FOUNDATIONS OF COMPUTING AND DECISION SCIENCES
Vol. 51 (2026) No. 1
ISSN 0867-6356
sciendo DOI: 10.2478/fcds-2026-0004 e-ISSN 2300-3405

Influence of Graphical Representation Type on Tessellated Geometry Classification

Maciej Majchrzak * Katarzyna Marciniak-Lukasiak †
Mateusz Jakubowski ‡ Piotr Lukasiak §

Abstract. This research investigates the impact of graphical representation on the accuracy of three-dimensional shape classification. 3D models, both scanned and modeled, are used in engineering, computer graphics and scientific data visualization. Various approaches are adopted in these fields to visually represent 3D geometry, utilizing solutions such as OpenGL and Direct3D, each with its distinct goal of achieving either real-time manipulation or photorealism. The purpose of this research was to determine the most effective graphical representation for categorizing mechanical components with a high degree of geometric similarity, such as beams and rods. The study examined various image representations and their combinations, obtained through the adjustment of rendering parameters and image compositing. In an effort to improve classification accuracy, novel techniques for addressing image recognition issues were developed and tested against commonly used image representation methods. This innovative approach proposed in the paper led to a 48% reduction in classification errors.

Keywords: 3D scanning, Image classification, Convolutional neural networks, Rendering (computer graphics), Computer aided design, machine learning

*Faculty of Computing and Telecommunications, Poznan University of Technology, Poznan, Poland, maciej.majchrzak@put.poznan.pl

†Faculty of Food Assessment and Technology, Warsaw University of Life Sciences, Warsaw, Poland, katarzyna_marciniak_lukasiak@sggw.edu.pl

‡Faculty of Mechanical Engineering, Poznan University of Technology, Poznan, Poland, mateusz.jakubowski@doctorate.put.poznan.pl

§Faculty of Computing and Telecommunications, Poznan University of Technology, Poznan, Poland, piotr.lukasiak@put.poznan.pl, corresponding author

1. Introduction

The classification of 3D models in various fields, such as engineering, computer graphics and medicine, is a challenging problem. The prevalence of CAD (Computer Aided Design), computer graphics, and medical imaging programs has resulted in an ever-increasing number of 3D models. The expanding use of 3D-scanning technology [21] and photogrammetry [10] is enlarging the collection of precise models of real objects with intricate and uneven shapes that previously could not be faithfully represented in 3D before. Especially in engineering, 3d scanning has found its application, making it possible to combine newly designed structures with existing ones. However, the challenge remains in classifying the elements of the scanned structure, arising from the fact that the scans are in the form of a point cloud. CAD programs can display them, but a human being is still needed to recognize the elements. The growing number of models leads to the need for categorizing techniques, and the development of Machine Learning methods to solve this issue began promptly [4]. Several classification methods have been developed that utilize the 3D geometry of a model, including voxel-based [44], feature-based [39], and graph-based [45]. Nevertheless, the adaptability of these methods is restricted by the different ways in which 3D geometry can be stored - two disjoint scans of the same geometry could produce a polygonal mesh having completely different topologies, even if they depict the same thing to the human eye (see Fig. 1).

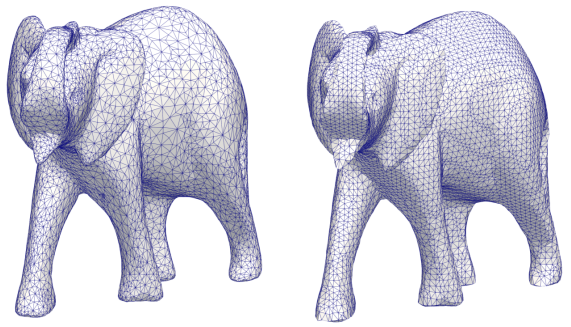

Figure 1. Two distinct tessellations of a 3D model. The left one was acquired through Catmull-Clarke subdivision [9] of a triangular mesh, and the right one was produced by voxel-based remeshing [55] of the mesh featured on the left. Although both meshes retain a similar number of nodes and appearance, they have greatly dissimilar topologies.

The limitations and scarce availability of these methods have led to the increased use of 2D image recognition for this purpose. The image recognition market is continuously growing, and this trend is expected to continue in the upcoming years [64]. The widespread application of this method in commerce, multimedia, banking, automotive, telecommunications, and medicine is due to the numerous available solutions and their relative ease of use. Point clouds generated by 3D-scanning can easily be

triangulated into a surface mesh and rendered into images. Consequently, this study aims to determine which 3D shape representation method yields the highest classification quality. The research involved generating an image dataset consisting of various graphical representations of mechanical components. Based on the results obtained for the basic representation types, new methods of representation were developed, some of which yielded higher classification accuracy. The resultant knowledge could enable the standardization of representation type, both in research and practical applications (see Fig. 2), allowing for a focus on other important aspects, making this research both novel and significant.

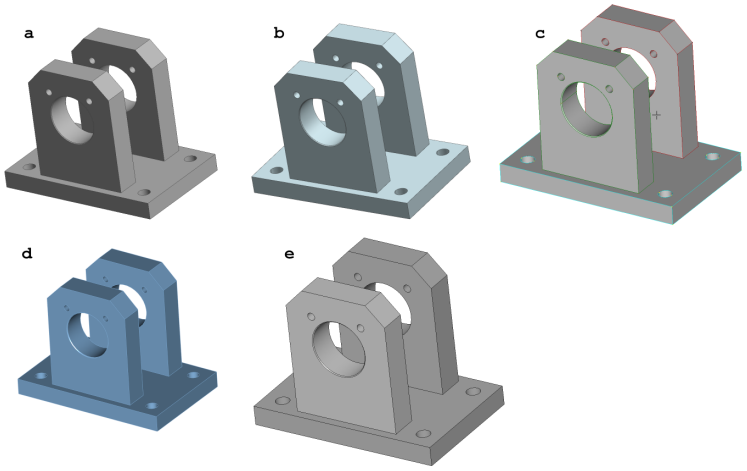

Figure 2. The standard representation types used in CAD/CAE programs exhibit a significant degree of variation in terms of appearance. The image presented provides a depiction of a basic assembly viewed using Autodesk Inventor (a), Siemens NX (b), Siemens Femap (c), Varicad (d), and CAD Assistant (e).

2. Related Works

Chen et al. introduced the Light Field Descriptor (LFD) in 2003 [11]. Their methodology involves producing light field images, which are sets of monochromatic 3D shape projections from various perspectives (see Fig. 2). Comparing these sets facilitated recognition and clustering of comparable shapes. The outcomes obtained were 25% superior to MPEG-7 Multiple View Descriptor [48] and 94% better than MPEG-7 Shape 3D descriptors [11]. In 2004, Lecun [38] conducted a comparison between Convolutional Neural Networks (CNNs) [24] and Support Vector Machines (SVM) [18] in terms of classification accuracy of images that represented 3D models. The models were categorized into 5 classes: four-legged animals, human figures, airplanes, trucks and cars. Images were created with solid and textured backgrounds, under various

lighting conditions and from different angles. It was found that in both cases, CNN outperformed SVMs resulting in a 7% error rate. The real-time version of the system achieved object classification at a rate of approximately 10 frames per second [38]. In 2015, Su et al. proposed the Multi-view Convolutional Neural Network (MVCNN) [65]. Their approach involves generating several graphical representations of a specific 3D model (see Fig. 3), feeding each image into CNNs, and then aggregating the resulting outputs in a pooling layer. The second CNN then performs classification. The MVCNN attained an accuracy of 87.2%, compared to human performance of 93% [65].

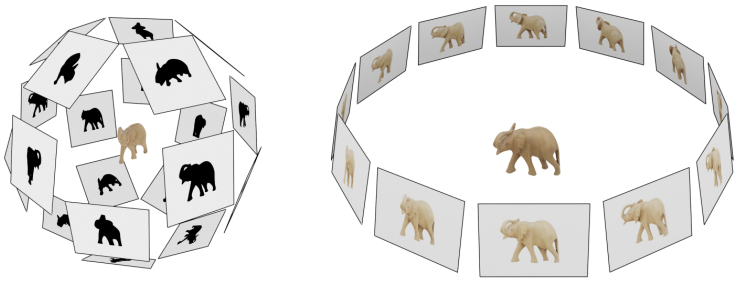

Figure 3. Multiple views were used to compare LFD (left) and MVCNN (right). LFD utilizes silhouettes of a solid, with monochromatic images taken from 20 viewpoints that are located at dodecahedron vertices. Conversely, MVCNN uses a shaded representation of the solid, with 12 viewpoints situated on a single plane.

Heisele et al. conducted a study investigating the possibility of training networks to recognize real-life objects in photos using 3D models [27]. The study addressed the impact of camera positioning and resolution on classification accuracy. Additionally, they found that the preparation of photorealistic scenes is a time-consuming process despite the short time it takes for rendering [27]. Sarkar et al. conducted an investigation [60] on the influence of the background and surface appearance of models on classification quality utilizing CNNs. However, their research did not pertain to the classification of 3D models themselves, but to the identification of genuine objects depicted in, say, catalog images. Furthermore, they noticed that 3D model detail has an impact on prediction quality, with the most successful outcomes associated with a fully photorealistic texture model [60]. Kanezaki et al. presented RotationNet [33], a CNN structure that can concurrently recognize an object's class and pose. Unlike MVCNN, RotationNet has the capability of using fewer images, rendering side views needless. Experiments on the ModelNet40 database produced an accuracy of 90.44% for a single perspective and 97.45% for 12 perspectives [33]. Multi-view Graph Attention Networks (MGAT) [76] is a method for using graphs in multi-view image classification. The method, presented by Xie et al., employs graph embedding to select a set of views that yield higher classification accuracy compared to a randomly selected set. Another approach for utilizing graphs in classifying 3D models is the 3D Shape Knowledge Graph [49], proposed by Nie et al. They introduced the idea of a geometric word that can describe a model attribute. Each 3D shape in the dataset

can be described using specific terminology, and their connections to one another are a subset of the overall geometric vocabulary. A groundbreaking technique, introduced by Li et al. in 2020, that converts CAD models into representations of the frequency domain [39]. Their approach involves searching for fingerprints generated from the spectrogram of the obtained frequency domain. Zeng et al. proposed the GA-MVCNN [80] approach which utilizes a multi-view strategy to develop a Hierarchical Graph Attention Based Multi-View Convolutional Neural Network. The approach employs Graph Attention Network (GAT) [76] to select views and reduce redundancy, and correlation-weighted feature aggregation to merge the selected views. The proposed model exceeds the performance of ResNet-34 by 1.6% in terms of accuracy [80]. In 2022, Khorolska et al [35] presented an approach similar to MVCNN, but with additional CNN utilization to combine multiple images into one for classification. Their architecture was tested on a ModelNet40 dataset, achieving a 93.01% accuracy compared to MVNN's 89.9% [35]. Researchers are increasingly interested in general image recognition, resulting in the emergence of new solutions frequently. The ImageNet dataset, which comprises more than 14 million images from over 20,000 categories, serves as a standard for evaluating the classification accuracy of convolutional neural networks (CNNs). In recent years, the record for accuracy of classification of this dataset has been broken many times. In 2019 by the EfficientNetB7 [67], in 2020 by BiT-L [36], NoisyStudent [75], FixEfficientNet-L2 [69] and Meta Pseudo Labels [54]. In 2021, ViT-G/14 [79] and CoAtNet [50] took the lead, followed by Model Soups [72] and CoCa [77] in 2022, and BASIC-L [12] in 2023. In recent years, there has been a noticeable decline in the frequency of major breakthroughs in image recognition - an example being that the two newer architectures than those mentioned earlier - GhostNet [43] and ConvNeXt-T-Hermite [34] - have not attained higher classification accuracy for the ImageNet set. However, this does not mean a decline in interest in the subject, as new architectures [41, 74] and improved versions of established solutions such as EfficientNetV2 [68] continue to appear.

Advances in image classification, such as the advent of vision transformers [70], have ensured that image-based 3D shape recognition methods continue to advance as well. Liu et al. proposed a method based on MVCNN utilizing voting-based view filtering to select the most informative views [42]. Their approach resulted in a 95.2% accuracy on the ModelNet40 dataset, surpassing other methods based on MVCNN [35]. In 2023 Ding et al. presented a method of classification of 3D models based on extraction of global and local features using Vision Transformers [20]. Wu et al. explored the robustness of 3d shape recognition methods against occlusion and previously unseen objects, while highlighting the importance of these methods with the ever-increasing number of 3D models available [73].

Significant advances in generative AI models in recent years have also brought new solutions in 3D shape classification. An example of the Analysis-by-Synthesis method is the DC3DO: Diffusion Classifier for 3D Objects [37] presented in 2024. The authors propose introducing 3D models into a generative model as input noise, along with various labels. Classification is based on comparing the differences between the input geometry and the model output for each class label. Diffusion-Based Classification (DiffClass) presented in the same year by Meng et al. [46] is another example.

The authors train the diffusion model so that the denoising trajectory is specific to a given class. Again, the classification is performed by comparing how well the model handles denoising points under different labels.
The increasing frequency of new and improved solutions promotes their usage in the classification of 3D models, as opposed to purpose-built architectures. Previous works on the classification of 3D models have examined the impact of lighting, background, texture [60], and camera settings [27], but there is a lack of research that concentrates solely on the influence of graphical representation type on classification accuracy.

3. Materials and Methods

Due to the lack of sufficiently large open datasets of 3D scans of mechanical components and to avoid the influence of scanning method, color, illumination or surface roughness, it was decided to conduct tests on tessellated models derived from CAD programs. The dataset utilized comprises 8 component classes, with each class associated with specific mechanical components (see Fig. 4). These components include:

- C-beams - a type of beam used in construction and mechanical engineering.
- flat bars - a flat strip of sheet metal
- hollow rectangular sections - popular in lighter structures and automotive applications
- I-beams and H-beams - the most widely used beam types in construction
- L-beams are a beam type commonly used in construction and mechanical engineering
- pipes are utilized as load-bearing components and for fluid transport
- round bars (rods) have both a construction and mechanical engineering application
- square bars are primarily used in mechanical engineering.

For each category, 250 components were generated, resulting in a perfectly balanced dataset of 2,000 components. The chosen categories are frequently used in construction, primarily in heavy industry, construction, or industrial installations. The components used in the study come from genuine industrial projects, specifically from waste incinerators. The dataset is comprised of real engineering models without any prior selection. This ensures that the models are tested in conditions that more closely resemble practical industrial applications. The selection of these categories was intended to restrict diversity in the dimensions of 3D geometry, as all eight categories contain elements with a uniform cross-section, and make the results less dependent on the overall shape of the component. The graphical representation was unified by

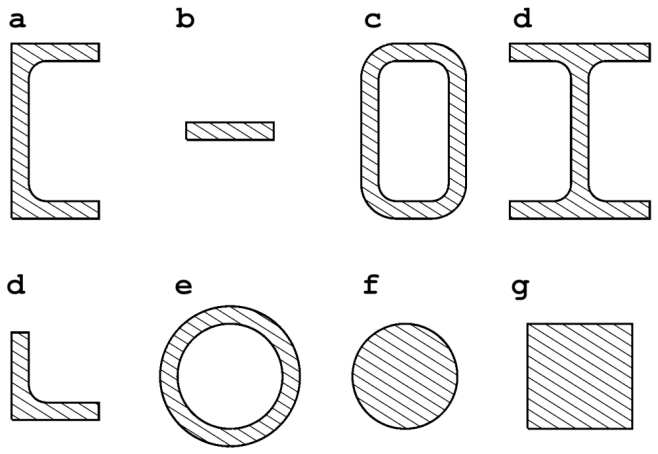

Figure 4. Classes of components, in sequence: c-beam (a), flat bar (b), hollow section (c), I-beam (d), L-beam (e), pipe (f), round bar (g) and square bar (h). These components have a constant cross-section along their length and are regularly used in both mechanical engineering and building construction. The dimensions of these components are commonly defined as length, width and height, or length and diameter for pipes.

focusing on one end of the classified component. The addition of mechanical components with more compact shapes, like nuts or bearings, would have complicated the interpretation of the results. This is because some of them would have been visible in full and some only partially. Displaying elongated components wholly is not feasible because very lengthy beams would appear as a solitary, broad line due to the restricted resolution of images required by popular Convolutional Neural Networks (see Fig. 5).

3.1. Export and import of CAD data

Geometry data describing structural components are saved in cad software formats, but not all of them allow for model transfers. Therefore, universal formats such as IGES [47] and STEP [30] are employed. IGES is based on ASCII, and it is an older format dating back to the punched card era. Its popularity fell significantly after the ISO 10303 standard describing the STEP format was published in 1994. Based on Boundary Representation (BREP) [30], the STEP file format is currently more widely used. BREP describes the exterior surfaces of a solid by defining all the component surfaces. Import and export of STEP files are supported by all 3D CAD programs, making them the industry standard for transferring CAD models between companies.

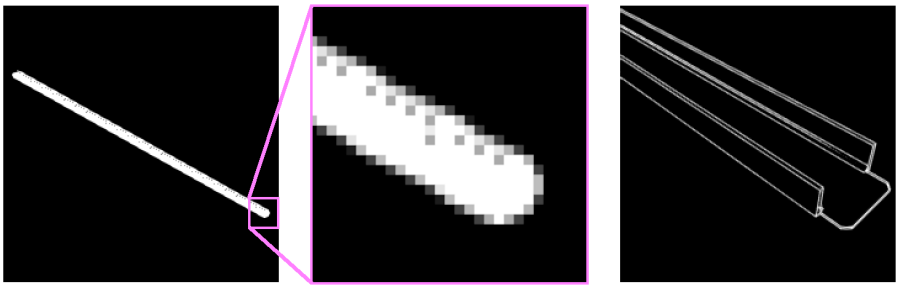

Figure 5. Long components are challenging to recognize with low-resolution images (in this case, 224x224 pixels) as the image needs to be scaled. On the right, there is an image of the same beam as on the left, but the camera only focused on its end, making the identification of the component's type possible.

This is due to the waning support for the outdated IGES format in contemporary CAD software. In this study, the STEP format files were worked on using Open Cascade Technology (OCCT, OCC), an open-source software development platform designed for 3D CAD, CAM and CAE [51]. Another advantage of utilizing this format is the reduced occurrence of errors during the importation process. The resulting models of the structure were obtained in the form of files containing complete assemblies of large sections of the entire plant. A Python script was developed for the current study [71], utilizing the pythonOCC library [53], which functions as a Python wrapper for OpenCASCADE C++ technology. The software extracted individual parts from the assembly and saved them in separate files. The names of these files were taken from the assembly file, and uniquely identified the class to which the component belonged (Fig. 6). Files displaying defects such as absent surfaces, unattached overlapping edges or non-manifold intersections between surfaces following importation have been rejected.

The study employed the Euler characteristic defined as

$$\chi = V - E + F, \tag{1}$$

which depicts the correlation between the number of surfaces F, edges E and vertices V [22]. For any convex polyhedron the value of χ equal to 2, a deviation from this value signifies non-manifoldness of the shell. As all the components share a constant cross-section, it was beneficial to orient them in the same direction to streamline further processing. To achieve this, a Python script was developed to leverage pythonOCC's abilities to evaluate the features of a solid, including surfaces, edges, and nodes. The longitudinal axis direction of each solid was determined by summing the lengths of all edges and cylindrical surface axes in the same orientation in three-dimensional space. Technical term abbreviations were explained upon first use. The direction which provided the highest combined length was designated as the longitudinal axis of the model. It was then rotated to coincide with the X axis of the Cartesian coordinate system [19] used in pythonOCC.

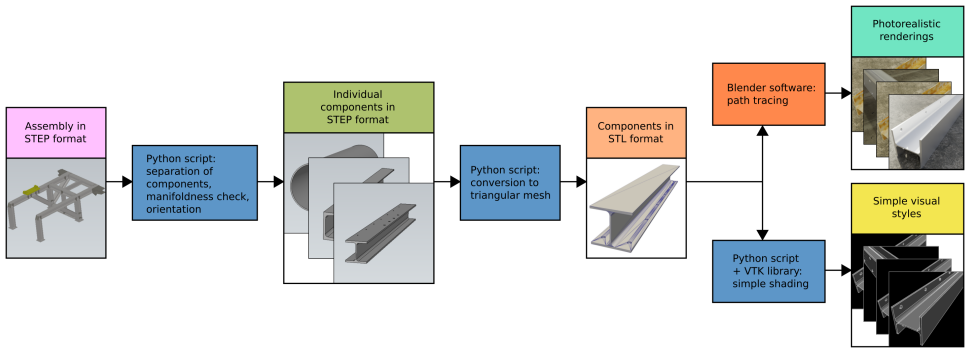

Figure 6. A diagram demonstrating the process of dataset preparation is presented. To begin, an assembly written in STEP format undergoes division into individual components with the aid of a Python script. This same script then checks the manifoldness of the 3D model whilst also orienting it correctly. Subsequently, a second Python script converts the components into STL format. The STL files are then used to generate images using Blender for photorealistic rendering, and a Python script using the VTK library for simple representations.

For pipes and cylindrical rods this method was effective. However, for other shapes, an additional step was needed to prevent an unusual and physically impossible alignment. This involved identifying the flat face with the largest surface area and orienting the model so that the normal direction coincided with the Z-axis (considered the vertical direction) while maintaining the lowest Z coordinate value. The resulting models were saved in STEP format in a separate directory. The next step was to convert the STEP files into a format that could be easily imported in computer graphics programs. The STL [1] format was chosen because of its widespread use in both graphics and CAD programs. In it, a surface is described using nodes and the triangles connecting them. The script prepared for this purpose opened all STEP files one by one and saved them in STL format, with an arbitrarily selected quality (defined as the acceptable deviation of tessellated surface from the input model).

3.2. Image preparation

Five types of graphical representation were utilized in the initial stage of the research, varying from basic to advanced (see Fig. 7):

- wireframe - where lines depict only the solid edges
- shaded - where the surfaces of the solid are visible, and colors vary depending on light incidence angles
- fast rendering - commonly used in graphics software for quick model visualization.

- photorealistic without background - physically correct rendering without background
- photorealistic rendering with background.

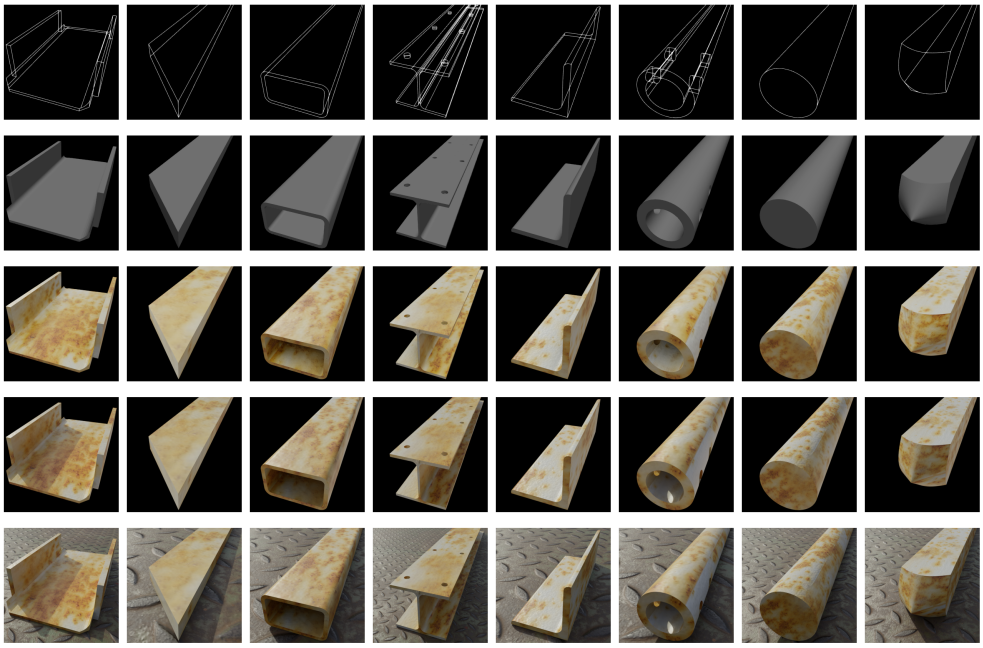

Figure 7. This picture demonstrates 5 basic graphical representation types, arranged from the top to the bottom as follows: wireframe, shaded, fast rendering, photorealistic without a background, and photorealistic with a background. In each column, one of the eight component categories is represented.

Blender takes advantage of the ability to define component surfaces in more detail by using the BSDF shader [3] based on the popular Disney Principled Shader [6]. This allows for the use of diffuse, roughness, normal and displacement textures with placement defined by uv-mapping [8]. The same shader was also employed for the surface upon which the components rest. Additionally, a panoramic environment map illuminated the scene using HDR [57]. This image is also presented in the background. The number of samples affected the quality of the rendering [32] (see Fig. 8), which was improved by using the built-in OpenImage denoiser [29]. All used textures, as well as the 3D model used in figure 5, have been sourced from Poly Haven website [78] and are licensed under Creative Commons 0 (CC0).

To generate images, STL files are loaded into Blender. Successive models from each class are then loaded and shifted relative to the previous one. This process facilitates selecting the currently visible component by moving the entire array of components in front of the camera. Four combinations of randomly generated camera

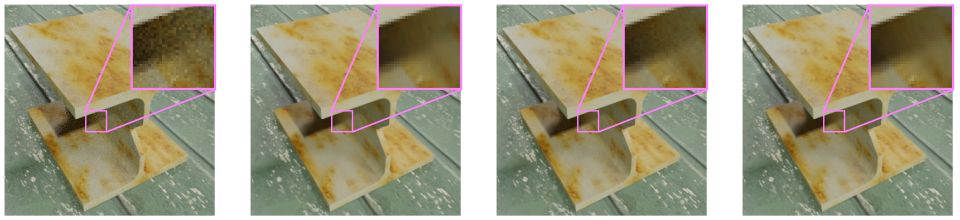

Figure 8. Rendered images were generated using 50, 50 (with denoiser), 500, and 500 (with denoiser) samples, arranged from left to right. The utilization of a denoiser enables a reduction in the number of samples by a factor of ten while still producing an image of comparable quality to those generated with significantly more samples. Nonetheless, expertise is required when implementing this technique since using an insufficient number of samples may result in blurring of the textures.

azimuth, ranging from 10 to 45 degrees, and elevation, ranging from 25 to 45 degrees are determined for each of the 250 class representatives (see Fig. 9).

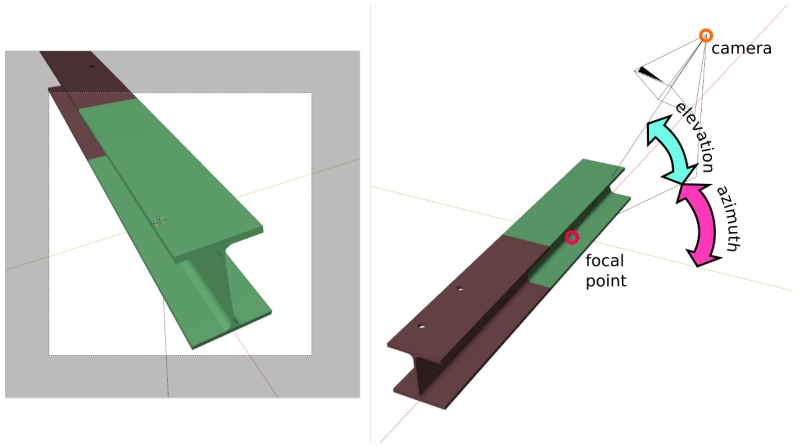

Figure 9. A slender component can be separated into two parts: the camera's focus falls upon the green section, while the remainder (red) is visible but disregarded in camera alignment. The object's perspective is defined by the azimuth (magenta) and elevation (cyan) angles.

The angles were selected based on a frequently encountered arrangement for photographing beams on the internet, while following objectivity principles. One out of four component materials, one of seven background materials and one of three environments are randomly selected for the renderings, as portrayed in Figure 10.

All components from each of the eight classes were rendered with the same settings, as demonstrated in Figure 6. The camera's position is adjusted according to the component size after being rotated by azimuth and elevation angles. For components

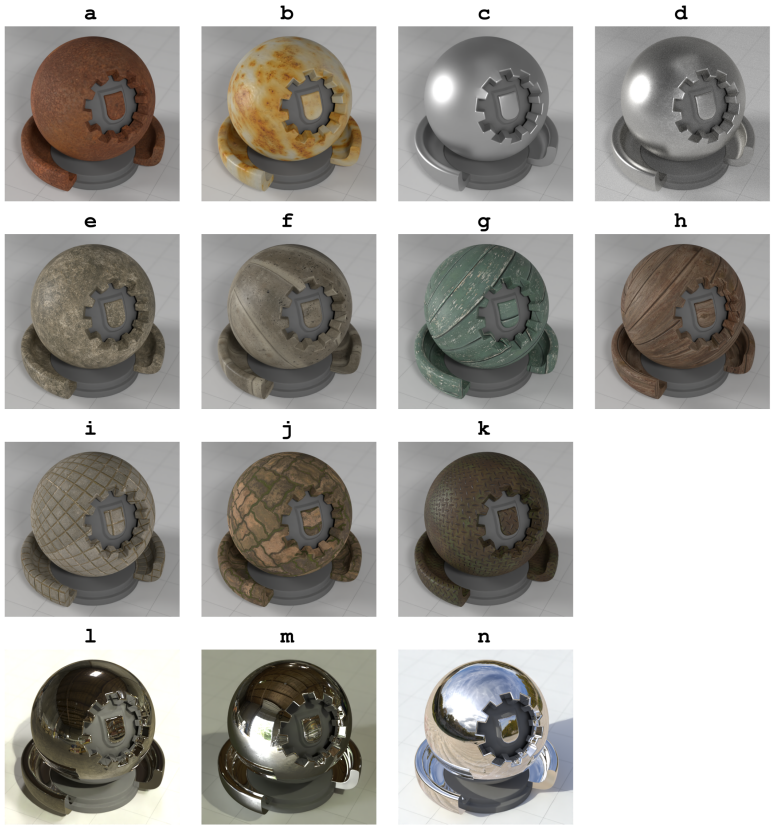

Figure 10. The top row displays four component materials: rusted steel (a), rust-speckled paint (b), aluminum (c), and stainless steel (d). The subsequent two rows exhibit seven background materials: poured concrete floor (e), concrete wall (f), painted wooden wall (g), wooden floor (h), pavement slabs (i), pavement tiles (j), and tread plate (k). Finally, the bottom row showcases reflections of environment maps, including factory interior (l), garage (m), and outdoor (n).

with a length to width or length to height ratio greater than 4:1, only a section of the component with a length equal to four times the width of the profile is focused on by the camera (see Fig. 9).

The animation's keyframes save the positions of the components, camera, and materials. Simultaneously, the same data is saved in a .csv file for generating images using the VTK library. The Python script performs the entire process, as Blender has its own Python API [17]. The animation is saved separately for fast rendering (Eevee renderer [25]) and both photorealistic representations (Cycles renderer [31]). The camera position settings that have been saved are then utilized in a Python script that employs the VTK library to produce graphical representations of the models. These images include both shaded and wireframe views. The intention behind using

identical views and materials for each class was to minimize the randomness of the results obtained.
The various methods of generating images varied significantly in the time required to prepare an 8,000-image dataset. Fast render took 310 minutes, photorealistic without background took 513 minutes, and photorealistic with background - the most time-consuming - took 639 minutes. The situation was entirely different for simpler representations obtained using the VTK library - for a single type, it took 4 minutes. The script generated all images for all 10 representation types in 32 minutes. Additionally, preparing the files required for renderings in Blender is a time-consuming process that requires experience.

3.3. Methods

The classification process was carried out using four commonly used neural network architectures:

- DenseNet201
- EfficientNetB4
- ResNet50V2
- Xception

The choice of multiple architectures was made to minimize the influence of the network's characteristics on the validity of the results.
One of these architectures is DenseNet [28], an Artificial Neural Network (ANN) that connects each layer to every other layer. This means that for each layer, the input consists of feature-maps from all previous layers. DenseNet has a significantly higher number of direct connections between layers compared to traditional CNNs. In this work, we utilized a variant of DenseNet201 which consists of 402 layers.
The EfficientNetB4 architecture used is one of the variants of EfficientNet, a family of networks proposed by Tan [67]. Their solution was based on using neural architecture search [81] to find combinations of basic parameters that describe the network architecture, such as depth, width, and resolution. Beginning with the baseline model, the architecture parameters were scaled using a non-uniform scale factor. One of the variants achieved the highest classification accuracy on the ImageNet [59] set at the time of publication, with a much smaller network size and faster inference time. Non-uniform scaling enables the identification of the optimal architecture at a given network size. EfficientNetB4 comprises 258 layers.
ResNet, an ANN architecture based on HighwayNet, was proposed by He [26]. In order to reduce the vanishing gradient problem, ResNet includes links that bypass some layers [63]. This work uses ResNet50V2 which consists of 103 layers.
Xception, a network architecture based on Inception [66], was proposed by Chollet [15]. The architecture of Xception, which consists of 81 layers, outperformed Inception V3 on the ImageNet dataset. Xception uses images with a resolution of 299x299

pixels, while the other networks use images with 224x224 pixels.
The library used in this work is TensorFlow [2], managed through the Keras [16] library of the Python language. The networks were trained on a workstation equipped with a GPU using the NVidia CUDA library. Pre-trained weights from the ImageNet dataset were used for all architectures, eliminating the need for model training and reducing network training time. The hyper-parameters also had identical values, as follows:

- learning rate = 0.01
- momentum = 0.3
- batch size = 32
- decay rate = 1

To reduce overfitting, a dropout layer with a dropout factor of 0.5 was added [62]. The last dense layer has a size of 128. The maximum number of epochs is 40, and an early-stopping mechanism was used to stop training if the loss did not drop for more than 3 epochs [7]. Stratified K-fold validation [58] with 10 folds was used to ensure proper model validation. This method ensures an even distribution of representatives from all classes in each fold. All tests were run with the random number generator seed set to the same value of 42, so that each iteration for each style was trained and tested on exactly the same set of models. This ensured that only the appearance of the image could influence the quality of the classification.

4. Results and Discussion

The first step was to examine the images generated using the five representation types. The main evaluation criterion was classification accuracy, but as this already varied by 98% for the basic types of graphical representation, it was decided to use an additional criterion - macro-averaged F1 score. The macro-averaged F1 score is calculated as the arithmetic mean of the F1 scores obtained for each class separately. This measure allows one to assess whether the model performs equally well for all classes, as its value drops significantly even if the classification accuracy is low for just one of the classes [52].

As the chosen initial representations are commonly used and the neural network architectures tested are state-of-the-art, the results of this first series of experiments will be taken as a baseline for the remaining tests. Table 1 includes the values of both classification quality measures and the number of misclassifications.

The ResNet architecture achieved the highest values of accuracy and Macro-F1 for both the Shaded representations, with 98.18% and 99.09% respectively, showing a negligible advantage over Wireframe (97.90% and 98.95%). The Fast render achieved an accuracy of 92.85% (Xception), while Photorealistic without background images

Table 1. Results obtained using 5 basic types. The values for accuracy, Macro-averaged F1 score, and number of misclassifications for each architecture are as follows: DenseNet (D), EfficientNet (E), ResNet (R), and Xception (X).

Representation Type	**Accuracy** [%]				**Macro-F1** [%]				**Misclassified**			
	D	E	R	X	D	E	R	X	D	E	R	X
Wireframe	97.75	97.00	97.90	96.73	98.88	98.5	98.95	98.36	180	240	168	262
Shaded	97.88	96.08	98.18	97.78	98.94	98.03	99.09	98.89	170	314	146	178
Fast render	92.40	91.00	92.58	92.85	96.19	95.48	96.28	96.42	608	720	594	572
Render w/o bg.	92.13	88.78	92.03	92.58	96.05	94.37	96.01	96.28	630	898	638	594
Render with bg.	81.95	84.23	81.78	83.45	90.92	92.09	90.85	91.69	1444	1262	1458	1324

(Xception) achieved a slightly lower value of 92.58%. The lowest accuracy result, at 83.45%, was obtained for photorealistic images with a background (Xception). There is a clear correlation between image accuracy and photorealism - the closer an image is to photorealism, the lower the accuracy. As the aim of this work is to determine the graphical representation type that provides the best classification accuracy, the focus was placed on the two most effective types in search of a better solution.

4.1. Causes of missclassification

To investigate the causes of misclassification, we examined the problematic images. As ResNet architecture yielded the best results for both representation types, we will concentrate on its outcomes (Fig. 11).

The potential reasons for misclassification errors include:

- the presence of large circular holes in non-pipe components machining of the end of a component that creates features characteristic of another class, such as a rectangular cutout in the end of a round bar or rounding of a square bar
- camera positioning that causes important details to be hidden
- blending of parallel planes caused by identical shading color
- undetected edges caused by a small angle between surfaces

It was found that the number of misclassified images for each group was low, with only 26 views misclassified in both wireframe (168 misclassified overall) and shaded (146 misclassified overall) representations. This indicates that most of the misclassification errors were not solely due to the shape of the component and the camera setting, but to their combination with a given graphical representation type.

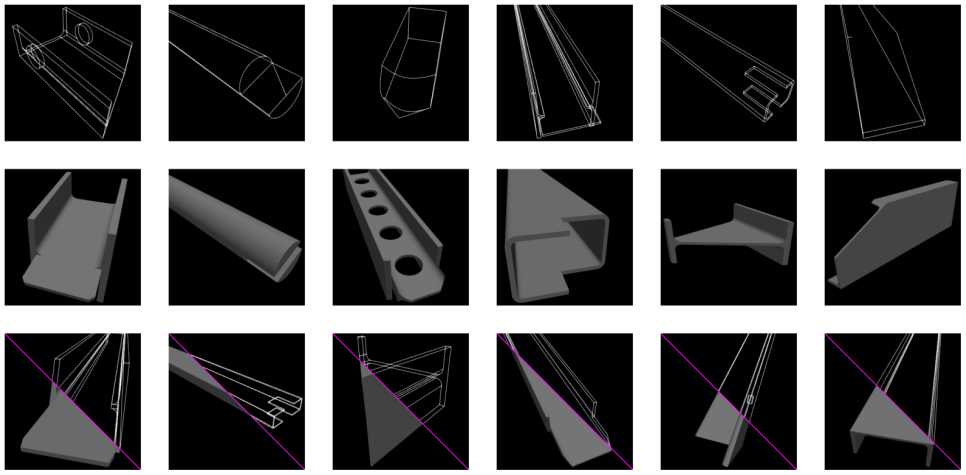

Figure 11. Examples of images that were misclassified using either wireframe representation (upper row), shaded representation (middle row), or both wireframe and shaded (lower row).

4.2. Additional types of graphical representation

Based on the results obtained in the first stage of the study, a decision was made to investigate the effect of anti-aliasing [23] (see Fig. 12) on the quality of classification by adding two more representation types:

- wireframe without anti-aliasing
- shaded without anti-aliasing

Table 2. Results obtained using 2 additional types.

Representation Type	Accuracy [%]				Macro-F1 [%]				Misclassified			
	D	E	R	X	D	E	R	X	D	E	R	X
Wireframe w/o AA	96.98	96.50	96.65	96.50	98.49	98.25	98.32	98.25	242	280	268	280
Shaded w/o AA	96.80	95.23	97.20	97.90	98.40	97.60	98.60	98.95	256	382	224	168

4.3. Complex representation types

Three new types, combining the features of wireframe and shaded representations, were proposed after it was inferred that there was a low overlap between views that were misclassified using these types of representation (see Fig. 14):

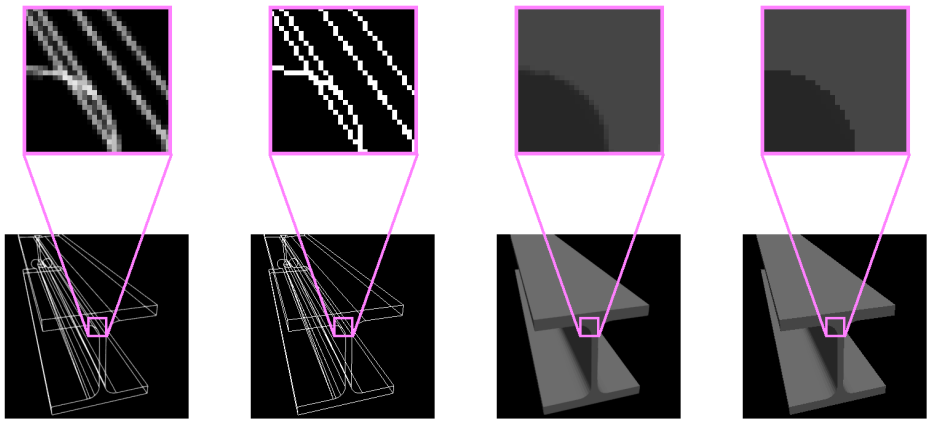

Figure 12. Differences between images generated with and without anti-aliasing.

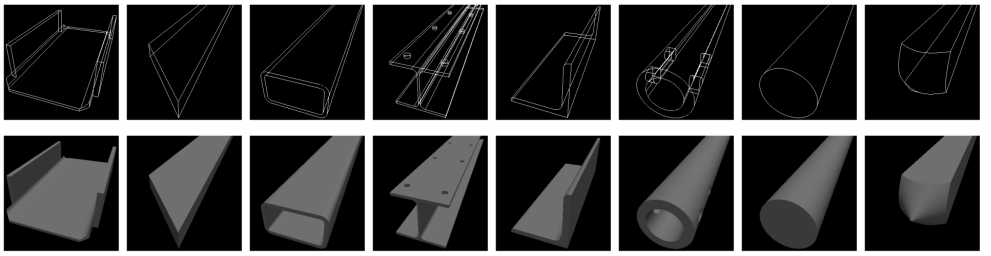

Figure 13. Two additional graphical representation type, wireframe without anti-aliasing (top row) and shaded without anti-aliasing.

- hidden
- overlay
- feature

The images were obtained by modifying the parameters of the Python script used for creating the 2 basic types.

4.4. Composite representation types

The architectures used in this work were designed for photo classification and utilize data saved in RGB format, with all color channels passed to the CNN in parallel. As wireframe and shaded produce grayscale images, each color channel contains the same information. To determine if this redundancy is necessary, three color channels were

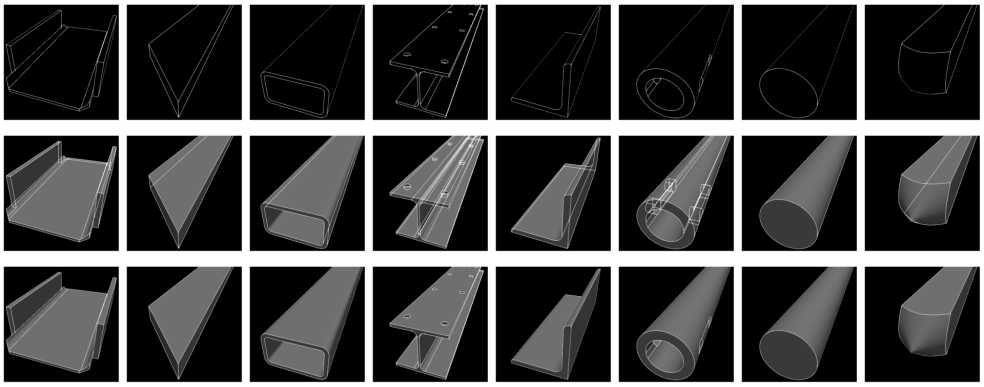

Figure 14. Three complex representation types. Hidden is similar in appearance to wireframe but does not contain lines obscured by the surface of the component. Overlay was created from superimposing wireframe on shaded - this was done not by merging the two images, but by changing the rendering parameters that caused all edges to be rendered offset above surfaces. The feature representation is similar to shaded, but with added feature edges.

Table 3. Results obtained using 3 complex representations.

Representation Type	**Accuracy** [%]				**Macro-F1** [%]				**Misclassified**			
	D	E	R	X	D	E	R	X	D	E	R	X
Hidden	96.50	95.55	96.40	96.85	98.25	97.77	98.2	98.43	280	356	288	252
Overlay	98.93	97.40	99.05	98.43	99.46	98.70	99.52	99.21	86	208	76	126
Feature	97.23	95.15	97.60	98.40	98.61	97.57	98.80	99.20	222	388	192	128

used as additional information transmitted to the neural network. For this purpose, four additional composite representation types have been developed. Each channel contains an image generated using a different method:

- composite 1: feature + wireframe + shaded
- composite 2: overlay + wireframe + shaded
- composite 3: overlay + feature + shaded
- composite 4: overlay + wireframe + feature

These new types are based on the two best basic representations (wireframe and shaded) and the two best composite representations (overlay and feature). Functions from the OpenCV library were used to create them [56]. Each image used in compositing is divided into three color channels - red (R), green (G), and blue (B). These channels contain the same values, as shown in the example of the shaded image in the

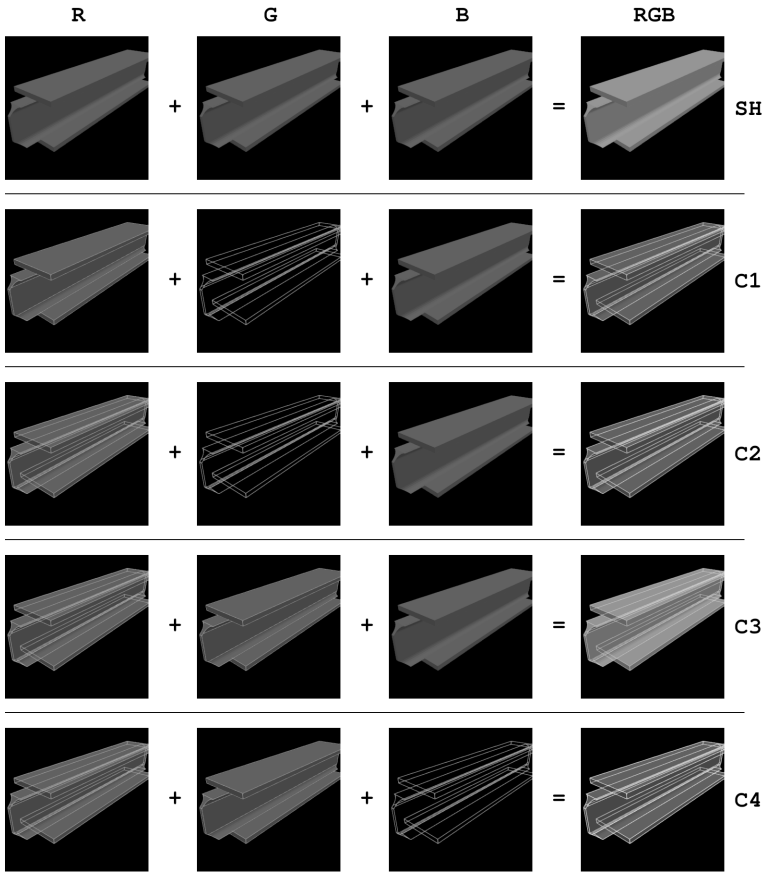

Figure 15. Separate color channels and an image combining them, in rows from the top: Shaded (SH), Composite 1 (C1), Composite 2 (C2), Composite 3 (C3) and Composite 4 (C4).

top row of Figure 15. A composite image is formed by merging three color channels, each obtained from a different input image.

The Overlay representation combined with DenseNet architecture produced the best results, with an accuracy of 99.05%. The Composite 1 representation, combined with ResNet, gave the second-best result with the accuracy of 98.93%, the same as for Overlay and DenseNet. No other composite types produced better results. Composite graphical representation types yielded better results for the EfficientNet and Xception architectures. For EfficientNet, Composite 4 achieved an accuracy of 98.25%, while for Xception, Composite 3 achieved an accuracy of 98.53%.

95% confidence interval was calculated for the best results, yielding ranges from 97.736% to 98.614% for shaded representation and ResNet architecture and from

98.685% to 99.415% for Overlay and ResNet architecture, allowing the assumption that overlay results are in fact better. In the case of Xception and EfficientNet architectures the confidence interval of the best of the simple representation was from 97.469% to 98.081% for the shaded representation and Xception architecture, while for the Xception and Composite 3 representation it ranged from 98.090% to 98.960%, also proving the advantage of this type of graphical representation.

Table 4. Results obtained using 4 composite representations.

Representation Type	**Accuracy [%]**				**Macro-F1 [%]**				**Misclassified**			
	D	E	R	X	D	E	R	X	D	E	R	X
C1	98.75	97.23	98.93	98.10	99.38	98.61	99.46	99.05	100	222	86	152
C2	98.83	97.75	98.60	98.08	99.41	98.88	99.30	99.04	94	180	112	154
C3	98.58	98.10	98.83	98.53	99.29	99.05	99.41	99.26	114	152	94	118
C4	98.75	98.25	98.58	98.23	99.37	99.13	99.29	99.11	100	140	114	142

Furthermore, the positive outcome achieved with the composite graphical representation types warrants exploring the potential for further modifications. It is possible for each of the color channels to display a distinct perspective of the classified component, which may enhance the accuracy of the classification (as demonstrated in MVCNN [65]). This method would utilize supplementary data stored in multiple views, without requiring the creation of a new, multi-view architecture. Composite images can be classified using a neural network architecture developed for single images. This approach is advantageous because methods developed for general image classification atract more research interest to researchers and new advances are more frequent.

Achieving a substantial decrease in misclassified images, despite the study's initial use of simple representations that already yielded an accuracy of over 98%, is a testament to the effectiveness of the proposed novel types. To further validate their performance, it would be beneficial to test the developed representations on a more diverse dataset, such as ModelNet40. Possible directions for further research include investigating why the composite representation types only yielded better results for EfficientNet and Xception architectures and testing the effect of camera angles on classification accuracy. In conclusion, we have established an inverse relationship between image photorealism and classification accuracy. The Overlay and Composite representations provided the best results, despite their appearance being less natural to the human eye than that of basic graphical representation types.

5. Conclusions

The problem of classification of objects presented in various images is well known and studied, but there are a wide range of aspects that have to be taken into account regarding specialized identification. The proposed study and methodology is a novel approach to analyze images based on the set of different image representation modes.

The study found that a simple graphical representation was superior in terms of accuracy and confidence in classifying 3D models, and that using an overlay representation reduced misclassification by 94% compared with a photo-realistic renderings with background. It may be assumed that using the most realistic images would result in the highest classification accuracy since the network architectures used were created to classify photographs. However, the obtained results indicate the opposite: the more photorealistic the image, the lower the classification accuracy. The advantage of simple representations over those that provide more realistic renderings can be attributed to the feature extraction process. Simple shading produces images with fewer gradients and complex textures, where surfaces typically have the same hue throughout, and gradients only occur on non-flat surfaces. The lower number of unique features in these images likely contributes to their higher accuracy.

Generating images using the VTK library has a major advantage over photorealistic renderings - it does not require complex configuration of lighting, materials, and parameters. Additionally, the generation of images takes only a fraction of a second. Furthermore, the VTK library does not require separate computer graphics software, making it easy to integrate with other applications. The findings indicate that selecting an appropriate graphical representation type can lead to achieving high classification accuracy, exceeding 99%, when using neural network architectures intended for photorealistic image classification. The overlay representation, which was developed for this study, resulted in a 94% reduction in misclassified images compared to the photorealistic image and a 48% reduction compared to the best of the basic types.

The study found that the graphical representation type most useful for image classification was one that appeared unnatural to the human eye. From the authors' perspective, this is the most important conclusion from the research conducted, suggesting the need to move away from an anthropocentric point of view in the research on convolutional neural networks. At the same time, it emphasizes the novelty of the presented research.

Future research could involve validating the developed graphical representation types using a larger and more diverse dataset of 3D models. It is worth considering applying the conclusions to improve photos or medical image classification or to check the possibility of classifying photographs using a model trained only on computer-generated images. Another follow-up could be to investigate whether similar results are obtained for different image classification methods, such as Vision Transformers.

Acknowledgment

This research was partially funded by the Ministry of Education and Science, grant no 0311/SBAD/0737 and "Applied Doctorate" program, DWD/5/0545/2021, 23.12.2021.

Data availability

The data presented in this study is available on request from the corresponding author.

Conflicts of interests

The authors declare no conflicts of interest.

References

[1] 3D Systems Inc., Stereolithography Interface Specification, Valencia, CA, June 1988.

[2] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y. and Zheng, X., "TensorFlow: A System for Large-Scale Machine Learning" in Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI '16). 2016.

[3] Asmail, C., "Bidirectional scattering distribution function (BSDF): a systematized bibliography" in J. Res. Natl. Inst. Standards Technol., vol. 96, 1991, pp. 215.

[4] Barutcuoglu, Z. and Decoro, C., "Hierarchical Shape Classification Using Bayesian Aggregation" in Proceedings - IEEE International Conference on Shape Modeling and Applications, SMI 2006, 2006, pp. 44-44.

[5] Blender Online Community, "Blender – a 3D modelling and rendering package". Amsterdam, Blender Institute, 2018.

[6] Burley, B., "Physically-Based Shading at Disney" in SIGGRAPH 2012, 2012.

[7] Caruana, R., Lawrence, S. and Giles, L., "Overfitting in Neural Nets: Backpropagation, Conjugate Gradient, and Early Stopping" in Proceedings of the 13th International Conference on Neural Information Processing Systems, (NIPS 2000), Denver, 2000, pp. 402-408, doi: 10.5555/3008751.3008807.

[8] Catmull, E., "A subdivision algorithm for computer display of curved surfaces", PhD thesis, University of Utah, 1974.

[9] Catmull, E. and Clark, J., "Recursively generated B-spline surfaces on arbitrary topological meshes", Computer-Aided Design, vol. 10 (6), 1978, pp. 350. doi:10.1016/0010-4485(78)90110-0.

[10] Chandler, J., "Effective application of automated digital photogrammetry for geomorphological research", Earth Surf. Process. Landforms, vol. 24, 1999, pp. 51-63.

[11] Chen, D., Tian, X., Shen, Y. and Ouhyoung, M., "On visual similarity based 3d model retrieval", Comput. Graph. Forum, vol. 22(3), 2003, pp. 223–232.

[12] Chen, X., Liang, C., Huang, D., Real, E., Wang, K., Liu, Y., Pham, H., Dong, X., Luong, T. and Hsieh, C., "Symbolic discovery of optimization algorithms", arXiv preprint arXiv:2302.06675, 2023.

[13] Chicco, D. and Jurman, G., "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation", BMC Genomics, vol. 21, 2020, doi:10.1186/s12864-019-6413-7.

[14] Chinchor, N. and Sundheim, B. M., "MUC-5 evaluation metrics" in Fifth Message Understanding Conference (MUC-5), Baltimore, 1993, pp. 69-78.

[15] Chollet, F., "Xception: Deep Learning with Depthwise Separable Convolutions", 2017, pp. 1800-1807, doi:10.1109/CVPR.2017.195.

[16] Chollet, F., "Deep Learning with Python", Manning Publications, 2017, ISBN 9781617294433.

[17] Conlan, C., "The Blender Python API: Precision 3D modeling and add-on development", Apress, USA, 2017, ISBN 1484228022, 9781484228029.

[18] Cortes, C. and Vapnik, V., "Support-vector networks" in Machine Learning, vol 20(3), 1995, pp. 273–297, doi:10.1007/BF00994018.

[19] Descartes, R., "Discourse on Method, Optics, Geometry, and Meteorology", Translated by Paul J. Oscamp (Revised ed.). Indianapolis, IN: Hackett Publishing, 2003, ISBN 978-0-87220-567-3. OCLC 488633510.

[20] Ding, B., Zhang, L., He, Y. and Qin, J., "3D Shape Classification Based on Global and Local Features Extraction with Collaborative Learning", The Visual Computer, vol. 40, no. 9, pp. 1-13, 2023, doi: 10.1007/s00371-023-03098-0.

[21] Edl, M., Mizerák, M. and Trojan, J., "3d laser scanners: history and applications", Acta Simulatio, vol. 4, 2018, pp. 1-5.

[22] Euler, L., "Elementa doctrinae solidorum" in Novi Commentarii Academiae Scientiarum Petropolitanae, 1758, pp.109–140.

[23] Freeman, H., "Computer processing of line drawing images" in ACM Computing Surveys, vol. 6(1), 1974, pp. 57–97, doi:10.1145/356625.356627. S2CID 18962414.

[24] Fukushima, K., "Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position" in Biological Cybernetics, vol. 36(4), 1980, pp. 193–202, doi:10.1007/BF00344251.

[25] Guevarra, E., "Modeling and Animation Using Blender: Blender 2.80: The Rise of Eevee", 2020, doi:10.1007/978-1-4842-5340-3.

[26] He, K., Zhang, X., Ren, S. and Sun, J., "Deep Residual Learning for Image Recognition" in IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Las Vegas, USA: IEEE, 2016, pp. 770–778, doi:10.1109/CVPR.2016.90. ISBN 978-1-4673-8851-1.

[27] Heisele, B., Kim, G. and Meyer, A. J., "Object Recognition with 3D Models" in Proceedings of the British Machine Vision Conference, London, England. 2009, pp. 1-11, doi:10.5244/C.23.29.

[28] Huang, G., Liu, Z., Van Der Maaten, L. and Weinberger, K. Q., "Densely Connected Convolutional Networks" in IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Honolulu, IEEE, 2017, pp. 2261–2269, doi:10.1109/CVPR.2017.243. ISBN 978-1-5386-0457-1.

[29] Intel Corporation (2019), "Open Image Denoise", Available: https://www.openimagedenoise.org, last accessed: March 4, 2025.

[30] International Organization for Standardization, Industrial automation systems and integration – Product data representation and exchange – Part 1: Overview and fundamental principles (ISO Standard No. 10303-1:1994), 1994.

[31] Jaros, M., Říha, L., Strakoš, P., Karasek, T., Vašatová, A., Jarošová, M. and Kozubek, T., "Acceleration of Blender Cycles Path-Tracing Engine Using Intel Many Integrated Core Architecture", 2015, pp. 86-97, doi:10.1007/978-3-319-24369-6_7.

[32] Kajiya, J. T., "The rendering equation" in Proceedings of the 13th annual conference on Computer graphics and interactive techniques, 1986, pp. 143-150.

[33] Kanezaki, A., Matsushita, Y. and Nishida, Y., "RotationNet: Joint Object Categorization and Pose Estimation Using Multiviews from Unsupervised Viewpoints", 2018, pp. 5010-5019, doi:10.1109/CVPR.2018.00526.

[34] Khalfaoui, H. I. and Kesselheim, S., "Learnable polynomial, trigonometric, and tropical activations". 10.48550/arXiv.2502.01247, 2025.

[35] Khorolska, K., Bebeshko, B., Desiatko, A. and Lazorenko, V., "3D Models Classification with Use of Convolution Neural Network" in proceedings of Information Technology and Implementation 2021, vol. 3179, 2021.

[36] Kolesnikov, A., Beyer, L., Zhai, X., Puigcerver, J., Yung, S., Gelly, S. and Houlsby, N., "Big Transfer (BiT): General Visual Representation Learning", 2020, doi:10.1007/978-3-030-58558-7_29.

[37] Koprucu, N., Nigam, M. S., Xu, S., Abere, B., Dominici, G., Rodriguez, A., Vadgam, S., Inal, B. and Tono, A., "DC3DO: Diffusion Classifier for 3D Objects", arXiv preprint arXiv:2408.06693, 2024, doi:10.48550/arXiv.2408.06693.

[38] LeCun, Y., Huang, F. J. and Bottou, L., "Learning methods for generic object recognition with invariance to pose and lighting" in Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit, 2004, pp. 11097-11104.

[39] Li, J., Sun, Q., Chen, K., Cui, H., Huangfu, K. and Chen, X., "3D Large-Scale Point Cloud Semantic Segmentation Using Optimal Feature Description Vector Network: OFDV-Net" in. IEEE Access, 2020, pp. 1-1, doi: 10.1109/ACCESS.2020.3044166.

[40] Li, W., Mac, G., Tsoutsos, N., Gupta, N. and Karri, R., "Computer aided design (CAD) model search and retrieval using frequency domain file conversion" in Additive Manufacturing, vol. 36, 2020, doi:10.1016/j.addma.2020.101554.

[41] Liu, Z., Mao, H., Wu, C.-Y., Feichtenhofer, C., Darrell, T. and Xie, S., 'A ConvNet for the 2020s' in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognition. (CVPR), 2022, pp. 11966–11976.

[42] Liu, Z., Zhang, Y., Jian, G. and Wang, S., "VFMVAC: View-Filtering-Based Multi-View Aggregating Convolution for 3D Shape Recognition and Retrieval", Pattern Recognition, vol. 129, pp. 108774, 2022, doi: 10.1016/j.patcog.2022.108774.

[43] Liu, Z., Hao, Z., Han, K., Tang, Y. and Wang, Y., "GhostNetV3: Exploring the Training Strategies for Compact Models". arXiv:2404.11202, 2024.

[44] Liu, Z., Song, W., Tian, Y., Ji, S., Sung, Y., Wen, L., Zhang, T., Song, L. and Gozho, A., "VB-Net: Voxel-Based Broad Learning Network for 3D Object Classification" in Applied Sciences, vol. 10(19), 2020, pp. 6735, doi: 10.3390/app10196735

[45] Mandelli, L. and Berretti, S., "CAD 3D Model classification by Graph Neural Networks: A new approach based on STEP format", arXiv:2210.16815v1, 2022.

[46] Meng, Z., Zhang, J., Yang, C., Zhan, Z., Zhao, P. and Wang, Y., "DiffClass: Diffusion-Based Class Incremental Learning", arXiv preprint arXiv:2403.05016, 2024, doi:10.48550/arXiv.2403.05016.

[47] Nagel, R., Braithwaite, W. and Kennicott, P., Initial Graphics Exchange Specification IGES, Version 1.0, National Bureau of Standards, Washington DC, NBSIR 80-1978, 1980.

[48] Yamada, A., Pickering, M., Jeannin, S., Cieplinski, L., Ohm, J.-R. and Editors, M., Eds., MPEG-7 Visual Part of Experimentation Model Version 8.0, Oct. 2000, ISO/IEC JTC1/SC29/WG11/N3673.

[49] Nie, W., Wang, Y., Song, D. and Li, W., "3D Model Retrieval Based on a 3D Shape Knowledge Graph" in IEEE Access, 2020, pp. 1-1, doi:10.1109/ACCESS.2020.3013595.

[50] Omar, K., Sakr, R. and Alrahmawy, M., "An ensemble of CNNs with self-attention mechanism for DeepFake video detection" in Neural Computing and Applications, vol. 36, 2023, doi:10.1007/s00521-023-09196-3.

[51] Open Cascade S.A.S.U. Open Cascade Technology. (2024) Available: https://www.opencascade.com. Last accessed: March 4, 2025.

[52] Opitz, J., "A Closer Look at Classification Evaluation Metrics and a Critical Reflection of Common Evaluation Practice", Transactions of the Association for Computational Linguistics, vol. 12, 2024, pp. 820-836, doi:10.1162/tacl_a_00675.

[53] Paviot, T., 2023, "python-OCC", https://github.com/tpaviot/pythonocc-core, Last accessed: March 4, 2025.

[54] Pham, H., Dai, Z., Xie, Q. and Le, Q. V., "Meta Pseudo Labels" in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), USA, 2021, pp.11557–11568.

[55] Phung, B. and Spear, A., "A voxel-based remeshing framework for the simulation of arbitrary three-dimensional crack growth in heterogeneous materials" in Eng. Fract. Mech. vol. 209, 2019, pp. 404–422.

[56] Pulli, K., Baksheev, A., Kornyakov, K. and Eruhimov, V., "Real-time Computer Vision with OpenCV" Queue. Vol. 10(4), 2012, pp.40:56, doi:10.1145/2181796.2206309.

[57] Robertson, M. A., Borman, S. and Stevenson, R. L., "Estimation-theoretic approach to dynamic range enhancement using multiple exposures" in Journal of Electronic Imaging, vol. 12(2), 2003, pp. 220–228, doi:10.1117/1.1557695.

[58] Rodríguez, J. and Lozano, J., "Repeated stratified k-fold cross-validation on supervised classification with naive Bayes classifier: An empirical analysis" in Fifth Spanish Workshop on Data Mining and Learning, TAMIDA 2007, 2007, pp. 65–74.

[59] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C. and Fei-Fei, L., "ImageNet Large Scale Visual Recognition Challenge". IJCV

[60] Sarkar, K., Varanasi, K. and Stricker, D., "Trained 3d models for CNN based object recognition" in Proceedings of the 12th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, 2017, pp. 130–137, doi:10.5220/0006272901300137.

[61] Schroeder, W., Martin, K. and Lorensen, B., "The Visualization Toolkit", 4th ed., Kitware, 2006, ISBN 978-1-930934-19-1.

[62] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R., "Dropout: A Simple Way to Prevent Neural Networks from Overfitting" in Journal of Machine Learning Research, vol. 15, 2014, pp. 1929-1958.

[63] Srivastava, R. K., Greff, K. and Schmidhuber, J. (2015). Highway Networks. arXiv:1505.00387.

[64] Statista. "Image Recognition - Worldwide". 2025. Available: https://www.statista.com/outlook/tmo/artificial-intelligence/computer-vision/image-recognition/worldwide, last accessed March 4, 2025.

[65] Su, H., Maji, S., Kalogerakis, E. and Learned-Miller, E. G., "Multi-view convolutional neural networks for 3d shape recognition", 2015, doi:10.1109/ICCV.2015.114.

[66] Szegedy, C., Liu, W., Jia, Y., Permanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. and Rabinovich, A., "Going deeper with convolutions" in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 1-9, doi:10.1109/CVPR.2015.7298594.

[67] Tan, M. and Le, Q., "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks" in Proceedings of the 36th International Conference on Machine Learning, vol. 97, 2019, pp. 6105-6114.

[68] Tan, M. and Le, Q., "EfficientNetV2: Smaller Models and Faster Training". International Conference on Machine Learning, 2021, pp. 10096-10106.

[69] Touvron, H., Vedaldi, A., Douze, M. and Jégou, H., "Fixing the train-test resolution discrepancy: FixEfficientNet", arXiv:1906.06423, 2020.

[70] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. and Polosukhin, I., "Attention is all you need", Advances in Neural Information Processing Systems, vol. 30, 2017.

[71] Venners, B., "The Making of Python: A Conversation with Guido van Rossum", Artima Developer 2003.

[72] Wortsman, M., Ilharco, G., Yitzhak Gadre, S., Roelofs, R., Gontijo Lopes, R., Morcos, A., Namkoong, H., Farhadi, A., Carmon, Y. and Kornblith, S., "Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time", arXiv preprint arXiv:2203.05482, 2022.

[73] Wu, Q., Ritchie, D., Savva, M. and Chang, A. X., "Generalizing Single-View 3D Shape Retrieval to Occlusions and Unseen Objects", arXiv, 2023.

[74] Xian, Z., Huang, R., Towey, D. and Yue, C., "Convolutional Neural Network Image Classification Based on Different Color Spaces," in Tsinghua Science and Technology, vol. 30, no. 1, 2025, pp. 402-417, doi: 10.26599/TST.2024.9010001.

[75] Xie, Q., E. Hovy, Luong, M.-T. and Le, Q., "Self-training with Noisy Student improves ImageNet classification" in Conference on Computer Vision and Pattern Recognition, 2019, pp. 10684–10695.

[76] Xie, Y., Zhang, Y., Gong, M., Tang, Z. and Han, C., "MGAT: Multi-view graph attention networks" in Neural Netw., vol. 132(12), 2020, pp. 180–189.

[77] Yu, J., Wang, Z., Vasudevan, V., Yeung, L., Seyedhosseini, M. and Wu, Y., "Coca: Contrastive captioners are image-text foundation models", arXiv:2205.01917, 2022.

[78] Zaal, G., "Poly Haven", Available: https://polyhaven.com, 2021. Last accessed: March 4, 2025.

[79] Zhai, X., Kolesnikov, A., Houlsby, N. and Beyer, L., "Scaling vision transformers", arxiv:cs.CV/2106.04560, 2021.

[80] Zeng, H., Zhao, T., Cheng, R., Wang, F. and Liu, J., (2021). "Hierarchical Graph Attention Based Multi-View Convolutional Neural Network for 3D Object Recognition" in IEEE Access, 2021, pp. 1-1, doi:10.1109/ACCESS.2021.3059853.

[81] Zoph, B. and Le, Q. V., "Neural architecture search with reinforcement learning", arXiv preprint arXiv:1611.01578, 2016.

Received 05.09.2025, Accepted 24.02.2026