Enhancing Aneurysm Analysis with Surface Features

Clément Hervé,[*] Paul Garnier,[†] Jonathan Viquerat,[‡] and Elie Hachem[§]
Mines Paris, PSL University, Centre for Material Forming (CEMEF),
UMR CNRS 7635, rue Claude Daunesse, 06904 Sophia-Antipolis, France
(Dated: June 25, 2025)

Intracranial aneurysms pose a significant risk, as they often remain asymptomatic until rupture, which can lead to life-threatening complications. In this study, we leverage novel surface features to enhance the performance of neural networks for classifying and segmenting 3D intracranial aneurysms, as well as for improving blood flow simulations using graph neural networks. Our results show that incorporating these features significantly boosts the effectiveness of 3D point cloud processing models, enabling them to match or even surpass the performance of state-of-the-art approaches. This advancement not only aids in the early detection and treatment of aneurysms but also contributes to a deeper understanding of their formation and progression. Integrating surface features into neural networks provides a way to improve 3D object processing, paving the way for future research in this area.

1. Introduction

Intracranial aneurysms are life-threatening conditions that often remain undetected until rupture. Early detection is crucial to prevent such events. The main challenge is that aneurysms typically present no symptoms before rupturing, and their discovery often occurs incidentally during medical examinations for unrelated reasons. The goal is to enhance detection accuracy using various imaging techniques, such as MRI scans. Furthermore, when aneurysms are detected, surgery is often required to prevent rupture. This surgery can be facilitated by using 3D simulations of blood flow, which help surgeons better understand aneurysm behavior and plan interventions accordingly.

By leveraging 3D modeling of the brain's vascular system reconstructed from MRA images, it becomes easier to identify aneurysms. With this approach in mind, we decided to train classification and segmentation algorithms using the Intra 3D dataset [1]. This dataset provides 3D models of both healthy blood vessels and aneurysms in different formats, including meshes and point clouds. It has been included in the large MedMNIST v2 database of 3D medical models [2]. New classification and segmentation models have been trained on this dataset, such as 3DMedPT [3]. We used these models to train neural networks such as PointNet [4] and PointNet++ [5], which are designed to process 3D point clouds. These networks can learn to classify aneurysms based on their geometric features, enabling more accurate detection.

For the simulation of blood flow, we used the AnXplore dataset [6], which contains 3D models of aneurysms and their corresponding blood flow simulations. It was created using 101 aneurysms from the Intra3D dataset [1]. This dataset is particularly useful for training graph neural networks [7] (GNNs) to predict blood flow patterns in aneurysms [8]. Encode-process-decode architectures combined with attention mechanisms [9] have been used to further enhance the quality of the simulation.

To extract the surface features, we employed TRELLIS [10], an algorithm capable of generating 3D objects from text, 2D images, or other 3D objects. TRELLIS is widely used as a benchmark in the field of 3D object and scene generation and has been compared in recent research such as Hunyuan3D 2.0 [11] and LT3SD [12]. It has also been used to generate 3D assets for other studies in engineering and physics, for motion planning tasks with IMPACT [13] or reconstructing deformable object motion with PhysTwin [14]. We reused the encoding part from 3D objects to obtain embeddings and point positions from each object to improve the performance of the models.

These extracted features were then used to boost neural network performance, improving both aneurysm classification and segmentation, as well as the accuracy of blood flow simulations. We also explored the effectiveness of using only these surface features for aneurysm classification, without relying on a dedicated point cloud processing neural network. Additionally, we performed a detailed analysis of the features themselves to better understand the information they contain. PCA and t-SNE visualizations were employed to analyze the structure of the feature space. PCA was also used to classify the aneurysms. Finally, we applied clustering techniques to the AnXplore dataset to examine whether aneurysms within the same clusters had similar shapes.

2. Related Algorithms

2.1. TRELLIS encoding

TRELLIS [10] is an advanced algorithm designed to generate 3D objects from various inputs, such as text prompts, multiple 2D views of an object, or directly from another 3D object. For the 3D object-to-3D object trans-

[*] clement.herve@etu.minesparis.psl.eu
[†] paul.garnier@minesparis.psl.eu
[‡] jonathan.viquerat@minesparis.psl.eu
[§] elie.hachem@minesparis.psl.eu

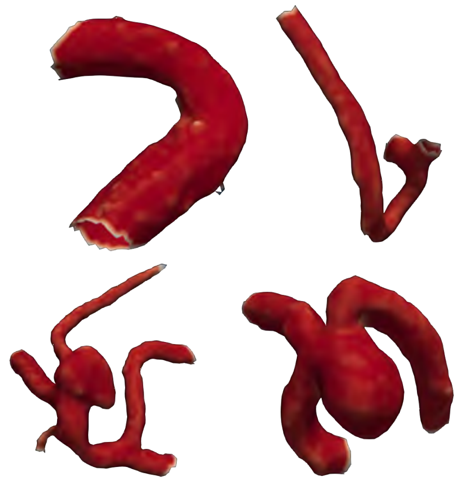


FIG. 1: Examples of intracranial aneurysms and vessels from the Intra 3D dataset.

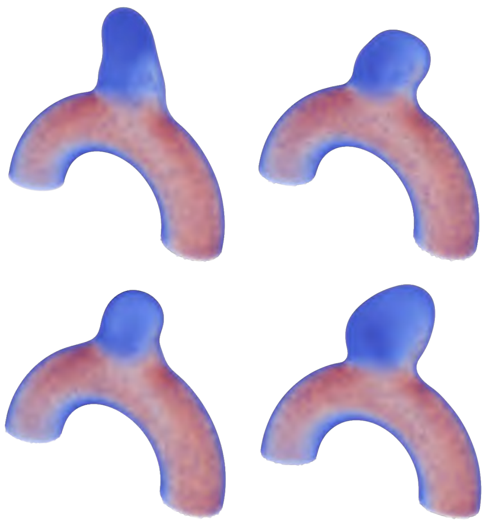


FIG. 2: Examples of aneurysms from the AnXplore dataset

formation, TRELLIS employs a sophisticated encoding process to extract detailed features from a mesh, enabling the recreation of the object with high precision. This process involves several key steps:

First, the 3D object is rendered from multiple angles to comprehensively capture its geometric details. These rendered views are then voxelized, converting the object into a grid of 3D pixels (voxels) that represent its structure. In our case, this grid is a $64 \times 64 \times 64$ array, where the active voxels correspond to the surface of the object. On average, each object contains approximately 5,000 active voxels.

The voxelized object is then processed to calculate features for each voxel. TRELLIS uses the voxelized representation combined with random views of the object, which are processed using a pre-trained DINOv2 autoencoder [15]. This mechanism enhances encoding efficiency by focusing on the most relevant parts of the object, ensuring that essential features are accurately captured.

The result of this encoding process is a point cloud, where each point is characterized by its position in 3D space and a set of features that describe its properties. This encoder can be used to extract features directly from a mesh, which can then be used to train a model.

This process is illustrated in Figure 3.

2.2. Point Cloud-Based Methods

Point cloud-based methods are widely used for 3D object classification and segmentation. We explored two models: PointNet [4] and PointNet++ [5]. These two benchmark algorithms are effectively used for processing 3D objects, for example in the Intra 3D dataset [1] or for non-medical data such as ModelNet40 [16]. PointNet is a deep learning architecture designed to process unordered 3D point clouds $\mathcal{X} = \{x_1, x_2, \dots, x_N\} \subset \mathbb{R}^d$, where each point x_i typically lies in $\mathbb{R}^3$. To maintain permutation invariance, the network applies a shared multi-layer perceptron (MLP) $\phi : \mathbb{R}^d \to \mathbb{R}^k$ to each point individually, producing features $h_i = \phi(x_i)$.

These features are then aggregated using a symmetric function to obtain a global feature vector:

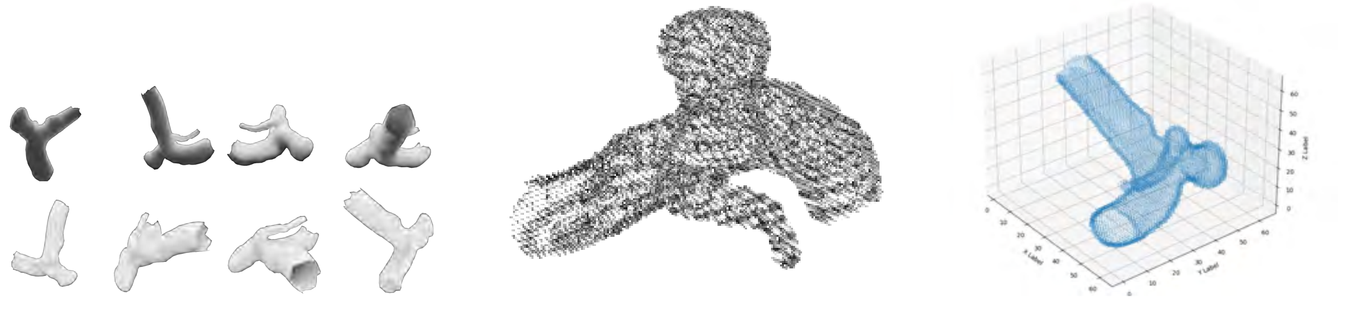
$$g = \max_{i=1,\dots,N} h_i.$$

This vector $g \in \mathbb{R}^k$ captures the overall structure of the point cloud and can be used for classification. For segmentation tasks, g is concatenated with each local feature h_i to form contextual descriptors $f_i = \mathrm{Concat}(h_i, g)$, which are then passed through another shared MLP to produce the output label for each point.

PointNet also uses a transformation network, or T-Net, which learns an affine transformation matrix T to align the input data or feature space. The transformation is applied as $x_i' = Tx_i$, and a regularization loss $\|I - TT^\top\|_F^2$ is added to encourage T to be close to orthogonal. This improves invariance to geometric transformations such as rotation and scaling. Overall, PointNet approximates a set function by combining shared point-wise functions with symmetric aggregation, enabling robust learning directly from raw point sets.

PointNet++ improves upon the original PointNet by incorporating local feature learning at different spatial scales. It processes a point cloud $\mathcal{X} = \{x_1, \dots, x_N\} \subset \mathbb{R}^d$ hierarchically using a series of *Set Abstraction* (SA) layers, each consisting of three key steps: sampling, grouping, and local feature extraction.

For sampling, a subset of points $\mathcal{X}' \subset \mathcal{X}$ is selected using Farthest Point Sampling (FPS) to serve as centroids for local neighborhoods. Then, for grouping, for each sampled point x_i', its local neighborhood $\mathcal{N}(x_i')$ is defined, typically using k-nearest neighbors. Finally, for feature extraction, a PointNet is applied in each local neighborhood to learn a feature representation



(a) Rendered views of 3D objects (b) Voxelized view (c) Point cloud

FIG. 3: Illustrations of rendered views, voxelized view, and point cloud of an aneurysm from the Intra 3D dataset.

$f_i = \mathrm{PointNet}(\mathcal{N}(x_i'))$. These features are then aggregated across neighborhoods to form a new, subsampled point cloud with refined features.

This process is repeated several times, producing an abstract representation of the point cloud. To recover point-wise features for segmentation tasks, PointNet++ employs *Feature Propagation* layers that use interpolation and skip connections to upsample and merge local and global information.

By modeling local geometric structures and using a multi-scale representation, PointNet++ effectively processes non-uniform point clouds and better captures spatial details, achieving higher performance for 3D classification and segmentation compared to PointNet.

2.3. Graph Neural Networks (GNNs)

Graph Neural Networks (GNNs) are a class of neural networks designed to operate on graph-structured data. They are particularly well-suited for tasks where the data can be represented as nodes and edges, such as 3D meshes.

The main idea behind GNNs is the message-passing mechanism, where each node in the graph iteratively updates its representation by aggregating information from its neighbors. This process can be broken down into three main steps: computing messages for each connection between nodes based on their features, aggregating these messages for each node from its neighbors, and updating the node's features based on the aggregated message and its current features.

This concept has been adapted for 3D aneurysm modeling to simulate hemodynamics in graph-physics [8]. The encode-process-decode framework facilitates the application of GNNs on mesh structures, enabling the simulation of blood flow in aneurysms. The performance of the simulations is further improved by using attention mechanisms [9] to focus on the most relevant parts of the mesh, enhancing both accuracy and efficiency.

For the simulation, graph-physics [8] represents the mesh by an undirected graph $G = (V, E)$ with nodes $V = \{x_i\}_{i=1}^N$, each $x_i \in \mathbb{R}^p$. The input is the matrix $X \in \mathbb{R}^{N\times p}$. The model follows an encode-process-decode architecture: the encoder maps X into a latent space $Z_0 = \mathrm{MLP}(X) \in \mathbb{R}^{N\times d}$ using two linear layers. The processor applies L transformer blocks; each block uses masked multi-head self-attention with the adjacency matrix $A \in \{0,1\}^{N\times N}$ as a mask, computed as

$$\mathrm{Attention}(Z) = \mathrm{softmax}\left(\frac{QK^\top \odot A}{\sqrt{d}}\right) V,$$

where Q, K, V are learned linear projections of Z. A Gated MLP with GeLU non-linearity follows, defined as

$$Z = W_f\big(\mathrm{GeLU}(W_l Z + b_l) \odot (W_r Z + b_r)\big) + b_f,$$

With residual connections and RMS normalization after each sub-layer. The decoder maps Z_L back to the output space via two linear layers.

To improve the receptive field and information flow, the adjacency matrix is augmented by dilated adjacency with k-hop neighbors, random edges added dynamically, and global attention connecting important nodes to all others.

3. Datasets

3.1. Intra 3D Dataset

The Intra 3D dataset [1] contains 3D models of intracranial aneurysms and healthy blood vessels. The classification dataset is imbalanced, with 1,694 healthy vessels and 215 aneurysms. There are also 116 annotated aneurysms used to train segmentation models, which can also be used to expand the classification dataset. The aneurysms and vessels are reconstructed from 2D MRA scans. The dataset is available in various formats, including meshes and point clouds, and is used for both classification and segmentation tasks. The original paper also presents classification and segmentation results using models such as PointNet [4], PointNet++ [5], PointCNN [17], SO-net [18], and PointConv [19].

3.2. AnXplore Dataset

The AnXplore dataset [6] contains 101 3D models of intracranial aneurysms, each with associated blood flow simulations. These aneurysms are extracted from the 116 annotated aneurysms in Intra3D [1]. The head of each aneurysm is isolated and placed on the same uniform vessel, meaning all aneurysms in AnXplore [6] are located on the same vessel but vary in shape and size. The dataset provides volumetric data from the blood flow simulations. For TRELLIS [10], we extract the surface mesh from the first time step of each simulation, as the aneurysm geometry does not change during the simulation.

4. Analysis of the TRELLIS features

We conducted several analyses on the TRELLIS features to better understand them and to evaluate how well these features represent the objects.

First, for each object, we computed different metrics. Since each point is described by an 8-dimensional feature vector, we calculated the mean, standard deviation, minimum, and maximum for each feature across all points in an object. This means each object is represented by four vectors of size 8. We then performed 2D PCA on these vectors for each category, first on the Intra3D dataset [1], and then on the combined Intra3D and AnXplore datasets [6]. Results for the mean and standard deviation features are shown in Figure 5; additional results are provided in the appendix.

To determine whether the information in other components could better separate aneurysms, we also performed t-SNE on the combined dataset, as shown in Figure 4 for the mean features.

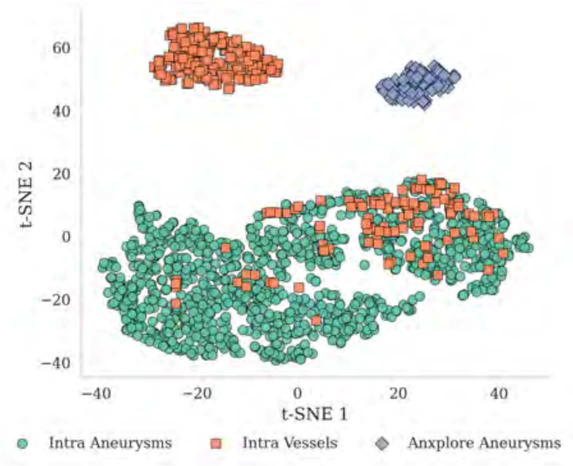


FIG. 4: Result of t-SNE on the mean features of the Intra3D and AnXplore datasets for classification.

We observe that for the 2D PCA, the mean and standard deviation are very effective at separating vessels and aneurysms, while the minimum and maximum are less so. However, the AnXplore dataset, which contains only aneurysms, is very well separated from the Intra3D dataset for three out of four metrics. With t-SNE, the separation is even clearer: AnXplore is always well separated from Intra3D, and within Intra3D, aneurysms, and vessels are extremely well separated using the mean and minimum of the features.

To further explore how the features and PCA can help classify aneurysms, we trained machine learning algorithms on the resulting 2D point clouds, as presented in the next section.

On the annotated aneurysms from Intra3D [1], we also performed a 2D PCA on the features for all points of a single element to observe the differences between features from the aneurysm part and the vessel part. However, this approach did not work well, and even a 2D t-SNE did not yield satisfactory results. We then tried computing the mean, standard deviation, minimum, and maximum for points from the aneurysm and vessel parts separately and performed t-SNE over the 116 annotated aneurysms. This proved to be very effective and showed that the two parts—the aneurysm and the vessel—are encoded very differently. Results are shown in Figure 6.

Finally, we performed 2D PCA over the mean, standard deviation, minimum, and maximum of the features from each element of the AnXplore dataset. We then ran clustering algorithms to see if aneurysms in the same cluster had similar shapes and sizes. The optimal number of clusters for the 116 elements was 15, and we observed interesting results: within each cluster, aneurysms were of similar size and shape, indicating that TRELLIS features are very effective at describing the shape of a 3D aneurysm. Results can be seen in Figure 7 for clusters based on the mean features.

5. Methodology

5.1. Classification

Four methods were selected for the classification process: PointNet [4], PointNet++ [5], a simple MLP, and direct use of PCA point clouds to train either an MLP or a logistic regression model. For the first two models, we trained them on the Intra3D dataset [1], first using only the data provided by the dataset, and then with the additional features from TRELLIS [10] to compare the performance of both with identical neural networks. For the MLP, we performed a single training using only the features (without positions) to evaluate the performance of a very simple algorithm. For the two models with PCA, we directly used the four PCA point clouds with the mean, standard deviation, minimum, and maximum values.

For the PointNet [4] model, we did not use the original architecture but a slightly improved version, incorporating message passing to aggregate the features but not using the T-net module. For the baseline results, we

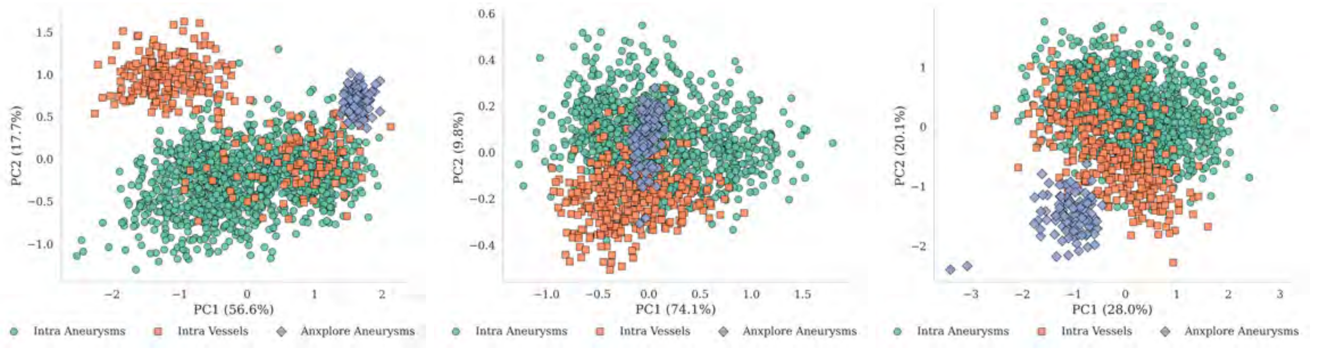


FIG. 5: Results of PCA on the Intra3D and AnXplore datasets. The left figure shows the mean features, the middle figure shows the standard deviation features and the right figure shows the minimum features.

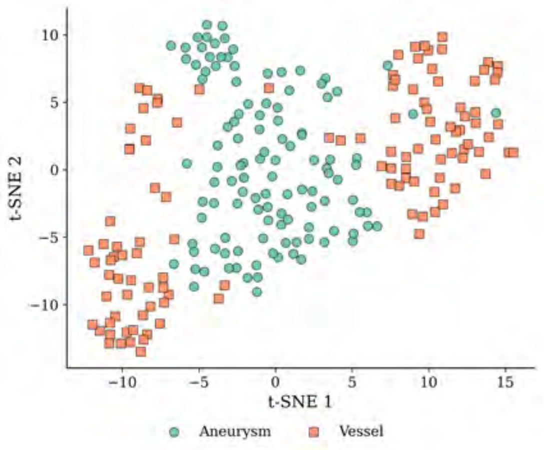


FIG. 6: t-SNE on annotated aneurysms from Intra3D [1] using the mean features of the aneurysm and vessel parts.

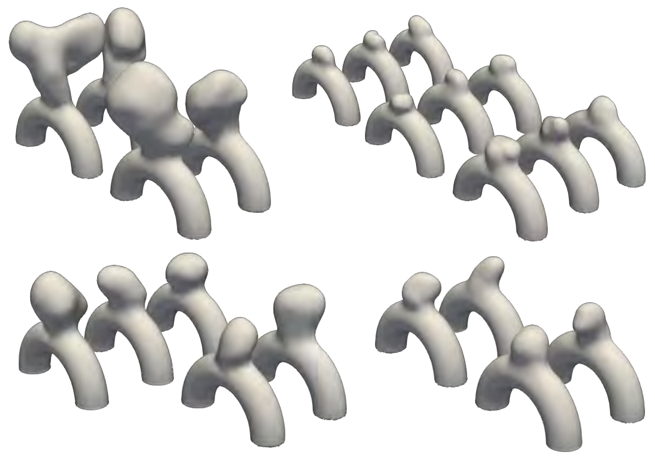


FIG. 7: Clustering results on AnXplore dataset

used the positions from the point cloud combined with the surface normal vector at each point as features. We then ran the same algorithm but replaced the normals with the TRELLIS [10] features at each point. The architecture consists of five modified PointNet layers, each with two layers of 64 neurons combined by ReLU. This is followed by a global max pooling layer and a linear layer of size 64 with an output of size 2, combined with a log softmax function. For each layer of message aggregation, the maximum number of neighbors is set to 16.

To implement PointNet++ [5], we directly reused the original architecture. We followed the same approach as with PointNet: first using the point coordinates and normals, and then another run using the point coordinates and the TRELLIS [10] features. The first layer is a set abstraction layer with an MLP consisting of one layer of 64 neurons and two layers of 128 neurons. The ratio for the farthest point sampling is set to 0.5, and the number of neighbors for the grouping phase is set to 64, within a radius of 0.2 (each element is rescaled to a normalized scale). The second layer is a set abstraction layer with two layers of 128 neurons and a final layer of size 256. The ratio for the farthest point sampling is set to 0.25, and the number of neighbors for the grouping phase is set to 64, within a radius of 0.4. Then, we have a global feature aggregation layer, using an MLP with one layer of 256 neurons, one layer of 512 neurons, and a final layer of size 1024. This is followed by a final MLP with one layer of 1024 neurons, one of 512 neurons, and an output layer of size 2, on which we used a 50% dropout, combined with a log softmax function.

The MLP was trained using only the features of the points, not their positions. Each point has 8 features extracted from the mesh. We used simple MLP blocks to process these features and classify the points as aneurysms or healthy vessels. The MLP architecture is divided into two modules: the first processes the features for each point and is an MLP with one layer of 8 neurons (the number of features per point), one layer of 16 neurons, one layer of 8 neurons, and a final layer of size 1. The second module is a global MLP that processes the output of the first module; it has one layer whose size is the number of points sampled from the point cloud, one layer of 256 neurons, one layer of 64 neurons, one layer of 16 neurons, and an output layer of size 2, combined with a softmax function. The first module is shared between all points, and the second module is shared between all elements of the dataset. Both MLPs are trained using 40% dropout. This architecture is shown in Figure 8.

The three models were trained for 100 epochs with a batch size of 16. We used the AdamW optimizer with a learning rate of 0.001 and a cosine weight decay schedule.

For the algorithms using PCA, we trained an MLP and performed logistic regression on the 2D point clouds. For both, we tried using only the mean feature, then another training with mean and standard deviation, and finally one with all four metrics. This was done to show how well the features can represent the data type and to determine which metrics from the set of features for each element of the dataset are most useful. The architecture of the MLP is a first layer of size 8 (the number of features per point), a second layer of size 16, five layers of size 64, one layer of size 32, and a final layer of size 2, combined with ReLU functions and a dropout of 40% during training. It is trained for 100 epochs with a learning rate of 0.01. The logistic regression is trained on the same data for 1000 iterations to ensure convergence.

5.2. Segmentation

For the segmentation process, we used two algorithms: PointNet [4] and PointNet++ [5]. For both, we adopted architectures similar to those used for classification. Specifically, for PointNet, we used the slightly improved version, modifying the last layer for per-point segmentation instead of a global pooling layer. For PointNet++, we used the original segmentation architecture. For both models, we compared performance using point clouds with surface normals as features and then with TRELLIS features.

Our models were trained for 200 epochs with a batch size of 8. We used the AdamW optimizer with a learning rate of 0.001 and a cosine weight decay schedule. For the PointNet [4] architecture, we used five PointNet layers, each with two layers of 32 neurons combined by ReLU. The last layer is a linear layer of size 32 with an output of size 2, combined with a log softmax function. For each layer of message aggregation, the maximum number of neighbors is set to 16.

For PointNet++ [5], the first layer is a set abstraction layer with three layers of 64 neurons and a final layer of size 128. The ratio for the farthest point sampling is set to 0.2, and the number of neighbors for the grouping phase is set to 64, within a radius of 0.2 (each element is rescaled to a normalized scale). The second layer is a set abstraction layer with three layers of 128 neurons and a final layer of size 256. The ratio for the farthest point sampling is set to 0.25, and the number of neighbors for the grouping phase is set to 64, within a radius of 0.4. Then, we have a global feature aggregation layer, using an MLP with one layer of 256 neurons, two layers of 512 neurons, and a final layer of size 1024. This is followed by three feature propagation modules: the first has three layers of 256 neurons, the second has a first layer of 128 neurons, a hidden layer of 256 neurons, and a final layer of size 128, and the last one has four layers of 128 neurons. Finally, each point is classified with a final MLP with three layers of 128 neurons and an output layer of size 2, combined with a log softmax function.

5.3. Graph Neural Networks for Blood Flow Simulation

For blood flow simulation, we used the AnXplore dataset [6] and the GNN architecture from graph-physics [8] without modifying the original architecture. The main modification to the original work was the inclusion of surface features from TRELLIS [10], combined with the original mesh features for one run, and another without those features.

6. Results

In this section, we present the results of our experiments on the Intra 3D dataset [1] and the AnXplore dataset [6]. We evaluate the performance of different models for classification, segmentation, and blood flow simulation. All classification and segmentation methods were evaluated using 5-fold cross-validation, and the results are averaged over the 5 folds.

The results of the classification experiments on the Intra 3D dataset [1] are shown in Table I. We compare the performance of PointNet [4], PointNet++ [5], MLP, and PCA-based methods. The results show that using TRELLIS features significantly improves the performance of both PointNet and PointNet++ compared to using only point coordinates and normals. The MLP model also performs well when trained on TRELLIS features, achieving high accuracy.

Our best results are obtained with PointNet++ using TRELLIS features, reaching a vessel accuracy of 98.97% and an F1 score of 97.82%. The best model in terms of aneurysm accuracy is a simple MLP trained with TRELLIS features, achieving an aneurysm accuracy of 97.33% and an F1 score of 95.32%. These results show that with TRELLIS features, we can outperform state-of-the-art models on the Intra 3D dataset [1], even with classic neural networks and not only with point cloud processing architectures.

The segmentation results on the Intra 3D dataset [1] are shown in Table II. We compare the performance of PointNet [4] and PointNet++ [5] models, both with and without TRELLIS features. The results indicate that both models achieve strong accuracy in segmenting aneurysms and healthy vessels, with PointNet++ generally outperforming PointNet. The addition of TRELLIS features consistently leads to a notable improvement in segmentation performance.

Our best results are achieved with PointNet++ using TRELLIS features. The difference between models us-

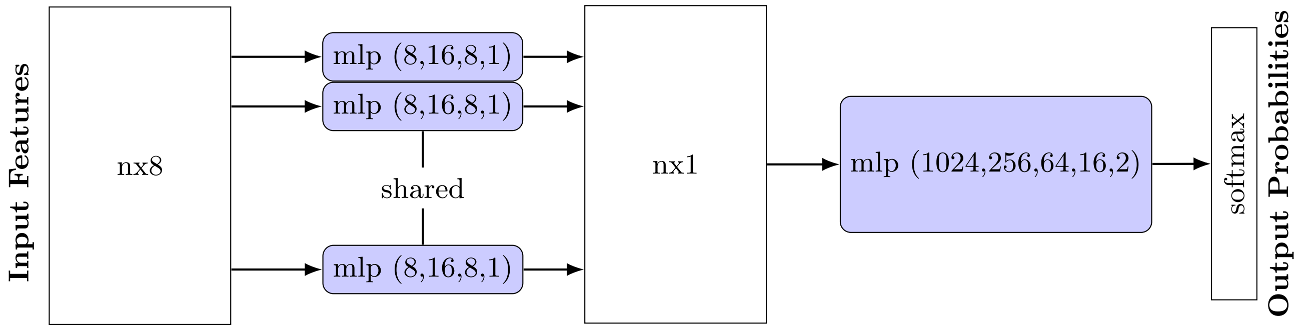


FIG. 8: MLP Architecture for Point Cloud Classification with TRELLIS Encoding

Model	Input	V. (%)	A. (%)	F1
MLP + feats	512	94.02	91.62	92.61
	1024	89.19	**97.33**	95.32
	2048	83.79	91.07	86.25
PointNet [4]	512	91.79	52.09	82.09
	1024	92.48	52.33	82.74
	2048	91.70	45.17	80.19
PointNet [4] + feats	512	98.26	93.34	91.16
	1024	98.69	93.10	97.42
	2048	**98.97**	94.10	**97.82**
PointNet++ [5]	512	90.30	52.59	81.30
	1024	89.69	53.01	81.06
	2048	90.57	51.29	81.05
PointNet++ [5] + feats	512	98.34	89.19	96.18
	1024	98.53	87.12	95.89
	2048	98.35	89.66	96.32
MLP on PCA	mean	**100**	51.56	87.66
	mean and std	99.56	68.44	92.01
	all metrics	98.70	**78.07**	**93.82**
Logistic Regression on PCA	mean	96.43	65.40	88.96
	mean and std	98.61	73.83	92.73
	all metrics	98.00	75.64	92.71

TABLE I: Comparison of the different models on the Intra 3D dataset. Results are mean values for vessel segment accuracy (V.), aneurysm segment accuracy (A.), and F1-score.

Model	Points	IoU V.	IoU A.	DSC V.	DSC A.
PointNet [4]	512	88.08	66.38	93.65	79.75
	1024	85.81	60.17	92.18	75.09
	2048	81.16	50.95	89.59	67.19
PointNet [4] + feats	512	90.51	72.71	95.01	84.19
	1024	90.49	72.31	95.00	83.86
	2048	90.00	71.01	94.73	83.00
PointNet++ [5]	512	87.87	64.75	93.52	78.50
	1024	88.52	66.68	93.91	79.90
	2048	89.33	67.98	94.30	80.89
PointNet++ [5] + feats	512	91.49	74.62	95.56	85.43
	1024	92.29	76.86	95.99	86.89
	2048	**93.55**	**79.98**	**96.67**	**88.87**

TABLE II: Comparison of the different models on the Intra 3D dataset. Results are mean values for IoU and DSC on vessels (V.) and aneurysms (A.).

ing only the original features (point coordinates and normals) and those using TRELLIS features is clear. Moreover, the models with TRELLIS features outperform the PointNet and PointNet++ results reported in [1] that use only the original features, confirming the effectiveness of TRELLIS features for segmentation tasks. However, we did not surpass the very best state-of-the-art segmentation results from [1] obtained with SO-net [18], but our results are very close, and slightly better in terms of aneurysm DSC.

The results of the GNN + Transformers model [8] on the AnXplore dataset [6] are shown in Table III. The model was trained using the features extracted from TRELLIS [10] and the mesh structure of the aneurysms. The results indicate that the model achieves a lower RMSE (Root Mean Square Error) across all time steps of the blood flow simulation with the addition of TRELLIS features compared to using only the original features (point coordinates and normals). The error is reduced by approximately 15%, demonstrating the effectiveness of these features for blood flow simulation tasks.

Additionally, this approach impacts the training time for the models. For point cloud processing, using the MLP allows us to reduce the training time by a factor of three compared to models like PointNet and PointNet++.

Another important aspect is the encoding time: encoding 400 objects with TRELLIS on an A100 GPU takes about 12 hours, as it is about five minutes per object. While this is a significant amount of time, it is a one-time process—after encoding, we can train different models on the resulting dataset. In comparison, running the entire pipeline of encoding and training on the full dataset would increase the total time by a factor of 30 compared to training point cloud processing models such as Point-

Model	All-Rollout RMSE	
	Mean	Std
S/n	7.57	1.103
S/n + feats	6.09	0.637
L/n	4.03	0.330
L/n + feats	**3.55**	**0.017**

TABLE III: Results of the GNN + Transformers model on the AnXplore dataset. The All-Rollout RMSE is computed over all time steps of the blood flow simulation.

Net or PointNet++ alone.

7. Conclusion

In this work, we explored the use of TRELLIS [10] features for 3D object classification and segmentation tasks, specifically on the Intra 3D dataset [1] and the AnXplore dataset [6]. We showed that using TRELLIS features significantly improves the performance of PointNet [4] and PointNet++ [5] compared to using only point coordinates and normals.

We also applied TRELLIS features to the AnXplore dataset [6] for blood flow simulation using Graph Neural Networks (GNNs), demonstrating the versatility of these features for different 3D tasks.

Moreover, we found that a simple MLP trained on TRELLIS features achieves competitive results, outperforming state-of-the-art models on the Intra 3D dataset [1]. We also analyzed the TRELLIS features, showing that they effectively capture the shape and size of aneurysms, enabling clustering based on these characteristics.

Overall, we conclude that TRELLIS features are a powerful tool for 3D object classification and segmentation, and offer a promising direction for future research in this area. The ability to encode complex 3D structures into meaningful features opens up new possibilities for improving the performance of various 3D models.

For future work, it would be interesting to apply TRELLIS features to other 3D object classification or segmentation tasks. For instance, PointNet or PointNet++ could be evaluated on different 3D object datasets. Comparing results across these datasets could help assess how well TRELLIS features generalize to various types of 3D objects and tasks. Additionally, testing other models such as SO-net [18] or PointCNN [17] with TRELLIS features could provide further insights into their effectiveness across different architectures.

To address the encoding time, a more detailed study could be conducted to determine how much the number of views can be reduced without negatively impacting the performance of the classification and segmentation models.

References

[1] Xi Yang, Ding Xia, Taichi Kin, and Takeo Igarashi. Intra: 3d intracranial aneurysm dataset for deep learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[2] Jiancheng Yang, Rui Shi, Donglai Wei, Zequan Liu, Lin Zhao, Bilian Ke, Hanspeter Pfister, and Bingbing Ni. Medmnist v2 - a large-scale lightweight benchmark for 2d and 3d biomedical image classification. *Scientific Data*, 10(1), January 2023. ISSN 2052-4463. doi:10.1038/s41597-022-01721-8. URL `http://dx.doi.org/10.1038/s41597-022-01721-8`.

[3] Jianhui Yu, Chaoyi Zhang, Heng Wang, Dingxin Zhang, Yang Song, Tiange Xiang, Dongnan Liu, and Weidong Cai. 3d medical point transformer: Introducing convolution to attention networks for medical point cloud analysis, 2021. URL `https://arxiv.org/abs/2112.04863`.

[4] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *CoRR*, abs/1612.00593, 2016. URL `http://arxiv.org/abs/1612.00593`.

[5] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *CoRR*, abs/1706.02413, 2017. URL `http://arxiv.org/abs/1706.02413`.

[6] Goetz A, Jeken-Rico P, Pelissier U, Chau Y, Sédat J, and Hachem E. Anxplore: a comprehensive fluid-structure interaction study of 101 intracranial aneurysms. 2024.

[7] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W. Battaglia. Learning mesh-based simulation with graph networks. 2021. URL `https://arxiv.org/abs/2010.03409`.

[8] Jonathan Viquerat Paul Garnier, Vincent Lannelongue and Elie Hachem. Training transformers to simulate complex physics.

[9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL `http://arxiv.org/abs/1706.03762`.

[10] Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang. Structured 3d latents for scalable and versatile 3d generation. *arXiv preprint arXiv:2412.01506*, 2024.

[11] Z. Zhao, Z. Lai, Q. Lin, Y. Zhao, H. Liu, S. Yang, Y. Feng, M. Yang, S. Zhang, X. Yang, H. Shi, S. Liu, J. Wu, Y. Lian, F. Yang, R. Tang, Z. He, X. Wang, J. Liu, X. Zuo, Z. Chen, B. Lei, H. Weng, J. Xu, Y. Zhu, X. Liu, L. Xu, C. Hu, S. Yang, S. Zhang, Y. Liu, T. Huang, L. Wang, J. Zhang, M. Chen, L. Dong, Y. Jia, Y. Cai, J. Yu, Y. Tang, H. Zhang, Z. Ye, P. He, R. Wu, C. Zhang, Y. Tan, J. Xiao, Y. Tao, J. Zhu, J. Xue, K. Liu, C. Zhao, X. Wu, Z. Hu, L. Qin, J. Peng, Z. Li, M. Chen, X. Zhang, L. Niu, P. Wang, Y. Wang, H. Kuang, Z. Fan, X. Zheng, W. Zhuang, Y. He, T. Liu, Y. Yang,

D. Wang, Y. Liu, J. Jiang, J. Huang, and C. Guo. Hunyuan3d 2.0: Scaling diffusion models for high resolution textured 3d assets generation, 2025. URL `https://arxiv.org/abs/2501.12202`.

[12] Quan Meng, Lei Li, Matthias Nießner, and Angela Dai. LT3SD: Latent Trees for 3D Scene Diffusion. *arXiv e-prints*, art. arXiv:2409.08215, September 2024. doi:10.48550/arXiv.2409.08215.

[13] Yiyang Ling, Karan Owalekar, Oluwatobiloba Adesanya, Erdem Bıyık, and Daniel Seita. Impact: Intelligent motion planning with acceptable contact trajectories via vision-language models, 2025. URL `https://arxiv.org/abs/2503.10110`.

[14] Hanxiao Jiang, Hao-Yu Hsu, Kaifeng Zhang, Hsin-Ni Yu, Shenlong Wang, and Yunzhu Li. Phystwin: Physics-informed reconstruction and simulation of deformable objects from videos, 2025. URL `https://arxiv.org/abs/2503.17973`.

[15] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2024. URL `https://arxiv.org/abs/2304.07193`.

[16] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1912–1920, 2015. doi:10.1109/CVPR.2015.7298801.

[17] Yangyan Li, Rui Bu, Mingchao Sun, and Baoquan Chen. Pointcnn. *CoRR*, abs/1801.07791, 2018. URL `http://arxiv.org/abs/1801.07791`.

[18] Jiaxin Li, Ben M. Chen, and Gim Hee Lee. So-net: Self-organizing network for point cloud analysis. *CoRR*, abs/1803.04249, 2018. URL `http://arxiv.org/abs/1803.04249`.

[19] Wenxuan Wu, Zhongang Qi, and Fuxin Li. Pointconv: Deep convolutional networks on 3d point clouds. *CoRR*, abs/1811.07246, 2018. URL `http://arxiv.org/abs/1811.07246`.

Appendix A: Comparisons of the Performances of the Different Models

Model	**Features**	**Input**	**V. (%)**		**A. (%)**		**F1**	
			mean	std	mean	std	mean	std
MLP	with	512	94.02	8.93	91.62	4.28	92.61	5.77
		1024	89.19	2.62	**97.33**	8.72	95.32	2.20
		2048	83.79	14.55	91.07	12.70	86.25	8.77
PointNet [4]	without	512	91.79	3.26	52.09	8.59	82.09	2.80
		1024	92.48	2.06	52.33	4.52	82.74	0.59
		2048	91.70	1.90	45.17	5.93	80.19	1.65
PointNet [4]	with	512	98.26	0.79	93.34	3.41	91.16	1.09
		1024	98.69	**0.55**	93.10	**1.76**	97.42	**0.52**
		2048	**98.97**	0.71	94.10	5.86	**97.82**	1.77
PointNet++ [5]	without	512	90.30	2.04	52.59	2.61	81.30	1.24
		1024	89.69	2.23	53.01	2.36	81.06	1.23
		2048	90.57	2.28	51.29	4.49	81.05	1.55
PointNet++ [5]	with	512	98.34	0.94	89.19	4.61	96.18	1.47
		1024	98.53	0.77	87.12	5.02	95.89	1.08
		2048	98.35	0.92	89.66	3.35	96.32	0.90
MLP on PCA	mean		**100**	**0.00**	51.56	9.21	87.66	2.73
	mean and std		99.56	0.39	68.44	9.10	92.01	2.21
	all metrics		98.70	1.58	**78.07**	8.55	**93.82**	2.03
Logistic Regression on PCA	mean		96.43	1.08	65.40	5.74	88.96	1.39
	mean and std		98.61	0.70	73.83	**4.63**	92.73	**1.21**
	all metrics		98.00	0.44	75.64	7.07	92.71	1.80

TABLE IV: Full comparison of the different models on the Intra 3D dataset. Results are mean and standard deviation (std) on vessels segment accuracy (V.), aneurysms segment accuracy (A.), and F1-score.

Model	Features	Input	IoU V. (%)		IoU A. (%)		DSC V. (%)		DSC A. (%)	
			mean	std	mean	std	mean	std	mean	std
PointNet [4]	without	512	88.08	1.74	66.38	3.84	93.65	0.99	79.75	2.54
		1024	85.81	1.13	60.17	3.31	92.18	0.66	75.09	2.64
		2048	81.16	2.04	50.95	7.97	89.59	1.26	67.19	7.45
PointNet [4]	with	512	90.51	1.94	72.71	**1.54**	95.01	1.08	84.19	**1.08**
		1024	90.49	2.13	72.31	4.87	95.00	1.17	83.86	3.25
		2048	90.00	1.90	71.01	3.84	94.73	1.06	83.00	2.63
PointNet++ [5]	without	512	87.87	2.80	64.75	5.50	93.52	1.62	78.50	4.05
		1024	88.52	1.08	66.68	5.67	93.91	0.60	79.90	4.17
		2048	89.33	1.21	67.98	3.69	94.30	0.68	80.89	2.60
PointNet++ [5]	with	512	91.49	0.77	74.62	3.34	95.56	0.42	85.43	2.22
		1024	92.29	0.87	76.86	2.81	95.99	0.47	86.89	1.80
		2048	**93.55**	**0.49**	**79.98**	1.99	**96.67**	**0.26**	**88.87**	1.23

TABLE V: Full comparison of the different models on the Intra 3D dataset. Results are mean and standard deviation (std) on vessels segment accuracy (V.), aneurysms segment accuracy (A.), and F1-score.

Appendix B: Supplementary figures on the analysis of the TRELLIS features

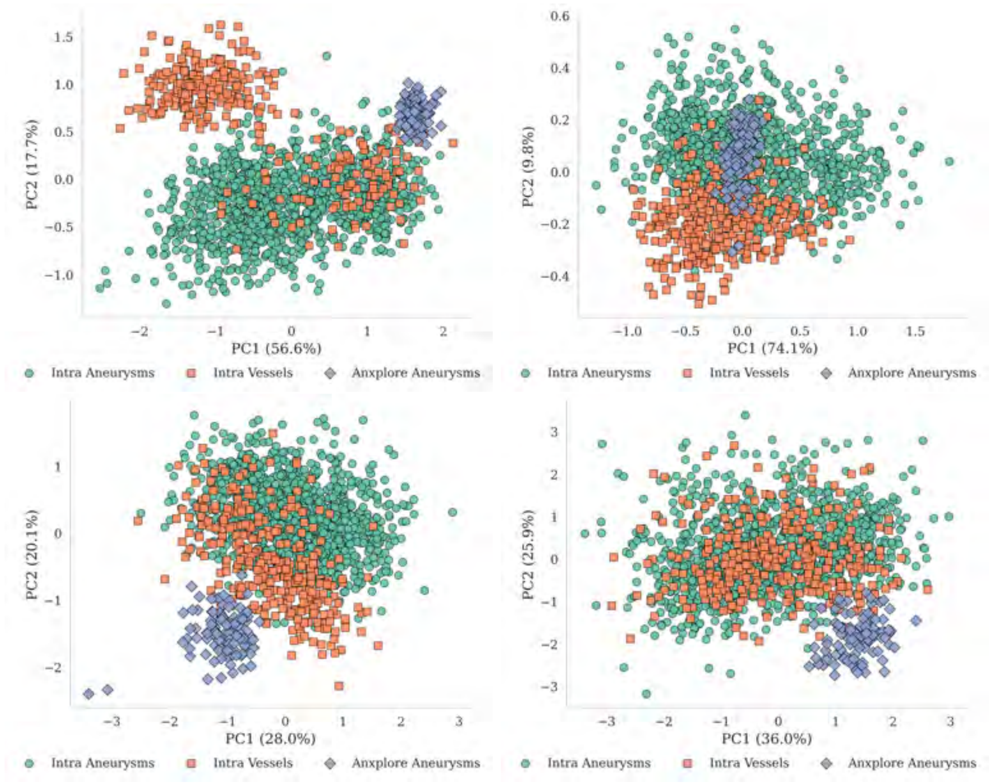


FIG. 9: Results of PCA on the Intra 3D dataset combined with the AnXplore dataset.

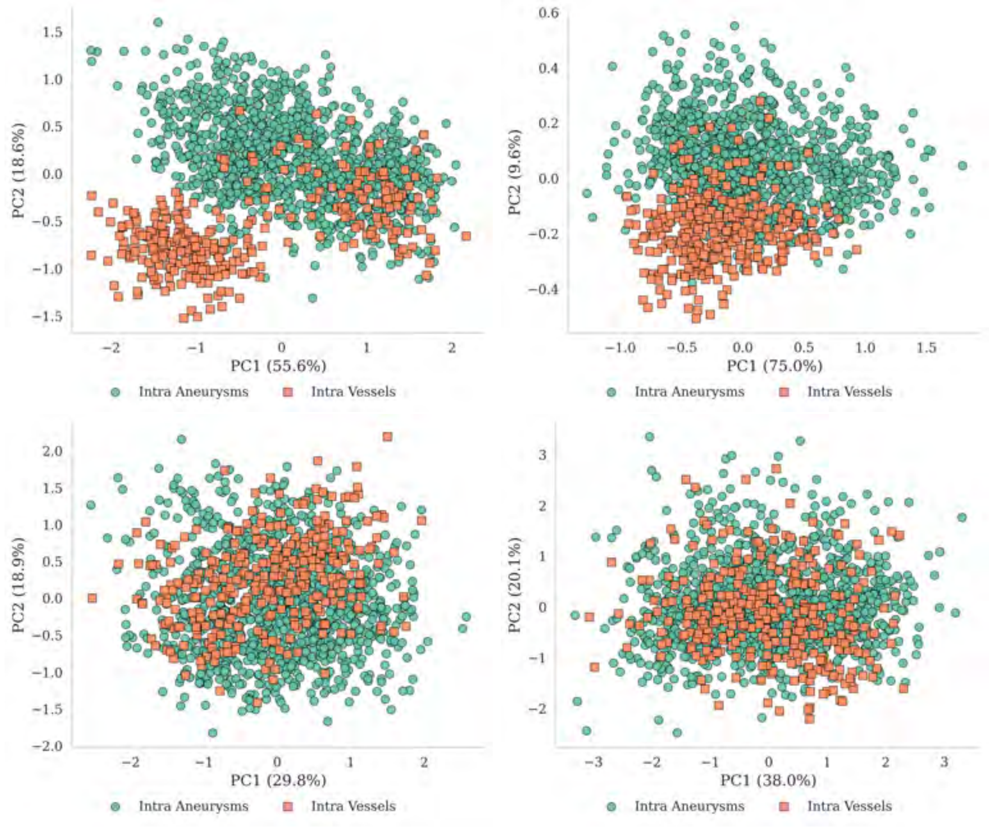


FIG. 10: Results of PCA on the Intra 3D dataset.

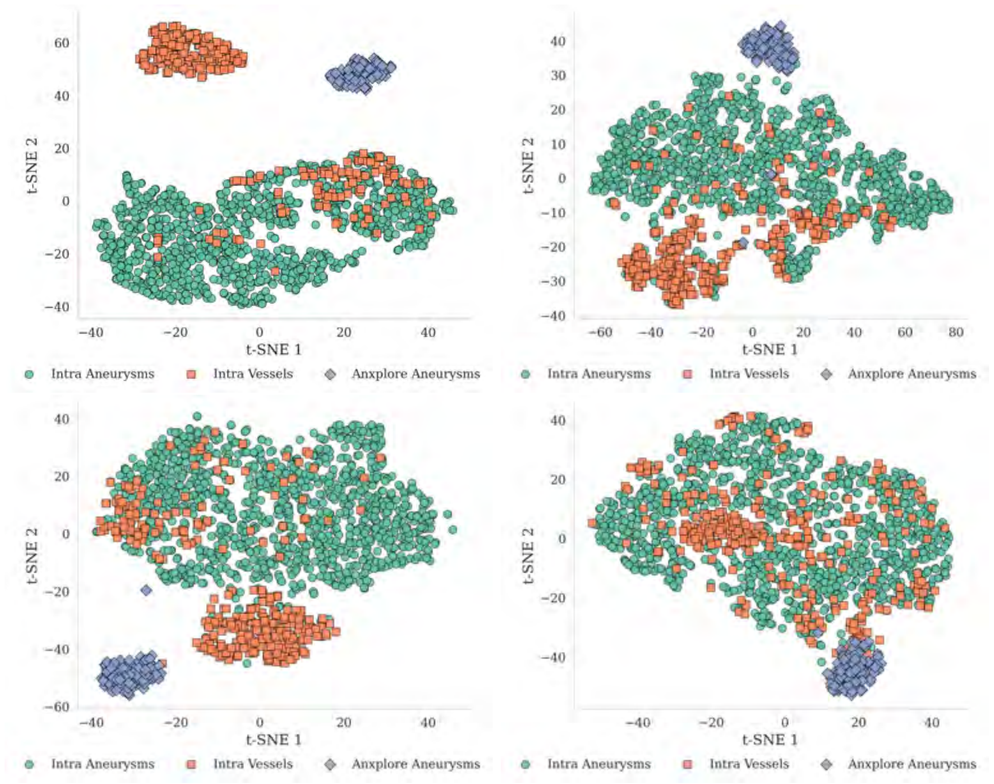


FIG. 11: Result of t-SNE on the Intra 3D dataset combined with the AnXplore dataset.

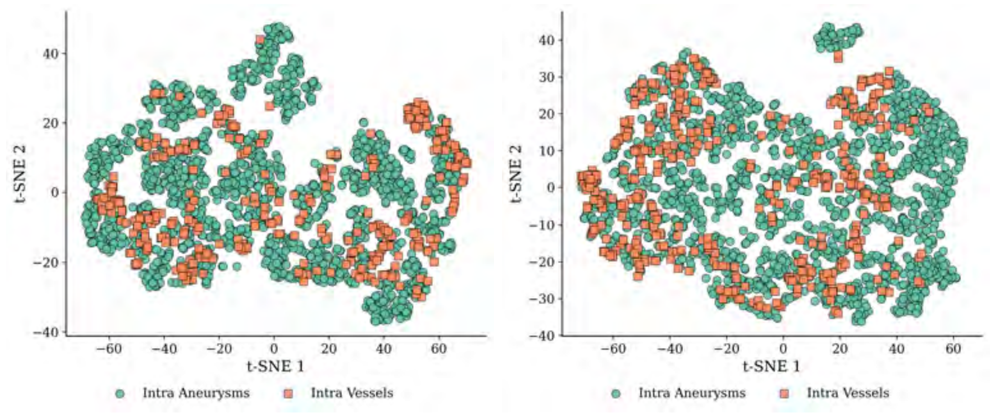


FIG. 12: t-SNE on all points of two different aneurysms

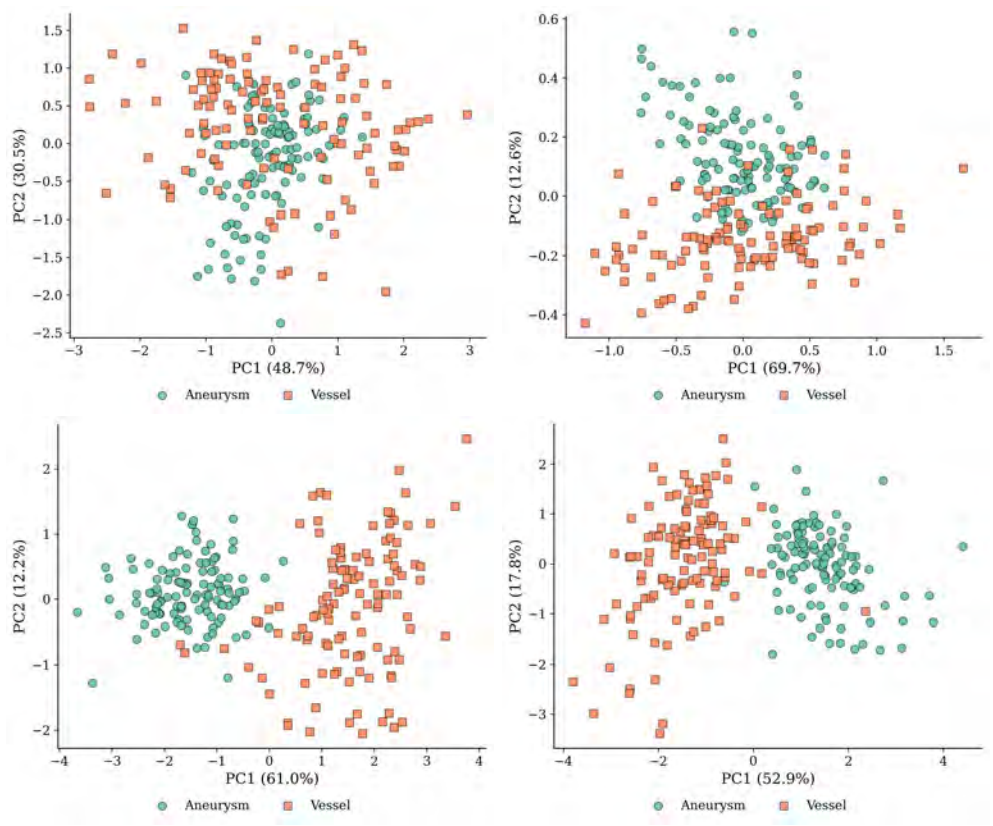


FIG. 13: PCA on annotated aneurysms

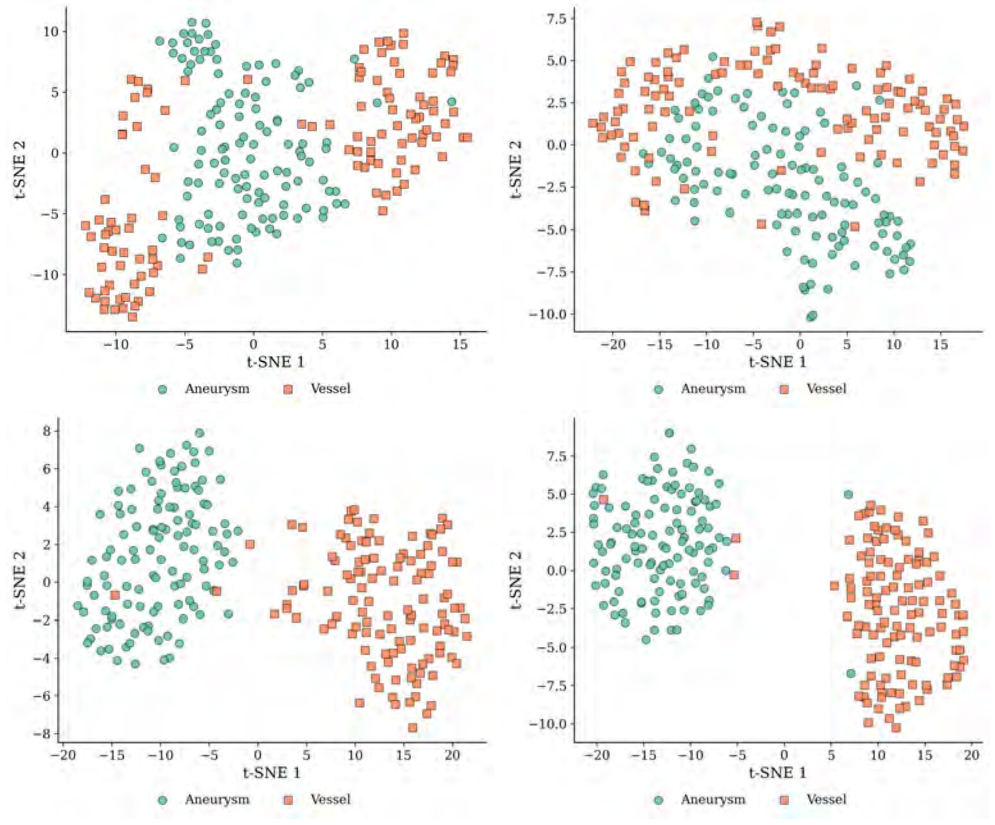


FIG. 14: t-SNE on annotated aneurysms

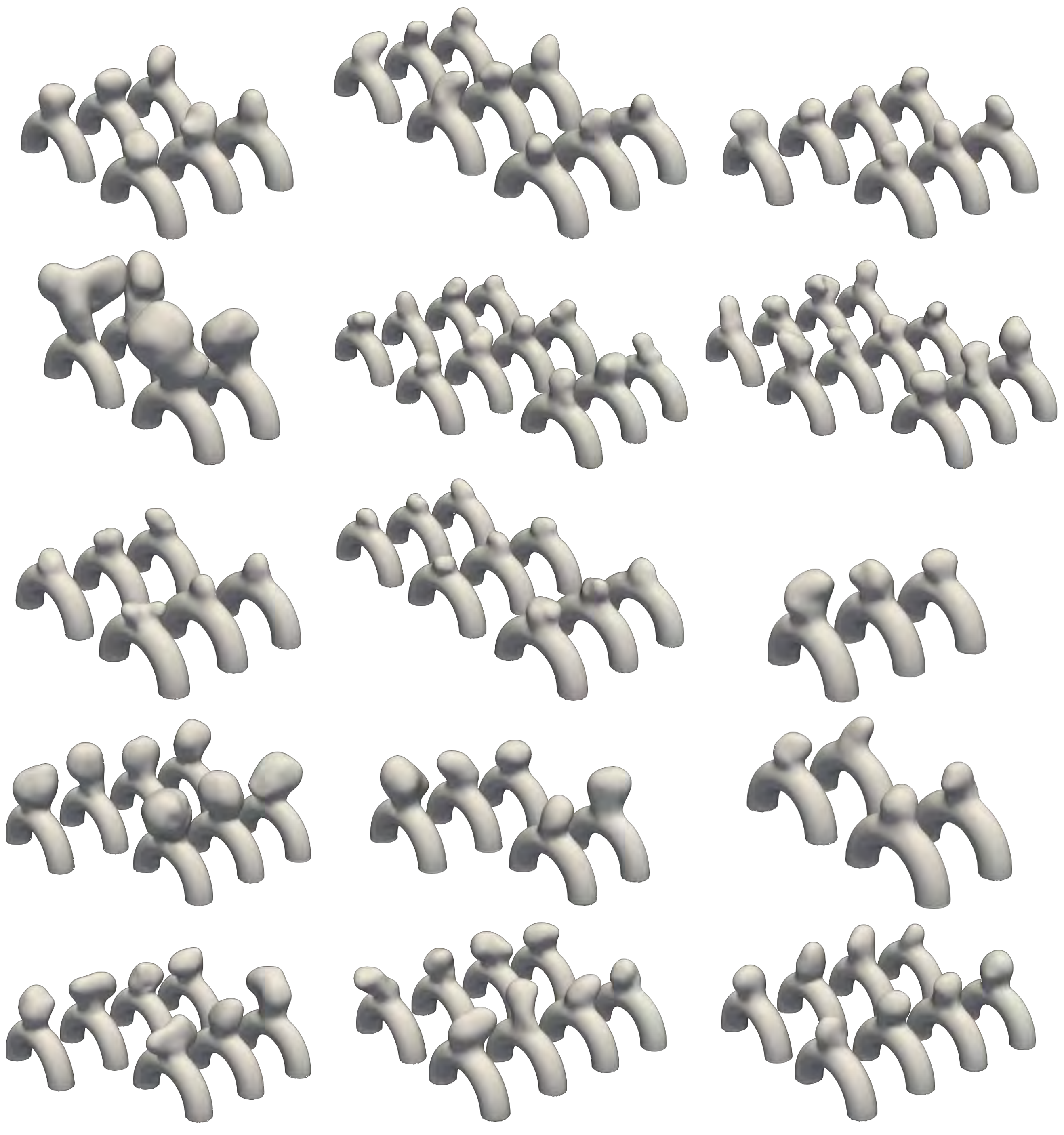


FIG. 15: Clustering results on AnXplore dataset