (2024) 34:19

https://doi.org/10.1007/s00006-024-01313-2

Advances in Applied Clifford Algebras

Exploring Quaternion Neural Network Loss Surfaces

Jeremiah Bill* and Bruce Cox

Abstract. This paper explores the superior performance of quaternion multi-layer perceptron (QMLP) neural networks over real-valued multi-layer perceptron (MLP) neural networks, a phenomenon that has been empirically observed but not thoroughly investigated. The study utilizes loss surface visualization and projection techniques to examine quaternion-based optimization loss surfaces for the first time. The primary contribution of this research is the statistical evidence that QMLP models yield smoother loss surfaces than real-valued neural networks, which are measured and compared using a robust quantitative measure of loss surface "goodness" based on estimates of surface curvature. Extensive computational testing validates the effectiveness of these surface curvature estimates. The paper presents a comprehensive comparison of the average surface curvature of a tuned QMLP model and a tuned real-valued MLP model on both a regression task and a classification task. The results provide strong support for the improved optimization performance observed in QMLPs across various problem domains.

Keywords. Quaternion neural networks, Dimensionality reduction, Surface curvature, Gradient descent optimization.

1. Introduction

Over the last several decades, advances in data collection, storage, and computational power have enabled a wide range of artificial intelligence and machine learning (AI/ML) applications in engineering and the sciences [11]. Deep learning methods in particular have grown increasingly popular due to their success across a broad range of problem sets. The use of modern neural networks such as GPT-4 [27] and other large language models (LLMs) has grown exponentially in recent years and has enabled advances in domains as

This article is part of the Topical Collection on Machine-Learning Mathematical Structures edited by Yang-Hui He, Pierre Dechant, Alexander Kasprzyk, and Andre Lukas.

*Corresponding author.

Birkhäuser

Published online: 24 April 2024

diverse as natural language processing, computer vision, time series analysis and forecasting, and robotic control.

Training such models involves iteratively solving an optimization problem in a very high-dimensional search space. While theory and intuition would suggest that minimizing a non-convex function in a high-dimensional space should yield poor results, in practice neural networks have achieved state-of-the-art results in many application domains (see [1] for a recent survey of state-of-the-art results). In addition, many recent works have demonstrated that neural networks constructed in higher dimensional algebras such as Clifford algebras and geometric algebras can reduce the dimensionality of large neural networks while simultaneously improving the optimization, training, and generalization performance of these neural networks as shown in [6,10,24].

Quaternion neural networks (QNNs) constitute the most popular Clifford algebra-based neural network implementation due to the ubiquity of quaternions in many engineering applications [8]. QNNs are neural network structures wherein the weights, biases, and inputs are all represented by quaternion numbers as opposed to real-valued numbers. QNNs have demonstrated consistent improved network accuracy and generalization performance in a variety of neural network applications versus their real-valued equivalents [36].

The QNN literature provides a wealth of empirical evidence demonstrating the superior performance of QNNs over real-valued neural networks [28]. However, the reasons behind this improved performance have remained largely unexplored. This work leverages loss surface visualization and projection techniques from the real-valued neural network literature to present the first ever exploration of quaternion-based optimization loss surfaces. The main contribution is the statistical evidence that tuned Quaternion Multi-Layer Perceptron (QMLP) models admit smoother loss surfaces then real-valued Multi-Layer Perceptron (MLP) neural networks. Towards this end the research develops a robust quantitative measure of loss surface "goodness" using estimates of surface curvature. This research conducts extensive computational testing to demonstrate the efficacy of the surface curvature estimates. Finally, this work presents robust statistical comparisons of the mean surface curvature of a tuned QNN model vs. a tuned real-valued neural network model on a regression task and a classification task. These experiments demonstrate that the QNN models admit a loss surface that is statistically significantly smoother than their real-valued counterparts, providing strong support for the improved optimization performance that QNNs have shown across myriad problem domains.

The rest of this article is organized as follows: Section 2 provides a brief overview of quaternion algebra, QNNs, neural network loss surface characterization, and surface curvature. Section 3 outlines the two test datasets used in this study, as well as details on each neural network architecture and the hyperparameter tuning process. Additionally, this section outlines the loss surface projection process, surface curvature estimates, and statistical tests employed throughout the study. Finally, Section 4 provides key results and relevant discussion, while Section 5 provides conclusions and suggestions for future research.

2. Background and Related Work

This section provides a brief overview of quaternion algebra and the construction of QNNs. Several relevant works from the loss surface visualization literature are presented as well as a brief introduction to surface curvature and differential geometry.

2.1. Quaternion Algebra

The quaternion numbers (denoted by $\mathbb{H}$) are a four-dimensional extension of the complex numbers. Each quaternion $\bar{q}$ consists of a real part and three imaginary parts, so that the quaternions form an isomorphism with $\mathbb{R}^4$ with basis elements 1, $\bar{i}$, $\bar{j}$, and $\bar{k}$:

$$\bar{q} = r + x\bar{i} + y\bar{j} + z\bar{k}. \tag{2.1}$$

In the discussion that follows and throughout the rest of this paper, we represent all quaternions with bar notation, while scalars in $\mathbb{R}$ are represented with lowercase, unbolded letters. We represent the single real and three imaginary components of a quaternion with the variables $r, x, y,$ and z, respectively.

Quaternions form a generalization of the complex numbers, wherein the three imaginary components $\bar{i}$, $\bar{j}$, and $\bar{k}$ follow the same construct as $\mathbf{i}$ in $\mathbb{C}$:

$$\bar{i}^{(2)} = \bar{j}^{(2)} = \bar{k}^{(2)} = -1. \tag{2.2}$$

However, the three imaginary basis components must also satisfy the following rules:

$$\bar{j}\bar{k} = -\bar{k}\bar{j} = \bar{i}, \tag{2.3}$$
$$\bar{k}\bar{i} = -\bar{i}\bar{k} = \bar{j}, \tag{2.4}$$
$$\bar{i}\bar{j} = -\bar{j}\bar{i} = \bar{k}. \tag{2.5}$$

These rules demonstrate that quaternion multiplication $\otimes$, known as the *Hamilton Product*, is non-commutative. The Hamilton Product is easily derived using the basis multiplication rules in Eqs. (2.3)–(2.5) and the distributive property. In reduced form, the Hamilton Product of two quaternions $\bar{q}_1$ and $\bar{q}_2$ is defined as:

$$\begin{aligned} \bar{q}_1 \otimes \bar{q}_2 := &(r_1 r_2 - x_1 x_2 - y_1 y_2 - z_1 z_2) \\ &+ (r_1 x_2 + x_1 r_2 + y_1 z_2 - z_1 y_2)\bar{i} \\ &+ (r_1 y_2 - x_1 z_2 + y_1 r_2 + z_1 x_2)\bar{j} \\ &+ (r_1 z_2 + x_1 y_2 - y_1 x_2 + z_1 r_2)\bar{k}. \end{aligned} \tag{2.6}$$

In addition, the element-wise (Hadamard) product, is defined as:

$$\bar{q}_1 \odot \bar{q}_2 := r_1 \cdot r_2 + (x_1 \cdot x_2)\bar{i} + (y_1 \cdot y_2)\bar{j} + (z_1 \cdot z_2)\bar{k}. \tag{2.7}$$

Similarly, quaternion addition is defined using the element-wise addition operation. For two quaternions $\bar{q}_1, \bar{q}_2 \in \mathbb{H}$, the sum $\bar{q}_1 + \bar{q}_2$ is defined as:

$$\begin{aligned}\bar{q}_1 + \bar{q}_2 := (r_1 + r_2) + (x_1 + x_2)\bar{i} \\ + (y_1 + y_2)\bar{j} + (z_1 + z_2)\bar{k}.\end{aligned} \tag{2.8}$$

The notion of a quaternion conjugate is analogous to that of a complex conjugate in $\mathbb{C}$. The conjugate of a quaternion $\bar{q} = r + x\bar{i} + y\bar{j} + z\bar{k}$ is given by $\bar{q}^* = r - x\bar{i} - y\bar{j} - z\bar{k}$. The norm of a quaternion is equivalent to the Euclidean norm in $\mathbb{R}$ and is given by

$$||\bar{q}|| := \sqrt{\bar{q}\bar{q}^*} = \sqrt{r^2 + x^2 + y^2 + z^2}. \tag{2.9}$$

With this quaternion norm, one can also define a notion of distance $d(\bar{q}, \bar{p})$ between two quaternions $\bar{q}$ and $\bar{p}$ as

$$d(\bar{q}, \bar{p}) := ||\bar{q} - \bar{p}||. \tag{2.10}$$

The quaternion norm is also used to define the multiplicative inverse of any quaternion:

$$\bar{q}^{(-1)} = \frac{\bar{q}^*}{||\bar{q}||^{(2)}}. \tag{2.11}$$

It is easy to verify that $\bar{q}^{(-1)}\bar{q} = \bar{q}\bar{q}^{(-1)} = 1$. In the special case where $\bar{q}$ is a unit quaternion (i.e., $||\bar{q}|| = 1$), then $\bar{q}^{(-1)} = \bar{q}^*$.

2.2. Quaternion Neural Networks

The mathematical machinery described in Sect. 2.1 provides all of the necessary components to build a basic quaternion neural network wherein the inputs, weights, and biases of the network are all composed of quaternions as opposed to real numbers. The basic Quaternion Multilayer Perceptron (QMLP) was first presented in [3]. The authors proposed a basic quaternion neural network model that generally mirrors the standard MLP with two major caveats.

First, the authors employed "split" or component-wise activation functions $\bar{\sigma}(\cdot)$ wherein a real-valued activation function $\sigma(\cdot)$ is applied to each component of a quaternion individually, i.e.,

$$\bar{\sigma}(\cdot) = \sigma(\cdot) + \sigma(\cdot)\bar{i} + \sigma(\cdot)\bar{j} + \sigma(\cdot)\bar{k}. \tag{2.12}$$

Second, [3] proposed a "pseudo-gradient" backpropagation method, wherein the gradient of a split activation function is computed in a similar component-wise fashion, i.e.,

$$\dot{\bar{\sigma}}(\cdot) = \dot{\sigma}(\cdot) + \dot{\sigma}(\cdot)\bar{i} + \dot{\sigma}(\cdot)\bar{j} + \dot{\sigma}(\cdot)\bar{k}. \tag{2.13}$$

Nitta [26] independently and concurrently proposed a QMLP model using the same split activation and pseudo-gradient construct as [3], with slight differences in the backpropagation derivation. The split quaternion activation functions employed by [3,26] were motivated by early work in complex multilayer perceptrons by [7], who found split activation functions to be necessary in the complex domain $\mathbb{C}$ to circumvent Liouville's Theorem [25], which states that any bounded entire function in the complex plane is constant. Arena et al. [2] proved a universal approximation theorem for complex-valued neural networks using split activation functions in $\mathbb{C}$, hence bypassing the issues posed by Liouville's Theorem for complex MLPs. Arena et al. [5] subsequently presented a succinct proof of a universal approximation theorem for QMLPs similar to the proof for the complex-valued case [2] and results for real-valued neural networks [13,18]. For a more detailed history and exposition of various quaternion neural network formulations, including [9]'s Clifford Multilayer Perceptron (CMLP), see [8].

2.3. Loss Surface Visualizations

Researchers have extensively studied the optimization dynamics of real-valued neural networks. Neural networks are trained in a supervised fashion by minimizing a parameterized loss function of the form:

$$L(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(x_i, y_i; \boldsymbol{\theta}), \tag{2.14}$$

where m represents the number of training samples in the dataset, $\boldsymbol{\theta}$ represents the weights of the neural network, and $\mathcal{L}(x_i, y_i; \theta)$ measures the error between the network's predicted value of the input x_i versus the known target value y_i. Several works have addressed neural network optimization from a theoretical perspective [12,14], requiring various assumptions on the nature or structure of the networks under study. An alternative method for evaluating the optimization dynamics of neural networks which does not require any assumptions on the structure or nature of the networks is to utilize low-dimensional projection techniques and surface visualizations of the parameterized loss function.

In [17], the authors provide one of the first attempts to visualize and characterize the loss functions of neural networks as a function of the network's trainable parameters. The authors perform linear interpolations of the loss function of several neural networks by interpolating between the initial weights and the final weights of each trained neural network. Plots of these linear interpolations provide insights into how neural networks evolve throughout the training process and provide insight into the local minima of the cost function of each network.

Subsequently, [32] utilized two-layer rectified linear unit (ReLU) networks and simple deep ReLU networks to provide several results regarding the geometry of neural network loss functions. The authors demonstrate that under mild conditions, random network initializations will result in a starting point for the network that has a continuous monotonically decreasing path to a global minimum. Additionally, under mild conditions, two-layer ReLU networks will initialize to a basin with good local minima. Keskar et al. [20] investigates the effects of batch size on the optimization process using the same visualization techniques presented in [17] to demonstrate the differences in minima found using large vs. small batch sizes. In [34], the authors investigate the role that batch normalization plays in improving neural network training performance. The authors show empirically and analytically that batch normalization increases the smoothness of both the loss function and the loss function gradients of neural networks. The authors show this empirically by measuring values of the loss function along the direction of the gradient for both batch norm and non-batch norm networks. Im et al. [19] investigates the impacts of different optimizers on the optimization process. The authors use three-dimensional subspace projections to provide deeper insights into the nature of the loss surfaces of various networks than are available using [17]'s two-dimensional weight interpolation plots. Im et al. [19] demonstrates that different optimizers appear to converge to different local minima during training.

While Im et al. [19]'s results demonstrate the differences in local minima found using different optimizers, the results in [33] indicate that these distinct local minima may in fact lie on a large, relatively flat, connected basin of an over-parameterized neural network's loss surface. Whereas many researchers attempt to visualize neural network loss surfaces using techniques similar to [17, 22] presents a different method for visualizing loss surfaces in two and three dimensions using a "filter normalization" technique. The authors demonstrate that [17]'s linear interpolation visualization technique can hide many of the non-convexities of a loss surface, thereby resulting in potentially misleading results. The authors demonstrate that their filter normalization visualization method results in more accurate portrayals of the true convexity of a loss surface for convolutional neural networks (CNNs), and the authors verify this by analyzing the eigenvalues of the Hessian of the loss function.

Finally, [15] introduces the notion of the "Goldilocks zone" in the loss surface of neural networks. The authors demonstrate through extensive experiments that there exists a region of high convexity/high positive curvature in the loss landscape that results in fast convergence of both fully-connected and convolutional neural networks. The size and location of this zone is dependent on the trace and the norm of the Hessian of the loss function. Additionally, the authors show empirically that many common weight initialization schemes result in a network initialized in the Goldilocks zone.

Advances in loss surface characterization and visualization techniques have greatly contributed to the conventional wisdom for training deep neural networks. For example, the relatively low non-convexity of high-dimensional neural network loss surfaces indicated in [17] may explain why over-parameterized neural networks exhibit better (albeit slower) training performance. Additionally, [22] highlights the need for skip connections in very deep neural networks and helps to explain why very deep networks experience unstable training performance beyond the vanishing/exploding gradients hypothesis. Keskar et al. [20] and Santurkar et al. [34] reinforce the importance of utilizing batch normalization and of using small mini-batch sizes when training with SGD. Finally, [15] sheds light on why nearly all of the commonly employed initialization methods result in better training performance. While these results represent a small subset of the loss surface characterization literature, each work listed highlights the relative importance of the topic in terms of a scientific understanding of the dynamics of neural network training and generalization performance. To date, no attempt has been made to characterize

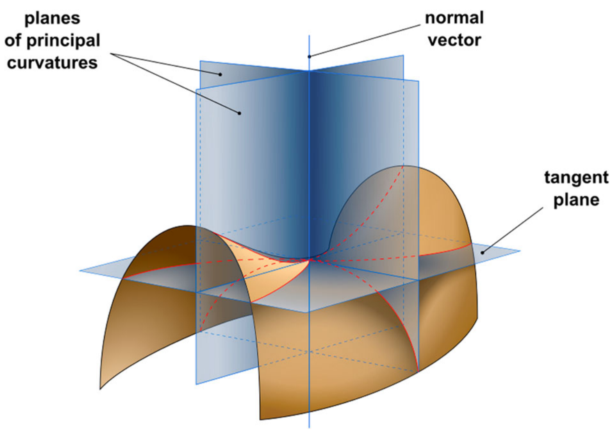

FIGURE 1. Surface with normal planes in direction of principal curvatures from [16]

or visualize the loss surfaces of QNNs. Since loss surface visualization techniques do not require any assumptions regarding the structure of the network under study, they provide an ideal method for examining some of the QNN optimization dynamics to better understand the improved performance that QNNs provide.

2.4. Surface Curvature

The curvature of a surface measures how much a surface bends by different amounts in various directions around a given point. Surface curvature is often expressed in terms of the two principal curvatures at a point, denoted k_1 and k_2, which are simply the minimum and maximum curvatures measured at a point. Formally, the principal curvatures of a surface at a point p are the minimum and maximum normal curvatures of the surface, which are found by measuring the smooth curve formed by intersecting a normal plane and the surface under study. A depiction of this process is shown in Fig. 1, and [21] provides an in-depth overview of surface curvature.

To date, applications of surface curvature in the deep learning literature have remained limited. Li et al. [22] utilize principal curvatures to validate their visualization technique, demonstrating that their constructed 2D surface plots are accurately depicting the convexity of the high-dimensional loss surface. In addition, [15] uses the notion of positive curvature at a point to indicate the convexity of the measured loss surface at a given point. However, both works measure curvature using the eigenvalues of the Hessian matrix of the loss function evaluated at a point. The explicit claim in both works is that the eigenvalues of the Hessian matrix correspond exactly to the measures of curvature at a given point. This statement is true at all critical points of a function f (i.e., all points where $\nabla f = 0$), but is not true in general, potentially resulting in misleading results.

This work directly estimates the curvature of the loss function at each point on a projected loss surface using surface variation. Surface variation provides a technique for estimating the curvature of an interpolated surface directly from a point cloud of data [29–31]. This method does not require any assumptions regarding the closeness of the Hessian eigenvalues to the principal curvatures at a point. Furthermore, this method can be computed directly for any projected loss surface, circumventing the need for a full quaternion Hessian matrix in the case of QNN loss surface curvature. As a measure of the amount of surface "bending" at each point on a surface, the principal curvature values provide an excellent metric for comparing the goodness of loss surfaces from an optimization perspective.

3. Experimental Methodology

This section outlines the methodology employed in the two supervised machine learning experiments employed in this research. Supervised machine learning is a subfield of machine learning wherein a model learns to predict outcomes based on a set of labeled input data. Supervised machine learning tasks can be divided into two categories: regression and classification. In regression tasks, the model predicts a continuous outcome, similar to ordinary least squares regression problems. Classification tasks require the model to predict discrete, categorical outcomes similar to categorical regression techniques such as logistic regression. This research employs a simple regression task and a simple binary classification task to compare the loss surfaces of a tuned real-valued neural network versus a tuned QNN. This section outlines the two datasets employed in the study as well as the loss surface projection technique, surface curvature estimate methodology, and hyperparameter tuning process utilized in this research. All computer experiments were performed on a desktop workstation with 256 GB of RAM and an AMD Epyc 7402p 24-core processor running Ubuntu 22.04.1 LTS. All coding was performed in Julia v1.9.0.

3.1. Binary Classification of Breast Cancer Samples

The Wisconsin breast cancer dataset is a binary classification dataset constructed from 569 samples of breast tissue [40]. Each sample contains 30 real-valued features describing various characteristics of the sample. The features are constructed via digitized images of a Fine Needle Aspirate (FNA) of breast tissue. The labels for each sample are either "M" for malignant or "B" for benign. The purpose of the dataset is to explore various classification techniques for categorizing a breast tissue sample as either malignant or benign based on the various input features. The dataset does not contain any missing values, although the two classes are unbalanced, containing 357 benign samples and 212 malignant samples. The full dataset is split 70%/15%/15% between training, validation, and test data. Prior to network training, each feature of the training dataset is standardized using Z-score normalization. Each feature of the test dataset is standardized using the mean and standard deviation values for each feature calculated from the training data.

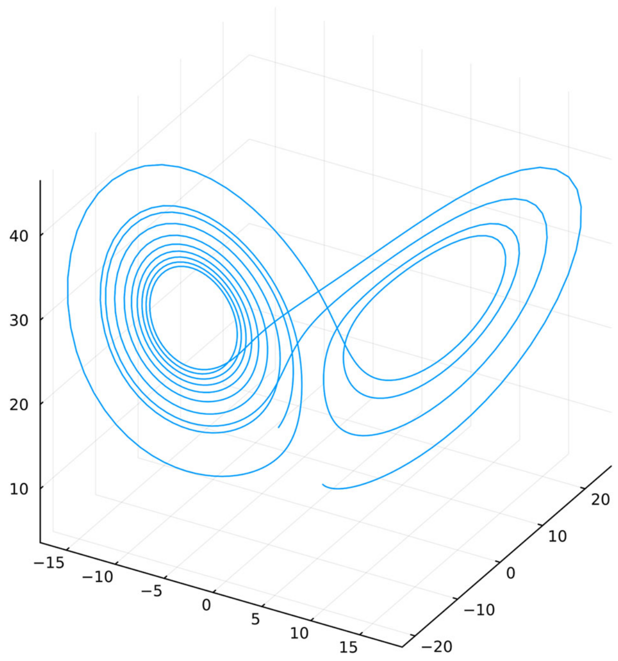

FIGURE 2. Static Lorenz attractor

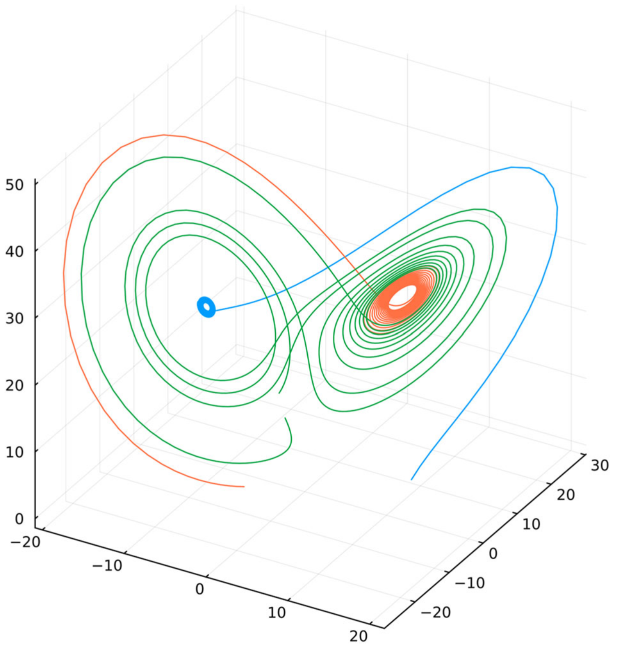

FIGURE 3. Impact of starting position

3.2. Chaotic Time Series Prediction

The Lorenz Attractor is a deterministic system of differential equations that was first presented by Edward Lorenz [23]. The attractor is a chaotic system, meaning that while it is deterministic, the system never cycles and never reaches a steady state. Additionally, the system is very sensitive to initial conditions. When represented as a set of 3-dimensional coordinates, the attractor produces a graph often referred to as the Lorenz butterfly. A static representation of this is shown in Fig. 2, while Fig. 3 shows the system's sensitivity to initial conditions.

The Lorenz Attractor is governed by the following system of differential equations:

$$\frac{dx}{dt} = \sigma(y - x), \tag{3.1}$$

$$\frac{dy}{dt} = \rho x - y - xz, \tag{3.2}$$

$$\frac{dz}{dt} = xy - \beta z, \tag{3.3}$$

where σ, ρ, and β are constants. In this experiment, $\sigma = 10$, $\rho = 28$, and $\beta = \frac{8}{3}$. Quaternions are naturally well-suited to predicting chaotic time series data since the problem involves a multidimensional input as well as a multidimensional output. Quaternion neural networks have proven quite successful at chaotic time series prediction based on small training datasets [4,5,37].

The data for the chaotic time series prediction experiment consists of 500 distinct time series generated using a fixed-timestep fourth-order Runge–Kutta Ordinary Differential Equation (ODE) solver with $dt = 0.01$. The data is split 70%/30% between training and test data. The starting point for each time series is randomly generated using a uniform $U[-10.0, 10.0]$ distribution for the x- and y-coordinates and a uniform $U[0.0, 10.0]$ distribution for the z-coordinates. The input to each neural network is a series of 10 timesteps, and the label or target values are the subsequent 1000 timesteps, measuring the ability of each network to learn the underlying dynamics of the chaotic system and make long range series predictions based on a short input.

3.3. Network Training and Hyperparameter Tuning

This research explores the loss surface of simple quaternion multilayer perceptrons (i.e. fully connected neural networks) compared to those of similarly-structured real-valued networks. The networks in both the regression and the classification tasks employed two hidden layers. In order to provide a robust comparison between network performance for each architecture, the QNNs and real-valued networks were tuned using the Hyperopt.jl hyperparamter tuning package. The tuning process iterated through 100 different random hyperparameter combinations for each architecture, searching over the parameter ranges shown in Table 1 for the QNN. The hyperopt search space for the real-valued network was identical except for the "Neurons" parameter, which represents the number of neurons in each hidden layer, which ranged from 40 to 200 neurons for the real-valued NN. This increased range accounts

TABLE 1. QNN hyperparameter search space

Factor	Type	Level 1	Level 2	Level 3
Activation func.	Categorical	Leakyrelu	Swish	Sigmoid
Optimizer	Categorical	SGD	Adam	
Neurons	Numeric	10	50	
Epochs	Numeric	100	250	
Learning rate	Numeric	1×10^{-5}	0.001	

for the fact that quaternions contain four real-valued components in each quaternion, and hence can encapsulate more information in a single number. This constitutes the sole difference between the hyperparameter search space for the QNNs versus the real-valued networks and reflects the reduction in dimensionality that quaternions provide in a neural network structure.

3.4. Loss Surface Projections

This research utilizes a loss surface projection process inspired by [22] with two key differences. First, this work employs a simple Gram-Schmidt orthogonalization process to ensure that the projection vectors are orthogonal. Second, since this work deals solely with fully-connected neural network models, the loss surface projections do not require any of the filter normalization techniques presented in [22] designed to account for convolutional neural network layers.

As indicated in Eq. (2.14), the loss function of a neural network is parameterized by the weights of a network, denoted by the weight vector $\boldsymbol{\theta}$. In order to project this high dimensional function to a low-dimensional representation for plotting and visualization, two direction vectors $\boldsymbol{\delta}$ and $\boldsymbol{\gamma}$ are randomly selected with $|\boldsymbol{\theta}| = |\boldsymbol{\delta}| = |\boldsymbol{\gamma}|$. These direction vectors are then orthogonalized using a simple Gram-Schmidt orthogonalization routine. Finally, a fine meshgrid of points is constructed along the ranges $-1 < \alpha < 1$ and $-1 < \beta < 1$ and the loss is measured along the projected dimensions at each point in the meshgrid. That is, using the parameterized form of a general loss function shown in Eq. (2.14),

$$f(\alpha, \beta) = L(\boldsymbol{\theta} + \alpha\boldsymbol{\delta} + \beta\boldsymbol{\gamma}). \tag{3.4}$$

2D surface plots of the resulting function values provide insight into the nature of the tuned minimizer represented by the weight vector $\boldsymbol{\theta}$. Since each plot is centered at $\boldsymbol{\theta}$, the surface plots provide an indication of how the loss of a network changes in the region surrounding the local minimizer in the weight space.

3.5. Surface Curvature Estimates and Statistical Comparisons

While low-dimensional projected loss surfaces provide insightful graphics with which to asses the visual differences between loss surface convexity and smoothness, they are limited in their overall utility. The projection method

Algorithm 1 Estimating surface curvature with surface variation

```
1: Initialize max distance $d_{max} = 0.015$
2: Initialize $n = 30$
3: Require $P = \{(\alpha, \beta, f(\alpha, \beta)) \mid -1 < \alpha, \beta, < 1\}$ for $f(\alpha, \beta)$ from Equation 2.14
4: for each $\mathbf{p} \in P$ do
5:     Select neighborhood $N$ of $n$ nearest points: $N = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n\}$.
6:     for $i \in \{1, 2, \ldots, n\}$ do
7:         Compute distance $d_i = ||\mathbf{p} - \mathbf{p}_i||_2$.
8:     end for
9:     Compute mean distance $\mu$.
10:    for $i \in \{1, 2, \ldots, n\}$ do
11:        if $d_i < d_{max}$: then
12:            $\xi_i = 1$
13:        else
14:            $\xi_i = exp(-\frac{d_i^2}{\mu^2})$
15:        end if
16:    end for
17:    Compute centroid of N: $\bar{\mathbf{p}} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{p_i}$.
18:    Compute weighted covariance matrix $\mathbf{C}$: $\mathbf{C} = \sum_{i=1}^{n} \xi_i \cdot (\mathbf{p}_i - \bar{\mathbf{p}}) \cdot (\mathbf{p}_i - \bar{\mathbf{p}})^T$.
19:    Find $\lambda_0 < \lambda_1 < \lambda_2$, eigenvalues of $\mathbf{C}$.
20:    Record $\boldsymbol{\sigma}_n(\mathbf{p}) = \frac{\lambda_0}{\lambda_0+\lambda_1+\lambda_2}$.
21: end for
22: Return $\boldsymbol{\sigma}_n(\mathbf{p})$ for each $\mathbf{p}$.
```

in Sect. 3.4 provides a crucial benefit over other existing approaches: the resulting point cloud of loss data in $\mathbb{R}^3$ provides a basis for estimating the magnitude of the surface curvature of the projected loss directly from the data using surface variation.

Surface variation at a point $\mathbf{p}$ in $\mathbb{R}^3$ with a neighborhood of size n is defined in terms of the eigenvalues of the covariance matrix of the local neighborhood around $\mathbf{p}$:

$$\boldsymbol{\sigma}_n(\mathbf{p}) = \frac{\lambda_0}{\lambda_0 + \lambda_1 + \lambda_2}, \tag{3.5}$$

where $\lambda_0 < \lambda_1 < \lambda_2$ are the eigenvalues of the covariance matrix for the point $\mathbf{p}$. [29] notes that surface variation is not an intrinsic property of point sampled surfaces but depends on the size of the neighborhood n chosen around each point. Further, this research assumes that the point sampled surface characterized by the loss surface projection in Sect. 3.4 provides a good representation of the true underlying projected surface. To improve the stability of the surface curvature estimates, this work employs the outlier weighting scheme presented in [30], which demonstrates that weighting all outlier points in the neighborhood around the point $\mathbf{p}$ results in better curvature estimates versus a naive approach when computing the covariance matrix around $\mathbf{p}$.

The full surface curvature estimation algorithm is shown in Algorithm 1. This research employs a fixed neighborhood $n = 30$ around each point for each curvature estimate. In addition, this work uses a $d_{max} = 0.015$ to determine the outliers in the neighborhood around each point.

This technique provides several benefits over other curvature estimation techniques from the loss surface characterization literature. First, this method does not require the Hessian of the loss function, so the computational cost for employing Algorithm 1 is tied solely to the fineness of the meshgrid, i.e. the number of points in the point cloud. Using the Hessian to characterize surface curvature is problematic for both performance and accuracy reasons, as noted in Sect. 2.3. Furthermore, since the surface variation is calculated directly from the projected surface values in $\mathbb{R}^3$, this method can be used to estimate the loss surface curvature values for real-valued neural networks, QNNs, and any higher dimensional Clifford algebra- or geometric algebra-based neural networks such as those surveyed in [6]. This is vital for assessing the structural advantages that high-dimensional algebraic networks provide over real-valued networks.

Finally, since loss surface projections depend on the selected orthogonal direction vectors delta and gamma each projection should be considered a sample of the actual loss surface. Performing repeated loss surface projections and curvature estimates allows for statistical comparison tests to assess statistical differences in the mean value of the curvature estimates across each surface. This research performs 30 replications of the loss surface projection process, estimating the associated curvature for each projected plot. The means of the curvature estimates are then compared between the QNN and the real-valued neural network for each experiment using a paired t-test, also known as the one-sample t-test. The paired t-test assumes the following:

- The observations being tested are all independent.
- The observations are approximately normally distributed.
- There are no outliers among the observations.

To ensure the independence of each observation, the random number streams used to generate the random projection vectors are carefully controlled. The normality assumption is checked using standard quantile-quantile (QQ) plots, and the outliers are assessed using standard box plots. The resulting statistical comparisons provide robust conclusions on the "smoothness" of QNN loss surfaces versus standard real-valued neural network loss surfaces between fully tuned models for each architecture.

4. Results and Discussion

The tuned QNN models for the breast cancer classification experiment and the time series prediction experiment outperformed the tuned real-valued models, consistent with many of the results highlighted in [28]. This section details the training characteristics of each model and provides measures of merit for the performance of each model. Furthermore, assessments of final

Table 2. Classification hyperparameter tuning results

	Model	
	QNN	NN
Hyperparameter	Value	Value
Activation function	Sigmoid	Sigmoid
Optimizer	SGD	SGD
Neurons	19	16
Epochs	460	500
Learning rate	0.0050	0.00062

model accuracy for each model are provided, as well as loss surface projections, curvature plots, and statistical comparisons of the resulting curvature estimates.

4.1. Breast Cancer Classification Results

The tuned hyperparameters for each classification model are shown in Table 2. The tuning process produced similar results for each model, but the QNN contained significantly fewer neurons per hidden layer and training epochs compared to the real-valued model. Figure 4 depicts the training and validation set loss for both models, capturing the improvement in validation set loss of the QNN model.

When evaluated on the test data, the QNN achieved higher classification accuracy and lower binary cross entropy loss compared to the standard real-valued neural network model. Figure 5 displays the Receiver Operating Characteristic (ROC) curve for both models. The QNN achieved an area under the ROC curve (AUROC) score of 0.975 while the NN model had an AUROC score of 0.969.

The loss surfaces of the QNN classification model and the real-valued neural network classification model illuminate the differences in model performance between the two classifiers. The projected QNN loss surface shown in Fig. 6 exhibits a lower maximum loss compared to the real-valued model, and it appears substantially smoother than the projected neural network surface shown in Fig. 7. The subsequent nine projected surfaces for the QNN model are shown in Fig. 10, portraying the projected minimizer from various angles. In each instance, the QNN classifier demonstrates smooth gradual changes in the projected space from the minimizer centered at $(0, 0)$.

The subsequent nine projected surfaces for the real-valued neural network classifier are shown in Fig. 11. In general, these surface projections exhibit steeper, rougher surfaces than their QNN counterparts around the real-valued minimizer centered at $(0, 0)$. The curvature plots shown in Figs. 8 and 9 quantify the smoothness present in each loss surface. The scale on Fig. 8 clearly demonstrates that in most cases, the projected QNN loss surfaces admits levels of curvature that are well below the curvature values shown for the projected neural network surfaces in Fig. 9. A careful evaluation of the curvature plots versus the loss surface projection plots indicate that the

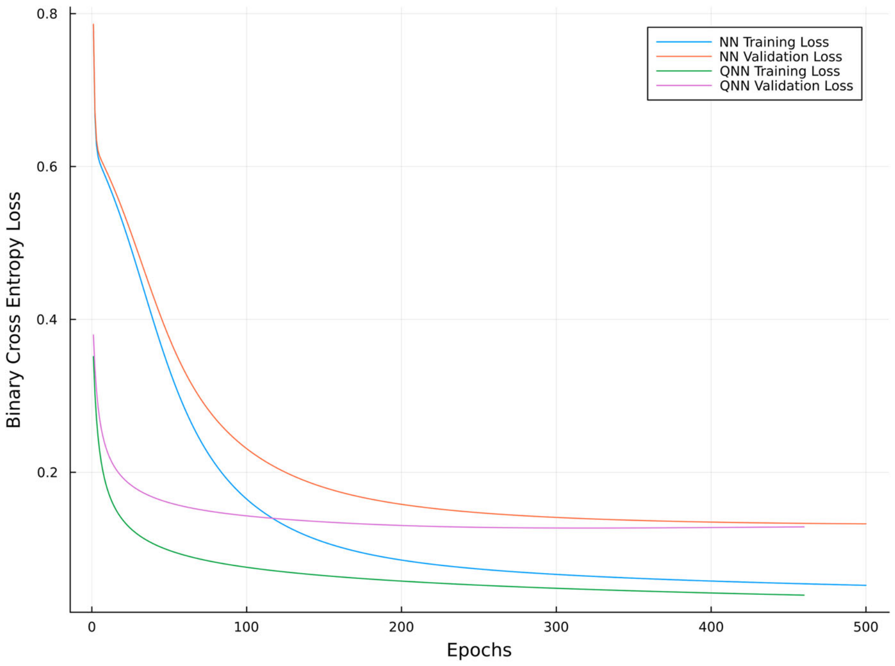

FIGURE 4. Breast cancer classifier QNN and NN training curves

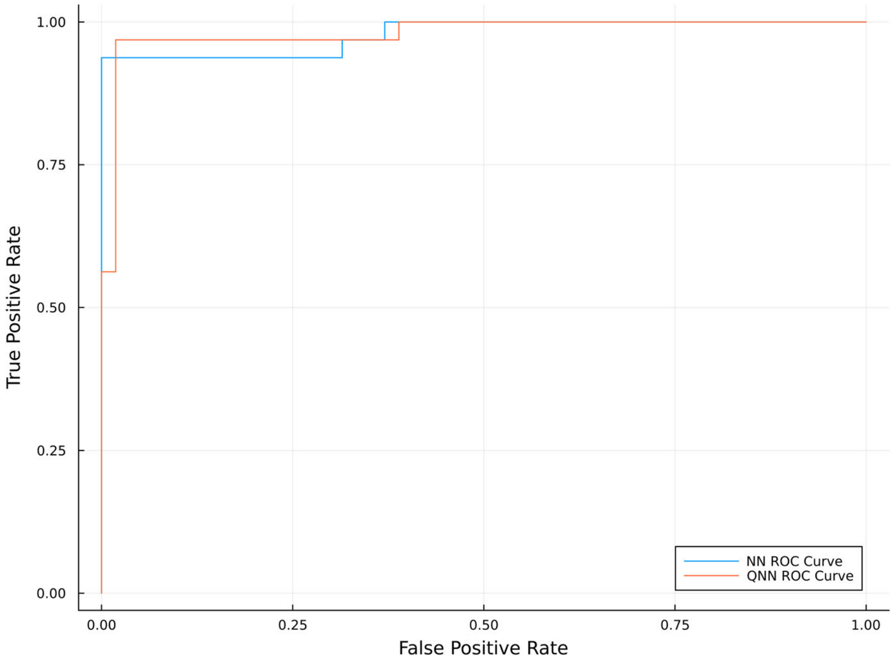

FIGURE 5. Receiver operating characteristic (ROC) curve for both classification models

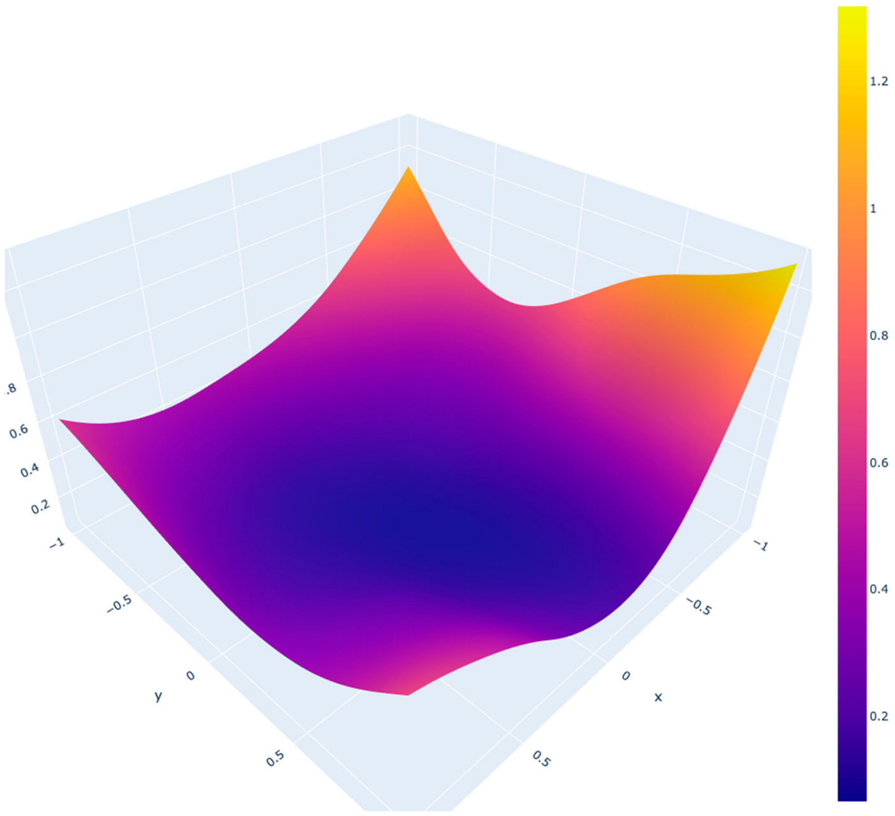

Figure 6. QNN classification 2D loss surface plot

regions of high curvature correspond exactly to the sharp bends or changes in each loss surface plot.

While the projected loss surfaces shown in Figs. 6 and 7 indicate that optimization of the QNN loss surface in the quaternion domain occurs over a smoother surface than the loss surface induced by the real-valued NN in $\mathbb{R}$, the surfaces also provide crucial insight into the generalization performance of the QNN versus the real-valued model. Each surface plot is constructed using the test set (or holdout) data from the classification experiment, hence providing an indication of how well each model generalizes to unseen data at the local minima and in a region around the local minima in the parameter space. The slower, smoother, and more consistent changes around the QNN minimizer showcased in Fig. 6 correlate with the improved generalization performance and higher AUROC and accuracy scores that the QNN model achieves on the holdout data from the experiment.

The QNN and real-valued network surface projections and curvature estimate experiments were repeated 30 times to facilitate robust statistical comparisons between the "smoothness" of the quaternion loss surfaces versus the real-valued loss surfaces. The mean curvature value across each projected

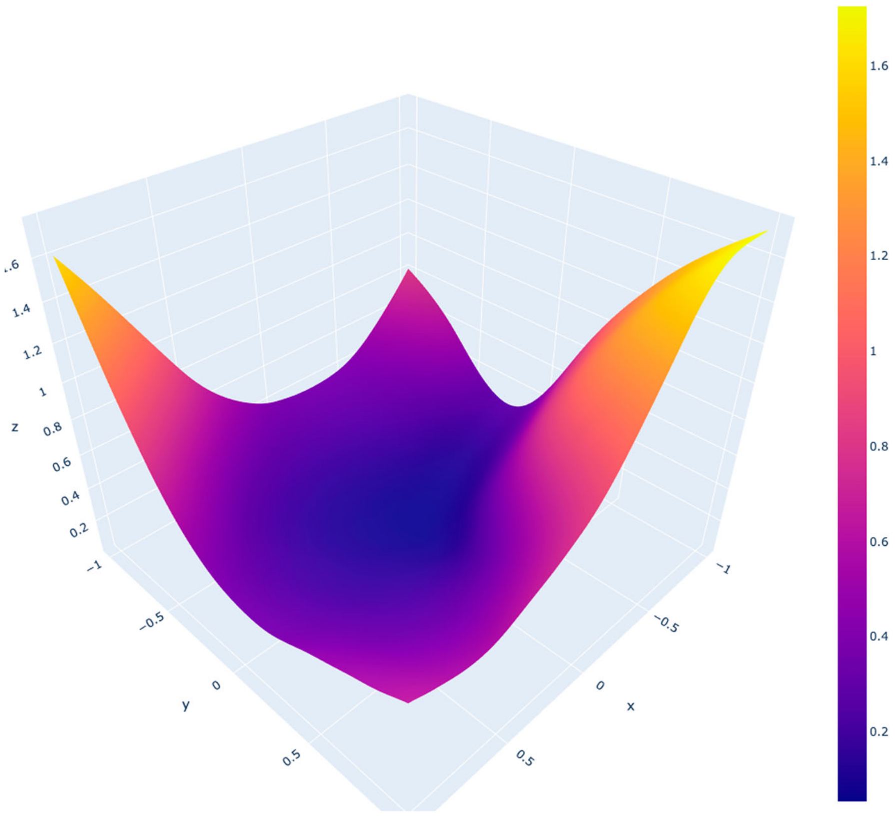

Figure 7. NN classification 2D loss surface plot

surface was calculated in order to construct paired differences for each run. The QQ plot of the paired differences is shown in Fig. 12 indicating the approximate normality of the data. Additionally, a box plot of the differences is shown in Fig. 13 demonstrating that there are no outliers in the paired difference data.

The results of the paired t-test are shown in Table 3. The null hypothesis of the test is that there is no difference between the mean curvature estimates between the two models which would indicate that the QNN and real-valued neural network model result in loss surface curvature values from the same population. The alternative hypothesis is that there are significant differences between the mean curvature estimates produced by the two models. At an $\alpha = 0.1$, the paired t-test indicates that there are statistically significant differences between the QNN mean curvature values and the real-valued network mean curvature values, with a p-value of 0.0874.

These test results indicate that the tuned QNN classification models used in this experiment admit a loss surface that is statistically significantly smoother than the tuned real-valued NN models. This lends credence to the claim that quaternion optimization occurs over a smoother loss surface. The

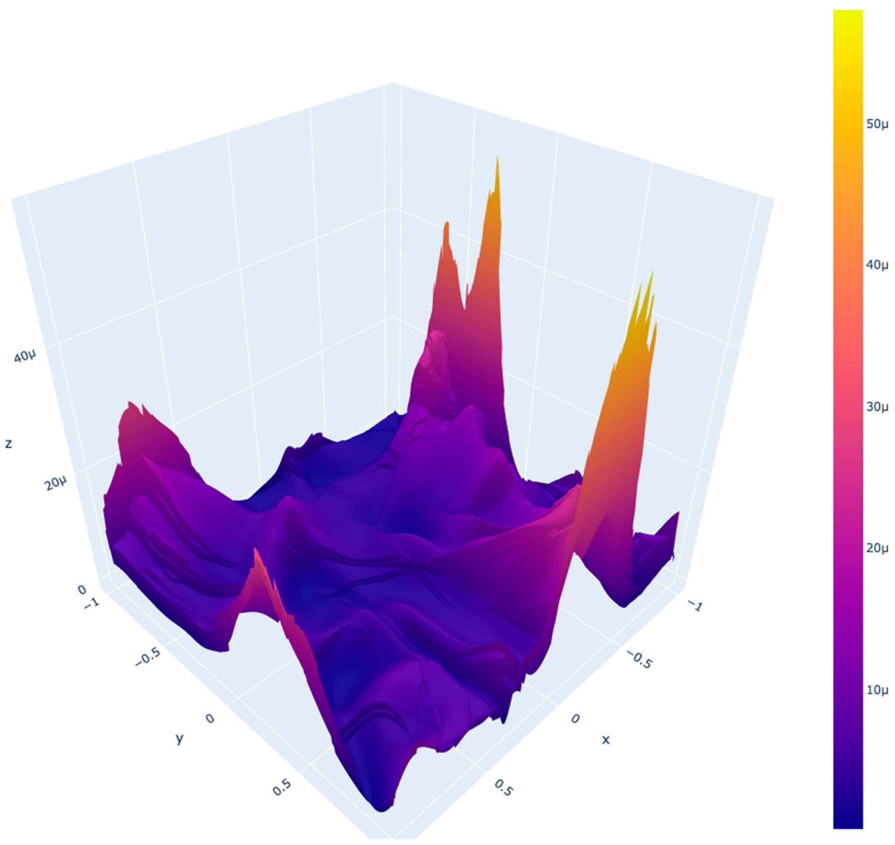

Figure 8. QNN classification curvature plot

smoothly varying surfaces also provides strong evidence to support the claim that quaternion optimization models provide better generalization performance versus equivalent real-valued models.

4.2. Time Series Prediction Results

Table 4 contains the final tuned hyperparameters for the QNN prediction model and the real-valued network prediction model. As with the classification experiment, the tuned hyperparameters for the prediction models shared many similarities, yet the QNN contained less than one fourth of the hidden neurons compared to the real-valued model. The training curves for the two models are shown separately in Figs. 14 and 15 due to the large differences in scale of the loss values between the two models. Finally, the test set accuracy scores for both models using a variety of metrics are shown in Table 5.

Figure 14 indicates that after the first training epoch, the QNN was able to achieve near-zero Mean Square Error loss on the 1000 step prediction task, effectively learning the complete underlying dynamics of the Lorenz attractor system. On the other hand, the real-valued model's test and training

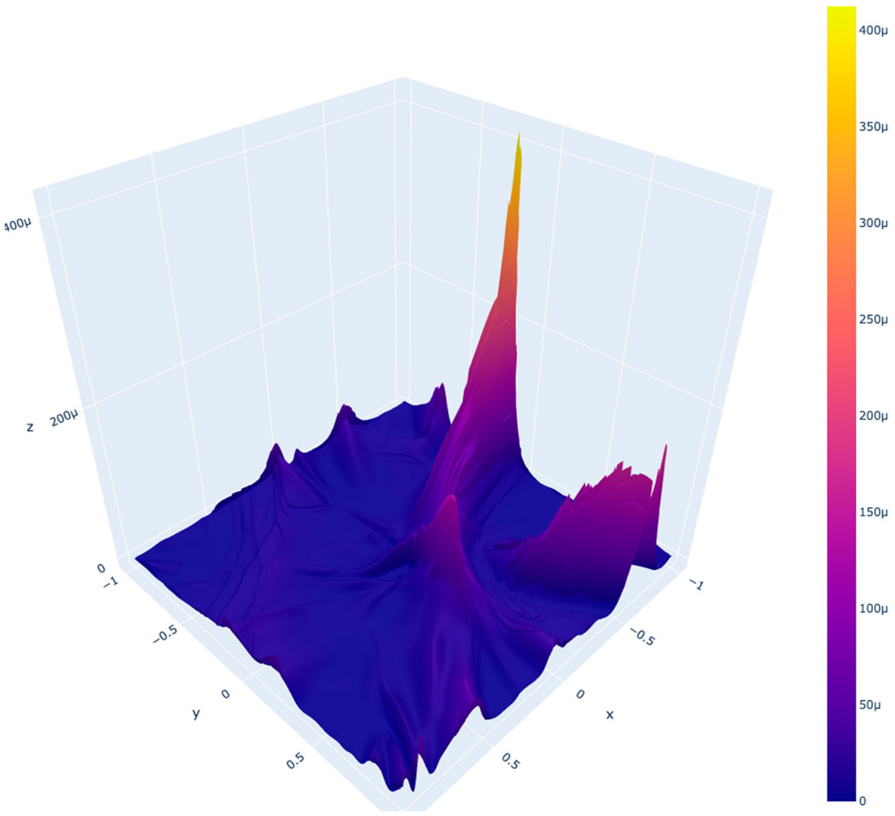

Figure 9. NN classification curvature plot

Table 3. Mean curvature paired t-test results for the classifier networks

Architecture	Mean	SD	t-statistic	p-value
NN	1.946×10^{-5}	1.47×10^{-5}	1.769	0.0874
QNN	1.382×10^{-5}	8.05×10^{-6}		

loss remained well above zero throughout the entire training process while overfitting slightly to the training data. These results are further illuminated by the test set error metrics shown in Table 5. The error values in Table 5 represent the average loss value across each time series in the test data (i.e. the mean of the mean error per time series). Across each metric, the QNN outperforms the real-valued model by a vast margin.

The projected loss surface plots for each model are shown in Figs. 18 and 19, respectively. Note that each figure displays the projected loss in terms of the root mean square error (RMSE), as the raw mean square error (MSE) values for the real-valued network grew rapidly in every direction away from

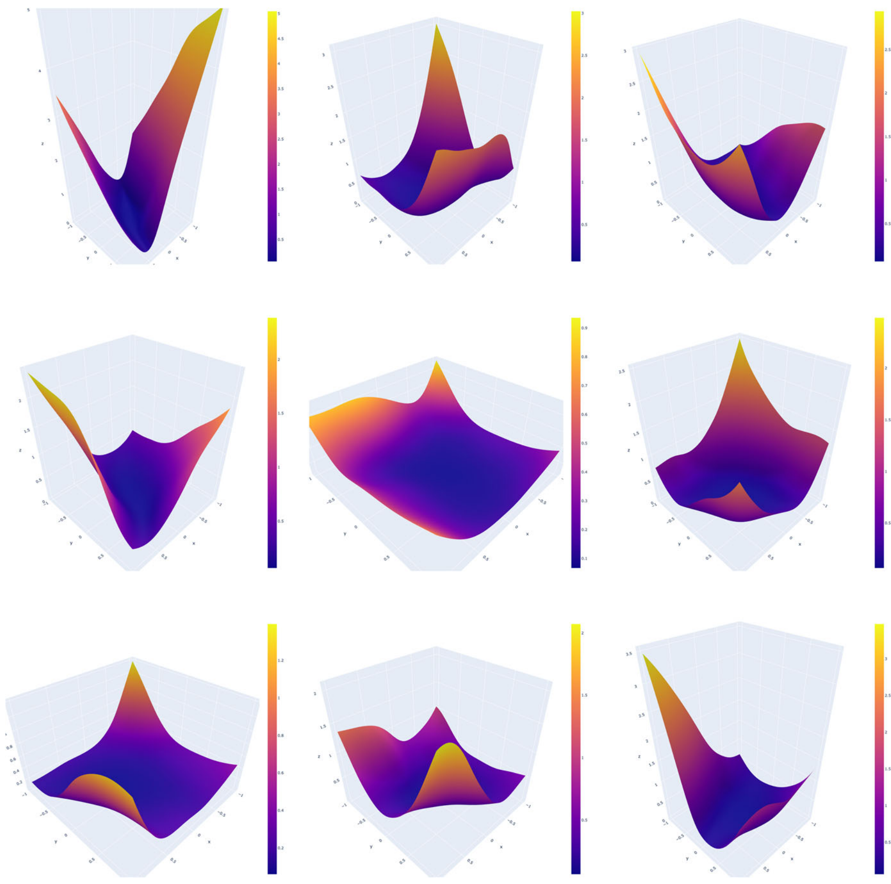

Figure 10. QNN classifier projected surface plots

Table 4. Prediction hyperparameter tuning results

	Model	
	QNN	NN
Hyperparameter	Value	Value
Activation function	Sigmoid	Swish
Optimizer	SGD	SGD
Neurons	32	148
Epochs	210	230
Learning rate	0.00273	0.00253

the minimizer centered at the origin. Even when viewed in terms of RMSE, the differences in scale between the QNN loss values and the real-valued network loss values are striking. The ability of the QNN to effectively replicate the underlying dynamics of the Lorenz system of differential equations enables

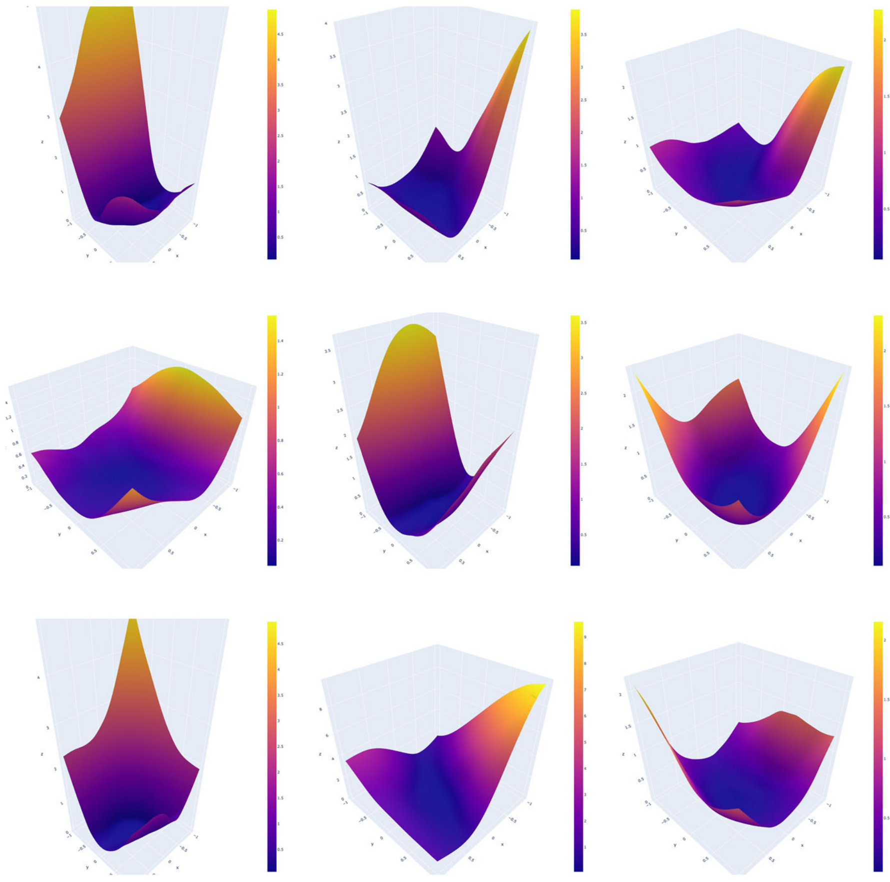

Figure 11. NN classifier projected surface plots

Table 5. Error metrics for the prediction models

Model	MSE	RMSE	MAE	MAPE
QNN predictor	1.63×10^{-13}	3.80×10^{-7}	1.25×10^{-7}	5.23×10^{-8}
NN predictor	33.757	5.456	3.763	1.400

the QNN to accurately predict the time series output based on a relatively short input of 10 timesteps.

The real-valued network does not appear to exhibit the same behavior and struggles to capture any of the relationships in the underlying system of equations. Plots of predicted time series values from the test set for each network are shown in Figs. 16 and 17. The QNN model produces a predicted time series that is indistinguishable from the true series. The real-valued model produces a predicted time series that generally follows the shape of the true series but contains a high degree of noise in each prediction.

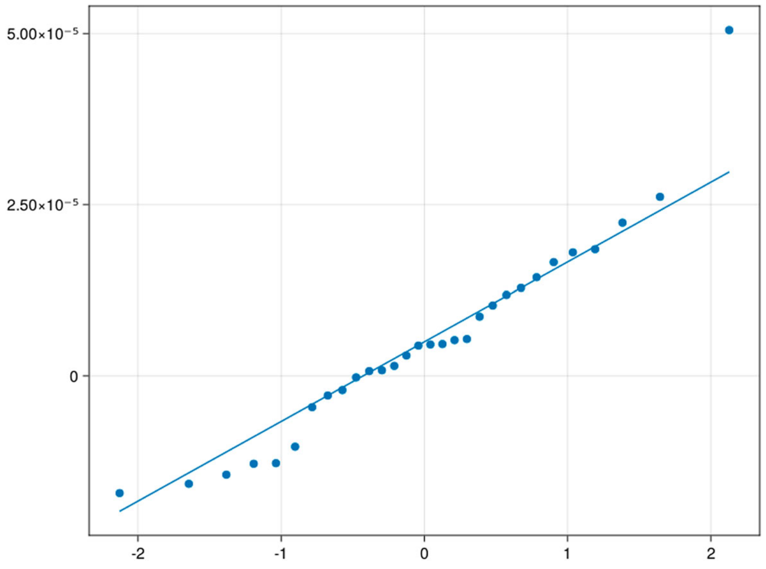

Figure 12. QQ plot of paired differences

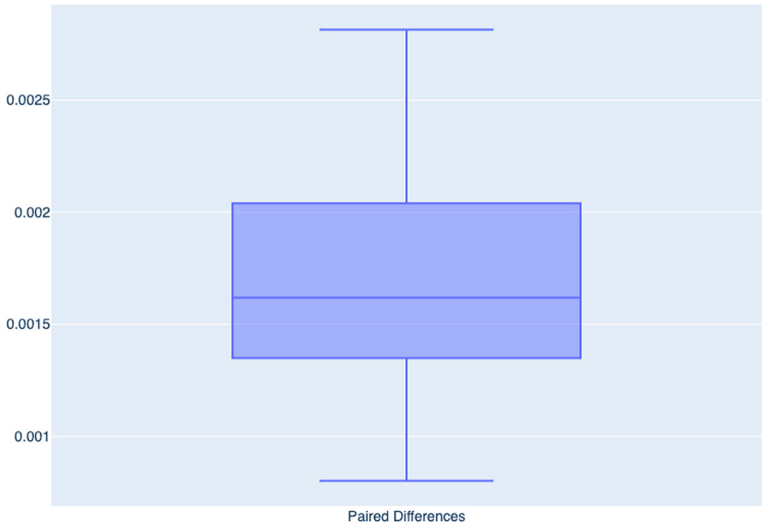

Figure 13. Boxplot of paired differences

The estimated surface curvature plots for both prediction models are shown in Figs. 20 and 21, respectively. Figure 20 captures the estimated curvature of the QNN model. Similar to the classification models, the estimated curvature values produce sharp spikes of curvature in areas that correspond to regions on the loss surface plots with sharp changes in the surface. In this

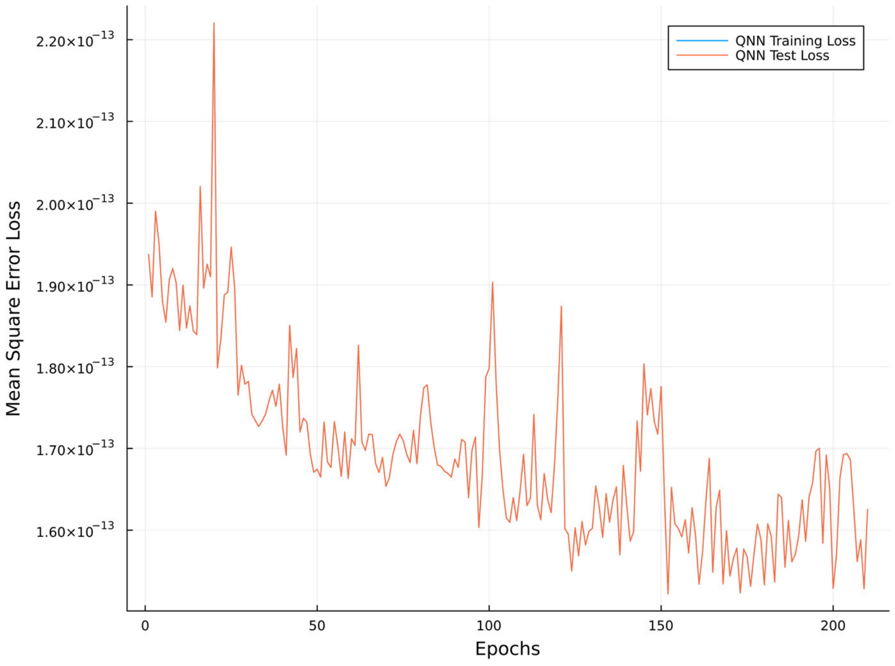

FIGURE 14. QNN training and test loss

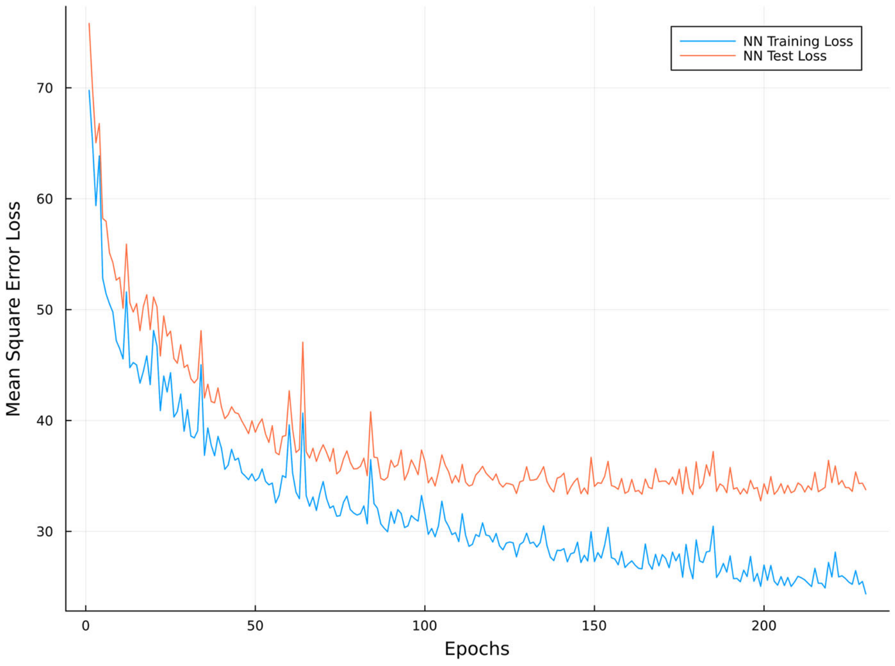

FIGURE 15. NN training and test loss

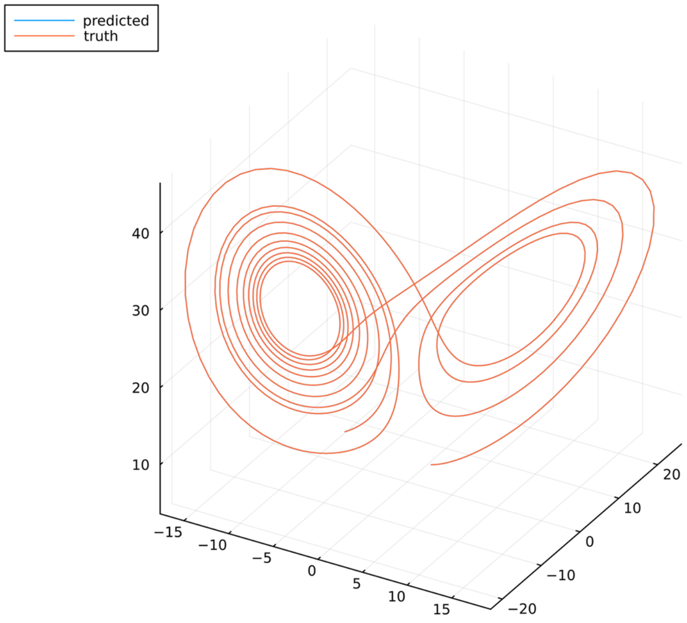

FIGURE 16. QNN predicted time series

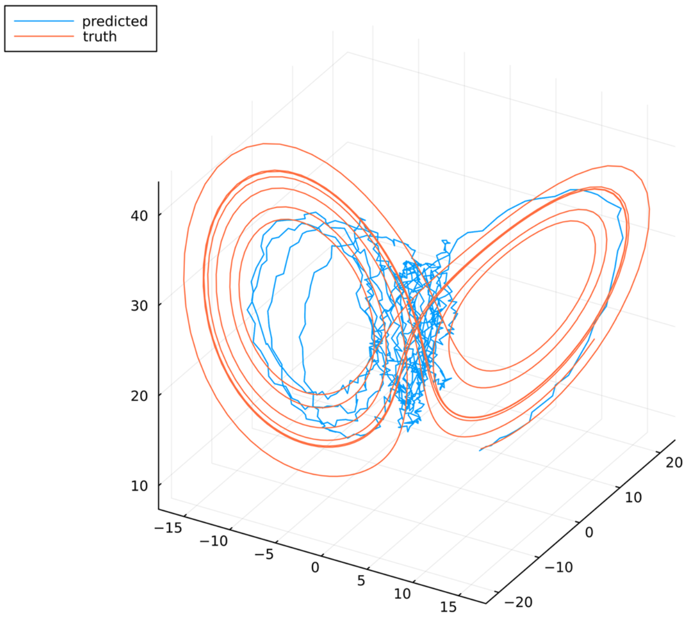

FIGURE 17. NN predicted time series

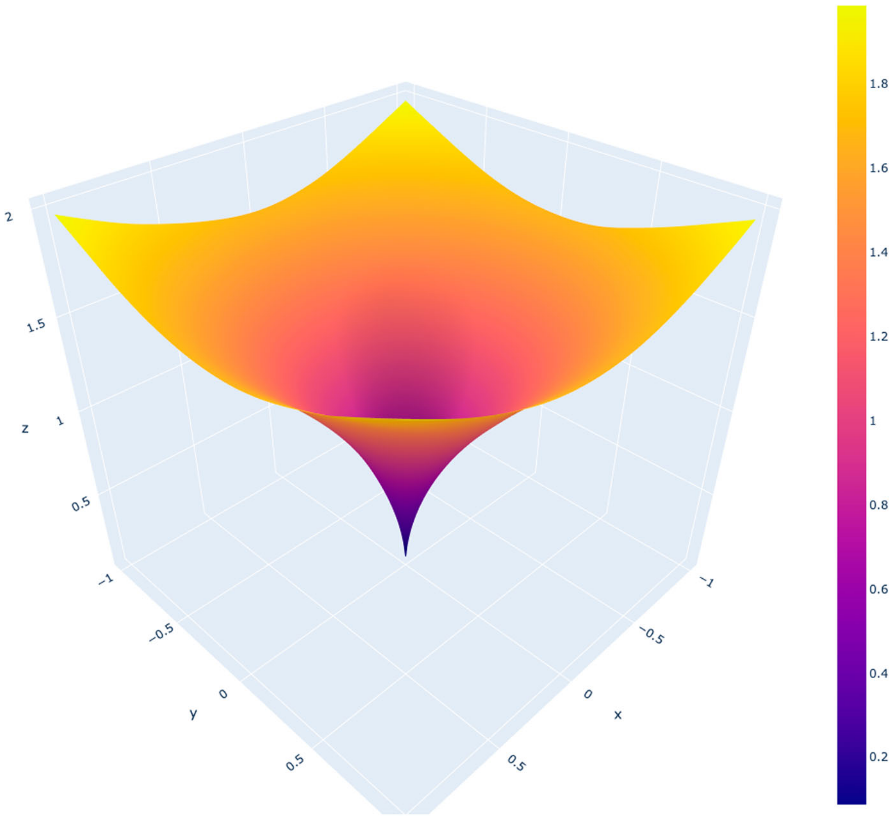

FIGURE 18. QNN prediction 2D loss surface plot

instance, the only region of high curvature is the very narrow area surrounding the origin.

The estimated curvature for the real-valued model is shown in Fig. 21. In contrast with the quaternion model, the real-valued curvature estimates contain a high degree of noise across the entire surface. The spikes in the curvature at each corner of the plot correspond to the rapid increase in loss values near the edges of the loss surface plot in Fig. 19. However, the large increase in curvature near the origin and the noise in the curvature estimates across the rest of the plot are masked by the sheer scale of the loss values. At the plot resolution, visual assessments of the "smoothness" of the loss surface prove to be misleading, demonstrating the need for quantitative measures of smoothness such as the curvature estimates which successfully reveal the true noise present across the loss surface.

The QQ plot and boxplot of the paired mean differences between the QNN curvature and real-valued network curvature values are shown in Figs. 22 and 23. The QQ plot does not show any cause for concern regarding the normality of the paired differences, while the boxplot indicates that there are no

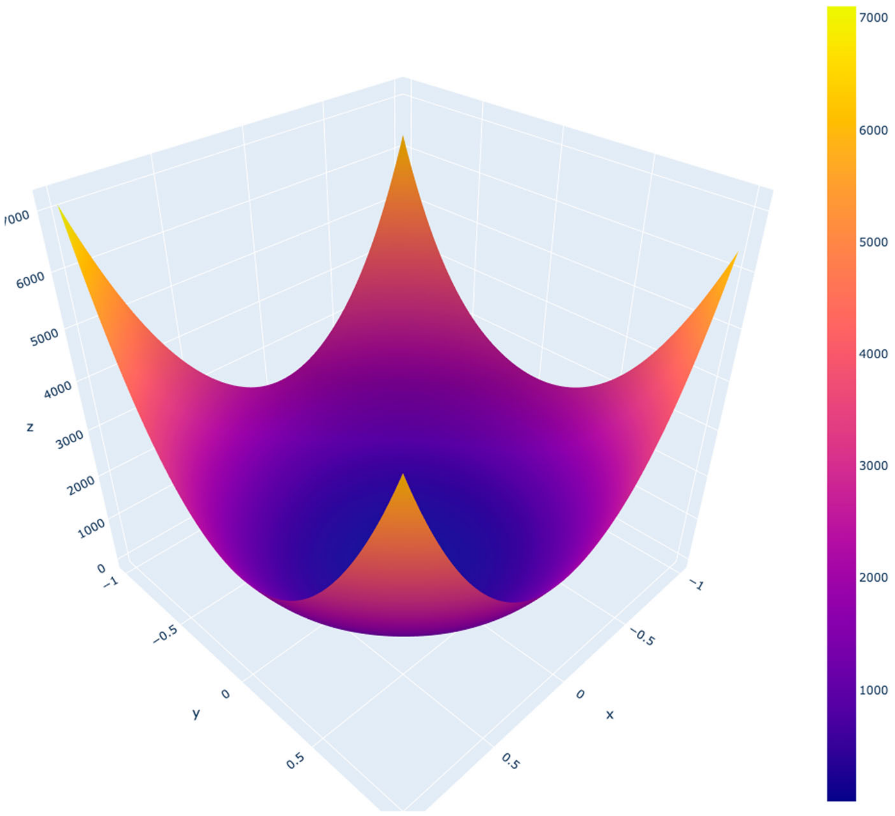

Figure 19. NN prediction 2D loss surface plot

Table 6. Mean curvature paired t-test results for the prediction networks

Architecture	Mean	SD	t-statistic	p-value
NN	0.0503	0.00167	164.129	$< 1 \times 10^{-43}$
QNN	0.000112	6.62×10^{-7}		

outliers among the difference data. The results of the paired *t*-test are summarized in Table 6. The *t*-test indicates that, with very high confidence, there are statistically significant differences in the curvature, and hence smoothness, of the quaternion loss surfaces versus the real-valued loss surfaces for the time series prediction models.

5. Conclusions and Future Work

The classification and prediction experiments in this study provide strong evidence to suggest that QNNs perform optimization over a smoother loss

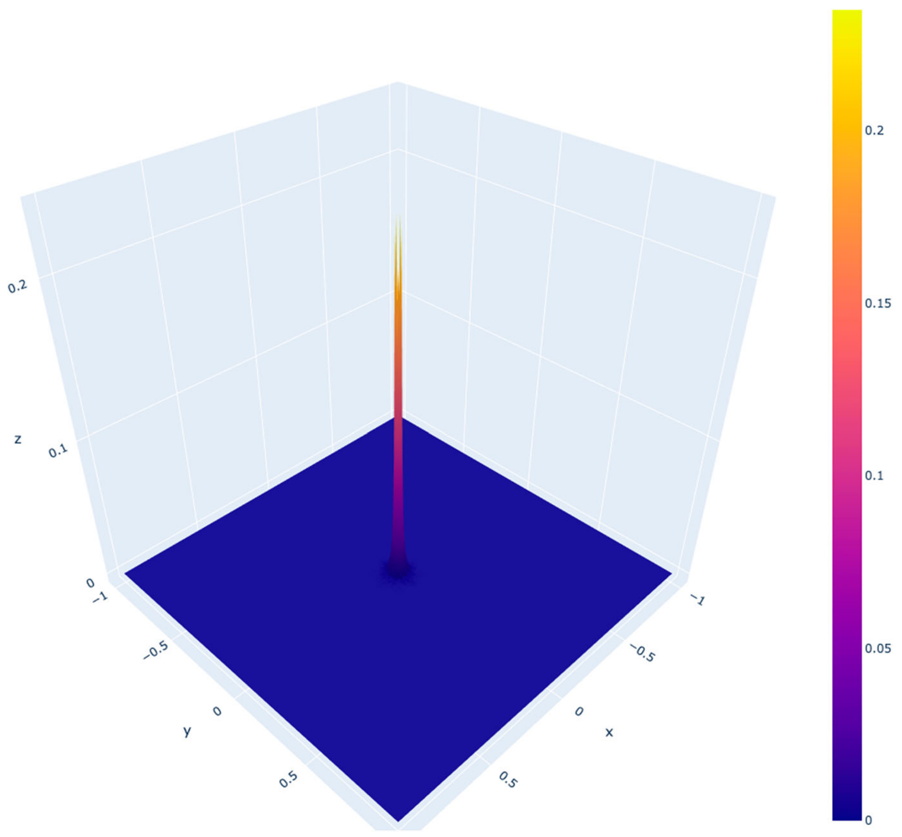

FIGURE 20. QNN prediction curvature plot

surface than standard real-valued neural networks. These results indicate that QNNs provide benefits both in terms of the structure that they provide to certain problems, such as color image processing tasks and any tasks involving rotations in $\mathbb{R}^3$, as well as the overall process of optimization itself. The quaternion loss surfaces characterized in this research provide potential insight into why QNNs have demonstrated consistent gains across a variety of accuracy and performance metrics in myriad problem domains, including in circumstances where QNNs provide no discernible structural benefit over standard real-valued neural networks.

In addition to these insights, the main contribution of this research is the introduction and evaluation of surface curvature estimates as a metric for characterizing the loss surfaces of any optimization problem. In contrast with prior methods, the surface curvature estimation algorithm presented in this work does not require any second order derivative information and instead relies solely on the projected loss surface point cloud data in $\mathbb{R}^3$. This provides a comparison metric that is valid for QNNs, geometric algebra-based networks, Clifford algebra networks, and any other optimization technique

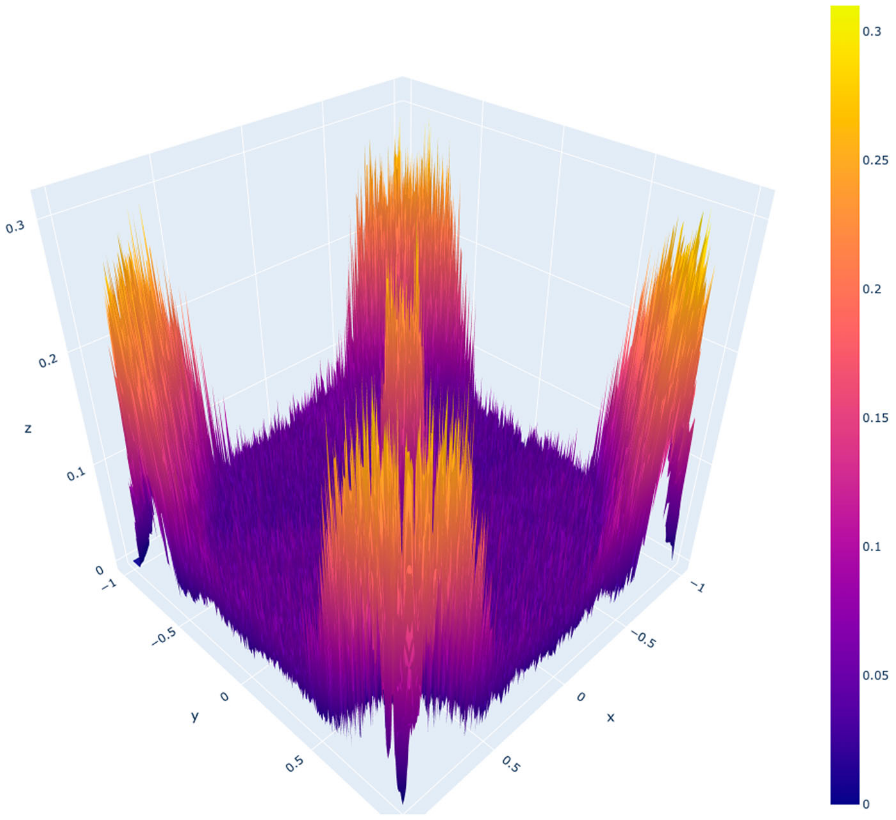

FIGURE 21. NN prediction curvature plot

that admits a high-dimensional loss function. This technique allows for robust comparisons of the effects that different architecture choices, hyperparameter settings, or optimization methods have on the resulting "smoothness" of a loss surface.

5.1. Limitations and Future Work

The experiments presented here are necessarily limited in scope. This research was restricted to an examination of loss surfaces of fully tuned and trained models. The QNN outperformed the real-valued model by such a significant margin in the regression experiment that the resulting loss and surface plots may be biased to favor the QNN model. The evaluation of loss surfaces of various models throughout the entirety of the training process remains an open question and is necessary for understanding not only the final performance of quaternion-based models, but also the progression of quaternion loss landscapes over the duration of the training process. In addition, all models assessed in this work utilized a fixed architecture with two hidden layers. Based on [22], the authors expect that deeper network architectures will lead

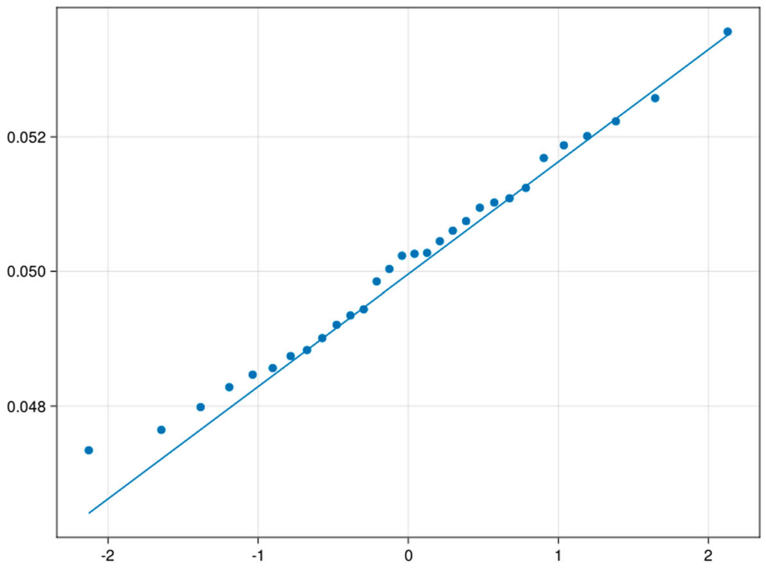

FIGURE 22. QQ plot of paired differences

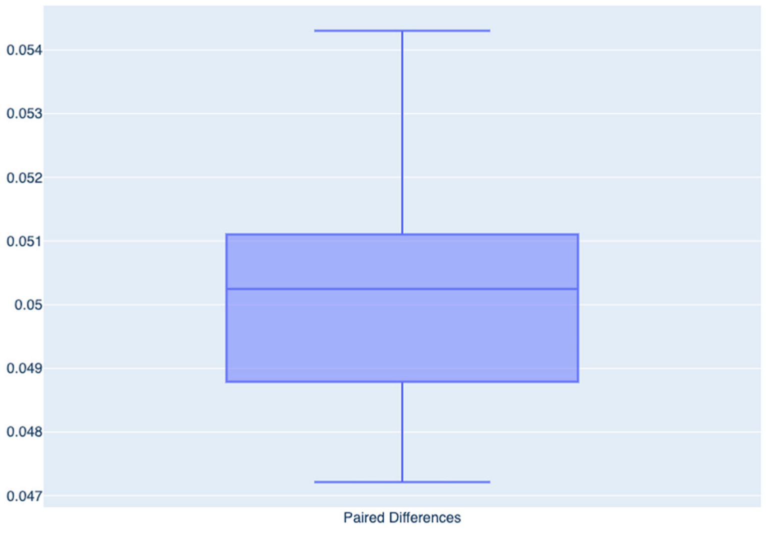

FIGURE 23. Boxplot of paired differences

to loss surfaces with greater degrees of surface curvature, and a formal experimental design sampling across a range of architectural choices (e.g., number of layers, number of neurons per layer) is logical follow-on research.

While regression and binary classification represent the two primary tasks of classical machine learning techniques, multi-class classification is a staple of modern machine learning research that has yet to be explored using the loss surface characterization and curvature estimation techniques presented here. In addition, this research did not consider any of the many specialized neural network architectures such as convolutional neural networks, recurrent neural networks, or transformer networks that have exploded in popularity in recent years. Quaternion versions of each of these architectures have been introduced in the literature and a full exploration of the optimization characteristics of these networks will be vital in understanding the benefits of such quaternion-based networks.

There is a growing interest in QNNs and other hypercomplex neural network architectures. However, there are two significant implementation aspects that are currently hindering progress in these key research areas; first, a lack of automatic differentiation tools for any hypercomplex algebras significantly slows the prototyping and experimentation of different neural network architectures. Second, a lack of optimized Basic Linear Algebra Subprogram (BLAS) routines for matrix multiplication in hypercomplex algebras significantly hampers any attempt to scale hypercomplex neural networks to large datasets. While researchers have attempted to address both of these issues (see [35,38,39]), progress remains limited in both areas. As research into hypercomplex neural networks expands, a robust automatic differentiation method and cross-platform BLAS routines will be vital in enabling the growth of the field.

Finally, the surface curvature estimation techniques presented here have broad applicability to a wide array of machine learning problems. While the primary use case presented in this research was towards evaluating the curvature of quaternion neural network loss surfaces where access to the full Hessian is limited, these techniques are by no means restricted to quaternion or hypercomplex algebraic networks. Indeed, these techniques are applicable to any high-dimensional optimization problem. Future applications of surface curvature estimates should explore the full impact that the hyperparameters of the curvature estimation algorithm have on the resulting curvature estimates, as well as the true nature of the relationship of the Hessian of the loss function to the curvature of the resulting surface.

Data and code files for this research are available upon request.

References

[1] Abiodun, O.I., Jantan, A., Omolara, A.E., Dada, K.V., Mohamed, N.A., Arshad, H.: State-of-the-art in artificial neural network applications: A survey. Heliyon **4**(11), e00938 (2018). https://doi.org/10.1016/j.heliyon.2018.e00938. http://www.sciencedirect.com/science/article/pii/S2405844018332067

[2] Arena, P., Fortuna, L., Re, R., Xibilia, M.: On the capability of neural networks with complex neurons in complex valued functions approximation. In: 1993 IEEE International Symposium on Circuits and Systems, vol. 4, pp. 2168–2171 (1993). https://doi.org/10.1109/ISCAS.1993.394188

[3] Arena, P., Fortuna, L., Occhipinti, L., Xibilia, M.: Neural networks for quaternion-valued function approximation. In: Proceedings of IEEE International Symposium on Circuits and Systems—ISCAS '94, vol. 6, pp. 307–310 (1994). https://doi.org/10.1109/ISCAS.1994.409587

[4] Arena, P., Baglio, S., Fortuna, L., Xibilia, M.: Chaotic time series prediction via quaternionic multilayer perceptrons. In: 1995 IEEE International Conference on Systems, Man and Cybernetics. Intelligent Systems for the 21st Century, vol. 2, pp. 1790–1794. IEEE, Vancouver (1995). https://doi.org/10.1109/ICSMC.1995.538035. http://ieeexplore.ieee.org/document/538035/

[5] Arena, P., Fortuna, L., Muscato, G., Xibilia, M.G.: Multilayer perceptrons to approximate quaternion valued functions. Neural Netw. **10**(2), 335–342 (1997). https://doi.org/10.1016/S0893-6080(96)00048-2. http://www.sciencedirect.com/science/article/pii/S0893608096000482

[6] Bayro-Corrochano, E.: A survey on quaternion algebra and geometric algebra applications in engineering and computer science 1995–2020. IEEE Access 1–1 (2021). https://doi.org/10.1109/ACCESS.2021.3097756 (conference name: IEEE Access)

[7] Benvenuto, N., Piazza, F.: On the complex backpropagation algorithm. IEEE Trans. Signal Process. **40**(4), 967–969 (1992). https://doi.org/10.1109/78.127967

[8] Bill, J., Cox, B.A., Champagne, L.: A comparison of quaternion neural network backpropagation algorithms. Expert Syst. Appl. 120448 (2023)

[9] Buchholz, S., Sommer, G.: On Clifford neurons and Clifford multi-layer perceptrons. Neural Netw. **21**(7), 925–935 (2008). https://doi.org/10.1016/j.neunet.2008.03.004. https://linkinghub.elsevier.com/retrieve/pii/S0893608008000592

[10] Cao, W., Li, S., Zhong, J.: Qmednet: a quaternion-based multi-order differential encoder-decoder model for 3D human motion prediction. Neural Netw. **154**, 141–151 (2022)

[11] Chollet, F.: Deep Learning with Python. Simon and Schuster, New York (2021)

[12] Choromanska, A., Henaff, M., Mathieu, M., Arous, G.B., LeCun, Y.: The loss surfaces of multilayer networks. In: Artificial Intelligence and Statistics, pp. 192–204. PMLR (2015)

[13] Cybenko, G.: Approximation by superpositions of a sigmoidal function. Math. Control Signals Syst. **2**(4), 303–314 (1989). https://doi.org/10.1007/BF02551274

[14] Dauphin, Y.N., Pascanu, R., Gulcehre, C., Cho, K., Ganguli, S., Bengio, Y.: Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. Adv. Neural Inf. Process. Syst. **27** (2014)

[15] Fort, S., Scherlis, A.: The goldilocks zone: towards better understanding of neural network loss landscapes. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33(01), pp. 3574–3581 (2019). https://doi.org/10.1609/aaai.v33i01.33013574. https://ojs.aaai.org/index.php/AAAI/article/view/4237 (number: 01)

[16] Gaba, E.: View of the planes establishing the main curvatures on a minimal surface. https://commons.wikimedia.org/wiki/File:Minimal_surface_curvature_planes-en.svg (2006)

[17] Goodfellow, I.J., Vinyals, O., Saxe, A.M.: Qualitatively characterizing neural network optimization problems. In: Technical Report. arXiv:1412.6544 [cs, stat] (2015)

[18] Hornik, K., Stinchcombe, M., White, H.: Multilayer feedforward networks are universal approximators. Neural Netw. **2**(5), 359–366 (1989). https://doi.org/10.1016/0893-6080(89)90020-8. http://www.sciencedirect.com/science/article/pii/0893608089900208

[19] Im, D.J., Tao, M., Branson, K.: An Empirical Analysis Of Deep Network Loss Surfaces, p. 20 (2017)

[20] Keskar, N.S., Mudigere, D., Nocedal, J., Smelyanskiy, M., Tang, P.T.P.: On large-batch training for deep learning: generalization gap and sharp minima. In: Technical Report. https://doi.org/10.48550/arXiv.1609.04836. arXiv:1609.04836 [cs, math] (2017)

[21] Kühnel, W., Hunt, B.: Differential Geometry: Curves-Surfaces-Manifolds. American Mathematical Society, New York (2005)

[22] Li, H., Xu, Z., Taylor, G., Studer, C., Goldstein, T.: Visualizing the loss landscape of neural nets. In: Advances in Neural Information Processing Systems, vol. 31. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2018/hash/a41b3bb3e6b050b6c9067c67f663b915-Abstract.html (2018)

[23] Lorenz, E.N.: Deterministic nonperiodic flow. J. Atmos. Sci. **20**(2), 130–141 (1963)

[24] Moya-Sánchez, E.U., Xambó-Descamps, S., Pérez, A.S., Salazar-Colores, S., Cortés, U.: A trainable monogenic convnet layer robust in front of large contrast changes in image classification. IEEE Access **9**, 163735–163746 (2021)

[25] Nelson, E.: A proof of Liouville's theorem. Proc. Am. Math. Soc. **12**(6), 995 (1961). https://doi.org/10.1090/S0002-9939-1961-0259149-4. https://www.ams.org/proc/1961-012-06/S0002-9939-1961-0259149-4/

[26] Nitta, T.: A quaternary version of the back-propagation algorithm. In: Proceedings of ICNN'95—International Conference on Neural Networks, vol. 5, pp. 2753–2756 (1995). https://doi.org/10.1109/ICNN.1995.488166

[27] OpenAI: Gpt-4 Technical Report (2023)

[28] Parcollet, T., Morchid, M., Linar's, G.: A survey of quaternion neural networks. Artif. Intell. Rev. **53**(4), 2957–2982 (2020). https://doi.org/10.1007/s10462-019-09752-1

[29] Pauly, M., Gross, M., Kobbelt, L.: Efficient simplification of point-sampled surfaces. In: IEEE Visualization, 2002. VIS 2002., pp. 163–170 (2002). https://doi.org/10.1109/VISUAL.2002.1183771

[30] Rusu, R.B., Marton, Z.C., Blodow, N., Dolha, M., Beetz, M.: Towards 3D point cloud based object maps for household environments. Robot. Auton. Syst.

56(11), 927–941 (2008). https://doi.org/10.1016/j.robot.2008.08.005. https://www.sciencedirect.com/science/article/pii/S0921889008001140

[31] Rusu, R.B.: Semantic 3D object maps for everyday manipulation in human living environments. KI-Künstliche Intell. **24**(4), 345–348 (2010). https://doi.org/10.1007/s13218-010-0059-6

[32] Safran, I., Shamir, O.: On the quality of the initial basin in overspecified neural networks. In: Proceedings of the 33rd International Conference on Machine Learning, pp. 774–782. PMLR (2016). https://proceedings.mlr.press/v48/safran16.html (ISSN: 1938-7228)

[33] Sagun, L., Evci, U., Guney, V.U., Dauphin, Y., Bottou, L.: Empirical analysis of the Hessian of over-parametrized neural networks. In: Technical Report. https://doi.org/10.48550/arXiv.1706.04454. arXiv:1706.04454 [cs] (2018)

[34] Santurkar, S., Tsipras, D., Ilyas, A., Madry, A.: How does batch normalization help optimization? In: Advances in Neural Information Processing Systems, vol. 31. Curran Associates, Inc. (2018). https://proceedings.neurips.cc/paper/2018/hash/905056c1ac1dad141560467e0a99e1cf-Abstract.html

[35] Tingelstad, L., Egeland, O.: Automatic multivector differentiation and optimization. Adv. Appl. Clifford Algebras (2017). https://doi.org/10.1007/s00006-016-0722-6

[36] Tripathi, B., Srivastava, N., Tiwar, A.K., Sharma, A.: Quaternion neural network architectures in various application domains. In: 2023 International Conference on IoT, Communication and Automation Technology (ICICAT), pp. 1–9. IEEE, New York (2023)

[37] Ujang, B.C., Took, C.C., Mandic, D.P.: Split quaternion nonlinear adaptive filtering. Neural Netw. **23**(3), 426–434 (2010)

[38] Williams-Young, D., Li, X.: On the Efficacy and High-Performance Implementation of Quaternion Matrix Multiplication. arXiv:1903.05575 [cs] (2019)

[39] Williams-Young, D.: Wavefunction91/HAXX (2020). https://github.com/wavefunction91/HAXX. Accessed 19 Apr 2017

[40] Wolberg, W., Mangasarian, O., Street, N., Street, W.: Breast Cancer Wisconsin (Diagnostic). UCI Machine Learning Repository (1995). https://doi.org/10.24432/C5DW2B

Jeremiah Bill and Bruce Cox
Air Force Institute of Technology
2950 Hobson Way
Wright-Patterson AFB
OH 45433
USA
e-mail: `jeremiah.bill@us.af.mil`

Bruce Cox
e-mail: `bruce.cox@afit.edu`

Received: July 28, 2023.
Accepted: January 5, 2024.