The Visual Computer (2024) 40:9017–9037
https://doi.org/10.1007/s00371-024-03292-8

ORIGINAL ARTICLE

CurveML: a benchmark for evaluating and training learning-based methods of classification, recognition, and fitting of plane curves

Andrea Raffo[1] · Andrea Ranieri[2] · Chiara Romanengo[2] · Bianca Falcidieno[2] · Silvia Biasotti[2]

Accepted: 22 January 2024 / Published online: 12 March 2024

Abstract

We propose CurveML, a benchmark for evaluating and comparing methods for the classification and identification of plane curves represented as point sets. The dataset is composed of 520*k* curves, of which 280*k* are generated from specific families characterised by distinctive shapes, and 240*k* are obtained from Bézier or composite Bézier curves. The dataset was generated starting from the parametric equations of the selected curves making it easily extensible. It is split into training, validation, and test sets to make it usable by learning-based methods, and it contains curves perturbed with different kinds of point set artefacts. To evaluate the detection of curves in point sets, our benchmark includes various metrics with particular care on what concerns the classification and approximation accuracy. Finally, we provide a comprehensive set of accompanying demonstrations, showcasing curve classification, and parameter regression tasks using both ResNet-based and PointNet-based networks. These demonstrations encompass 14 experiments, with each network type comprising 7 runs: 1 for classification and 6 for regression of the 6 defining parameters of plane curves. The corresponding Jupyter notebooks with training procedures, evaluations, and pre-trained models are also included for a thorough understanding of the methodologies employed.

Keywords Curves · Bézier · Dataset · Machine learning · Classification · Regression · Fitting

1 Introduction

Advances in the development of text and image recognition methods have been aided by the availability of high-quality training datasets and not just algorithmic advances. Today, it is widely recognised that the most important component for building good machine learning models is training data. This practice of systematically building datasets to achieve better model performance is often called data-centric AI. Because of this, various scientific groups have begun prioritising the creation of new datasets and benchmarks to achieve significant results even in problems other than image or text recognition, such as reverse engineering for computer-aided design [12, 24, 25], road safety [11, 17], pose estimation [29], hand gestures recognition [5], partial shape retrieval [27], and the analysis of macromolecules [8, 20, 21], to name a few. Additionally, calls are also periodically launched to encourage the development of new datasets and benchmarks within specialised conferences or workshops, such as TRECVID [1] for videos, MIREX [2] for music, and SHREC [3] for 3D shapes.

In this panorama, there is a lack of adequate datasets and benchmarks of plane curves. Having a benchmark to perform machine learning tasks on plane curves represented as sets of points is essential for ensuring objective evaluation, consistency, fair comparison, and advancement of the computer graphics and numerical analysis fields. This work fills the gap in the literature by introducing *CurveML*, a large

Andrea Raffo, Andrea Ranieri, and Chiara Romanengo contributed equally to this paper.

✉ Chiara Romanengo
chiara.romanengo@ge.imati.cnr.it

Andrea Raffo
andrea.raffo@ibv.uio.no

Andrea Ranieri
andrea.ranieri@cnr.it

Bianca Falcidieno
bianca@ge.imati.cnr.it

Silvia Biasotti
silvia@ge.imati.cnr.it

[1] Department of Biosciences, University of Oslo, Postboks 1066, Blindern 0316, Oslo, Norway

[2] Istituto di Matematica Applicata e Tecnologie Informatiche "E. Magenes", Consiglio Nazionale delle Ricerche, via De Marini 16, Genova 16149, Italy

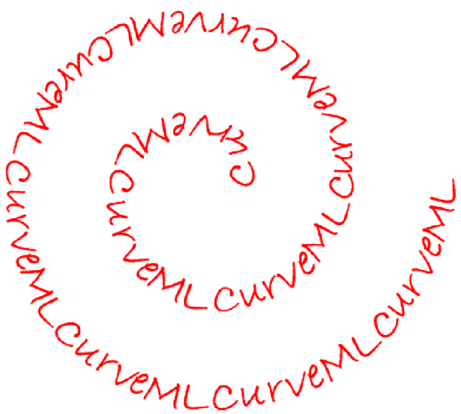


Fig. 1 Logo of the CurveML benchmark, available at https://gitlab.com/4ndr3aR/CurveML

benchmark to evaluate methods for classifying and identifying plane curves on point sets. Figure 1 shows the CurveML logo and its GitHub repository.

CurveML aims at providing the ground to assess if a method outperforms another one in tasks related to curve classification, (parameter) regression and fitting. Here, we adopt the following terminology: *classification* is the process of assigning labels (i.e. the types of curves) to input point sets; *parameter regression*, in this paper sometimes called just *recognition*, deals with the identification and estimation of the parameters that uniquely generated a given point set in terms of some parametric representations; *fitting* is the task of finding the best approximation of the input point set for some criteria. In this respect, it is clear that a good approximation does not always correspond to parameters that are close to ground truth values, as it could happen, for example, if the considered model is overfitting the data.

Additionally, our benchmark guarantees consistency in the evaluation process by providing a standard set of criteria for measuring the performance of different methods. This helps eliminate bias and ensures that the results are reliable and reproducible. The aim is also to yield a fair comparison of different methods by ensuring they are evaluated using the same dataset and evaluation metrics, thus avoiding situations where one method appears to perform better simply because it was tested on a simpler dataset or with different evaluation metrics. Finally, but certainly not least, we intend to provide researchers with tools with which they can evaluate their approaches and identify areas for improvement, thus supporting progress in the fields of computational geometry and machine learning by promoting the development of new and better curve classification and identification methods.

1.1 Related datasets

There are very few datasets of plane curves targeted to support learning-based tasks. We can cite LengthNet [18], a dataset of 500k point sets created for learning the length of planar sampled curves. These curves were created by considering four standard geometric shapes, such as circles, straight lines, triangles, and rectangles, which are scaled, rotated, translated, and randomly segmented. In some datasets, the curve is interpreted as the boundary of basic geometric shapes like FlatShapeNet [22], which includes eight data classes, each representing a type of shape (circle, kite, parallelogram, rectangle, rhombus, square, trapezoid, triangle). FlatShapeNet is a dataset composed of 20k images generated for an educational game. Comparable to FlatShapeNet is the 2D geometric shape dataset [13], which consists of nine data classes (triangle, square, pentagon, hexagon, heptagon, octagon, nonagon, circle, and star). Each class is composed of 10k images and it is generated to train machine learning models to perform geometric shape recognition tasks. Instead, no dataset complete with parametric equations and ground truth has been proposed for plane curves. This motivation led us to develop our CurveML dataset, which, in terms of breadth, diversity, and richness of ground truth, takes inspiration from the ABC dataset [12], a big CAD model dataset for geometric deep learning.

1.2 Contribution

In CurveML, we have selected some families of curves characterised by a representative shape in an atlas of plane curves, see [26], and we enriched this dictionary including Bézier curve and composite Bézier curves, made of 2 or 3 pieces. The dataset contains 520k point sets, generated from the parametric representations of curves and then perturbed, considering different types of artefacts. The way these objects were created makes the benchmark easily expandable since we can select any other curve of interest from the atlas [26] and insert it into our dataset.

CurveML is designed to be also used by machine learning methods: in fact, the dataset comes in the form of a training, a validation, and a test set. A ground truth file is associated with each point set in the training set. It contains information depending on the curve type: geometric parameters for plane curves and control points for Bézier curves.

The benchmark is equipped with some performance measures able to evaluate the quality of algorithms or machine learning models in performing the classification, regression, and fitting of the different types of curves in the dataset.

CurveML has been tested on classification and regression tasks. Indeed, we offer a collection of corresponding examples that illustrate curve classification and parameter regression problems with two popular deep learning models, namely ResNet [9] and PointNet [19]. In addition, we release a set of Jupyter notebooks and baseline pre-trained models to provide a starting point for researchers interested in using

the dataset and to show the quality of the ML/DL methods obtainable from training on it.

1.3 Outline

The rest of the paper is organised as follows. In Section 2, we introduce the pipeline necessary to generate point sets in our dataset, describing the type of curves selected to create the data, and we illustrate the information contained in the ground truth files. Section 3 introduces the performance measures chosen to evaluate the quality of the classification, recognition, and approximation of plane curves. Finally, Sect. 4 focuses on a test of the proposed benchmark with a learning-based approach. Closing remarks end the paper.

2 The dataset

The dataset comprises a total of 520k point sets: 280k are generated from eight families of curves found in [26], each of which was chosen because of the distinctive shapes it contains; 240k originate from Bézier or composite Bézier curves.

2.1 Type of curves

In this section, we describe the curves in our dataset in terms of their parametric equations and properties.

Families of curves

We have here selected families of curves that admit parametric representations depending on at most three parameters when given in some canonical forms. The generic parametric equation of any of these families of curves is

$$\mathbf{P}(t) := \begin{cases} x(t) = f(t, \mathbf{a}) \\ y(t) = g(t, \mathbf{a}) \end{cases} \tag{1}$$

where f and g are continuous functions of a common variable $t \in I \subseteq \mathbb{R}$, and $\mathbf{a}$ is a vector containing scalar values that will be hereafter called *geometric parameters*. The families of curves we consider are the *egg of Keplero*, *mouth curve*, *cissoid of Diocle*, *hypocycloid*, *citrus curve*, *Cassinian oval*, *Archimedean spiral* and *geometric petal*, see Table 1. In the following, we detail their main characteristics:

- The Egg of Keplero, mouth curve, and cissoid of Diocle depend on a single parameter a, with $a > 0$. The egg of Keplero is symmetric with respect to the $x-$axis and bounded; in particular, it is contained in the rectangle $[0, a] \times [-\frac{3\sqrt{3}}{16}a, \frac{3\sqrt{3}}{16}a]$. The mouth curve is a symmetric and bounded curve contained in the square $[-a, a] \times [-a, a]$. The cissoid of Diocle is an unbounded curve symmetric with respect to the $y-$axis.
- The hypocycloid is written in terms of the parameters a and n, the latter being the number of cusps. It is a bounded and central-symmetric curve with n axes of symmetry. This family contains two sub-families of well-known curves: the *deltoid* when $n = 3$, and the *astroid* if $n = 4$.
- The Citrus curve, Cassinian oval and Archimedean spiral depend on two parameters a and b. The citrus curve is a symmetric and bounded curve, contained in the rectangle $[-\frac{a}{2}, \frac{a}{2}] \times [-\frac{a}{8b}, \frac{a}{8b}]$. The Cassinian oval is a bounded curve symmetric with respect to the Cartesian axes. We only consider the case where $a \geq b$ and then the curve is connected. It is a rather large family and includes several types of shapes. Specifically, in case $a = b$ the curve coincides with the well-known *lemniscate of Bernoulli*, while in case $a \geq \sqrt{2}b$ it is convex. It has a shape similar to an oval until it reaches a shape similar to that of an ellipse or a circle as the value of a increases. Finally, the Archimedean spiral is connected and not limited, originating from the point $(a, 0)$.
- Geometric petal depends on the parameters a, b and n, where the latter indicates the number of petals. It is a central-symmetric curve contained in a circle with a radius $a + b$, and it has n axes of symmetry. We only consider the case where $a > b$, since there are no self-intersection points.

The parametric equations of these families are given in Table 1, together with an illustrative example per family.

While restricting our attention to a limited number of families, it is worth noting that the dataset is easily extendable by selecting additional families of plane curves, of which the literature is pretty rich: see, for example, [15, 16, 26].

Bézier curves

In applications like computer-aided design and computer graphics, a popular way of modelling a shape is to represent its outer curve as a patchwork of parametric polynomial pieces. One of the first and most popular examples is that of Bézier curves.

To construct a Bézier curve of degree $d \in \mathbb{N}/\{0\}$, we start with an interval $I = [0, 1]$ and $d + 1$ control points $\mathbf{C}_0, \ldots, \mathbf{C}_d$ in $\mathbb{R}^2$. The curve has the equation

$$\mathbf{P}(t) := \sum_{k=0}^{d} \mathbf{C}_k B_k(t) \tag{2}$$

where $B_k(t) := \binom{d}{k} t^k (1-t)^{d-k}$ are the Bernstein polynomials and where $t \in I$.

Table 1 Families of curves considered in this benchmark, together with their parametric representations

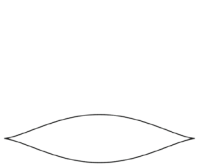
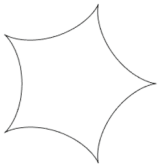
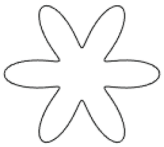
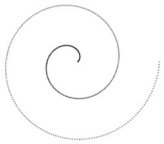
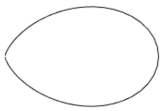
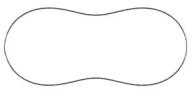
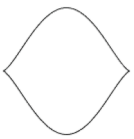
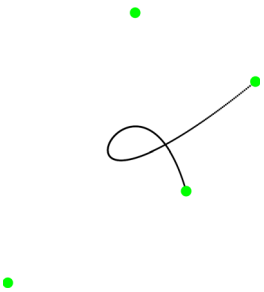
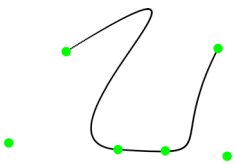
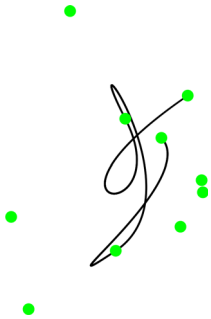
citrus curve	hypocycloid	geometric petal	Archimedean spiral
			
$\mathbf{P}(t) := \begin{cases} x = t - \frac{a}{2} \\ y = \pm\sqrt{\frac{(a-t)^3 t^3}{a^4 b^2}} \end{cases}$	$\mathbf{P}(t) := \begin{cases} x = (a - \frac{a}{n})\cos t + \frac{a}{n}\cos(n-1)t \\ y = (a - \frac{a}{n})\sin t + \frac{a}{n}\sin(n-1)t \end{cases}$	$\mathbf{P}(t) := \begin{cases} x = (a + b\cos nt)\cos t \\ y = (a + b\cos nt)\sin t \end{cases}$	$\mathbf{P}(t) := \begin{cases} x = (a+b)\cos t + \frac{a}{n}\cos t \\ y = (a+b)\cos t + \frac{a}{n}\sin t \end{cases}$
egg of Keplero	Cassinian oval	mouth curve	cissoid of Diocle
			
$\mathbf{P}(t) := \begin{cases} x = \frac{a}{(1+t^2)^2} - \frac{a}{2} \\ y = \frac{at}{(1+t^2)^2} \end{cases}$	$\mathbf{P}(t) := \begin{cases} x = t \\ y = \pm\sqrt{\sqrt{4b^2t^2 + a^4} - t^2 - b^2} \end{cases}$	$\mathbf{P}(t) : \begin{cases} x = a\cos t \\ y = a\sin^3 t \end{cases}$	$\mathbf{P}(t) : \begin{cases} x = 2a(\tan t - \frac{1}{2}\sin 2t) \\ y = 2a\sin^2 t \end{cases}$

Table 2 Examples of Bézier and composite Bézier curves (2 or 3 pieces) of degree 3 considered in this benchmark. Control points are shown in green

Bézier – 1 piece	Bézier – 2 pieces	Bézier – 3 pieces
		

Multiple pieces (i.e. Bézier curves) have to be glued together with at least C^0 continuity to represent complex shapes. Composite Bézier curves have equations

$$\mathbf{P}^{(\mu)}(t) = \sum_{k=0}^{d} \mathbf{C}_k^{(\mu)} B_k(t), \tag{3}$$

for $\mu = 1, \ldots, N$, being N the number of pieces. We remind the reader that, to ensure that the Bézier curves $\mathbf{P}^{(\mu)}$ and $\mathbf{P}^{(\mu+1)}$ join continuously at one end, the last control point of one curve must coincide with the first control point of the other one, i.e. $\mathbf{C}_d^{(\mu)} = \mathbf{C}_0^{(\mu+1)}$. To have G^1 continuity, it is further required that

$$\mathbf{C}_{d-1}^{(\mu)}, \mathbf{C}_d^{(\mu)} = \mathbf{C}_0^{(\mu+1)}, \mathbf{C}_1^{(\mu+1)}$$

are collinear.

We consider Bézier curves and composite Bézier curves made of 2 or 3 pieces while leaving the user the possibility of increasing their number. Examples of curves and their control points are shown in Table 2.

2.2 Data generation process—curve sampling

The point sets are obtained from the parametric representations considered in this benchmark using the following standard procedures.

Families of curves

For a given family of curves, point sets are sampled by using its parametric representation. As mentioned in Sec-

tion 2.1, parametric representations are in their canonical form, i.e. centred at the origin of the coordinate axes or/and with axes of symmetry coinciding with the x- and y-axes, depending on the geometric characteristics of curves. The geometric parameters that define each equation are assigned by randomly sampling uniform distributions. Subsequently, we randomly apply translations and/or rotations to put the sampled curve in a general position.

Bézier curves

We sample Bézier curves and composite Bézier curves via the well-known De Casteljau algorithm (see, for example, [7]).

Control points are here obtained by randomly sampling a bidimensional uniform distribution over the region $[-1, 1] \times [-1, 1]$. Points over a Bézier curve are initially computed by evaluating it at evenly spaced parameter values between 0 and 1. C^0 continuity between a pair of pieces $\mathbf{P}^{(\mu)}$ and $\mathbf{P}^{(\mu+1)}$ is imposed by setting $\mathbf{C}_d^{(\mu)} = \mathbf{C}_0^{(\mu+1)}$. To ensure G^1 continuity between the end point of $\mathbf{P}^{(\mu)}$ and the starting point of $\mathbf{P}^{(\mu+1)}$, we project $\mathbf{C}_1^{(\mu+1)}$ orthogonally to the line defined by the control points $\mathbf{C}_{d-1}^{(\mu)}$ and $\mathbf{C}_d^{(\mu)}$.

2.3 Data generation process—point set artefacts

Finally, some data perturbations can be applied to each point set. Specifically, the following cases are considered:

- *A0—Clean.* No perturbation is applied.
- *A1—Uniform or non-uniform downsampling.* In the case of uniform downsampling, we keep one point every m of the initial sampling of the curve, where m is randomly chosen among the values 2, 4, and 10. In the case of non-uniform downsampling, each point of the curve is associated with a uniformly sampled random number $p_i \in [0, 1]$; point $\mathbf{P}(t_i)$ is discarded if

$$p_i > 1 - \frac{0.8}{N_{\text{points}}} i,$$

  being N_{points} the number of points in the point sets and $t_i \leq t_{i+1}$ for all i.
- *A2—Global or local uniform noise.* The noise is obtained by sampling uniform distributions of the form $\mathcal{U}(-\frac{1}{20}, \frac{1}{20})$. It is applied to a fixed percentage of the points, chosen within the entire point set or only in a selected portion of it.
- *A3—Global or local Gaussian noise.* The noise is obtained by sampling a normal distribution with mean 0 and standard deviation 0.02, and applying such values to a fixed percentage of the points—chosen w.r.t. the entire point set or to a selected portion only.
- *A4—Global or local uniform outliers.* Outliers are introduced by moving 15% of the points—locally or globally—by a uniformly-sampled contribution between 0.09 and 0.12 in each coordinate direction. These values were chosen considering the used bounding box, as they correspond to $4.5 - 6\%$ of its size.
- *A5—Combination of artefacts.* The perturbations provided above are randomly combined to obtain more complex scenarios.

Figure 2 shows examples of data perturbations applied to the point sets for each curve type and provides an overall view of the dataset's content.

Each point set is provided in a CSV file named `point_set_perturbed.csv`, listing the triplets of coordinates, one point per line. A set of additional information is associated with each point set. Specifically, we provide the original clean point set in a CSV file `point_set_clean.csv`; the images 500×500 pixels generated from the two point sets as a pair of PNG files; and a CSV file `details_reproducibility.csv`, where we report the information about the type of artefact applied.

2.4 Ground truth

A ground truth file is associated with each point set, with information that depends on the curve type.

Families of curves

In the case of a family of curves, the information necessary to correctly recognise a point set is the rotation angle α and the translations TrX and TrY along the $x-$ and $y-$axis with respect to the canonical form of the selected family, the values of the geometrical parameters $\bar{a}$ and $\bar{b}$, and the numbers $\bar{n}$ of petals or cups. For this reason, the GT file `parameters.csv` contains a 2×6 cells structure. The first row lists the parameter labels $[angle, trans\_x, trans\_y, a, b, n]$, while the second row contains the corresponding values $[\alpha, TrX, TrY, \bar{a}, \bar{b}, \bar{n}]$.

Bézier curves

In the case of Bézier curves, the GT file `CPs.csv` contains the control points, one per row.

2.5 Dataset representativeness

For the first version of CurveML, we focused on curves that are non-trivial and not readily available in other popular datasets such as [4], [13], [22]. Hence, we omitted simpler shapes such as circles, stars, squares, triangles, and pentagons, and instead we selected families of curves that

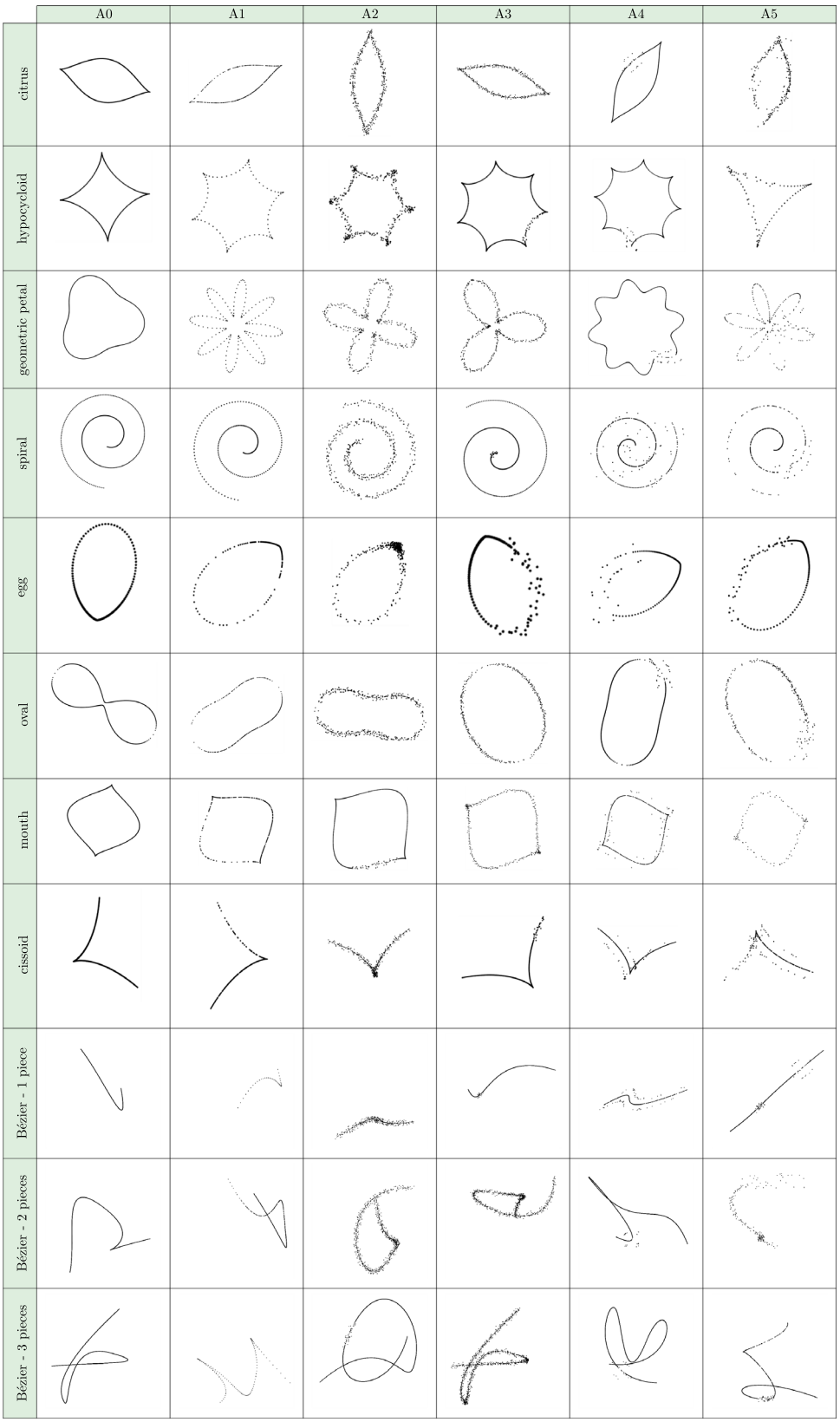


Fig. 2 Random point sets from the data set under each category. Columns identify the artefacts types: none (A0), uniform or non-uniform downsampling (A1), global or local uniform noise (A2), global or local Gaussian noise (A3), global or local uniform outliers (A4), a combination of different artefacts (A5). Rows correspond to the types of curves listed in Section 2.1

Table 3 Overview of the properties for the families of curves

Curve type	Open	Closed	Bounded	Unbounded	Piecewise
citrus	–	✓	✓	–	–
hypocycloid	–	✓	✓	–	–
geom. petal	–	✓	✓	–	–
Arch. spiral	✓	–	–	✓	–
egg	–	✓	✓	–	–
oval	–	✓	✓	–	–
mouth	–	✓	✓	–	–
cissoid	✓	–	–	✓	–
Bézier	✓	–	✓	–	✓

mathematically encode a wide range of variability in terms of unique patterns and distinctive geometric characteristics. Our goal was to ensure that the dataset is representative of various desirable features, including closed, open, bounded, and unbounded shapes—see Table 3. Additionally, we also accounted for data perturbation by including point sets that exhibit different types of downsampling (uniform or non-uniform), noise (local or global, Gaussian or uniform), and outliers (local or global), possibly combined into more complex artefacts. By including these artefacts, we aimed to provide a more realistic and challenging scenario.

3 Evaluation measures

Together with the dataset, we also release various quality measures from [6, 14] that permit the quantitative evaluation of methods and algorithms for curve classification, parameter recognition, and the approximation of plane curves.

3.1 Classification measures

To illustrate the classification performance of a method, we consider the so-called *confusion matrix* (CM) [14]. It is a square matrix that has the same order as the number of classes present in the dataset. The elements located on the CM diagonal represent the number of true positives, i.e. the number of items that were accurately classified as belonging to their corresponding ground truth classes. Off-diagonal elements indicate the count of items incorrectly classified by the model. Said otherwise, $\mathrm{CM}(i, i)$ is the number of elements correctly identified as members of class i, while $\mathrm{CM}(i, j)$, with $j \neq i$, is the count of elements wrongly labelled as belonging to class j rather than to class i. It goes without saying that ideal classifiers have diagonal classification matrices.

True Positive and Negative Rates.

The true-positive rate (TPR), also known as recall, assesses the capability of a model to accurately detect positive instances, such as the percentage of class i elements correctly identified as class i elements. Similarly, the true-negative rate (TNR) evaluates the model's ability to recognise negative instances, such as the percentage of class j elements (with $j \neq i$) correctly classified as such. These metrics are also referred to as sensitivity and specificity in statistical analysis.

Positive and negative predictive values.

In addition to TPR and TNR, other metrics can help us determine the probability that a model will return a correct prediction. These metrics are the positive predictive value (PPV) and the negative predictive value (NPV). PPV, also called precision, is the proportion of true positives to the total number of items classified as positives by the model; likewise, NPV represents the ratio of true negatives to the total number of items classified as negatives by the model.

Accuracy. Accuracy, here denoted by ACC, refers to the frequency at which a model makes accurate predictions. Specifically, it is calculated as the ratio of correct predictions the model makes to the total number of predictions.

F1-Score F1-Score is a metric that combines both precision (i.e. PPV) and recall (i.e. TPR) into a single value. It is the harmonic mean of precision and recall, which gives equal weight to both metrics. The F1-Score ranges from 0 to 1, with higher values indicating better performance.

3.2 Parameter recognition measures

To analyse the estimations made on point sets representing families of curves and Bézier curves, we exploit the information contained in the `parameters.csv` and `CPs.csv` files. We can do two types of analyses: a global one, by considering all predictions, and a local one, by examining predictions corresponding to only one type of family at a time.

Let N be the number of files in the test set. Given a specific parameter under study (angle, translation along a principal direction, control point along a principal direction, etc.), we define the vector $\boldsymbol{\gamma}$ as the float vector $1 \times N$ that contains the ground truth values (for files where that parameter is meaningful) or zero (otherwise).

We denote by γ_i the exact parameter value for the i-th curve in the test set; similarly, $\hat{\gamma}_i$ is the parameter estimate for curve number i. To assess the quality of recognition, we chose the following measures that apply to all parameters:

– *Mean Square Error* (MSE)

$$\mathrm{MSE} := \frac{1}{N}\sum_{i=1}^{N}(\gamma_i - \hat{\gamma}_i)^2.$$

– *Mean Absolute Error* (MAE)

$$\text{MAE} := \frac{1}{N}\sum_{i=1}^{N}|\gamma_i - \hat{\gamma}_i|.$$

– *Pearson's correlation coefficient* ρ

$$\rho := \frac{cov(\boldsymbol{\gamma}, \hat{\boldsymbol{\gamma}})}{\sigma(\boldsymbol{\gamma})\sigma(\hat{\boldsymbol{\gamma}})},$$

i.e. the covariance of the vectors $(\boldsymbol{\gamma}, \hat{\boldsymbol{\gamma}})$ of the estimated and exact parameters divided by the product of their standard deviations. The result always has a value in $[-1, 1]$.

– *Spearman's rank correlation coefficient* is defined as the Pearson's correlation coefficient between the rank values of $\boldsymbol{\gamma}$ and $\hat{\boldsymbol{\gamma}}$.

– *Coefficient of determination* R^2 *score* is the proportion of the variation in the dependent variable (here the unknown parameters that we aim at estimating) that is predictable by the model.

3.3 Approximation measures

To quantify the approximation accuracy of a specific curve, we first generate a dense set of points $\mathcal{C}_{clean}$ that represents the original curve, exploiting the parameter in the file `parameters.csv` or the control points in the file `CP.csv` and the parametric representation of the curve in Eq. 1 or 2-3, respectively. Then, we consider the parameters or the control points estimated by an algorithm and, in a similar way, we can obtain a sampling $\mathcal{C}$ of the corresponding curve exploiting the parametric representations in Eq. 1 or 2-3, respectively. Based on the point set representation of the ground truth curve and the inferred one, we propose the following two measures to evaluate the approximation accuracy of the curve $\mathcal{C}$:

– *Mean Fitting Error* (MFE), defined as

$$\text{MFE}(\mathcal{C}, \mathcal{C}_{clean}) := \frac{1}{|\mathcal{C}|}\sum_{\mathbf{x}\in\mathcal{C}} d_2(\mathbf{x}, \mathcal{C}_{clean})/l,$$

where d_2 is the Euclidean distance, and l is the diagonal of the axis-aligned minimum bounding box containing $\mathcal{C}$, which makes the measure independent from the curve size;

– *Directed Hausdorff distance*, defined as

$$d_{\text{dHaus}}(\mathcal{C}, \mathcal{C}_{clean}) := \max_{\mathbf{x}\in\mathcal{C}} \min_{\mathbf{y}\in\mathcal{C}_{clean}} \|\mathbf{x} - \mathbf{y}\|_2.$$

To make the measure independent from the curve size, we normalise it with respect to the diagonal l of the minimum bounding box containing $\mathcal{C}$.

4 A practical use case: applying neural networks to curve classification and regression problems

As a driving example of how the dataset can be used in machine learning tasks we train and evaluate a set of deep learning models, see Section 4.1. Models are used to classify the families of curves and to perform simple single-value regression tasks on their parameters, see Section 4.2.

We chose the univariate regression approach because our main interest is to provide robust baselines on which anybody could then further work and improve, and to demonstrate that the dataset is learnable "as-is" by deep learning models without any special cleaning techniques of the input or output data.

Moreover, had we chosen the multivariate regression approach, the neural network architecture should have had multiple regression heads in output to the encoder, and this would have significantly increased the amount of VRAM needed and the computation budget (already sufficiently high given the number of images).

4.1 Description of the learning-based methods

To evaluate the dataset's quality, we employed CurveML to train models belonging to two distinct families of architectures: ResNet [9] and PointNet [19]. We first present a set of convolutional neural network (CNN) models trained to perform classification or image regression tasks using the ResNet architecture. We then demonstrate how CurveML can be effectively learned by a family of PointNet-based models to accomplish the same tasks.

4.1.1 Training of models based on the ResNet architecture

The first deep learning (DL) method presented in this section exploits a standard "frozen" ResNet-101 [9] backbone, therefore with untrainable weights on the convolutional layers, and only performs the fine-tuning of a standard classification or regression head, according to the task involved. By allowing just over 2 million out of 44 million parameters to be trained, this transfer-learning approach saves computational power to train only what is beneficial to the problem.

The training took place solely on the geometric images ($280k$ samples, in two versions, with and without noise) for a total of $560k$ images divided into training/validation/test sets with a ratio of $0.8/0.1/0.1$[1]. Before feeding the neural network, the images were scaled to $250px$. No other type of data augmentation was used. As a result, it was possible to

[1] Although the split was sequential for the number of directories, due to the random extraction process of the curve parameters, the dataset has been effectively split randomly over the three sets.

obtain robust baselines that are highly replicable and more comparable between the different training runs, also with the expectation that these baselines can be used in the future for training or validating more complex models.

The training took place on two machines: the first with a single Nvidia RTX A5000 GPU with 24 Gb of VRAM, the second with a single Nvidia RTX 2070 Super GPU with 8 Gb of VRAM using the popular `Fast.ai` [10] and `Pytorch` libraries. `Fast.ai`'s convenient APIs allow to download the pre-trained backbone and weights from the `Torchvision`'s model `zoo` very simply and automatically. Also, the `Fast.ai` library automatically initialises the new head with random weights, generating the output layer of the neural network according to the type of data passed to its `DataLoaders`: one single neuron with a `SigmoidRange` activation function in the case of a regression problem, *n_classes* neurons—with *n_classes* equal to the number of classes—with a `Softmax` activation function in the case of a classification problem.

To maximise the level of automation during the training of the network, a set of `Fast.ai` callbacks was used to perform the early stopping of the training (with patience = 5, i.e. 5 epochs without improving the validation loss of the network for the regression problems and patience = 2 for the classification problem) and to save the best model of the training round and then reload it for validation and testing. We also used the `Weights & Biases` (wandb.ai) report callback to constantly monitor the different training runs remotely via browser.

The loss function used was MSE (mean square error) for the regression problems and cross-entropy loss for the classification problem. We used $bs = 32$, and the learning rates were set automatically using `Fast.ai`'s function `lr_find` to $slice(5e-4, 1e-3)$.

We used a wide set of metrics described in Section 3.2 to assess the learning progress of the model. All training code and pre-trained models are available here for download: https://gitlab.com/4ndr3aR/CurveML.

4.1.2 Training of models based on the PointNet architecture

To train the PointNet models [19], we chose a Pytorch-based implementation available on GitHub [28]. Just around 3.5 million trainable parameters make up the architecture implemented in the repository, as described in the original PointNet paper [19]. To indicate the scale of the difference, PointNet has an order of magnitude less trainable parameters and, hence, a lower computational cost for training than the ResNet-101, which we selected for image-based training and has slightly under 45 million trainable parameters.

To further speed up the training process, we loaded into a `Pandas DataFrame` all the 560.000 point sets (in their x, y, z form) and all the labels and parameters normally stored in the dataset as single files. We subsequently saved it in compressed `LZMA2` format (`.xz` extension). This allows the entire point-based dataset (without images) to be stored in less than 2 Gb of space and subsequently loaded entirely in RAM to maximise training speed.

As for the ResNet-based models, also for PointNet we execute 7 distinct experiments, one for classification and 6 for the regression of the 6 different parameters of the plane curves. In all experiments, the network backbone remains the same, only the output layer and the activation function change. For classification, 8 neurons are created and instantiated, one for each of the 8 classes of plane curves currently present in CurveML, followed by a `ReLU` (*Rectified Linear Unit*) type activation function and a `log_softmax` operation. For the regression, a single neuron is created, followed by a `sigmoid_range` function (a *sigmoid* rescaled between `limit_low` and `limit_high`).

As for the loss functions, for classification we left the original Negative Log Likelihood loss (`nll_loss`) of the Pytorch-based PointNet implementation, while for regression we chose the Mean Squared Error loss (`MSELoss`)[2].

The training of this class of models used the same split adopted for the training of the ResNet-101 models, choosing `Adam` as the optimiser, setting the learning rate to 1e-3, the weight decay to 1e-4, and training for 200 epochs. The checkpoint with the lowest validation loss (or with the highest validation accuracy for the classification problem) was selected as the best model for evaluation on the test set. All training runs were performed on a single Nvidia RTX 2070 Super GPU with 8 Gb of VRAM and took approximately 12 hours to complete.

4.2 Performance analysis

The classification models show near-perfect image classification performance across families of curves for both the ResNet-101 classifier and the PointNet classifier.

As visible in Table 4 and 5, both the models are close to 100% accuracy on both the validation and the test sets. The confusion matrices in Figs. 3 and 4 show the classification capabilities of both models in absolute value: on the validation set, the ResNet-101 model misclassifies a total of 4 samples out of 56k, while on the test set 6 samples are misclassified (again out of a total of 56k).

The PointNet-based classifier performs very well but comparatively slightly worse than the ResNet-based classifier. As shown in Fig. 4, the model fails to classify 17 samples out of 56k both on the validation and test set. Figure 5 shows the top losses for the ResNet-based classifier, i.e. the "most

[2] It has to be noted that the regression implementation was not originally present in the *PointNet/PointNet++ Pytorch* repository, so we just replaced the classification head with a single-neuron regression head.

Table 4 Performance of the ResNet classification model on the validation and test sets

Data set	Loss	TPR	TNR	PPV	NPV	ACC	F1
validation	2e-4	0.999	0.999	0.999	0.999	0.999	0.999
test	3e-4	0.999	0.999	0.999	0.999	0.999	0.999

Table 5 Performance of the PointNet classification model on the validation and test sets

Data set	Loss	TPR	TNR	PPV	NPV	ACC	F1
validation	1e-3	0.999	0.999	0.999	0.999	0.999	0.999
test	1e-3	0.999	0.999	0.999	0.999	0.999	0.999

misclassified" samples, for validation and test sets, respectively. As can be seen, the few misclassified curves are heavily undersampled and with a low signal-to-noise ratio, probably even for a trained human eye. Practically the same goes for the PointNet-based classifier: the worst classified curves are almost all barely distinguishable (Fig. 6).

For what concerns the ResNet-based model, the loss curves for training and validation (Fig. 7) on the classifi-

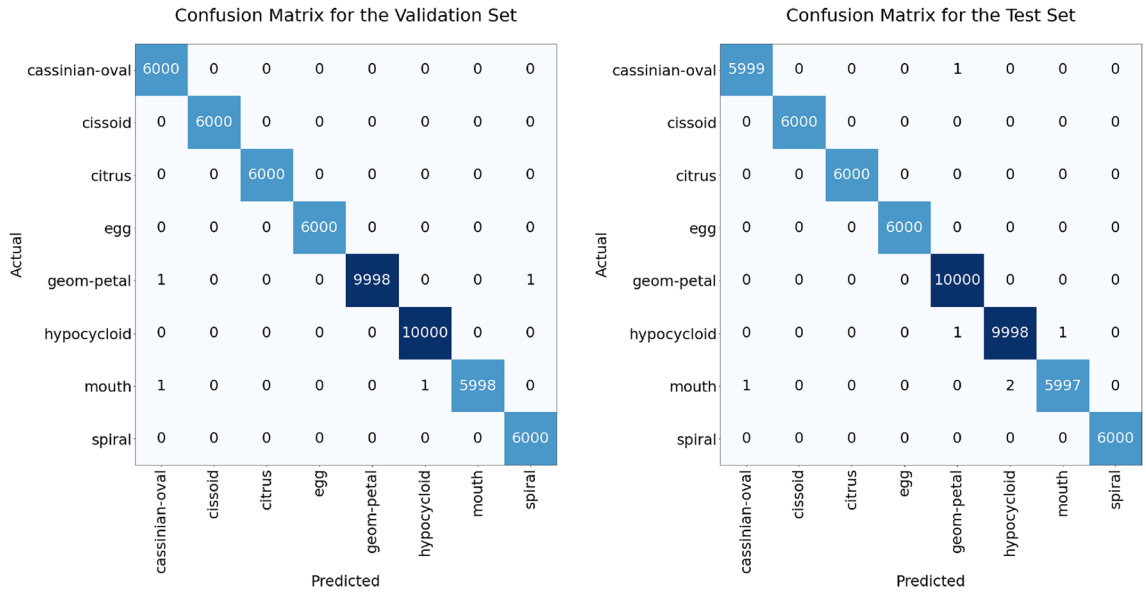


Fig. 3 Confusion matrices for the ResNet classifier on the validation set (left) and the test set (right)

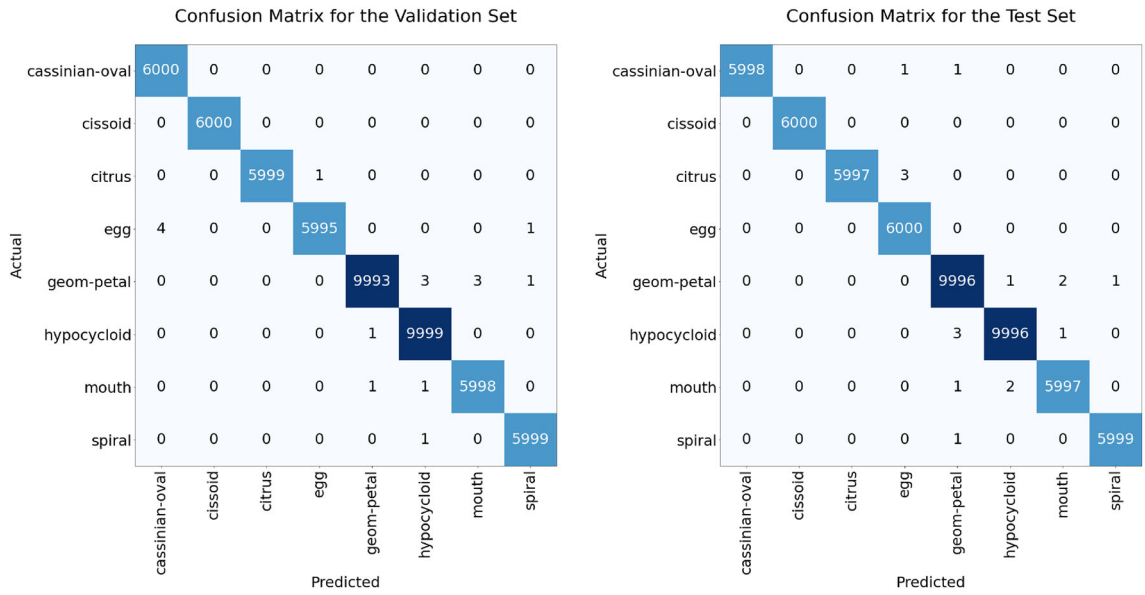


Fig. 4 Confusion matrices for the PointNet classifier on the validation set (left) and the test set (right)

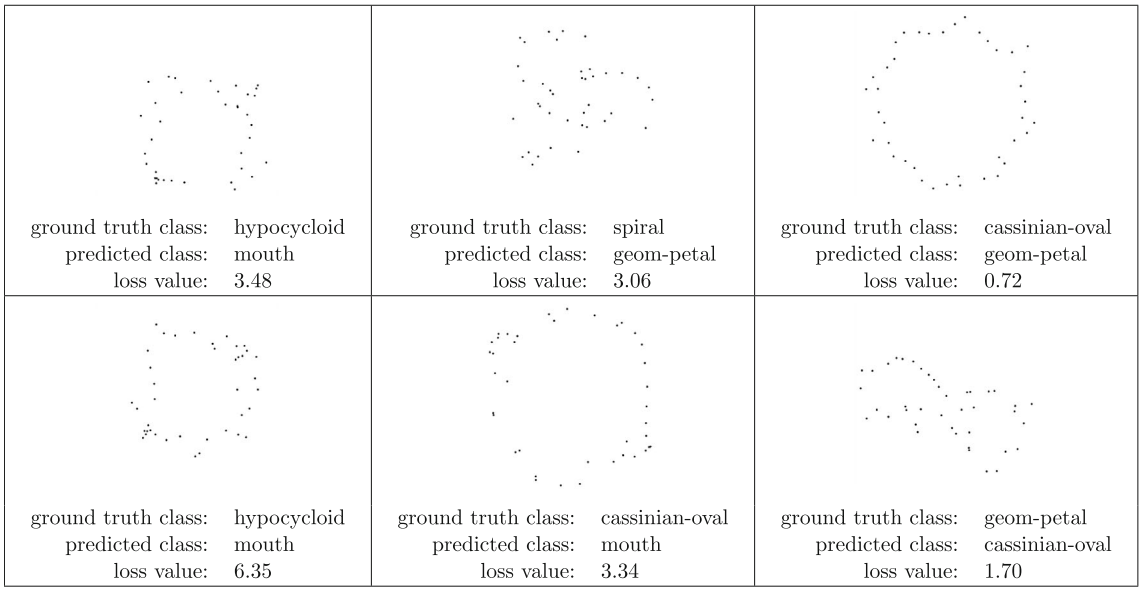


Fig. 5 Top losses (i.e. worst classified samples) for the ResNet-101 classification model, including loss values for each sample. The upper row shows the top losses in the validation set, and the bottom row shows the top losses in the test set

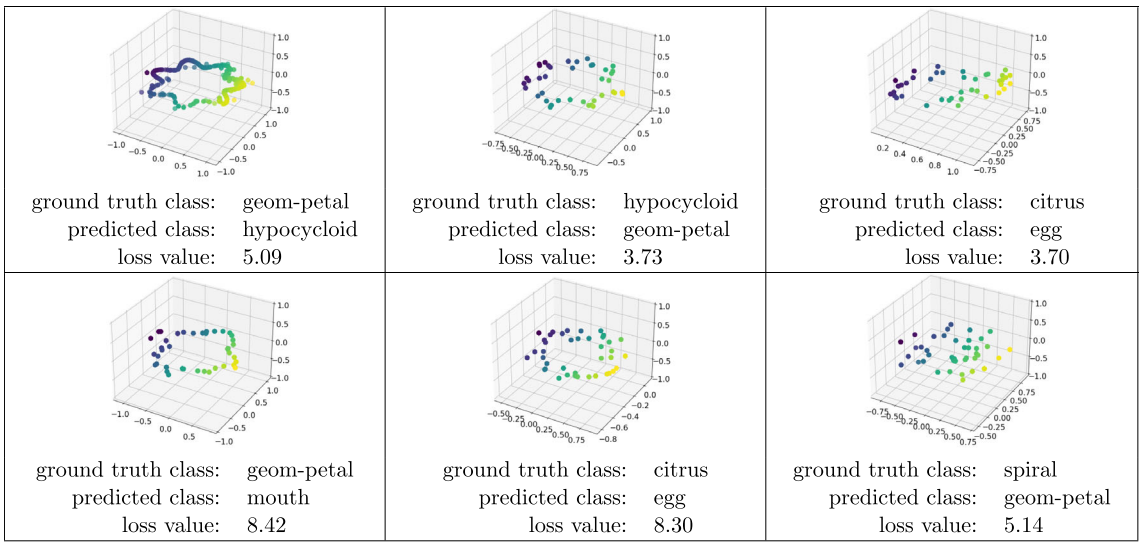


Fig. 6 Top losses (i.e. worst classified samples) for the PointNet classification model, including loss values for each sample. The upper row shows the top losses in the validation set, and the bottom row shows the top losses in the test set

cation problem (label: **Classification Training/Validation Loss**) show very well the near-zero cumulative loss values achieved by the model.

As shown in Table 6, in all the regression experiments, the training converges with a high correlation coefficient between predictions and ground truth ($r > 0.9$). However, convergence times vary significantly from experiment to experiment, ranging from only 8 epochs for the a parameter (label: **"a" Training/Validation Loss** in Fig. 7) up to 23 epochs for the α parameter (label: **"angle" Training/Validation Loss** in Fig. 7).

Regarding the PointNet training runs, Fig. 8 shows the accuracy of the classification model as the number of training epochs increases. In this case, it can be seen that around 50 epochs, the accuracy[3] is already very close to 100% (label:Training/Validation Acc.).

Figure 9 instead shows the variation of training and validation losses for the different PointNet regression models as the number of epochs increases. Also in this case it can be

[3] The training script of PointNet's Pytorch implementation that was used to train the classification model only provided accuracy but not cumulative loss.

Table 6 Performance of the ResNet regression models on the validation and test sets. The loss represents the MSE (Mean Square Error) between ground truth and predictions

Data set	Problem	Validation Loss	MAE	RMSE	R^2 Score	Pearson Corr. Coeff.	Spearman Coeff.
Validation	TrX	0.000	0.010	0.013	0.994	0.998	0.998
	TrY	0.000	0.010	0.013	0.994	0.997	0.996
	a	0.002	0.030	0.042	0.994	0.997	0.996
	b	0.001	0.029	0.033	0.980	0.997	0.935
	n	0.022	0.079	0.149	0.997	0.999	0.856
	α	127.965	3.865	11.312	0.984	0.992	0.986
Test	TrX	0.000	0.010	0.013	0.994	0.998	0.998
	TrY	0.000	0.010	0.013	0.994	0.997	0.996
	a	0.002	0.030	0.042	0.994	0.997	0.996
	b	0.001	0.029	0.033	0.980	0.997	0.935
	n	0.025	0.079	0.157	0.997	0.999	0.856
	α	144.798	3.921	12.033	0.982	0.991	0.986

seen how the loss on the α parameter (label: **"angle" Training/Validation Loss**) is orders of magnitude higher than that of the other parameters. On the contrary, the loss concerning the parameter "n" (label: **"n"Training/Validation Loss**) converges very quickly to zero, since "n" is a discrete parameter and therefore carries very little information[4].

Again in Fig. 9, in the last two graphs, it can be seen that the models of the parameters "a" and "b" show a convergence that can be defined as "normal", while the models relating to the parameters "x" and "y" are very interesting, since their validation loss is consistently lower than their training loss and around the 75th epoch it even becomes significantly lower. It is noteworthy that the loss—in both training and validation—is considerably higher for these two parameters compared to the others.

Also for the case of PointNet, Table 7 shows the complete set of metrics obtained on validation and test sets. The only parameter to visibly suffer from the use of PointNet seems to be α. Not only are there no other glaring differences to be noted, but rather the similarity of the statistics that two radically different approaches—which employ two neural networks with a radically different "model capacity"—have on the same dataset, is striking.

Finally, it is useful to keep in mind that the current version of the training scripts does not exploit symmetries, and the angle is severely penalised by this lack of information. In fact, small variations in prediction vs. ground truth around the symmetry angle correspond to high losses which hurt the training to some extent. Unfortunately, this limitation is not solvable without also using the information about the class of the figure during training, given that each class of figure has its symmetry which can be different from the others.

Further evaluation is needed to understand how to exploit these symmetries best. Without further experimentation, it is not possible to establish a priori whether it is sufficient to train a "global model" capable of estimating class and all parameters simultaneously, or whether it is necessary (and more convenient) to have a small classification model upstream of the pipeline that passes its classification output as input to parameter estimation models. Indeed, the more than satisfactory results obtained with PointNet and its low computational cost make us lean towards this latter solution, but further tests are needed.

Figures 10, 11, 12, 13, 14, 15 and 16 show several examples of classification and single-parameter regression performed by the different ResNet and PointNet models.

Figure 10 showcases the classification capabilities of ResNet and PointNet models on CurveML, with both models achieving nearly perfect performance. This high accuracy suggests that the visual cues necessary for classification are well-captured by both architectures, indicating their effectiveness for this type of task.

Figure 11 focuses on the regression of the α parameter, revealing a notable distinction between the two models. ResNet's mean squared error (MSE) is significantly lower than that of PointNet, showing ResNet's superior ability to capture rotational variations within the plane curves. This could be attributed to ResNet's convolutional structure, which is inherently more sensitive to rotational changes in image data. Figure 12 examines the n parameter regression, which, as previously said in a note, could potentially be treated as a classification problem. Again, the models perform similarly, with ResNet exhibiting a slightly lower error

[4] We could easily have replaced the regression task for the parameter "n" into a classification task, but it was also useful as a canary test to estimate the convergence of the network on *CurveML* after a few epochs.

Fig. 7 Training and validation loss curves of the different ResNet models: (**a**, **b**) classification problem, (**c**, **d**); regression of the parameter n; (**e**, **f**) regression of the parameter α ; (**g**, **h**) regression of the parameters TrX, TrY, a, b. In the legend, x and y refer to TrX, TrY, respectively. The x-axis represents the number of epochs, while the y-axis is the loss value (scalar)

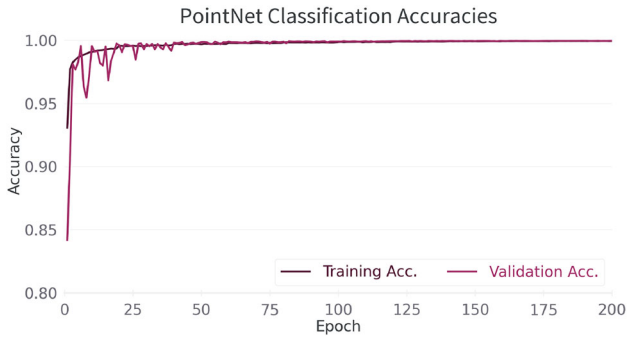


Fig. 8 Training and validation accuracies for the PointNet classification model. The x-axis represents the number of epochs, while the y-axis is the accuracy value (scalar)

rate which in any case is not perceptible simply by examining some randomly chosen samples.

Figures 13 and 14 explore the regression of the *TrX* and *TrY* parameters, respectively. Both models yield close results, with ResNet having a marginally lower regression error. The models' comparable performance on these translation parameters suggests that both networks are adequately learning to identify spatial displacements.

Finally, Figs. 15 and 16 delve into the regression of parameters a and b whose primary visual effect is to alter the size of the plane curves. ResNet's performance is marginally better for both parameters. Also in this case, this could be due to the larger number of ResNet parameters and the greater pretraining to which it was subjected.

In conclusion, ResNet generally performs somewhat better than PointNet for all parameters, except for the α parameter, where its MSE is noticeably lower. A not insignificant aspect that emerges from these experiments is that the training, validation, and test set are overall homogeneous in terms of the information contained and its variability. This is confirmed by the comparable performance obtained on the validation and test set.

Table 8 provides various statistics of the measures introduced in Section 3.3 to quantify the fitting quality of the ResNet regression model, considering the whole test set. The approximation measures are computed by exploiting the parameters predicted by the regression model since they can be inserted into the parametric representation of Eq. 1 to obtain a dense sample of the predicted curve. The statistics considered in this analysis are the first, second, and third quartiles, the mean value, and the standard deviation for each type of family in our dataset. More specifically, these quartiles split a set of sorted real numbers into four parts of approximately equal cardinality: the first (Q1) and the third (Q3) quartiles are defined as the values such that,

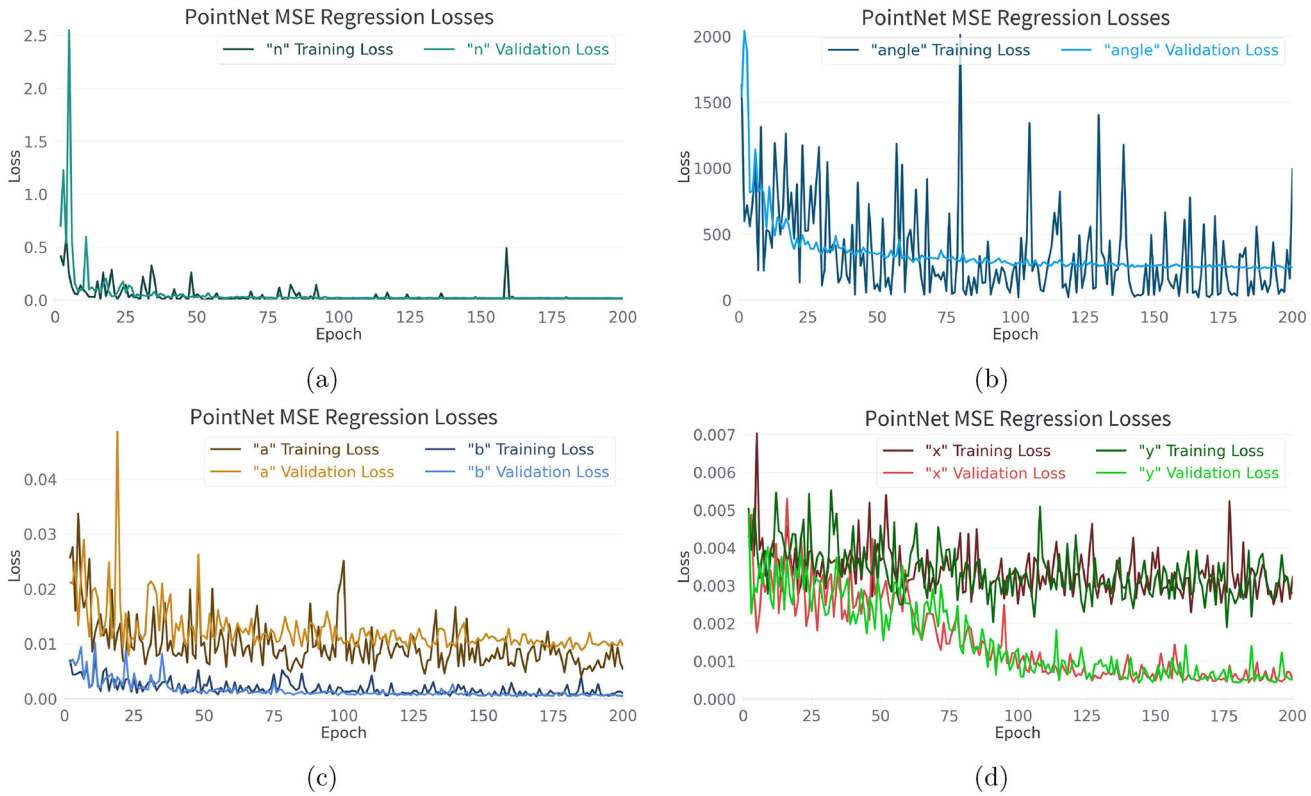


Fig. 9 Training and validation losses (MSE) of the different PointNet models. In the legend, x and y refer to TrX, TrY respectively. The x-axis represents the number of epochs, while the y-axis is the loss value (scalar)

Table 7 Performance of the PointNet regression models on the validation and test sets. The loss represents the MSE (mean square error) between ground truth and predictions

Data set	Problem	Validation Loss	MAE	RMSE	R^2 Score	Pearson Corr. Coeff.	Spearman Coeff.
Validation	TrX	0.000	0.016	0.021	0.985	0.992	0.991
	TrY	0.000	0.015	0.021	0.985	0.992	0.990
	a	0.008	0.056	0.089	0.971	0.987	0.975
	b	0.001	0.014	0.023	0.990	0.995	0.933
	n	0.017	0.063	0.132	0.998	0.999	0.856
	α	247.200	7.505	15.723	0.970	0.989	0.987
Test	TrX	0.000	0.016	0.021	0.985	0.992	0.991
	TrY	0.000	0.015	0.021	0.985	0.993	0.991
	a	0.008	0.057	0.089	0.971	0.987	0.975
	b	0.001	0.014	0.023	0.990	0.995	0.933
	n	0.022	0.064	0.147	0.997	0.999	0.856
	α	262.875	7.624	16.213	0.968	0.987	0.986

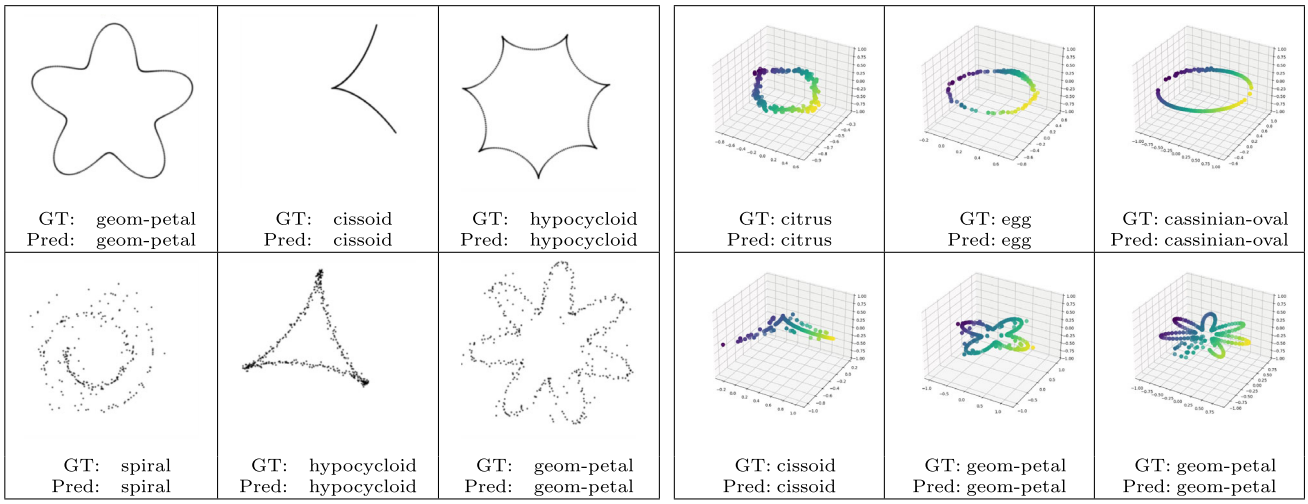


Fig. 10 Inference results for the ResNet (left) and PointNet (right) classification models. The top labels represent the ground truth (GT), while the bottom ones represent the model prediction (Pred)

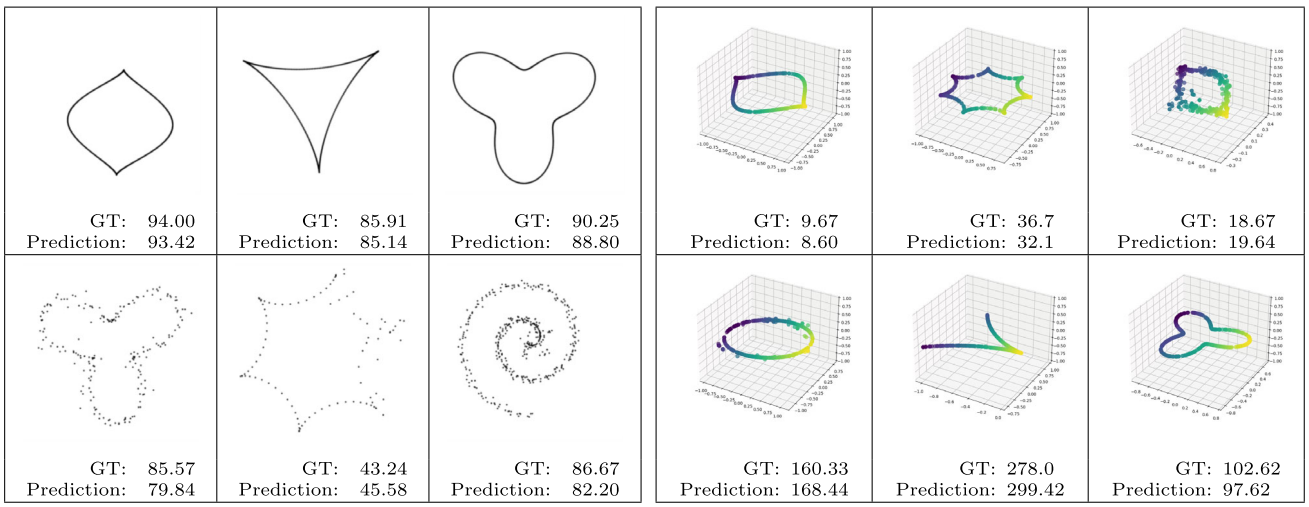


Fig. 11 Inference results for the ResNet (left) and PointNet (right) regression models for the parameter α. The top labels represent the ground truth (GT), while the bottom ones represent the model prediction (Pred)

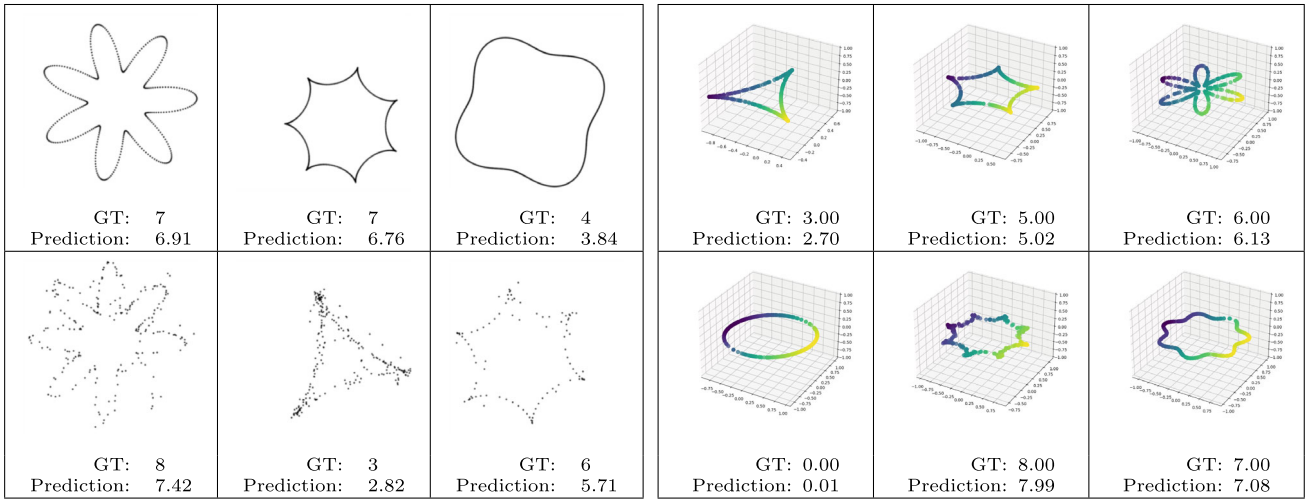


Fig. 12 Inference results for the ResNet (left) and PointNet (right) regression models for the parameter n. The top labels represent the ground truth (GT), while the bottom ones represent the model prediction (Pred)

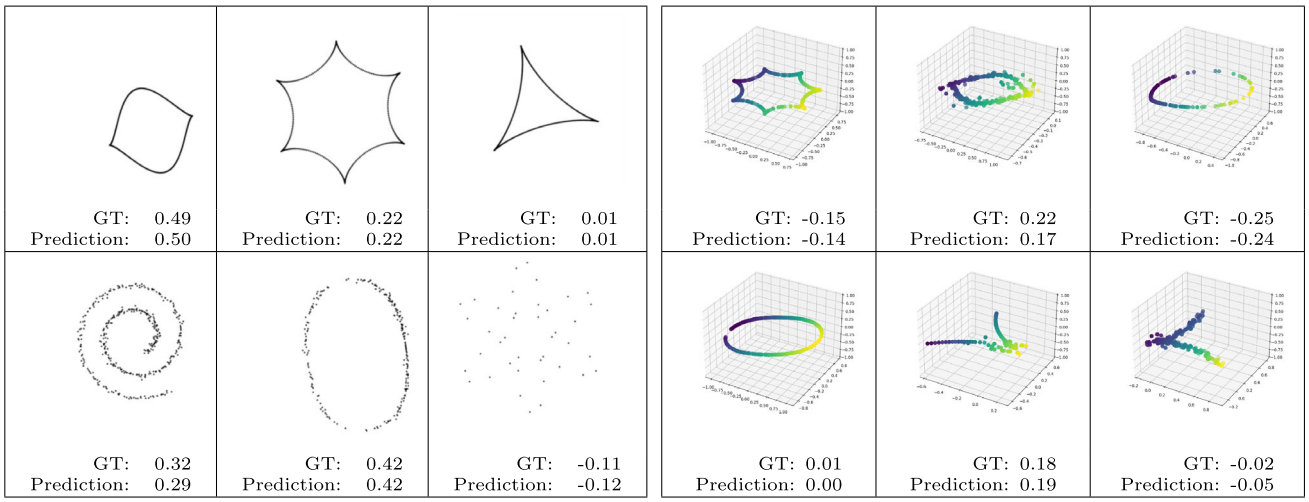


Fig. 13 Inference results for the ResNet (left) and PointNet (right) regression models for the parameter TrX. The top labels represent the ground truth (GT), while the bottom ones represent the model prediction (Pred)

respectively, 25% and 75% of the numbers lie below them; the second quartile (Q2) is the median of the set. Quartiles' significance is due to their ability to identify possible outliers. Based on these statistics, we can conclude that the best results in terms of fitting error are achieved in the case of mouth curves, while the worst values are obtained in the case of Archimedean spirals. With the only exception of the latter family, the quartiles of the MFEs lie under the order of magnitude of 10^{-2}, while the quartiles of the Hausdorff distances lie under the order of magnitude of 10^{-1}. Figure 17 shows some examples of fitting results. Specifically, following the notation introduced in Section 3.3, it displays in black a dense point set $\mathcal{C}_{clean}$ of the original curve, generated starting from the ground truth parameters, and in red the point set $\mathcal{C}$ generated starting from the parameter obtained in the regression step. The values of the MFE and d_{dHaus} for these examples are, respectively, 0.0085 and 0.0185 for the hypocycloid, 0.0054 and 0.0105 for the cissoid, 0.0062 and 0.0120 for the mouth curve, 0.0075 and 0.0310 for the geometric petal.

5 Conclusions

In this paper, we propose the CurveML benchmark, a dataset of 520k curves, and related metrics to develop and compare methods and models for the classification and identification of plane curves represented by point sets. CurveML is easily extendable with new curves; it is open and available at the following URL: https://gitlab.com/4ndr3aR/CurveML.

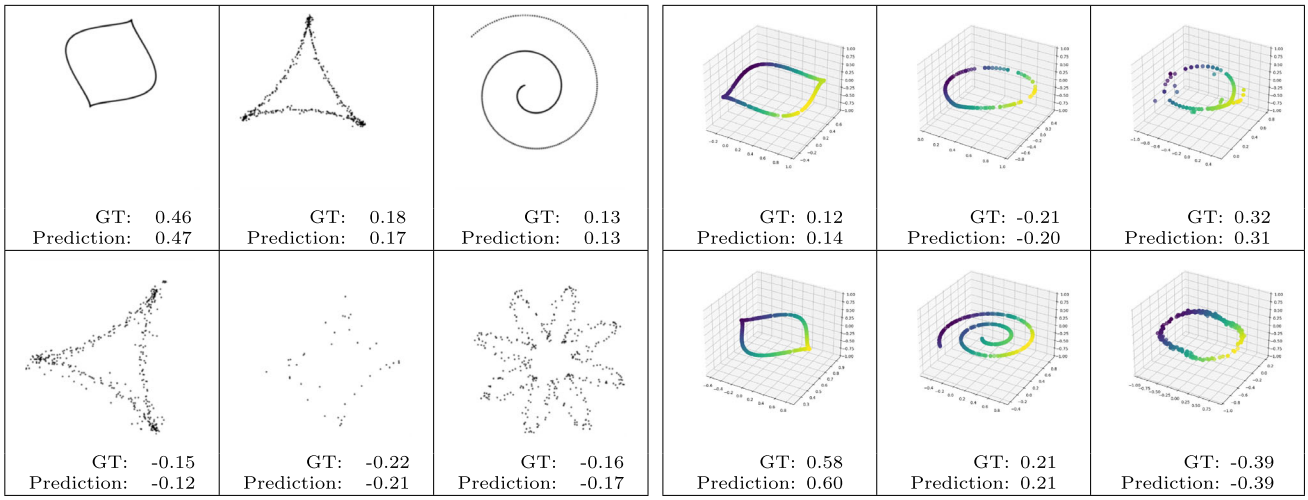


Fig. 14 Inference results for the ResNet (left) and PointNet (right) regression models for the parameter TrY. The top labels represent the ground truth (GT), while the bottom ones represent the model prediction (Pred)

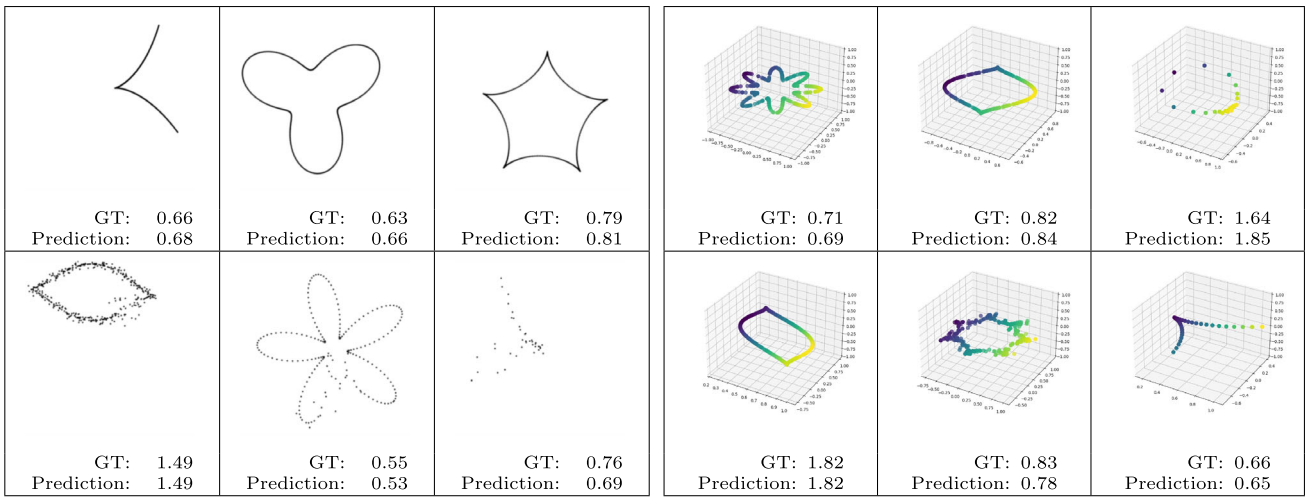


Fig. 15 Inference results for the ResNet (left) and PointNet (right) regression models for the parameter a. The top labels represent the ground truth (GT), while the bottom ones represent the model prediction (Pred)

We focused on generating and curating the CurveML dataset because we believe that, in this time of rapid evolution of Computer Vision, a new dataset that could fill the gap in plane curve classification, regression, and fitting is something the scientific community can use and benefit from it. We believe that this benchmark will impact every task that benefits from a pre-classification of the type of curve in input. This would be beneficial, for instance, to feed the Hough transform with the family of curves and an estimate of the parameters in which to find a solution [23].

Moreover, the problem of approximation of Bézier curves with deep learning methods is still totally open, and CurveML offers the ground truth and evaluation measures for experimenting with novel and automatic approximation techniques.

Our benchmark was designed to be representative of several curve types. We selected a few families of closed, open, bounded, and unbounded curves from the atlas (see [26]). This dictionary has been enriched by adding Bézier curves and composite Bézier curves of 2 or 3 parts. We also perturbed them by introducing some noise or various kinds of artefacts.

Being CurveML split into training, validation, and test sets, it is particularly suitable for machine learning tasks. We train two very different families of deep learning models to perform simple curve classification and single-parameter regression tasks. With these models, we show that our dataset is consistent, and it can be effectively used to train both models that require images and models that work on point clouds and establish a baseline for developing future methods.

Moreover, to promote replicability and future research, we are making all material available: in addition to the dataset, we provide its generating scripts and both the pre-trained

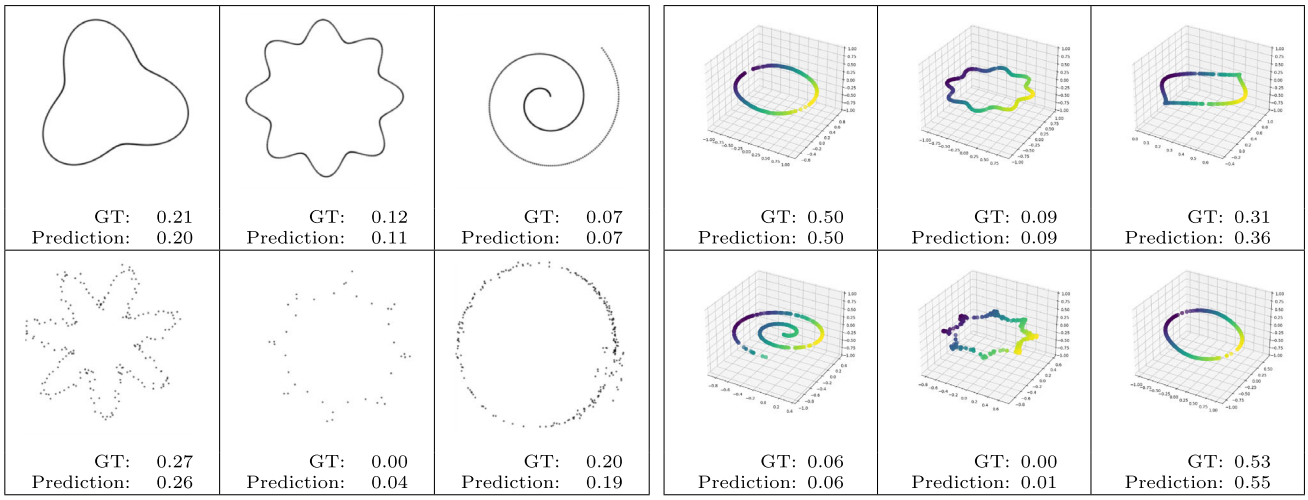


Fig. 16 Inference results for the ResNet (left) and PointNet (right) regression models for the parameter b. The top labels represent the ground truth (GT), while the bottom ones represent the model prediction (Pred)

Table 8 Statistics of the approximation (aka fitting) errors for the whole test set

	Type of curve	Q1	Q2	Q3	mean	std
MFE	citrus	0.0086	0.0121	0.0169	0.0143	0.0103
	hypocycloid	0.0098	0.0142	0.0218	0.0183	0.0160
	geometric petal	0.0112	0.0149	0.0211	0.0167	0.0072
	Archimedean spiral	0.1115	0.1268	0.1446	0.1283	0.0243
	egg of Keplero	0.0107	0.0170	0.0253	0.0193	0.0123
	Cassinian oval	0.0083	0.0121	0.0183	0.0148	0.0121
	mouth curve	0.0075	0.0103	0.0144	0.0118	0.0066
	cissoid of Diocle	0.0095	0.0148	0.0238	0.0181	0.0132
d_{dHaus}	citrus	0.0236	0.0334	0.0456	0.0379	0.0253
	hypocycloid	0.0239	0.0332	0.0455	0.0387	0.0320
	geometric petal	0.0290	0.0353	0.0444	0.0387	0.0145
	Archimedean spiral	0.2550	0.2781	0.3024	0.2788	0.0372
	egg of Keplero	0.0216	0.0302	0.0408	0.0335	0.0186
	Cassinian oval	0.0201	0.0274	0.0389	0.0327	0.0211
	mouth curve	0.0170	0.0240	0.0339	0.0274	0.0154
	cissoid of Diocle	0.0286	0.0458	0.0706	0.0545	0.0417

models and the Jupyter notebooks needed to replicate our training process. Finally, we provide a set of performance metrics able to quantitatively evaluate algorithms and deep learning models for curve classification, parameter inference, and approximation.

To the best of our knowledge, CurveML is the first dataset designed for learning tasks that contain families of parametric curves and even single and composite Bézier curves with control points. While our driving models reach an impressive classification rate with very few misclassifications on noisy data, our experiments show that parameter regression is more challenging. In particular, given that in this preliminary work we have not dealt with the regression of the parameters of Bézier curves, it would be highly valuable to carry out a study similar to the one conducted in Section 4 to extend the learning models trained up to now. Among the key challenges in implementing a neural network to classify, fit and recognise Bézier curves, there is a wide variety of curve configurations. Bézier curves lack a distinctive shape, unlike the other curve families considered in this paper. They can vary greatly in complexity, degree, and number of control points, making it difficult for the neural network to learn a curve representation. Moreover, the quantity of parameters needed to fit a (composite) Bézier curve increases quickly with the degree and number of pieces and can eventually get to the point where the problem becomes unmanageable.

In the future, it would also be interesting to try to train a final classifier–regressor capable of providing all the labels

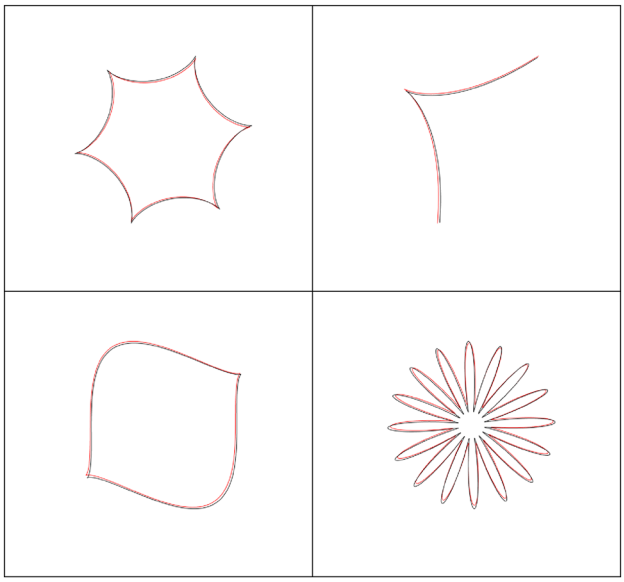


Fig. 17 Visualisation of some fitting results. Following the notation introduced in Section 3.3, the red curve represents the point set $\mathcal{C}_{clean}$, while the black curve represents the point set $\mathcal{C}$

and parameters of the whole dataset at once, regardless of the family type. Indeed, it is extremely challenging to perform regression on these types of parameters: for this reason, we believe that the size and complexity of our dataset are ideal for developing and stress-testing future robust models and curve processing algorithms.

Acknowledgements The present study has been funded by the European Union - NextGenerationEU and by the Ministry of University and Research (MUR), National Recovery and Resilience Plan (NRRP), Mission 4, Component 2, Investment 1.5, project "RAISE - Robotics and AI for Socio-economic Empowerment" (ECS00000035). This work has been partially developed in the CNR-IMATI research activities DIT.AD004.100 and DIT.AD021.080.001. This study has also been carried out in the framework of the activities of NRRP CN MOST—Sustainable Mobility Center, Spoke 7—"Connected Networks and Smart Infrastructure", whose financial support is gratefully acknowledged.

Funding Open access funding provided by Consiglio Nazionale Delle Ricerche (CNR) within the CRUI-CARE Agreement.

Data availability statement The dataset generated and analysed during the current study are available in the CurveML repository, https://gitlab.com/4ndr3aR/CurveML, together with the Python code used for training and data analysis.

Declaration

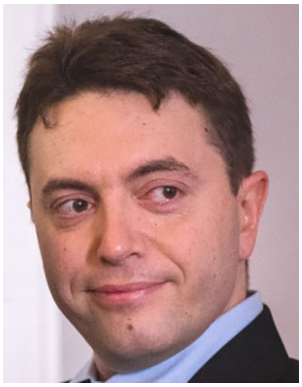
Conflict of interest The authors declare that they have no conflict of interest.

References

1. TRECVid - TREC Video Retrieval Evaluation. Available at https://trecvid.nist.gov (2001-2023)
2. MIREX - Music Information Retrieval Evaluation eXchange. Available at https://www.music-ir.org/mirex (2005-2023)
3. SHREC - SHape REtrieval Contest. Available at https://www.shrec.net (2008-2023)
4. Four Shapes. Available at https://www.kaggle.com/datasets/smeschke/four-shapes (2017)
5. Caputo, A., Giachetti, A., Giannini, F., Lupinetti, K., Monti, M., Pegoraro, M., Ranieri, A.: SFINGE 3D: A novel benchmark for online detection and recognition of heterogeneous hand gestures from 3D fingers' trajectories. Computers & Graphics **91**, 232–242 (2020). https://doi.org/10.1016/j.cag.2020.07.014
6. Deza, M.M., Deza, E.: Encyclopedia of Distances. Springer, Berlin Heidelberg (2009)
7. Farin, G.: Curves and Surfaces for CAGD: A Practical Guide. The Morgan Kaufmann Series in Computer Graphics. Elsevier Science (2001)
8. Gagliardi, L., Raffo, A., Fugacci, U., Biasotti, S., Rocchia, W., Huang, H., Amor, B.B., Fang, Y., Zhang, Y., Wang, X., Christoffer, C., Kihara, D., Axenopoulos, A., Mylonas, S., Daras, P.: SHREC 2022: Protein-ligand binding site recognition. Computers & Graphics **107**, 20–31 (2022). https://doi.org/10.1016/j.cag.2022.07.005
9. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. CoRR **abs/1512.03385** (2015)
10. Howard, J., Gugger, S.: Fastai: A layered API for deep learning. Information **11**(2), 108 (2020). https://doi.org/10.3390/info11020108
11. Juneja, A., Singla, S.K., Kumar, V.: HUDRS: hazy unpaired dataset for road safety. The Visual Computer **39**(9), 3905–3922 (2023). https://doi.org/10.1007/s00371-022-02534-x
12. Koch, S., Matveev, A., Jiang, Z., Williams, F., Artemov, A., Burnaev, E., Alexa, M., Zorin, D., Panozzo, D.: ABC: A Big CAD Model Dataset For Geometric Deep Learning. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2019). https://doi.org/10.1109/CVPR.2019.00983
13. Korchi, A.E., Ghanou, Y.: 2D geometric shapes dataset - for machine learning and pattern recognition. Data in Brief **32**, 106090 (2020)
14. Kuhn Max;Johnson, K.: Applied Predictive Modeling. Springer New York (2018)
15. Lawrence, J.D.: A catalog of special plane curves. Courier Corporation (2013)
16. Lockwood, E.H.: A book of curves. Cambridge University Press (1961)
17. Moscoso Thompson, E., Ranieri, A., Biasotti, S., Chicchon, M., Sipiran, I., Pham, M.K., Nguyen-Ho, T.L., Nguyen, H.D., Tran, M.T.: SHREC 2022: Pothole and crack detection in the road pavement using images and RGB-D data. Computers & Graphics **107**, 161–171 (2022). https://doi.org/10.1016/j.cag.2022.07.018
18. Or, B., Amos, I.: LengthNet: Length Learning for Planar Euclidean Curves. In: P. Frosini, D. Giorgi, S. Melzi, E. Rodolà (eds.) Smart

Tools and Apps for Graphics - Eurographics Italian Chapter Conference. The Eurographics Association (2021). https://doi.org/10.2312/stag.20211472
19. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: PointNet: Deep learning on point sets for 3D classification and segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 77–85. IEEE Computer Society, Los Alamitos, CA, USA (2017). https://doi.org/10.1109/CVPR.2017.16
20. Raffo, A., Fugacci, U., Biasotti, S.: GEO-Nav: A geometric dataset of voltage-gated sodium channels. Computers & Graphics **115**, 285–295 (2023). https://doi.org/10.1016/j.cag.2023.06.023
21. Raffo, A., Fugacci, U., Biasotti, S., Rocchia, W., Liu, Y., Otu, E., Zwiggelaar, R., Hunter, D., Zacharaki, E.I., Psatha, E., Laskos, D., Arvanitis, G., Moustakas, K., Aderinwale, T., Christoffer, C., Shin, W.H., Kihara, D., Giachetti, A., Nguyen, H.N., Nguyen, T.D., Nguyen-Truong, V.T., Le-Thanh, D., Nguyen, H.D., Tran, M.T.: SHREC 2021: Retrieval and classification of protein surfaces equipped with physical and chemical properties. Computers & Graphics **99**, 1–21 (2021). https://doi.org/10.1016/j.cag.2021.06.010
22. Rivaldo, M.G.: FlatShapeNet. Available at https://github.com/reevald/FlatShapeNet/ (2022)
23. Romanengo, C., Biasotti, S., Falcidieno, B.: Recognising decorations in archaeological finds through the analysis of characteristic curves on 3D models. Pattern Recognition Letters **131**, 405–412 (2020)
24. Romanengo, C., Raffo, A., Biasotti, S., Falcidieno, B., Fotis, V., Romanelis, I., Psatha, E., Moustakas, K., Sipiran, I., Nguyen, Q.T., Chu, C.B., Nguyen-Ngoc, K.N., Vo, D.K., To, T.A., Nguyen, N.T., Le-Pham, N.Q., Nguyen, H.D., Tran, M.T., Qie, Y., Anwer, N.: SHREC 2022: Fitting and recognition of simple geometric primitives on point clouds. Computers & Graphics **107**, 32–49 (2022). https://doi.org/10.1016/j.cag.2022.07.004
25. Romanengo, C., Raffo, A., Qie, Y., Anwer, N., Falcidieno, B.: Fit4CAD: A point cloud benchmark for fitting simple geometric primitives in CAD objects. Computers & Graphics **102**, 133–143 (2022). https://doi.org/10.1016/j.cag.2021.09.013
26. Shikin, E.V.: Handbook and atlas of curves. CRC Press (1995)
27. Sipiran, I., Meruane, R., Bustos, B., Schreck, T., Li, B., Lu, Y., Johan, H.: A benchmark of simulated range images for partial shape retrieval. The Visual Computer **30**(11), 1293–1308 (2014). https://doi.org/10.1007/s00371-014-0937-2
28. Yan, X.: Pointnet/pointnet++ pytorch. https://github.com/yanx27/Pointnet_Pointnet2_pytorch (2019)
29. Zhao, X., Li, Q., Chao, Y., Wang, Q., He, Z., Liang, D.: RT-less: a multi-scene RGB dataset for 6D pose estimation of reflective texture-less objects. The Visual Computer (2023). https://doi.org/10.1007/s00371-023-03097-1

Andrea Raffo is a postdoctoral candidate in Bioinformatics at the University of Oslo, having obtained his PhD in Mathematics from the same institution in 2022. His research interests revolve around geometric modelling and pattern recognition, with a focus on various application domains such as biosciences, computer graphics, and computer-aided design.



Andrea Ranieri is a researcher at CNR-IMATI where he works with Deep Learning mainly in the field of Computer Vision. Andrea has a degree in Computer Engineering, a PhD in Space Science and Engineering and an extremely varied technological background ranging from networking and distributed systems to robotics and perception. He participated in many Italian and European research projects and co-authored many scientific publications in conference proceedings and international journals. His research interests range from semantic segmentation problems on complex datasets to self-supervised learning and generation of synthetic datasets.

Chiara Romanengo has a PhD in Mathematics and Applications and is currently a research fellow at CNR-IMATI. Her research interests include curves and surfaces, algebraic geometry, features on 3D models, and geometric modelling. She investigates methods for the identification and recognition of characteristic parts on the surface of 3D models based on the Hough transform technique.

Bianca Falcidieno is an associate Research Director at CNR-IMATI. She has been leading and coordinating research at the international level in Computational Mathematics, Computer Graphics, and Shape Modeling. She authored over 250 peer-reviewed publications and books. She is a Eurographics Fellow and a Pioneer of the Solid Modeling Association. She was awarded the Eurographics Gold Medal and the Tosiyasu Kunii Award for her outstanding scientific contributions to computer graphics and shape modelling.

Silvia Biasotti is a Research Director at the Institute of Applied Mathematics and Information Technologies of the National Research Council of Italy (CNR-IMATI). Her research interests include shape modelling; similarity reasoning in the presence of incomplete or partial information; pattern, colour and texture recognition on surfaces; and change detection on 3D data; the exploration of large collections of 3D models. She was in charge of several CNR-IMATI research activities on surface similarity reasoning and is involved in the creation of datasets and benchmarks, in particular, she organised several tracks of the SHape REtrieval Contest (SHREC).