CURVATURE-BASED MACHINE LEARNING METHOD FOR AUTOMATED SEGMENTATION OF DENDRITIC SPINES

ABDEL KADER A. GERALDO[†], MICHAEL A. CHIRILLO[§], KRISTEN M. HARRIS[‡], AND THOMAS G. FAI*

Abstract. Recent advances in connectomics have been led by high-resolution reconstruction of large volumes of neural tissues using electron microscopy (EM), providing unprecedented insights into brain structure and function. Dendritic spines—dynamic protrusions on neuronal dendrites—play crucial roles in synaptic plasticity, influencing learning, memory, and various neurological disorders. However, current spine analysis methods often rely on manual annotation of subcellular features, limiting their ability to handle the complexity of spines in dense dendritic networks. This paper introduces a novel automated computational framework that integrates discrete differential geometry, machine learning, and 3D image processing to analyze dendritic spines in these intricate environments. By generating distributions of spine morphology from high resolution images including many thousands of spines, our approach captures subtle variations in spine shapes, offering a nuanced understanding of their roles in synaptic function. This framework is tested on multiple EM datasets, with the aim of enhancing our understanding of synaptic plasticity and its alterations in disease states. The proposed method is poised to accelerate neuroscience research by providing a scalable, objective, and comprehensive solution for spine analysis, uncovering insights into the role of spine geometry for neural function.

1. Introduction

Recent advances in electron microscopy and connectomics have made it possible to reconstruct astonishing large volumes of neural tissue at nanometer resolution (EM) [25, 29]. These breakthroughs have opened new opportunities for studying brain microstructures, particularly dendritic spines, dynamic protursion on dendrites where most excitatory synapses in the brain occur. The morphology of dendritic spines—encompassing shape, size, and density—is highly plastic and closely tied to fundamental neurological processes such as learning and memory [6, 19, 23, 46, 59].

Quantitative studies have shown that fine-scale geometric features of spines, including head volume, neck length, and curvature, are strongly associated with synaptic strength, developmental state, and functional specialization [37–39]. Changes in spine structure have also been linked to drug abuse, environmental influences, and a wide range of neurodevelopmental, neurodegenerative, and psychiatric disorders [16, 35, 57].

Accurate detection and analysis of dendritic spines in three-dimensional reconstructions is therefore essential for advancing our understanding of synaptic organization and plasticity. However, spine identification remains challenging due to their heterogeneous morphologies,

Date: April 9, 2026.

[†]Department of Mathematics, Brandeis University, Waltham, MA, USA.

[§]Department of Biological Sciences, University of Rhode Island, Kingston, RI, USA.

[‡]Department of Neuroscience, University of Texas at Austin, Austin, TX, USA.

*Department of Mathematics and Volen Center for Complex Systems, Brandeis University, Waltham, MA, USA (tfai@brandeis.edu).

dense clustering, and local curvature variations. Manual annotation is highly time-intensive and impractical for large-scale datasets, underscoring the need for automated and reliable computational methods. Several segmentation approaches have been proposed in recent years [4, 5, 41, 45, 47, 50, 52, 54, 56], yet challenges remain in achieving both accuracy and scalability.

In particular, many existing 3D CNN–based methods [2, 10, 34], operate directly on voxel representations, which require dense volumetric grids and therefore scale poorly to large EM datasets. This voxel dependence leads to substantial memory overhead, slow inference, and difficulty capturing fine geometric details such as thin spine necks. More broadly, large EM volumes contain millions of surface elements, creating significant computational constraints for voxel-level processing or repeated high-resolution inference. Together, these challenges underscore the need for automated approaches that incorporate geometric cues and remain robust across large, heterogeneous datasets.

To address these challenges, we introduce a method for automated spine segmentation that combines discrete differential geometry with deep learning. Our framework begins by preprocessing 3D reconstructions of dendritic segments to smooth the triangulated mesh, reduce noise, and enhance surface quality. From the resulting meshes, we extract geometric descriptors such as Gaussian and mean curvature, along with additional features including distances from the dendritic shaft skeleton and clustering-based descriptors. Together, these features provide a rich representation of both local and global morphology.

Building on this geometric foundation, we developed a series of deep neural network (DNN) architectures to evaluate how feature enrichment impacts segmentation performance. The baseline model, $\mathbf{DNN}_1$, relies primarily on curvature-based descriptors. $\mathbf{DNN}_2$ extends this by incorporating distance-to-skeleton features, improving the separation of spines from shafts. Finally, $\mathbf{DNN}_3$ integrates a set of enriched geometric and topological descriptors, enabling the network to capture subtle morphological variations and complex spine arrangements. This progression of models demonstrates how systematically incorporating new features enhances both training convergence and segmentation accuracy.

2. Methods and materials

In this section, we employ differential geometry to design a deep neural network architecture for the segmentation of dendritic spines. As a first step, we analyze how curvature can contribute to the characterization of dendritic morphology. Building on this analysis, we then develop and explore a deep neural network that leverages the geometric properties of dendrites to achieve accurate segmentation. For convenience, a summary table of the symbols and variables used throughout Sections 2 is provided in Supplemental Table S1.

2.1. Dendritic Morphology Analysis Using Discrete Differential Geometry.

Segmentation of dendritic shafts and spines can be guided by their differential geometric properties, particularly Gaussian and mean curvature. These curvature measures capture local shape variations that distinguish the roughly cylindrical shaft from protruding spines. However, raw curvature values obtained directly from EM reconstructions are often noisy due to the sectioning process and mesh irregularities. To address this, we first smooth the dendritic triangulated surface mesh using discrete differential geometry techniques, then compute curvature values, and finally enhance these through image processing filters. This three-step process—smoothing, curvature computation, and enhancement–provides robust geometric descriptors that form the basis for accurate segmentation.

2.1.1. *Mean and Gaussian curvature.* In this section, we motivate the use of Gaussian and mean curvature values as key variables for segmenting the dendritic shaft. In the study of dendritic morphology, the shaft is typically modeled as an approximately cylindrical structure from which spines protrude. The discrete mean $\mathbf{H}$ and Gaussian curvatures $\mathbf{K}$ computation details are provided in Section S5.

The Gaussian curvature provides insight into the surface's shape at different points. At hyperbolic points, where the surface curves in opposite directions (saddle-like), the Gaussian curvature is negative. Conversely, at elliptical points, where the surface curves uniformly in the same direction (dome-like), the Gaussian curvature is positive. This contrasts with the cylindrical shaft, along which the Gaussian curvature is zero. Applying these principles to dendritic morphology, we observe that along the spine neck and at the intersection between the neck of a spine and the dendritic shaft, the surface exhibits a saddle shape, leading to negative Gaussian curvature. On the other hand, the spine head, with its more spherical or dome-like structure, exhibits positive Gaussian curvature.

The mean curvature further characterizes the surface. On a concave surface, the mean curvature is positive, while on a convex surface, it is negative. This distinction helps to identify regions of significant shape change. For instance, at the transition from the dendritic shaft to the spine neck, the surface is concave, leading to relatively positive mean curvature. In contrast, the spine head, which is a convex region, has relatively negative mean curvature. These curvature measures are most useful for segmentation when the triangular mesh is sufficiently smooth, as smoothness preserves the overall geometry of the dendritic branches while reducing discretization noise. Under these conditions, the curvature fields more accurately capture the cylindrical shape of the dendritic shaft and the geometric deviations introduced by spine necks and spine heads. As a result, the smoothed curvature values provide a more reliable representation of the underlying morphology, which is valuable in distinguishing the shaft from protruding spines.

2.1.2. *Smoothing the Triangulated Surface Mesh.* To analyze the triangulated surface of the dendritic mesh, we first address the inherent roughness introduced by the EM sectioning process. We apply discrete differential–geometry techniques–specifically a discrete approximation of mean-curvature (Willmore) flow–to smooth the triangular surface mesh.

We define k_{bend} as the bending coefficient of the surface. The curvature energy over the surface A is

$$\mathbf{W}_{bend}(\mathbf{X}) = \frac{k_{bend}}{2} \int_A \mathbf{H}^2(\mathbf{X})\, dA, \tag{1}$$

and the associated bending force is obtained as the negative gradient of this energy:

$$\mathbf{F}_{bend}(\mathbf{X}) = -\nabla_{\mathbf{X}} \mathbf{W}_{bend}(\mathbf{X}). \tag{2}$$

The surface is then evolved by a gradient-descent step,

$$\frac{\partial}{\partial t}\mathbf{X} = \mathbf{F}_{bend}(\mathbf{X}), \tag{3}$$

which produces a smoothed version of the mesh. In practice, this smoothing is implemented using standard discrete differential-geometry operators. Although the mesh consists of discrete triangles, smoothing is well-defined: each vertex exchanges curvature information only with the vertices in its 1-ring neighborhood (the set of triangles directly adjacent to it), as illustrated in Figure S1. Updating each vertex using curvature information from its neighbors provides a discrete analogue of smoothing on a continuous surface. This suppresses

high-frequency numerical noise introduced by the discretization while preserving the underlying geometric structure of the dendritic mesh. Further numerical details, following the approach of [55], are provided in Section S5.

The results of performing this smoothing are illustrated in Figure 1, where we compute the Gaussian and mean curvature of a segment of spiny dendrite. The data was obtained from high-resolution 3D EM reconstructions of dendritic segments in the CA1 region of the hippocampus from P21 rats. Slices were subjected to *in vitro* theta-burst stimulation to induce LTP or to control stimulation, as described in [7–9]. Comparing the curvature before and after smoothing, we observe that the processed mesh exhibits a more easily interpretable profile of the dendritic surface, which can be exploited to improve the accuracy of spine-shaft segmentation.

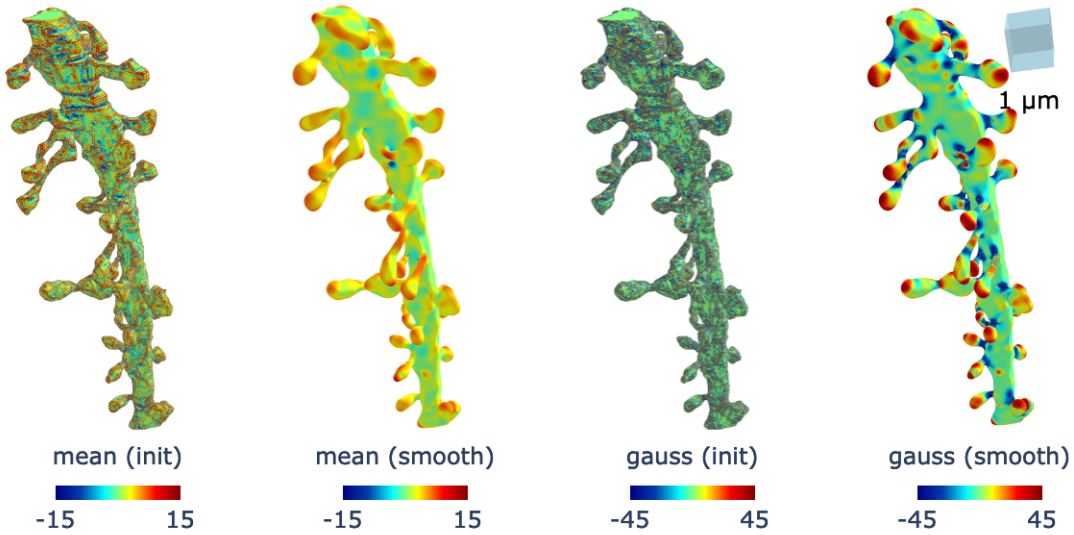

FIGURE 1. Effect of curvature smoothing on dendritic meshes. The plots labeled (init) and (smooth) show the initial dendritic curvature and the curvature after smoothing, respectively. The plots labeled mean and gauss correspond to the mean curvature and Gaussian curvature of a segment of dendrite. Smoothing enhances the curvature profile, producing a more easily interpretable pattern in the mesh that can be effectively leveraged to improve segmentation accuracy. For visualization, Gaussian curvature values are thresholded to remain within an absolute value of 45, while the mean curvature are constrained within an absolute value of 15, thereby highlighting the most relevant mesh faces. The triangular mesh is obtained from CA1 dendritic reconstructions [7–9].

2.1.3. *Enhancement of Curvature through Image Processing.* Here, we provide an intuition for using image processing techniques to enhance the mean and Gaussian curvature values. The goal is to emphasize regions of the surface with significant curvature changes, which are more likely to correspond to dendritic shafts or spines, thereby facilitating segmentation. This forms the foundation for the machine learning methods that will be developed in the segmentation algorithm.

First, let us define the following sigmoidal function, which is often used in image processing to normalize and enhance contrast:

$$\zeta(x) = \frac{1}{1 + \exp(-x)}.$$

This transformation maps all real values into the interval $(0, 1)$. For large positive x, $\zeta(x) \to 1$, while for large negative x, $\zeta(x) \to 0$. Around $x = 0$, the function has its steepest slope, which enhances small variations near zero and makes them more distinguishable.
Applying this transformation to the mean curvature $\mathbf{H}$ and the Gaussian curvature $\mathbf{K}$, we obtain:

$$\widetilde{\mathbf{H}} = \zeta(a_{\mathbf{H}}\mathbf{H} + b_{\mathbf{H}}), \quad \widetilde{\mathbf{K}} = \zeta(a_{\mathbf{K}}\mathbf{K} + b_{\mathbf{K}}), \tag{4}$$

where $a_{\mathbf{H}}$, $b_{\mathbf{H}}$, $a_{\mathbf{K}}$, and $b_{\mathbf{K}}$ are empirically chosen parameters used to emphasize specific geometric features of the dendritic mesh. For example, in the neck region of a spine, we expect relatively high negative Gaussian curvature. By appropriately choosing $a_{\mathbf{K}}$ and $b_{\mathbf{K}}$, these negative values are pushed toward the lower end of the sigmoid, making them stand out more clearly during segmentation.
In contrast, along the cylindrical shaft, the Gaussian curvature is close to zero. Proper tuning ensures that values near zero are mapped consistently with the shaft, so that these regions are correctly identified. For mean curvature, which distinguishes concave and convex regions, the transformation highlights transitions: concave regions (positive mean curvature) are enhanced toward higher sigmoid values, while convex regions (negative mean curvature) are mapped toward lower values.
In practice, within this paper, the quantities $\widetilde{\mathbf{H}}$ and $\widetilde{\mathbf{K}}$ are not computed explicitly. Their formulation serves only to motivate the use of a DNN as an approximation function for the segmentation task, with a sigmoidal activation used in the output layer. The last two panels of Figure 3 show the spine and shaft probability outputs of the DNN that will be examined next. These results demonstrate a substantial improvement in segmentation quality compared to the manually tuned baseline. The overall processing pipeline—from mesh smoothing and geometric feature extraction to DNN-based prediction—is summarized in the flowchart shown in Figure S2.

2.2. **Deep Neural Network Approach for Spine and Shaft Analysis.** In the previous section, we provided an intuitive explanation of how segmentation can be enhanced using image processing techniques. We then introduced empirical filtering parameters that can improve segmentation quality. Instead of manually selecting these parameters and explicitly computing $\widetilde{\mathbf{H}}$ and $\widetilde{\mathbf{K}}$, we can leverage Deep Neural Networks (DNNs) to learn an optimal segmentation approximation function based on $\mathbf{H}$ and $\mathbf{K}$. At the same time, DNNs allow us to incorporate non-linearities that further enhance segmentation performance.
We analyze three different DNN architectures. The first network is designed to support the second by assisting in the extraction of the shaft skeleton, while the third network relies solely on external Python libraries for skeletonization. Machine learning methods have also been employed for dendrite segmentation of dendritic spines obtained from confocal reconstruction images using Convolution Neural Networks CNNs [52]. In this work, we adopt a simple DNN architecture inspired by the physics-informed neural network (PINN) model [22, 24, 30, 32, 33, 42–44], which has been widely applied to approximate ordinary and partial differential equations. These models leverage their well-known ability to serve as universal function approximators [22]. Given the geometrical characteristics of the dendritic triangular mesh, this approximation capability of DNNs constitutes a fundamental tool in

our algorithm. Here, we present the architectures that produced the best results among the different trials.

2.2.1. *Deep Neural Network (*$\mathbf{DNN}_1$*) using the Gaussian and Mean Curvature.* As for the sigmoidal function (4) introduced in the previous section, various model parameters require fine-tuning to enhance the shaft segmentation process. Rather than manually selecting these parameters and computing $\widetilde{\mathbf{H}}$ and $\widetilde{\mathbf{K}}$, we train a deep neural network in a two-step procedure.

We first train a deep neural network, denoted as $\mathbf{DNN}_1$, whose architecture is illustrated in Figure 2. The input layer receives both the mean curvature $\mathbf{H}$ and the Gaussian curvature $\mathbf{K}$ values of the dendritic triangular mesh (after smoothing), together with their squared terms to capture higher-order variations. The input features are preprocessed such that the Gaussian curvature values and their squared terms are thresholded to remain within an absolute value of 45. Specifically, for $\mathbf{K} = (\mathbf{K}_1, \mathbf{K}_2, \ldots, \mathbf{K}_m)$, we enforce $\mathbf{K}_i \in [-45, 45]$. We then compute $\mathbf{K}_i^2$ and threshold its value to the range $[0, 45]$. Similarly, the mean curvature values and their squared terms are constrained within an absolute value of 15, following the same procedure as in the Gaussian curvature case.

These thresholds are applied because curvature values can sometimes become very large, and such extreme values do not significantly improve prediction accuracy but can instead cause instability during training and inference. To further reduce instability, any NaN values, if present, are replaced with the average of the corresponding curvature values.

The network consists of four hidden layers, each containing fifty neurons with ReLU activation functions to introduce non-linearity [30]. The output layer employs a Sigmoid activation function, as described in Section 2.1.3, to classify vertices into dendritic shafts and spines.

The segmentation produced by $\mathbf{DNN}_1$ is not fully satisfactory, as regions within spines are sometimes misclassified as part of the shaft. This misclassification arises because certain spines regions exhibit relatively flat curvature, making them appear similar to shaft regions.

2.2.2. *Skeletonization.* In this section, we describe the process used to obtain the skeletonization of dendritic meshes. is an image processing technique that reduces binary shapes to thin, single-pixel-wide lines while preserving their topological structure and connectivity [31, 60].

The first step in building the skeleton is to ensure that the mesh is *watertight*, meaning that it forms a completely closed surface with no gaps, holes, or disconnected edges. A watertight mesh guarantees a well-defined interior and exterior, which is essential for reliable geometric processing and for preserving the topological structure during skeletonization. To achieve this, we wrap the existing mesh using the algorithm described in Section S6.1. This procedure converts the surface into a uniformly sampled point cloud, estimates and orients normals, and then applies Poisson surface reconstruction to generate a closed, watertight representation.

In addition to Poisson reconstruction, we also evaluated the *alpha-wrap* method provided by `PyMeshLab` [11], which is based on the classical α-shape formulation [15]. Alpha wrapping produces well-behaved closures and can preserve fine geometric detail; however, in our experiments it was substantially slower than Poisson surface reconstruction for meshes of this size and resolution. For this reason, we adopted Poisson reconstruction as our primary

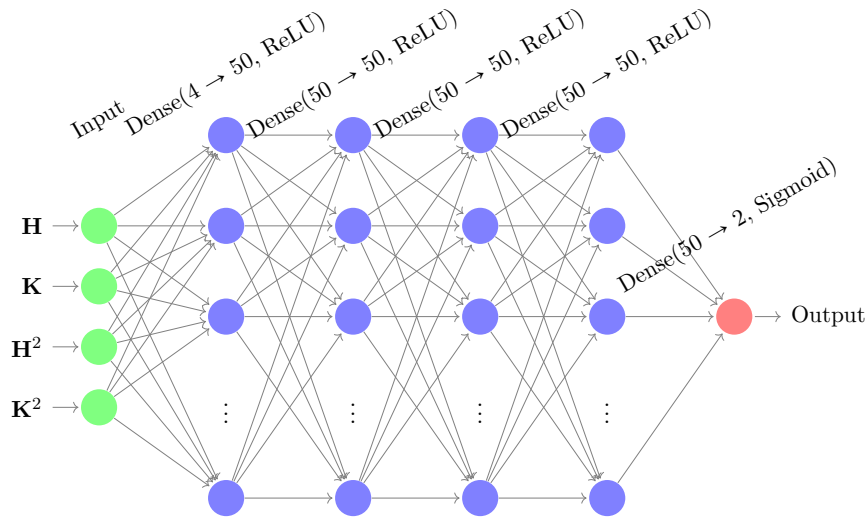

FIGURE 2. Deep Neural Network ($\mathbf{DNN}_1$) architecture used to segment the dendrite shaft. The input layer consists of the mean curvature $\mathbf{H}$ and the Gaussian curvature $\mathbf{K}$, computed after smoothing, and their squares. The network includes four hidden layers, each with fifty neurons and ReLU activation functions. The output layer employs a sigmoidal activation function.

wrapping method. For completeness, the alpha-wrap implementation is also included in Section S6.1.

Once a watertight mesh is obtained, we compute its skeleton using the `scikit-image` skeletonization package, which implements algorithms from [31,60]. Further implementation details are provided in Section S6. This simplified representation captures the essential branching geometry of dendrites for subsequent segmentation and morphological analysis.

2.2.3. *Deep Neural Network ($\mathbf{DNN}_2$) using Shaft Skeleton.* We can enhance dendritic spine–shaft segmentation by incorporating the shaft skeleton. This involves using the distance between the shaft skeleton vertices and the dendritic branch as an additional input to the DNN. For this purpose, we use the shaft segmented with $\mathbf{DNN}_1$ together with the skeletonization procedure outlined in Section 2.2.2.

To begin, we consider the shaft-segmented meshes obtained from $\mathbf{DNN}_1$. Since this mesh results from the dendritic branch after spine regions have been segmented, it is no longer watertight. Therefore, we apply the procedure described earlier to make the mesh watertight. Once the shaft mesh skeletonization is completed, we compute the distance $\mathbf{D}$ between the shaft skeleton and the dendritic branch mesh vertices. Here, for each vertex $\mathbf{X}_l$, $l \in \{1, 2, \ldots, n\}$ in the dendritic triangular mesh with n vertices, and $\{\mathbf{V}_1, \mathbf{V}_2, \ldots, \mathbf{V}_v\}$ the set of shaft skeleton vertices, we compute the Euclidean norm:

$$\mathbf{D}_l \;=\; \min_{1 \leqslant j \leqslant v} \left\| \mathbf{V}_j - \mathbf{X}_l \right\|_2 , \qquad \mathbf{D} \;=\; \big(\mathbf{D}_1, \mathbf{D}_2, \ldots, \mathbf{D}_n\big). \tag{5}$$

With this additional information, we develop an improved deep neural network, $\mathbf{DNN}_2$, whose architecture is shown in Figure S3. This model is similar to $\mathbf{DNN}_1$, except that it incorporates the new input feature. As in the earlier model, the input layer receives both

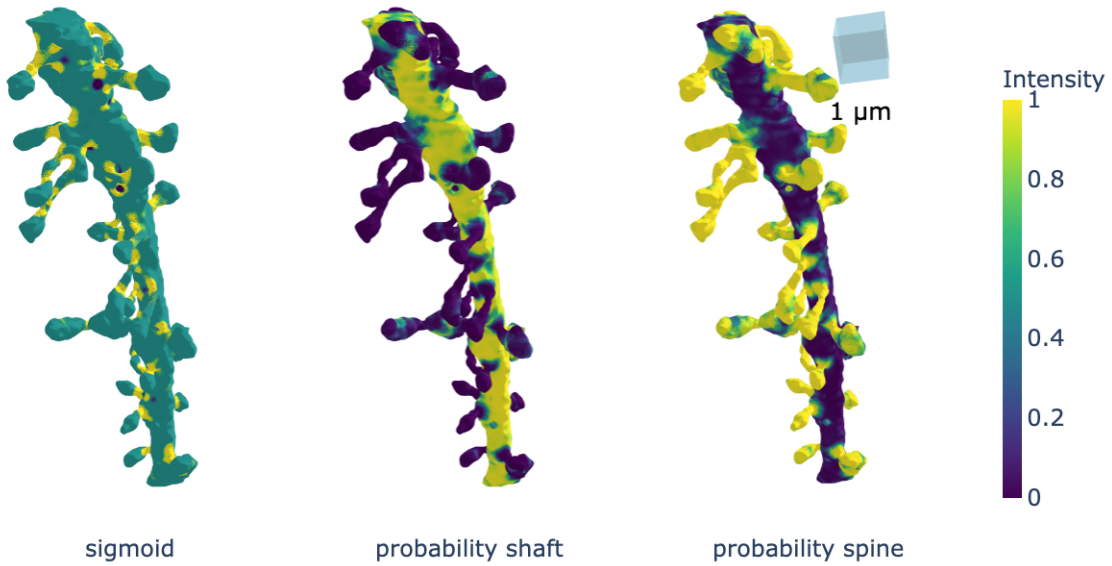

FIGURE 3. Visualization of geometric-based and learned probability representations used for dendritic spine segmentation. The panel labeled (sigmoid) shows the sigmoid of a linear combination of mean intensity and Gaussian curvature, which highlights the neck region and serves as a handcrafted baseline cue for segmentation. The panels labeled probability shaft and probability spine display the class-specific probability outputs of the network $\mathbf{DNN}_3$ for shaft and spine, respectively. Compared with the (sigmoid) representation, the learned probabilities provide a clearer and more discriminative separation of spine and shaft regions. The triangular mesh is obtained from CA1 dendritic reconstructions [7–9].

the Gaussian and mean curvature values of the dendrite, along with the squared values of these curvatures as additional features. The network consists of four hidden layers, each containing fifty neurons with ReLU activation functions to introduce non-linearity. The output layer employs a Sigmoid activation function, as discussed in Section 2.1.3, to classify vertices into dendritic shafts or spines.

2.2.4. *Dendrite Spine-Shaft Segmentation Using Dendritic Branch Regions (*$\mathbf{DNN}_3$*).* To improve segmentation accuracy, we introduce a deep neural network ($\mathbf{DNN}_3$) that incorporates regional information from dendritic branch segments. These segments are derived from the dendrite skeleton, which serves as a structural reference for spatial organization. For this model, once the skeleton is extracted, we compute the shortest distance from each mesh vertex to its nearest skeleton point. These distances are then clustered using the K-means algorithm to partition the dendritic mesh into multiple regions. This regional segmentation enables us to distinguish between different parts of the dendrite shaft, spine neck, and spine head—regions that are critical for accurate classification.
As shown in Figure 4, to capture variations in dendritic morphology we apply K-means clustering with multiple values of k. We denote these segmentation features by $\mathbf{S}^k$, where $k \in \{2, 3, \ldots, 10\}$ corresponds to the number of clusters. In particular, the value of each feature $\mathbf{S}^k$ is given by the corresponding cluster label. Therefore this model incorporates

9 additional scalar features. In this study, we explore values of k ranging from 2 to 10 to provide a multi-scale representation of regional structure.

In addition to these region-based segmentation features, we incorporate Gaussian curvature and its squared value as geometric descriptors. These measures are particularly effective for identifying neck regions, as discussed in earlier sections. We intentionally exclude mean curvature as we have observed that the mean curvature input increases the probability of misclassifying all dome-like regions as spines, leading to false positives. By omitting mean curvature and focusing on more discriminative features, $\mathbf{DNN}_3$ achieves improved segmentation performance compared to the baseline models.

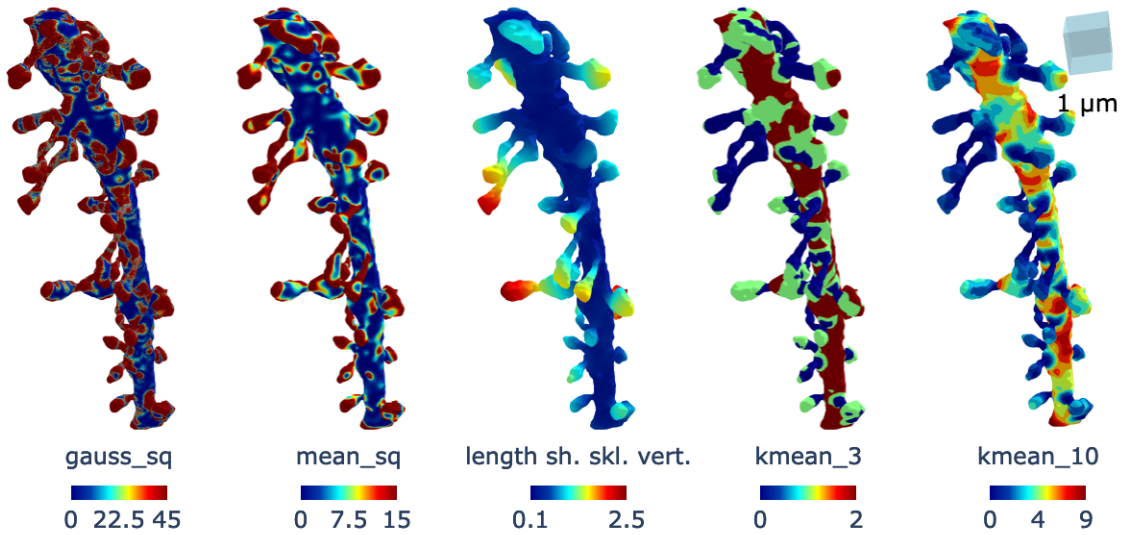

FIGURE 4. Additional features used in the DNN training. The plot labeled gauss_sq and mean_sq show respectively the square of the Gaussian and mean curvature of the dendrite. The plot labeled length sh. skl. vert. indicates the distance from the shaft skeleton to the dendritic mesh vertices, denoted by $\mathbf{D}$. The plot labeled kmean_k illustrates the k-subdivision from K-means clustering, corresponding to the set $\mathbf{S}^k$. The triangular mesh is obtained from CA1 dendritic reconstructions [7–9].

Note that in practice, the K-means algorithm is not deterministic because its initial values depend on a random seed. To mitigate this and obtain consistent region assignments, we perform 50 independent clustering runs and select the labeling corresponding to the run with the lowest final K-means inertia. Since inertia measures the within-cluster compactness optimized by K-means, choosing the run with minimal inertia yields the most stable and coherent clustering among all runs. This procedure effectively filters out seed-dependent fluctuations and provides reproducible region assignments without requiring post hoc consensus voting.

We also considered using the silhouette score as an alternative selection criterion; however, computing silhouette values for all vertices is prohibitively expensive due to the large number of mesh vertices. For this reason, inertia-based selection offered a practical and computationally efficient solution.

2.3. **Loss Function.** We use the binary cross-entropy (BCE) to compute the loss function, and its derivation is presented below.
Let us consider a set of m dendritic meshes, denoted as $\mathcal{D} = (\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_m)$, where each mesh $\mathcal{D}_i$ consists of n_i vertices. For the training set, we assume that each $\mathcal{D}_i$ has a ground-truth classification matrix $\mathbf{Y}_i \in \{0,1\}^{n_i \times 2}$, which represents the classification of vertices in $\mathcal{D}_i$ in one-hot encoding, such that:

$$\mathbf{Y}_{i,j} = \begin{cases} (0,1) & \text{if vertex } j \text{ of } \mathcal{D}_i \text{ belongs to the dendritic shaft,} \\ (1,0) & \text{otherwise.} \end{cases}$$

We define $\mathbf{Y} = (\mathbf{Y}_1, \mathbf{Y}_2, \dots, \mathbf{Y}_m)$ as the set of all annotation matrices. Next, we define the set of mean curvature vectors for all dendritic meshes as $\mathbf{H} = (\mathbf{H}_1, \mathbf{H}_2, \dots, \mathbf{H}_m)$ and the set of Gaussian curvature vectors as $\mathbf{K} = (\mathbf{K}_1, \mathbf{K}_2, \dots, \mathbf{K}_m)$. For each mesh $\mathcal{D}_j$, $j \in \{1, 2, \dots, m\}$, the mean curvature vector is given by $\mathbf{H}_j = (\mathbf{H}_{1,j}, \mathbf{H}_{2,j}, \dots, \mathbf{H}_{n_j,j})$, and the Gaussian curvature vector by $\mathbf{K}_j = (\mathbf{K}_{1,j}, \mathbf{K}_{2,j}, \dots, \mathbf{K}_{n_j,j})$, where $\mathbf{H}_{i,j}$ and $\mathbf{K}_{i,j}$ denote the mean and Gaussian curvatures of $\mathcal{D}_j$ at vertex i, for $i \in \{1, 2, \dots, n_j\}$.
Additionally, we define the distance vector $\mathbf{D} = (\mathbf{D}_1, \mathbf{D}_2, \dots, \mathbf{D}_m)$, representing the distances between the central curve and the vertices, as well as additional structural descriptors $\mathbf{S}^k$, where $k \in \{2, 3, \dots, 10\}$. For each mesh $\mathcal{D}_j$, we write $\mathbf{S}^k_j = (\mathbf{S}^k_{1,j}, \mathbf{S}^k_{2,j}, \dots, \mathbf{S}^k_{n_j,j})$.
We now define the feature vector for vertex i of mesh j as

$$\mathbf{Z}_{i,j} = \big(\mathbf{H}_{i,j}, \mathbf{H}^2_{i,j}, \mathbf{K}_{i,j}, \mathbf{K}^2_{i,j}, \mathbf{D}_{i,j}, \mathbf{S}^2_{i,j}, \mathbf{S}^3_{i,j}, \dots\big),$$

and the corresponding feature matrix for dendritic mesh $\mathcal{D}_j$ as

$$\mathbf{Z}_j = \big(\mathbf{H}_j, \mathbf{H}^2_j, \mathbf{K}_j, \mathbf{K}^2_j, \mathbf{D}_j, \mathbf{S}^2_j, \mathbf{S}^3_j, \dots\big).$$

Our objective is to find a function

$$f_\theta : \mathbf{Z}_j \mapsto \mathbf{Y}_j \in [0,1]^{n_j \times 2}, \quad \forall j \in \{1, 2, \dots, m\},$$

that best approximates the ground-truth classification by minimizing the following loss function:

$$\mathcal{L} = \min_\theta \left\| f_\theta(\mathbf{Z}) - \mathbf{Y} \right\|,$$

where $\|\cdot\|$ denotes an appropriate norm measuring the discrepancy between the predicted and actual classifications. In particular, we use a weighted sum of the binary cross-entropy (BCE), which encourages a more balanced optimization process. The binary cross-entropy formula is given by:

$$\mathrm{BCE}(y, \hat{y}) = -\left(y \log(\hat{y}) + (1-y)\log(1-\hat{y})\right).$$

The final loss function is defined as:

$$\mathcal{L}_\theta = \frac{1}{m} \sum_{j=1}^{m} \left(\frac{1}{n_j} \sum_{i=1}^{n_j} \Big(\mathrm{BCE}(\mathbf{Y}_{i,j}, f_\theta(\mathbf{Z}_{i,j})) \cdot w_j \Big)^2 \right).$$

Here the weights w_j is empirically determined.
Note that the mean curvature $\mathbf{H}$ and Gaussian curvature $\mathbf{K}$ used during training are computed after smoothing the curvature fields on the triangular dendritic-branch meshes. The sole purpose of this smoothing is to obtain stable and interpretable curvature estimates that enhance the segmentation performance during training. As described in Section 2.1.2, smoothing the curvature improves the readability of the curvature profile and reduces

noise-driven artifacts. Thus, the smoothing step in our pipeline serves only as a feature-enhancement operation, not as a geometric modification of the underlying dendrite.
Importantly, all manual annotations were performed on the actual, unsmoothed dendritic branches, ensuring that the ground-truth labels correspond to the true biological geometry. During training, the network is therefore supervised to segment the real dendritic shaft and spine structures. After training, all dimensional measurements and quantitative analyses are performed on the original triangulated meshes, not on the smoothed versions.

2.4. **Spine and Shaft Detection.** After the training step of the algorithm, the next stage of spine–shaft segmentation involves applying the model for classification as well as performing additional post-processing steps to first isolate the shaft and then the spines. In this section, we analyze the processes required to obtain a reliable segmentation.

2.4.1. *Grouping Dendritic Mesh Parts into Connected Vertices.* After training the DNN, the first step is to classify the vertices of a given test dendritic mesh into *shaft* and *spine* categories. The resulting classification can then be grouped into connected components of spine vertices and shaft vertices.
As a consequence of the sigmoidal activation function applied to the final layer, the output of the DNN is a probability matrix $\overline{\mathbf{Y}} = f_\theta(\mathbf{Z}) \in [0,1]^{n\times 2}$, where each row corresponds to a vertex and contains the probabilities of that vertex being classified as a *spine*, $\widetilde{\mathcal{X}}_{\text{spine}}$ (first column greater than the second), or a *shaft*, $\widetilde{\mathcal{X}}_{\text{shaft}}$ (second column greater than or equal to the first). Formally, we define:

$$\begin{aligned} \widetilde{\mathcal{X}}_{\text{spine}} &= \{\mathbf{X}_i \in \mathcal{D} \mid b\,\overline{\mathbf{Y}}(i,1) > a\,\overline{\mathbf{Y}}(i,0)\}, \\ \widetilde{\mathcal{X}}_{\text{shaft}} &= \{\mathbf{X}_i \in \mathcal{D} \mid b\,\overline{\mathbf{Y}}(i,1) \leqslant a\,\overline{\mathbf{Y}}(i,0)\}. \end{aligned} \tag{6}$$

Here, a and b are user-defined weighting parameters that control the decision boundary. In our experiments, we fixed $b = 0.81$ and initially set $a = 3$. If the resulting segmentation produced too many shaft labels, we increased a; if too many spine labels were produced, we decreased a. This adjustment allowed us to fine-tune the spine–shaft separation based on the model's output distribution.
Next, we describe how to group the predicted dendritic parts—classified as either *spine* or *shaft*—into subgroups of connected components. Let each vertex $\mathbf{X}_i$ have an associated set of neighboring vertices denoted by $\mathcal{N}_i$:

$$\mathcal{N}_i = \{\mathbf{X}_j \mid \mathbf{X}_j \text{ is in the 1-ring neighborhood of } \mathbf{X}_i\}.$$

We define a *connected group* of vertices $\mathcal{G}_i \subseteq \mathcal{D}$ such that for any pair $\mathbf{X}_p, \mathbf{X}_q \in \mathcal{G}_i$, there exists a path of vertices $\mathbf{X}_{i_1}, \mathbf{X}_{i_2}, \ldots, \mathbf{X}_{i_m} \in \widetilde{\mathcal{X}}_{\text{part}}$ satisfying:

$$\mathbf{X}_{i_1} = \mathbf{X}_p, \quad \mathbf{X}_{i_m} = \mathbf{X}_q, \quad \mathbf{X}_{i_\ell} \in \mathcal{N}_{i_{\ell-1}} \quad \text{for all } \ell = 2, \ldots, m.$$

Here, $\widetilde{\mathcal{X}}_{\text{part}}$ denotes the set of vertices classified as a given part type–either $\widetilde{\mathcal{X}}_{\text{spine}}$ for spines or $\widetilde{\mathcal{X}}_{\text{shaft}}$ for shafts. The subset of *part-classified* vertices within this group is then:

$$\widetilde{\mathcal{Y}}^i_{\text{part}} = \left\{ \mathbf{X}_j \in \widetilde{\mathcal{X}}_{\text{part}} \;\middle|\; \mathbf{X}_j \in \mathcal{G}_i \right\}.$$

Finally, the complete set of part-classified vertices $\widetilde{\mathcal{Y}}_{\text{part}}$ can be decomposed into disjoint set of connected vertices, where connectivity is defined by neighborhood overlap:

$$\widetilde{\mathcal{Y}}_{\text{part}} = \left\{ \widetilde{\mathcal{Y}}^i_{\text{part}} \;\middle|\; \mathbf{X}_i \in \widetilde{\mathcal{X}}_{\text{part}} \right\}.$$

2.4.2. *Spine–Shaft Detection.* Using the subgroups defined above of connected components, we now define the segmentation of the dendritic mesh into spines and shaft. Let $\widetilde{\mathcal{Y}}_{\text{shaft}}$ denote the set of connected components classified as *shaft*. In practice, some of these components may actually be parts of spines, so it is crucial to separate them from the true shaft set for accurate segmentation.

To achieve this, we identify the largest connected component in $\widetilde{\mathcal{Y}}_{\text{shaft}}$ and designate it as the *entire shaft*. All remaining connected components in $\widetilde{\mathcal{Y}}_{\text{shaft}}$ are then assumed to belong to spines.

The set of vertices belonging to the shaft is defined as:

$$\mathcal{Y}^0_{\text{shaft}} = \max_{\widetilde{\mathcal{Y}}^i_{\text{shaft}} \subset \widetilde{\mathcal{X}}_{\text{shaft}}} \left| \widetilde{\mathcal{Y}}^i_{\text{shaft}} \right|,$$

where $|\cdot|$ denotes the cardinality of the set.

Once the entire shaft is defined, spine segmentation proceeds by removing the shaft vertices from the dendrite vertex set, reclassifying them as spine vertices and then reapplying the connected-component grouping process described in Section 2.4.1 to the remaining vertices. The set of segmented spines is defined as:

$$\mathcal{Y}_{\text{spine}} = \left\{ \mathcal{Y}^i_{\text{spine}} \mid \mathbf{X}_i \in \mathcal{X}_{\text{spine}} \right\},$$

where each individual spine component is given by:

$$\mathcal{Y}^i_{\text{spine}} = \left\{ \mathbf{X}_j \in \mathcal{D} \backslash \mathcal{Y}^0_{\text{shaft}} \mid \mathbf{X}_j \in \mathcal{G}_i \right\}.$$

2.5. **Dataset Description.** This section describes the datasets of dendritic segments used for training and testing our algorithm. To mitigate overfitting, the training and testing sets were independent and originate from different animals. This separation ensures that the model learns generalizable features of dendritic structures rather than memorizing patterns specific to a single specimen. Table S3 provides a consolidated overview of the datasets used for training and testing, including their sources, annotation strategy, and selection criteria.

2.5.1. *Training dataset.* The training dataset used as ground truth in this paper comprises six high-resolution 3D EM reconstructions of dendrites from the CA1 region of the hippocampus from rats at postnatal day 21 (P21), an immature developmental stage. Brain slices from these animals either underwent *in vitro* theta-burst stimulation to induce long-term potentiation (LTP) or received control stimulation, as described in [7–9].

For the training dataset, the meshes were manually annotated by experts, where spine and shaft segmentations were performed on all six dendritic branch reconstructions. Each spine was stored as an individual `.obj` file in watertight format. The mesh corresponding to the shaft of each dendritic segment was provided separately, along with the complete mesh of the dendritic segment. Because spine and shaft meshes were stored independently, we wrapped the spines and shaft with a tight mesh from the exterior in order to generate a unified dendritic mesh using the algorithm in Section S6.1. We then used the independent spine meshes to label the constituent parts of the new wrapped dendritic mesh. To achieve this, we employed a KD-Tree–based nearest-neighbor search to map spine vertices to the wrapped dendritic mesh. Specifically, for each set of spine vertices in the individual dataset `vertices`, we queried a KD-Tree built from the wrapped dendritic mesh vertices to identify all branch vertices within a radius threshold r_{th}:

$$\texttt{vertices_appr} = \texttt{list}\left(\texttt{set}\left(\texttt{np.concatenate}\left(\texttt{kdtree.query_radius}(\texttt{vertices}, r_{\text{th}})\right)\right)\right).$$

Here, `query_radius` returns the indices of dendrite mesh vertices that lie within a distance r_{th} of each spine vertex. By aggregating these indices, we obtain the set of dendrite vertices corresponding to the annotated spine regions on the wrapped mesh. This mapping step allows us to merge the spine and shaft meshes into a single labeled dendritic mesh, which is then used for both training and validation.

2.5.2. *Training Data Augmentation.* Only six dendritic branches are available for training, so we expanded the dataset by applying a series of geometric transformations. First, we computed the PCA of each mesh and projected its vertices onto the corresponding PCA plane. We then applied random scaling to the vertices in two separate steps, introduced random rotations in two additional steps, and added random translations. These transformations produced four augmented versions of each original mesh, resulting in a total of 30 dendritic branches available for training and validation. Of these, 22 branches were used for training, and the remaining 8 were used for validation.

2.5.3. *Testing dataset.* For testing, we used data from the axon–spine coupling study, which includes a complete nanoconnectomic 3D reconstruction of hippocampal neuropil obtained via serial EM [20]. This dataset contains four independent annotations–two performed by each of two annotators [3]–providing detailed segmentation of dendritic spines. Out of 151 meshes, we selected 28 spiny dendritic branches for testing based on annotation consensus. Specifically, we included only those meshes where two or more annotators agreed on the presence of at least one dendritic spine. Additionally, for each spine mesh identified by the annotators, we retained only those meshes that were consistently labeled as spines by at least two of the selected annotations. This ensured that the testing set reflected a high-confidence subset of spiny dendritic structures.
It is important to note that some structure exhibiting spine-like morphology were not annotated as spines in the test set for two reasons. First, biologically, the presence of a synaptic area in the spine head is essential for defining a spine. Thus, even if a mesh appears spiny, it is not considered a spine if no synapse is present. Second, some meshes were excluded due to incomplete reconstruction–parts of the spine may have been cut off or lost during imaging.
In contrast, the training dataset does not apply as strict criteria for spine identification as the test set. As a result, the testing dataset introduces additional segmentation challenges that are not reflected in the training data. Our algorithm does not explicitly account for these differences, which may influence performance.

2.5.4. *Independent dataset.* In addition to the training and testing datasets, we also used dendritic segments from the large-scale nanoconnectomic EM reconstruction of mouse neocortex by Kasthuri et al. [25]. This dataset provides densely reconstructed neuropil at nanometer resolution but does not include manual spine or shaft annotations. We therefore use it solely for qualitative visualization (Figure 7) to demonstrate that our method generalizes to independently acquired EM volumes. These data are not used for training, validation, or quantitative evaluation.

3. Results

This section presents a comprehensive evaluation of our algorithm's performance in segmenting dendritic shafts and spines. We first describe the identification of dendritic shafts

using geometric properties and a neural network-based approach. We then assess the ability of the algorithm to segment dendritic spines, highlighting both its strengths and limitations. Figure 7 illustrates the predicted dendritic shafts and spines from datasets from Kasthuri et al. [25], demonstrating that our method successfully identifies spines along the dendritic shaft by leveraging geometric properties such as Gaussian and mean curvature. However, misclassification can occur when the dendritic structure deviates significantly from the idealized cylindrical shape, underscoring the need for further improvements to the method.
To make these methods widely accessible, we have posted an open-source code repository on GitHub: GitHub:curvature-based-dendrite-segmentation.

3.1. **DNN Prediction Results Analysis.** In this section, we present and analyze the prediction results of the three DNNs, $\mathbf{DNN}_1$, $\mathbf{DNN}_2$, and $\mathbf{DNN}_3$. The methodology is outlined in Section 2.2.4, and the evaluation here focuses on four key metrics: the training loss, the DICE coefficient (Dice Similarity Coefficient), the Jaccard index IoU (Intersection over Union), and the AUC (Area Under the ROC Curve, where ROC denotes the Receiver Operating Characteristic).

3.1.1. *Analysis of the performance.* The training loss curves for the three models are shown in Figure S4. Among the three models, $\mathbf{DNN}_2$ achieves the most effective convergence, reaching a minimum loss of approximately 2.44, closely followed by $\mathbf{DNN}_3$ with a minimum loss of about 3.56. In contrast, the baseline architecture $\mathbf{DNN}_1$ shows the weakest convergence, with a substantially higher final loss of 6.73 throughout training. These results highlight the importance of the additional features and architectural refinements incorporated into $\mathbf{DNN}_2$ and $\mathbf{DNN}_3$ for improving optimization stability.
To quantitatively assess segmentation accuracy, we computed the IoU, DICE, and AUC scores between predicted and annotated vertices for the dendritic shaft and spines on both the training and test sets. The performance curves are presented in Figure S5. For $\mathbf{DNN}_2$, the IoU converges to average values of approximately 0.70 for the shaft and 0.81 for the spines, with corresponding DICE scores of about 0.88 and 0.92, respectively. $\mathbf{DNN}_3$ achieves comparable performance across both IoU and DICE metrics, whereas $\mathbf{DNN}_1$ lags behind with average IoU values of 0.55 (shaft) and 0.75 (spines), along with lower DICE scores overall. The AUC curves further support these trends: both $\mathbf{DNN}_2$ and $\mathbf{DNN}_3$ maintain consistently high AUC values throughout training, while $\mathbf{DNN}_1$ again shows weaker discriminative performance. Importantly, the similarity between training and testing curves across all metrics indicates good generalization and minimal overfitting.
Overall, these results demonstrate a clear progression in performance from $\mathbf{DNN}_1$ to $\mathbf{DNN}_3$. The incorporation of additional geometric and topological features in $\mathbf{DNN}_2$ and $\mathbf{DNN}_3$ substantially improves convergence and segmentation accuracy, underscoring the value of feature enrichment in dendritic spine detection.

3.2. **Dendritic Spine Detection Analysis.** Following dendritic shaft segmentation, we applied the spine detection algorithm described in Section 2.4 to identify spines. For each detected spine, we computed the IoU to evaluate segmentation accuracy. In addition, we calculated the IoU for the union of all detected spines to assess overall performance. We then computed accuracy, precision, recall, and F1-score to further quantify classification performance.

The segmentation results produced by the **DNN** models are shown in Figure 5. A clear qualitative progression is evident from $\mathbf{DNN}_1$ to the more advanced architectures $\mathbf{DNN}_2$ and $\mathbf{DNN}_3$. The $\mathbf{DNN}_1$ model frequently misclassifies portions of the shaft as spines, resulting in fragmented and noisy predictions. Incorporating additional geometric features—such as the distance to the shaft skeleton—allows $\mathbf{DNN}_2$ to reduce these errors and produce a more coherent separation between shaft and spine regions. Building on this, $\mathbf{DNN}_3$ integrates richer geometric and topological descriptors, enabling it to capture subtle curvature variations and resolve complex spine clusters more effectively. These improvements are visible not only in the 3D mesh renderings but also in the 2D slice, where the highlighted region clearly shows the reduction in mislabeling from $\mathbf{DNN}_1$ to $\mathbf{DNN}_3$. Overall, the predicted segmentations from $\mathbf{DNN}_3$ align most closely with the expert annotation. These qualitative trends are consistent with the quantitative results in Table S4, where both $\mathbf{DNN}_2$ and $\mathbf{DNN}_3$ outperform $\mathbf{DNN}_1$ across all evaluation metrics.

3.2.1. *Quantitative Evaluation and Analysis.* We evaluated segmentation performance using multiple complementary metrics, including AUC, Precision, Recall, and F1-score, computed under both IoU and DICE criteria, each with Standard and Union variants. These results, summarized in Table S4, enable a detailed comparison between our geometric-informed models ($\mathbf{DNN}_1$, $\mathbf{DNN}_2$, $\mathbf{DNN}_3$) and established CNN baselines (U-Net, VoxNet, VGG16-FCN). Details on the training procedures and evaluation of the three CNN models are provided in Section S9. The weighting parameters used in the geometric-informed models are specified in (6), where the values of a for each model are listed and the parameter b is fixed at 0.81.

Among the CNN baselines, U-Net achieves the highest AUC (0.792), indicating strong global discriminative ability. However, AUC alone does not fully capture fine-scale segmentation quality. When examining Precision, Recall, and F1-score under the IoU and DICE criteria, the geometric-informed model $\mathbf{DNN}_3$ consistently delivers the strongest performance. Under the IoU–Union metric, $\mathbf{DNN}_3$ attains the highest Recall (0.873) and F1-score (0.845), indicating that it captures spine regions more completely than both the CNN baselines and the other geometric-informed variants. Its DICE–Union performance is even stronger, with an F1-score of 0.857 and a Recall of 0.895.

In contrast, $\mathbf{DNN}_1$ and $\mathbf{DNN}_2$ exhibit more limited recall, particularly under the Standard variants of IoU and DICE, where both models tend to miss a substantial fraction of true spines despite maintaining moderate-to-high precision. VoxNet and VGG16-FCN perform noticeably worse across all metrics, with lower AUC values (0.682 and 0.466, respectively) and substantially reduced F1-scores, especially under the IoU criterion. These results highlight the limitations of intensity-based CNN architectures when applied to thin, high-curvature structures such as dendritic spines.

The Union variants of IoU and DICE consistently yield higher Recall and F1-scores across all models, reflecting their ability to more fairly evaluate clustered or contiguous spine regions. This is particularly relevant in dense dendritic environments, where individual spine boundaries may be ambiguous. Under these more biologically meaningful metrics, $\mathbf{DNN}_3$ shows the clearest advantage, outperforming all baselines by a substantial margin.

Overall, the quantitative results demonstrate a clear progression in performance from $\mathbf{DNN}_1$ to $\mathbf{DNN}_3$, with $\mathbf{DNN}_3$ providing the most reliable and complete segmentation of dendritic spines. The strong performance of $\mathbf{DNN}_3$ under both IoU and DICE—especially in the

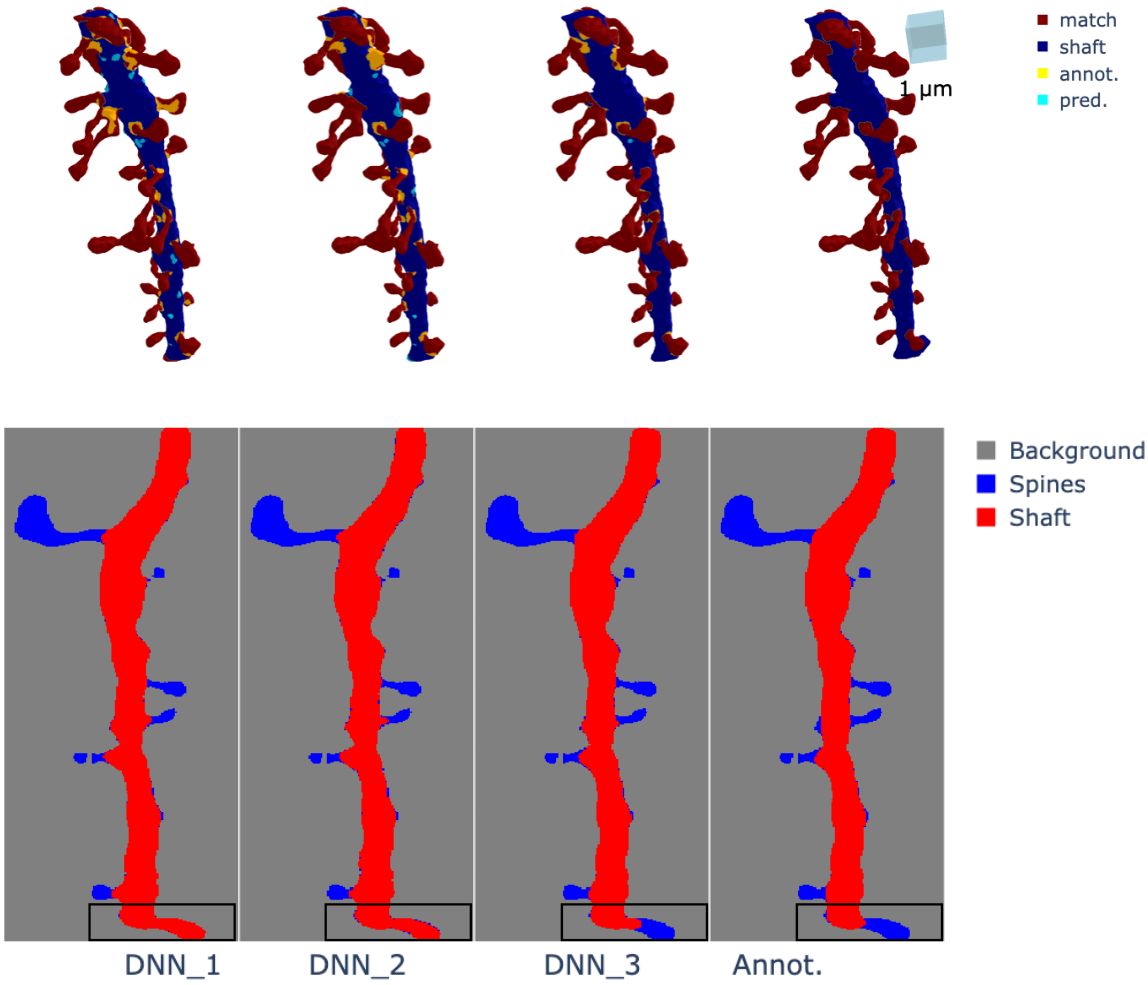

FIGURE 5. Comparison of segmentation results obtained using $\mathbf{DNN}_1$, $\mathbf{DNN}_2$, and $\mathbf{DNN}_3$. The first three columns correspond to the predictions from $\mathbf{DNN}_1$, $\mathbf{DNN}_2$, and $\mathbf{DNN}_3$, respectively, and the final column shows the expert spine annotation used as ground truth. The top row displays the 3D mesh representations, while the bottom row shows 2D slices of the same regions. In the meshes, red vertices indicate correct spine predictions that match the annotation, sky-blue vertices represent shaft regions misclassified as spines, and yellow vertices represent spine regions misclassified as shaft. $\mathbf{DNN}_1$ produces noisier results with more misclassifications, particularly in clustered regions. $\mathbf{DNN}_2$ reduces shaft-to-spine errors, and $\mathbf{DNN}_3$ achieves the most accurate segmentation overall, with fewer misclassifications and the closest agreement with the expert annotation. The slice highlights the region where mislabeling is most apparent, especially for $\mathbf{DNN}_1$ and $\mathbf{DNN}_2$. The triangular mesh is obtained from CA1 dendritic reconstructions [7–9].

Union variants—underscores the value of incorporating geometric curvature cues for accurate spine–shaft segmentation.

3.3. **Dendrite Spine Segmentation without Smoothing.** In the segmentation of dendrite triangular meshes, curvature smoothing is typically required. However, when the number of vertices is large–more than 200,000–the process becomes computationally expensive, taking more than 20 hours to produce a smooth mesh. We note that by using $\mathbf{DNN}_3$, the computational time can be reduced by performing segmentation without smoothing, while still achieving a level of accuracy comparable to that obtained with smoothing.
We evaluated the effectiveness of curvature smoothing with $\mathbf{DNN}_3$ on both the training and test datasets and compared the accuracy. The computed accuracy for the test dataset is 0.658 (with and without smoothing), while for the training dataset it is 0.704 with smoothing and 0.780 without smoothing. These results show that smoothing the curvature of the triangular mesh when using $\mathbf{DNN}_3$ is unnecessary, and segmentation may even perform better without it.
This phenomenon may be due to the fact that the nine region-based segmentation features $\mathbf{S}^k$ enable the **DNN** to detect the neck area without requiring an enhanced curvature profile produced by smoothing. When we tested the same process on other **DNN** models, the accuracy was zero in both cases.

3.4. **Application to Dendritic Surface Meshes with Many Vertices.** We next use our algorithm to segment the spines of a dendritic mesh dataset containing a large number of vertices and multiple branches.
In particular, the dataset we study is from the reconstruction of a sub-volume of mouse neocortex [25]. We analyzed a dendritic mesh with 5,387,879 vertices and 10,777,005 faces. This dendritic mesh is about 13.42 times larger than the training dataset used in the previous sections (401,371 vertices and 802,750 faces). While most of the training and testing datasets contain only one main dendritic branch with attached spines, this dataset has nine branches with multiple spines. This highlights how much more complex this dataset is compared to the training and testing data.
Because this dataset is relatively large, the segmentation process differs slightly from that used in training. Computations were performed on a MacBook Pro with an Apple M4 Max chip and 64 GB of memory. When we attempted to segment the large dendritic mesh directly, the process failed due to computational constraints. Therefore, we reduced the size of the mesh using the simplification algorithm described in Section S6.2. The mesh was simplified to nearly the size of the largest training dataset, given (about 449,109 vertices and 900,000 faces), and we then proceeded with the same testing process as before.
The segmentation results are shown in Figure 7. Overall, $\mathbf{DNN}_3$ achieves the strongest performance, correctly labeling the majority of spines across the dendritic mesh. However, it still misclassifies a small region of the shaft as spines, as illustrated in the bottom panel (C). This error likely arises from local geometric similarities between the shaft surface and spine neck regions, particularly where the mesh bends or narrows.
In contrast, $\mathbf{DNN}_1$ also performs reasonably well and produces few large false-positive spine regions. However, closer inspection of panel (A) shows that $\mathbf{DNN}_1$ frequently misclassifies spine necks as shaft, and in some cases entire spines are labeled as shaft. These errors are consistent with the limited feature set used by $\mathbf{DNN}_1$, where Gaussian and mean curvature alone do not provide sufficient spatial context for reliable spine–shaft discrimination.
Finally, $\mathbf{DNN}_2$ exhibits the weakest performance on this dataset. It correctly identifies spines in only a subset of regions, while misclassifying large contiguous portions of the spines as shaft. This behavior suggests that the combination of shaft–skeleton distance,

Gaussian curvature, and mean curvature features is insufficient to capture the complex local geometry present in this large-scale reconstruction.

3.5. **Dendritic Spine Morphologic Parameters.** In this section, we use the segmentation results obtained from our algorithm to compute several morphological parameters of the dendritic meshes in both the training and testing datasets. These parameters include the neck diameter, head diameter, and spine length.

The computation of head and neck diameters was performed using the spine skeleton. This skeleton was obtained by identifying the closest point of the triangular mesh skeleton to each vertex of the segmented spine mesh. To determine the neck and head diameters, we considered each segmented spine's triangular mesh. For each vertex, we computed the shortest distance to its nearest skeleton point, as described in equation (5), while computing $\mathbf{D}$. This procedure yielded a thickness profile along the spine. The neck radius was then defined as the minimum of these computed averages, starting from the skeleton vertices closest to the dendritic shaft. Conversely, the head diameter was defined as the maximum computed distance, measured from the skeleton vertices at the farthest point from the shaft. Additionally, to compute the length of the spine, we used the spine skeletonization and interpolated additional points along the skeleton using a spline technique. This interpolation increases the accuracy of the computed distances by smoothing and interpolating along the dendritic spine skeleton [13, 14, 53]. Additional details are provided in Section S7. The spine length was then computed as the sum of Euclidean distances between consecutive interpolated points:

$$L = \sum_{i=1}^{N-1} \|\mathbf{V}_{i+1} - \mathbf{V}_i\|_2,$$

where $\mathbf{V}_i$ denotes the i-th interpolated vertex.

The distribution of morphological parameters in the training and large dendritic mesh datasets is shown in Figure 6. In the first column, the head diameter is plotted against the neck diameter of dendritic spines, while in the second column the spine volume is plotted against the spine area, with a colormap encoding spine length. For clearer visualization, some outliers were removed. Marginal histograms of head and neck diameters are also included to facilitate interpretation.

Quantitatively, the segmented training dataset contains 210 spines, with an average neck diameter of $0.166 \pm 0.0969\,\mu$m, an average head diameter of $0.512 \pm 0.101\,\mu$m, and an average spine length of $1.35 \pm 0.694\,\mu$m. The average spine volume is $0.122 \pm 0.121\,\mu\text{m}^3$ and the average spine area is $2.30 \pm 1.83\,\mu\text{m}^2$, yielding an average area-to-volume ratio of 12.9. Importantly, the relationship between spine volume and area is very strong, as reflected by a high coefficient of determination ($R^2 = 0.916$).

In contrast, the segmented large dendritic mesh dataset contains 562 spines, with an average neck diameter of $0.148 \pm 0.0787\,\mu$m, an average head diameter of $0.661 \pm 0.164\,\mu$m, and an average spine length of $5.57 \pm 5.34\,\mu$m. The average spine volume is $0.090 \pm 0.085\,\mu\text{m}^3$ and the average spine area is $1.53 \pm 1.15\,\mu\text{m}^2$, yielding an average area-to-volume ratio of 13.9. Here too, the relationship between spine volume and area remains strong, with a coefficient of determination of $R^2 = 0.847$.

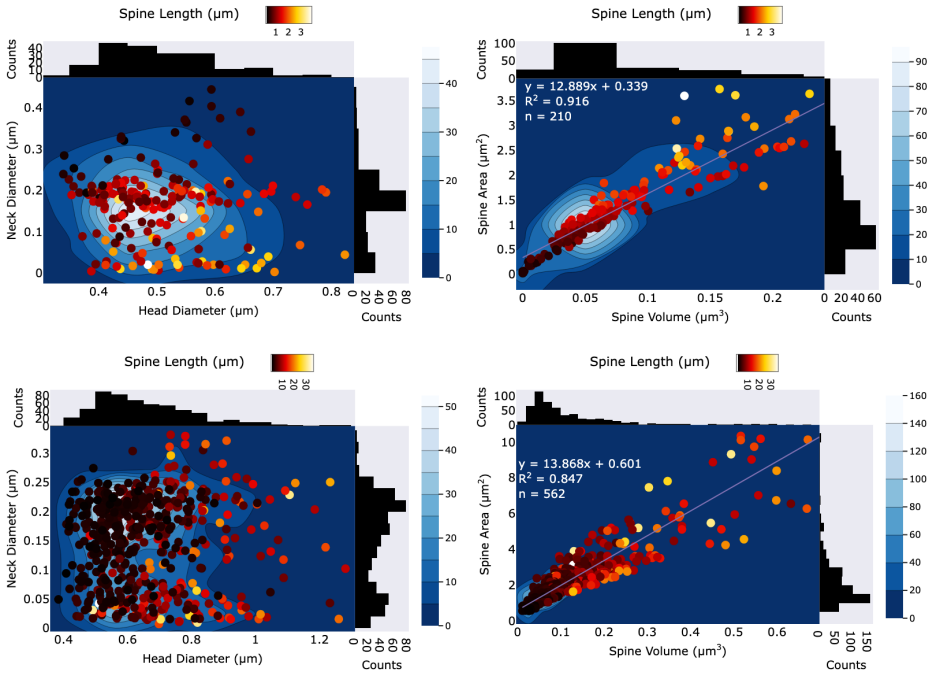

FIGURE 6. Distribution of morphological parameters in the training dataset and in the large dendritic mesh dataset, obtained using $\mathbf{DNN}_3$. In the first column, head diameter is plotted against neck diameter of dendritic spines, while in the second column spine volume is plotted against spine area. A colormap encodes spine length. The segmented training dataset contains 210 spines, with an average neck diameter of $0.166 \pm 0.0969\,\mu$m, an average head diameter of $0.512 \pm 0.101\,\mu$m, and an average spine length of $1.35 \pm 0.694\,\mu$m. The average spine volume is $0.122 \pm 0.121\,\mu\text{m}^3$ and the average spine area is $2.30 \pm 1.83\,\mu\text{m}^2$, yielding an average area-to-volume ratio of 12.9 with a coefficient of determination $R^2 = 0.916$. The segmented large dendritic mesh dataset contains 562 spines, with an average neck diameter of $0.148 \pm 0.0787\,\mu$m, an average head diameter of $0.661 \pm 0.164\,\mu$m, and an average spine length of $5.57 \pm 5.34\,\mu$m. The average spine volume is $0.090 \pm 0.085\,\mu\text{m}^3$ and the average spine area is $1.53 \pm 1.15\,\mu\text{m}^2$, yielding an average area-to-volume ratio of 13.9 with a coefficient of determination $R^2 = 0.847$. The dataset is derived from the Nanoconnectomic 3D EM reconstruction of hippocampal neuropil from the axon–spine coupling study [3].

4. Discussion

Despite its overall promise, our algorithm encounters challenges in segmenting spines with complex morphologies and spatial arrangements. Below, we discuss three key failure cases, illustrated in Figures 8. The consistently high precision suggests that when a spine is detected, it is almost always correct (few false positives). However, the moderate recall for IoU alone indicates that some spines are missed (false negatives), potentially affecting downstream morphological analyses. The substantial performance improvement with

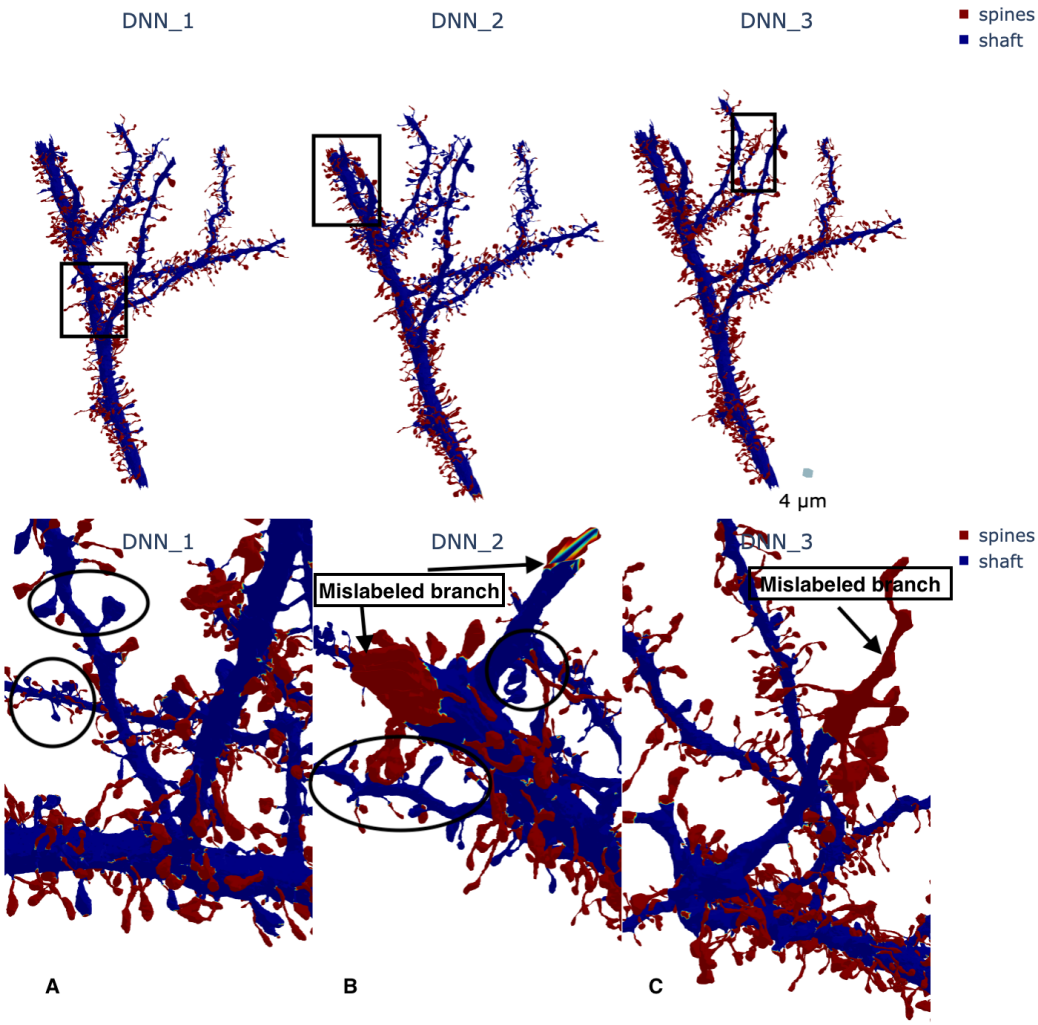

FIGURE 7. Comparison of segmentation results obtained using $\mathbf{DNN}_1$, $\mathbf{DNN}_2$, and $\mathbf{DNN}_3$ on a large-scale nanoconnectomic EM reconstruction of mouse neocortex from Kasthuri et al. [25]. In all panels, blue indicates the dendritic shaft mesh, while red highlights the segmented spines. The top panel shows the full dendritic segment with predictions from the three models. All models correctly identify the majority of spines, but notable differences emerge upon closer inspection. Three boxed regions highlight areas of disagreement, which are shown in detail in the bottom panels. Panel (A) shows spine meshes that $\mathbf{DNN}_1$ incorrectly labels as shaft. Panel (B) illustrates dendritic sub-branches that $\mathbf{DNN}_2$ misclassifies as spines. Panel (C) shows a similar misclassification pattern for $\mathbf{DNN}_3$, where local curvature and branching geometry lead to spurious spine predictions.

union-based IoU suggests that a more inclusive segmentation strategy enhances detection reliability.

4.1. **Limitations in Spine Group Segmentation.** Dendritic spines frequently appear in dense clusters, complicating their segmentation. These spine groups are two or more spines whose vertices are directly connected, and after the shaft is removed they still group together. Figure 8 shows a region containing tightly packed spines where the algorithm struggles to distinguish individual structures due to overlapping curvature features. This limitation highlights the need for improved clustering-based segmentation techniques or the integration of additional geometric descriptors capable of differentiating individual spines in high-density regions.

4.1.1. *Misclassification of Dendritic Shaft Regions as Spines.* Another failure mode involves the erroneous classification of dendritic shaft regions as spines, as depicted in Figure 8[F]. This misclassification likely stems from local curvature variations that resemble spine-like features. To mitigate this issue, we propose the incorporation of spatial continuity constraints, contextual neighborhood information, or multi-scale curvature analysis to better distinguish dendritic shafts from true spine structures.

4.1.2. *Scalability and Large Dataset Limitations.* A further limitation arises from the scale of the datasets used in training and evaluation. High-resolution 3D EM reconstructions of dendritic segments generate extremely large meshes, often containing millions of vertices. Processing such datasets requires substantial computational resources, both in terms of memory and runtime. This constraint limits the feasibility of applying our method to very large-scale reconstructions or to entire brain regions without significant preprocessing or downsampling. Moreover, the need to balance mesh resolution with computational efficiency may lead to the loss of fine structural details, particularly in thin spine necks or small protrusions. Addressing this limitation will require the development of more efficient algorithms, parallelized implementations, or hierarchical multi-resolution approaches that can scale to increasingly large datasets while preserving biologically relevant detail.

5. Conclusion

In this work, we developed and evaluated three geometric-informed deep neural network architectures for dendritic shaft and spine segmentation. Our results demonstrate a clear progression in performance from the baseline $\mathbf{DNN}_1$ to the enhanced models $\mathbf{DNN}_2$ and $\mathbf{DNN}_3$, with $\mathbf{DNN}_3$ achieving the strongest overall segmentation accuracy. By incorporating curvature-based and topology-aware features, these models substantially improve both training stability and predictive performance, particularly in regions with dense spine clusters or complex branching patterns.

A key advantage of the geometric-informed approach is its computational efficiency relative to volumetric 3D CNNs. CNN-based models require dense voxel representations that scale cubically with resolution, making them memory intensive and slow when applied to high-resolution EM reconstructions. In contrast, our geometric models operate directly on mesh-derived features, avoiding voxelization and enabling faster inference on large or irregular geometries. As a result, $\mathbf{DNN}_3$ not only surpasses all CNN baselines in accuracy but also scales more effectively to high-resolution meshes, making it the preferred choice for detailed EM datasets where preserving fine structural detail is essential.

Overall, our findings highlight the potential of feature-enriched geometric learning for robust dendritic spine detection. By improving segmentation accuracy, computational efficiency, and scalability, these models provide a strong foundation for downstream morphological

analyses and quantitative studies of synaptic connectivity. Continued refinement of geometric descriptors, multi-resolution strategies, and post-processing methods will further enhance the reliability and applicability of this framework to increasingly large and complex neural reconstructions.

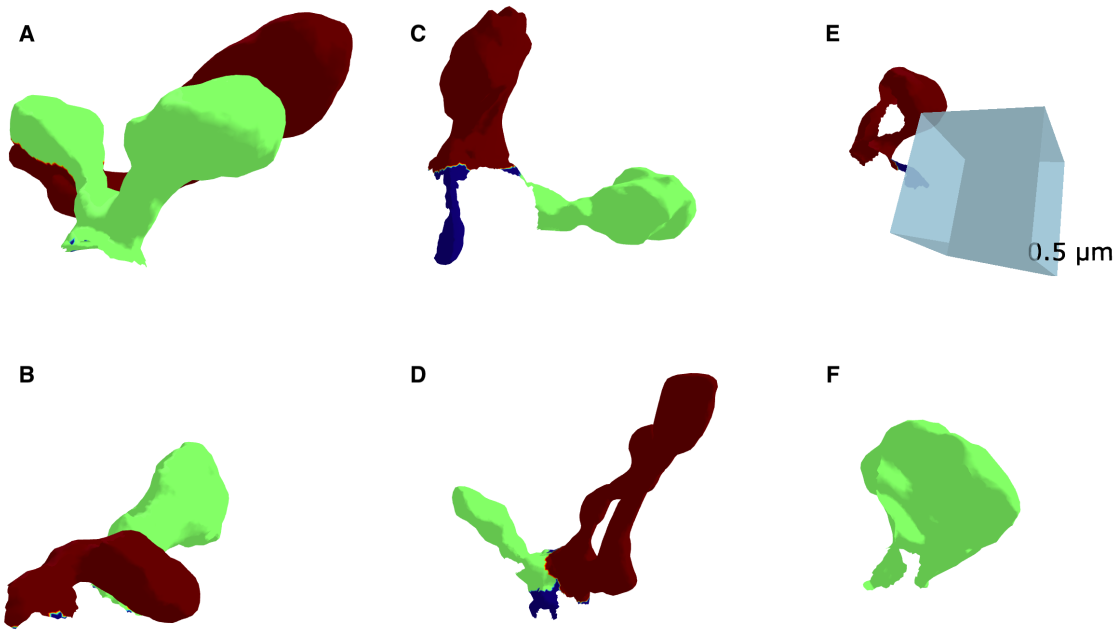

FIGURE 8. Examples of misclassification encountered during spine segmentation. In the figure, blue indicates the shaft mesh, while other colors indicate spine meshes that were misclassified. Panels (A–D) illustrate cases where multiple spines were incorrectly merged and classified as a single spine. Panel (E) depicts a region of the dendritic shaft that was mistakenly labeled as a spine, while Panel (F) shows a fraction of a spine that was only partially labeled. These examples highlight the challenges of accurately separating spines in dense clusters. The triangular mesh is obtained from CA1 dendritic reconstructions [7–9].

Acknowledgments

We acknowledge Myles Joyce for help with spine segmentation. This work was supported by NSF Technology Hub #1707356, NSF NeuroNex2 #2014862, NSF award 2219894, and NIH grants R01MH095980 and R56MH139176 to KMH and NIH grant T32 NS007292 to AKAG. We acknowledge use of the Brandeis High Performance Computing Cluster (HPCC) which is partially supported by the NSF through DMR-MRSEC 2011846 and OAC-1920147.

Declaration of Interests

The authors declare no competing interests.

References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, *Tensorflow: a system for large-scale machine learning*, in Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation, OSDI'16, USA, 2016, USENIX Association, p. 265–283.

[2] W. Al-Khater and S. Al-Madeed, *Using 3d-vgg-16 and 3d-resnet-18 deep learning models and fabemd techniques in the detection of malware*, Alexandria Engineering Journal, 89 (2024), pp. 39–52.

[3] J. Bartol, Thomas M, C. Bromer, J. Kinney, M. A. Chirillo, J. N. Bourne, K. M. Harris, and T. J. Sejnowski, *Nanoconnectomic upper bound on the variability of synaptic plasticity*, eLife, 4 (2015), p. e10778.

[4] S. Basu, P. K. Saha, M. Roszkowska, M. Magnowska, E. Baczynska, N. Das, D. Plewczynski, and J. Wlodarczyk, *Quantitative 3-D morphometric analysis of individual dendritic spines*, Sci. Rep., 8 (2018).

[5] S. Bernal-Garcia, A. P. Schlotter, D. B. Pereira, A. J. Recupero, F. Polleux, and L. A. Hammond, *A deep learning pipeline for accurate and automated restoration, segmentation, and quantification of dendritic spines*, Cell Reports Methods, (2025), p. 101179.

[6] J. Bourne and K. M. Harris, *Do thin spines learn to be mushroom spines that remember?*, Current opinion in neurobiology, 17 (2007), pp. 381–386.

[7] J. N. Bourne and K. M. Harris, *Coordination of size and number of excitatory and inhibitory synapses results in a balanced structural plasticity along mature hippocampal ca1 dendrites during ltp*, Hippocampus, 21 (2011), pp. 354–373.

[8] C. Bromer, T. M. Bartol, J. B. Bowden, D. D. Hubbard, D. C. Hanka, P. V. Gonzalez, M. Kuwajima, J. M. Mendenhall, P. H. Parker, W. C. Abraham, T. J. Sejnowski, and K. M. Harris, *Long-term potentiation expands information content of hippocampal dentate gyrus synapses*, Proc. Natl. Acad. Sci. U. S. A., 115 (2018), pp. E2410–E2418.

[9] M. A. Chirillo, M. S. Waters, L. F. Lindsey, J. N. Bourne, and K. M. Harris, *Local resources of polyribosomes and ser promote synapse enlargement and spine clustering after long-term potentiation in adult rat hippocampus*, Scientific reports, 9 (2019), p. 3861.

[10] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger, *3d u-net: Learning dense volumetric segmentation from sparse annotation*, in Medical Image Computing and Computer-Assisted Intervention – MICCAI 2016, S. Ourselin, L. Joskowicz, M. R. Sabuncu, G. Unal, and W. Wells, eds., Cham, 2016, Springer International Publishing, pp. 424–432.

[11] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia, *Meshlab: an open-source mesh processing tool.*, vol. 1, 01 2008, pp. 129–136.

[12] M. Dawson-Haggerty et al., *Trimesh: A python library for loading and using triangular meshes.* https://trimesh.org, 2019.

[13] P. Dierckx, *Algorithms for smoothing data with periodic and parametric splines*, Computer Graphics and Image Processing, 20 (1982), pp. 171–184.

[14] P. Dierckx, *Curve and surface fitting with splines*, in Monographs on numerical analysis, 1996.

[15] H. Edelsbrunner, D. Kirkpatrick, and R. Seidel, *On the shape of a set of points in the plane*, IEEE Transactions on Information Theory, 29 (1983), pp. 551–559.

[16] J. C. Fiala, J. Spacek, and K. M. Harris, *Dendritic spine pathology: cause or consequence of neurological disorders?*, Brain research reviews, 39 (2002), pp. 29–54.

[17] M. Garland and P. S. Heckbert, *Surface Simplification Using Quadric Error Metrics*, Association for Computing Machinery, New York, NY, USA, 1 ed., 2023.

[18] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. Del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, *Array programming with NumPy*, Nature, 585 (2020), pp. 357–362.

[19] K. Harris and S. Kater, *Dendritic spines: cellular specializations imparting both stability and flexibility to synaptic function*, Annual review of neuroscience, 17 (1994), pp. 341–371.

[20] K. M. Harris, J. Spacek, M. E. Bell, P. H. Parker, L. F. Lindsey, A. D. Baden, J. T. Vogelstein, and R. Burns, *A resource from 3d electron microscopy of hippocampal neuropil for user training and tool development*, Scientific Data, 2 (2015), p. 150046.

[21] A. E. Hoerl and R. W. Kennard, *Ridge regression: Biased estimation for nonorthogonal problems*, Technometrics, 12 (1970), p. 55.

[22] K. Hornik, M. Stinchcombe, and H. White, *Multilayer feedforward networks are universal approximators*, Neural Networks, 2 (1989), pp. 359–366.

[23] E. R. Kandel, Y. Dudai, and M. R. Mayford, *The molecular and systems biology of memory*, Cell, 157 (2014), pp. 163–186.

[24] G. Karniadakis, Y. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, *Physics-informed machine learning*, Nature Reviews Physics, (2021), pp. 1–19.

[25] N. Kasthuri, K. Hayworth, D. Berger, R. Schalek, J. Conchello, S. Knowles-Barley, D. Lee, A. Vázquez-Reina, V. Kaynig, T. Jones, M. Roberts, J. Morgan, J. Tapia, H. Seung, W. Roncal, J. Vogelstein, R. Burns, D. Sussman, C. Priebe, H. Pfister, and J. Lichtman, *Saturated reconstruction of a volume of neocortex*, Cell, 162 (2015), pp. 648–661.

[26] M. Kazhdan, M. Bolitho, and H. Hoppe, *Poisson surface reconstruction*, in Proceedings of the Fourth Eurographics Symposium on Geometry Processing, SGP '06, Goslar, DEU, 2006, Eurographics Association, p. 61–70.

[27] M. Kazhdan and H. Hoppe, *Screened poisson surface reconstruction*, ACM Trans. Graph., 32 (2013).

[28] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, CoRR, abs/1412.6980 (2014).

[29] M. Kuwajima, J. M. Mendenhall, and K. M. Harris, *Large-volume reconstruction of brain tissue from high-resolution serial section images acquired by SEM-based scanning transmission electron microscopy*, in Nanoimaging, Methods in molecular biology (Clifton, N.J.), Humana Press, Totowa, NJ, 2013, pp. 253–273.

[30] Y. LeCun, Y. Bengio, and G. Hinton, *Deep learning*, Nature, 521 (2015), pp. 436–44.

[31] T. Lee, R. Kashyap, and C. Chu, *Building skeleton models via 3-d medial surface axis thinning algorithms*, CVGIP: Graphical Models and Image Processing, 56 (1994), pp. 462–478.

[32] H. Li, Z. Zhang, T. Li, and X. Si, *A review on physics-informed data-driven remaining useful life prediction: Challenges and opportunities*, Mechanical Systems and Signal Processing, 209 (2024), p. 111120.

[33] L. Lu, X. Meng, Z. Mao, and G. E. Karniadakis, *Deepxde: A deep learning library for solving differential equations*, SIAM Review, 63 (2021), pp. 208–228.

[34] D. Maturana and S. Scherer, *Voxnet: A 3d convolutional neural network for real-time object recognition*, in 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2015, pp. 922–928.

[35] P. Merino-Serrais, R. Benavides-Piccione, L. Blazquez-Llorca, A. Kastanauskaite, A. Rabano, J. Avila, and J. DeFelipe, *The influence of phospho-tau on dendritic spines of cortical pyramidal neurons in patients with alzheimer's disease*, Brain, 136 (2013), pp. 1913–1928.

[36] M. Meyer, M. Desbrun, P. Schröder, and A. H. Barr, *Discrete differential-geometry operators for triangulated 2-manifolds*, in Visualization and Mathematics III, H.-C. Hege and K. Polthier, eds., Berlin, Heidelberg, 2003, Springer Berlin Heidelberg, pp. 35–57.

[37] N. Ofer, R. Benavides-Piccione, J. DeFelipe, and R. Yuste, *Structural analysis of human and mouse dendritic spines reveals a morphological continuum and differences across ages and species*, eNeuro, 9 (2022).

[38] N. Ofer, D. R. Berger, N. Kasthuri, J. W. Lichtman, and R. Yuste, *Ultrastructural analysis of dendritic spine necks reveals a continuum of spine morphologies*, Developmental Neurobiology, 81 (2021), pp. 746–757.

[39] N. Ofer and O. Shefi, *Axonal geometry as a tool for modulating firing patterns*, Applied Mathematical Modelling, 40 (2016), pp. 3175–3184.

[40] B. O'Neill, *Elementary Differential Geometry*, Academic Press, 1997.

[41] E. Pchitskaya, P. Vasiliev, D. Smirnova, V. Chukanov, and I. Bezprozvanny, *SpineTool is an open-source software for analysis of morphology of dendritic spines*, Sci. Rep., 13 (2023), p. 10561.

[42] M. Raissi, P. Perdikaris, and G. Karniadakis, *Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations*, Journal of Computational Physics, 378 (2019), pp. 686–707.

[43] C. Rao, H. Sun, and Y. Liu, *Physics-informed deep learning for incompressible laminar flows*, Theoretical and Applied Mechanics Letters, 10 (2020), pp. 207–212.

[44] E. Samaniego, C. Anitescu, S. Goswami, V. Nguyen-Thanh, H. Guo, K. Hamdia, X. Zhuang, and T. Rabczuk, *An energy approach to the solution of partial differential equations in computational mechanics via machine learning: Concepts, implementation and applications*, Computer Methods in Applied Mechanics and Engineering, 362 (2020), p. 112790.

[45] M. S. Smirnov, T. R. Garrett, and R. Yasuda, *An open-source tool for analysis and automatic identification of dendritic spines using machine learning*, PLoS One, 13 (2018), p. e0199589.

[46] N. Spruston, *Pyramidal neurons: dendritic structure and synaptic integration*, Nature Reviews Neuroscience, 9 (2008), pp. 206–221.

[47] R. Su, C. Sun, C. Zhang, and T. D. Pham, *A novel method for dendritic spines detection based on directional morphological filter and shortest path*, Computerized Medical Imaging and Graphics, 38 (2014), pp. 793–802.

[48] J. M. Sullivan, *Curvatures of smooth and discrete surfaces*, 2007.

[49] R. Tibshirani, *Regression shrinkage and selection via the lasso*, Journal of the royal statistical society series b-methodological, 58 (1996), pp. 267–288.

[50] A. Ustinova, E. Volkova, A. Rakovskaya, D. Smirnova, O. Korovina, and E. Pchitskaya, *Generate and analyze three-dimensional dendritic spine morphology datasets with SpineTool software*, Curr. Protoc., 4 (2024), p. e70061.

[51] S. van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, T. Yu, and scikit-image contributors, *scikit-image: image processing in python*, PeerJ, 2 (2014), p. e453.

[52] I. Vidaurre-Gallart, I. Fernaud-Espinosa, N. Cosmin-Toader, L. Talavera-Martínez, M. Martin-Abadal, R. Benavides-Piccione, Y. Gonzalez-Cid, L. Pastor, J. DeFelipe, and M. García-Lorenzo, *A deep learning-based workflow for dendritic spine segmentation*, Frontiers in Neuroanatomy, Volume 16 - 2022 (2022).

[53] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1. 0 Contributors, *SciPy 1.0: fundamental algorithms for scientific computing in Python*, Nature Medicine, 17 (2020), pp. 261–272.

[54] F. W. Vogel, S. Alipek, J.-B. Eppler, P. Osuna-Vargas, J. Triesch, D. Bissen, A. Acker-Palmer, S. Rumpel, and M. Kaschube, *Utilizing 2d-region-based CNNs for automatic dendritic spine detection in 3D live cell imaging*, Scientific Reports, 13 (2023), p. 20497.

[55] C.-H. Wu, T. G. Fai, P. J. Atzberger, and C. S. Peskin, *Simulation of osmotic swelling by the stochastic immersed boundary method*, SIAM Journal on Scientific Computing, 37 (2015), pp. B660–B688.

[56] X. Xiao, M. Djurisic, A. Hoogi, R. W. Sapp, C. J. Shatz, and D. L. Rubin, *Automated dendritic spine detection using convolutional neural networks on maximum intensity projected microscopic volumes*, Journal of Neuroscience Methods, 309 (2018), pp. 25–34.

[57] W. Yu and B. Lu, *Synapses and dendritic spines as pathogenic targets in alzheimer's disease*, Neural plasticity, 2012 (2012), p. 247150.

[58] C. Yuksel, *Sample elimination for generating poisson disk sample sets*, Comput. Graph. Forum, 34 (2015), p. 25–32.

[59] R. Yuste, *Dendritic spines and distributed circuits*, Neuron, 71 (2011), pp. 772–781.

[60] T. Y. Zhang and C. Y. Suen, *A fast parallel algorithm for thinning digital patterns*, Commun. ACM, 27 (1984), pp. 236–239.

[61] Q.-Y. Zhou, J. Park, and V. Koltun, *Open3D: A modern library for 3D data processing*, arXiv:1801.09847, (2018).