UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

3D SHAPE RETRIEVAL USING NEURAL NETWORKS

TESIS PARA OPTAR AL GRADO DE
DOCTOR EN COMPUTACIÓN

ARNIEL LABRADA DENIZ

PROFESOR GUÍA:
BENJAMÍN BUSTOS CÁRDENAS

MIEMBROS DE LA COMISIÓN:
IVÁN SIPIRÁN MENDOZA
JUAN BARRIOS NÚÑEZ
THEOHARIS THEOHARIS

SANTIAGO DE CHILE
2024

Resumen

RECUPERACIÓN DE OBJETOS 3D USANDO REDES NEURONALES

En los últimos años se han observado avances significativos en las tareas de recuperación, clasificación y segmentación de modelos 3D. Las representaciones tradicionales como las nubes de puntos y las mallas poligonales si bien son adecuadas para la renderización, presentan desafíos para las tareas mencionadas debido a su complejidad y redundancia. Esta tesis se enfoca en varias tecnicas para la representación de los modelos 3D asi como diferentes técnicas de recuperación de los mismos, centrándose en la recuperación intermodal utilizando image views.

Comenzamos nuestra investigación utilizando técnicas de representación de nivel medio, como bag-of-words combinado con features tradicionales como por ejemplo filtros de Gabor. Luego, avanzamos hacia métodos de deep learning para manejar modelos 3D representados como conjuntos de image views. Proponemos una arquitectura novedosa, que combina redes neuronales convolucionales (CNN) y autoencoders para calcular embeddings de los modelos 3D a partir de image views, con el objetivo de capturar información semántica para la evaluación de similitudes.

Ampliamos el trabajo a la recuperación de modelos 3D basados en imágenes, revelando desafíos para encontrar un espacio conjunto para embeddings de imágenes y modelos 3D. Finalmente proponemos una arquitectura end-to-end para aprender a comparar imágenes y modelos 3D directamente.

Los objetivos de la tesis incluyen el desarrollo de métodos basados en redes neuronales para la recuperación de formas 3D y la recuperación de formas 3D intermodal, evaluación comparativa, establecimiento de métricas, comparación con métodos tradicionales y pruebas en escenarios reales.

Abstract

Significant advancements have been witnessed in 3D model retrieval, classification, and segmentation tasks in recent years. While suitable for rendering, traditional representations like point clouds and polygon meshes present challenges for the tasks above due to their complexity and redundancy. This thesis delves into various 3D model representations and retrieval techniques, focusing on cross-modal retrieval using image views.

We begin our research using mid-level representation techniques such as bag of words combined with hand-engineered features. Then, we progress towards deep learning methods to handle 3D models represented as sets of image views. We propose a novel architecture, combining Convolutional Neural Networks (CNNs) and Autoencoders to compute 3D model embeddings from image views, aiming to capture semantic information for similarity assessment.

We extend the work to image-based 3D model retrieval, revealing challenges in finding a joint space for image and 3D model embeddings. Finally, we propose an end-to-end architecture to learn how to directly compare images and 3D shapes.

The thesis objectives include developing neural network-based methods for 3D shape retrieval and cross-modal 3D shape retrieval, benchmarking, metric establishment, comparison with traditional methods, and real scenario testing.

TABLE OF CONTENT

List of Tables

List of Figures

Chapter 1

INTRODUCTION

Many advances have been made in tasks like 3D model retrieval, 3D model classification, and 3D model segmentation during the last few years. Typical 3D representations such as point clouds, voxels, and polygon meshes are primarily suitable for rendering purposes. At the same time, their use for cognitive processes (retrieval, classification, segmentation) is limited due to their high redundancy and complexity. This work explores different techniques for 3D model representation, 3D model retrieval, and cross-domain 3D model retrieval using image views to represent the 3D models.

We start our research using mid-level representation techniques, such as the bag of words combined with hand-engineered features extracted with image descriptors. Then, we use this representation in the 3D model retrieval task. Subsequently, we expand our research to deep representation using deep learning techniques to propose a learning architecture to handle 3D models represented as sets of image views as input. This architecture combines standard architectures, like Convolutional Neural Networks and Autoencoders, for computing 3D model embeddings using sets of image views extracted from the 3D models. We aim to represent a 3D model as a vector with enough information to substitute the 3D model for high-level tasks. Since this vector is a learned representation that tries to capture the relevant characteristics of a 3D model, we show that the embedding representation conveys semantic information that helps deal with the similarity assessment of 3D objects. We compare our proposed embedding technique with state-of-the-art techniques for 3D model retrieval using the ShapeNet dataset. We show that the embeddings obtained with our proposed architecture allow us to acquire a high effectiveness score in both normalized and perturbed versions of the ShapeNet dataset.

We also extend our research to other subtasks like sketch-based 3D model retrieval and image-based 3D model retrieval. In the last case, we use our proposed embedding-based model to compute embeddings from images and 3D shapes and then perform the retrieval. This approach is usual in other domains, such as image retrieval, because pre-trained networks already give a good representation for metric learning. We evaluated this approach for image-based 3D shape retrieval, and we observed poor performance. This result indicates that metric learning struggles to find a suitable joint space for image embeddings and 3D models embeddings. Inspired by these insights, we propose a second model that is an end-to-end architecture. This end-to-end model directly learns how to compare images and 3D shapes by guiding the feature extraction to the construction of common feature space. Our experimental evaluation shows that the end-to-end architecture is more effective than the embedding-based model for the retrieval task.

1.1 Information Retrieval

Information Retrieval (IR) is an interdisciplinary field that focuses on efficiently searching and retrieving relevant information from large datasets. This discipline is essential in various applications, including 3D object retrieval, where the goal is to find relevant three-dimensional models in a database based on user needs and queries. Before delving into 3D object retrieval, it is crucial to understand the fundamental concepts of Information Retrieval.

User Information Need: IR begins with a user's need for information. Users seek information to fulfill their goals, whether finding answers to specific questions, exploring topics, or retrieving relevant resources.

Types of Queries: Queries are users' expressions to search for information. They can be text queries, queries based on visual content, or, in the context of 3D object retrieval, queries based on geometric or structural features.

Relevance: Relevance is a key concept in IR. It refers to how a retrieved document or object satisfies the user's information needs. Evaluating relevance is essential to determine the effectiveness of a retrieval system.

Ranking: In IR, search results are presented in an order that reflects their perceived relevance. IR systems use ranking algorithms to rank retrieved documents or objects based on their estimated relevance.

Similarity: Similarity measures the closeness between the user's query and the documents or objects in the database. In 3D object retrieval, similarity can be based on geometric features such as shape and structure.

Descriptor: A descriptor is a structured or semantic representation of a document or object. In 3D object retrieval, descriptors may include metadata, geometric features, and other relevant attributes.

Index: IR systems use indexes to expedite searching and retrieval. An index is a data structure that enables quick access to relevant documents or objects based on search terms or query features.

Search: Searches involve retrieving relevant information from a database using queries. Search techniques vary depending on the type of information and system requirements.

In the context of 3D object retrieval, these Information Retrieval concepts are crucial for designing effective systems that can retrieve relevant three-dimensional models for users. By understanding these concepts, we can develop algorithms and techniques that enhance the accuracy and efficiency of 3D object retrieval in practical applications.

1.1.1 3D shape retrieval

The problem of 3D shape retrieval refers to searching for and retrieving a specific three-dimensional object or relevant 3D models from a collection of 3D models. To further understand the problem, we will describe some of the basic concepts of information retrieval adapted to our 3D shape Retrieval proposal. The user can be a researcher, a designer, or any entity interested in finding specific 3D models. Then, the query is usually a 3D object, but in other more specific tasks like image-based 3D shape retrieval, the goal is to retrieve the 3D objects more similar to a query image.

We use several standard benchmarks for 3D shape retrieval and classification for all our experiments. These benchmarks have their own train/validation/test split, and each 3D model is labeled with their respective class. This means that given a specific query, an object in the dataset is relevant if labeled with the same class as the query.

Finally, to evaluate the performance of our retrieval system, we use several metrics based on a ranked list like Precision-Recall plot (PR), Mean Average Precision MAP, Nearest Neighbor NN, First Tier FT, Second Tier ST, and Discounted Cumulative Gain DCG.

1.2 Objectives

In this section, we present the general and specific objectives of the proposed thesis.

1.2.1 General objective

The general objective of the proposed thesis is to develop methods to solve the problem of 3D shape retrieval using artificial neural network techniques, as well as to compare the performance of these methods with that of the traditional methods of the area.

1.2.2 Specific objectives

- Develop methods to solve the 3D shape retrieval (cross-modality 3D shape retrieval, partial 3D shape retrieval) tasks using artificial neural networks.
- Select and implement the current methods that perform best in the area of 3D shape retrieval (cross-modality 3D shape retrieval, partial 3D shape retrieval).
- Establish standard benchmarks for experimentation.
- Establish metrics to measure the performance of the methods
- Make comparisons between the proposed and most important methods in the area.
- Test the proposed methods in real scenarios and refine them to improve the results if necessary.

1.3 Hypothesis

By using artificial neural networks combined with techniques for the representation of 3D models, it is possible to develop methods to solve 3D shape retrieval tasks that improve the performance of current methods and solve some open problems in the areas of cross-modal 3D shape retrieval and partial 3D shape retrieval.

1.4 Research problem

Since the outstanding results obtained by AlexNet [42] for image classification in 2012, the architectures of artificial neural networks, specifically Convolutional Neural Networks (CNNs) [45], have been continuously improving to solve visual computing tasks. Especially in the image processing field, CNNs has been used to solve the most relevant tasks, outperforming the results obtained with the area's standard techniques. These tasks include retrieval, classification, segmentation, and computation of image embeddings, among others.

Due to this remarkable performance, researchers have extended the use of CNNs to other fields, such as 3D model processing. In recent years, some works propose using these networks to solve classification and retrieval of 3D models with excellent results [37]. However, deep learning in the field of 3D models is still a relatively under-researched topic, and some 3D model tasks, such as computing 3D model embeddings, have received little to no attention.

textcolorblueBy researching these topics, we aim to create, help, or enhance several real-life applications related to 3D data. For instance, two applications motivated this research.

The first consists of automatically adding the existing metadata from an image in the semantic web to new 3D shapes in the semantic web. To accomplish these, we must match 3D models with images to make this process automatic, enhancing the metadata for 3D shapes on the web.

The second task is to create 3D catalogs of museums from existing 2D catalogs. Once again, the goal is to automatically match 3D models with images so that, given a 3D shape, we can extract all the pertinent information from the existing 2D catalog by finding the image more similar to the 3D shape.

We also worked on other real-life applications during our research. Precisely, in the work “A sketch-aided retrieval approach for incomplete 3D objects”. This work defines an appropriate workflow for content-based retrieval of 2D image data from incomplete 3D objects. The workflow is built around a human-in-the-loop approach, allowing experts to provide sketch-aids for adding missing shape information, query weighting, and visual result comparison. Domain experts can accurately estimate missing parts of Cultural Heritage objects. The basic idea of sketch-aids is to allow users to create additional object structures, filled by a texture inpainting step, which serves as input for content-based retrieval. This work was applied to archaeological object comparison with promising results.

1.4.1 3D shape retrieval

In this work, we propose an artificial neural network architecture for computing 3D model embeddings using sets of image views extracted from them. For this purpose, we first render a set of image views from the 3D models. We propose an artificial neural network module to process these image view sets to consume them as input. Our proposed architecture combines this module with other standard artificial neural network architectures, like CNNs and Autoencoders, to obtain the 3D model embeddings. This proposal simultaneously learns features from the whole set of views instead of using the standard approach in the field to combine the features from each view by incorporating view pooling layers [93]. This traditional approach needs to compute each view’s features separately before combining them using the view pooling layer, thus making the training and inference of these networks a time and resource-demanding process. On the other hand, our approach learns features directly from a single tensor with all the views. Besides, we do not need to specify an order for the image views. We will show that our proposed architecture is significantly more time and resource-efficient than the standard approach.

Since 3D models are a complex and space-consuming type of data, their processing is often challenging. However, our proposal goal is to compile into single vectors as much information as possible of the 3D models. We can substitute the 3D models with these vectors in further tasks such as retrieval, cross-modal retrieval, classification, segmentation, and others.

We present five main contributions:

- We propose a convolutional architecture that uses image views as input for computing 3D model embeddings by combining different deep learning architectures and imposing restrictions on the network to improve the quality of the embeddings.
- We use a network architecture that does not require a view pooling layer, making our network faster and easier for training and inference computing.
- We propose techniques for selecting the image views taken from the 3D model as input for our architecture.
- We analyze the performance of different convolutional architectures for 3D model embedding.
- We study the relationship between the quality and the cost of computing the 3D model embeddings as the number of image views increases by changing the numbers of image views $(4, 8, 16)$ to represent the models, and we also evaluate two different techniques for selecting the image views.

Finally, we show how our proposed 3D shape embedding method can benefit 3D model retrieval. To achieve this goal, after we compute the vector representation of the 3D models, we use the cosine distance as the similarity metric between the vectors. With this methodology, we show that we can obtain competitive results compared with the state-of-the-art techniques for the 3D model retrieval task.

1.4.2 Image-based 3D shape retrieval

3D model retrieval is the problem of searching for an object in a 3D model collection. Instead of using a textual query that describes what the user needs, we are interested in the case where the query is an object similar to the 3D models in the collection. For example, in traditional 3D model retrieval, the search starts with a 3D shape as a query object. While this is a common approach in multimedia information retrieval, known as query-by-example, there could be a contradiction in the assumption that the user has available a 3D shape for making the query. Indeed, the query-by-example approach supposes that there is a mechanism to obtain or search the 3D query object before using the retrieval algorithm. From the user perspective, for many practical applications of 3D object retrieval, it would be easier if the search algorithm receives an image as input. Thus, a relevant problem is finding a 3D model given an image that depicts it, which we call "image-based 3D shape retrieval".

Applying deep learning techniques for solving the 3D model retrieval problem is a current interesting trend, as this type of techniques has shown promising results in several computer vision tasks during the last decade [37, 34]. So, the question is how to train a deep artificial neural network capable of finding relevant 3D objects given an image as a query. In a deep learning context, one challenge is to have good representations for both the query image and relevant 3D models such that these representations share a common place in the feature space (also known as the latent space). To solve this, we need to find an appropriate artificial neural network architecture that allows us to train it so that it learns to match images with 3D objects effectively.

Another challenge related to this problem is the data. Images and 3D objects are complex data types. In addition, these data may be subject to several types of transformations (scale, translation, orientation, mirroring, noise, etc.), so there is a crucial need to develop retrieval algorithms that are robust to these transformations. Moreover, in our case, the retrieval algorithms must tackle the invariance problem in both domains (image and 3D shape) simultaneously, making the search problem even harder than when the user uses the same type of object as in the collection for processing a query.

In this work, we study and propose deep learning models for image-based 3D shape retrieval. The first model is based on computing embeddings for both the 3D shapes and the images, which can be used to implement the retrieval phase. In this model, we use an image-view approach for representing the 3D shapes. However, the experimental evaluation of this model shows that it has poor performance in solving the image-based 3D shape retrieval problem. This result indicates that metric learning struggles to find a suitable joint space for image embeddings and 3D model embeddings. Therefore, we need to explore novel architectures for improving the effectiveness of the retrieval.

The insights obtained during the evaluation of our first model led us to propose a second model, which is an end-to-end architecture that directly learns how to match images and 3D shapes. We show that the development of this second model requires us to present solutions to a number of technical difficulties, as each data object is represented as a multichannel data source but with (most probably) a different number of channels. We explain how to tackle all these difficulties. Finally, we conduct an experimental evaluation with this second model and show that the end-to-end architecture is far more effective than the embedding-based architecture for the image-based 3D shape retrieval task.

Our contributions in this field are:

- We propose a deep learning architecture for image-based 3D shape retrieval based on computing embeddings of the images and 3D shapes.
- We propose an end-to-end architecture for image-based 3D shape retrieval.
- We present an experimental evaluation of both proposed retrieval methods, concluding that the end-to-end architecture is a more effective model for the retrieval task.

1.5 Publications

Along with our research work, we contribute to the scientific community the following papers:

- A convolutional architecture for 3D model embedding using image views. Arniel Labrada, Benjamin Bustos, and Ivan Sipiran. The Visual Computer, pages 1–15, 2023 [43].
- HateU at SemEval-2022 Task 5: Multimedia Automatic Misogyny Identification. Aymé Arango, Jesus Perez Martin, and Arniel Labrada. In Guy Emerson, Natalie Schluter, Gabriel Stanovsky, Ritesh Kumar, Alexis Palmer, Nathan Schneider, Siddharth Singh, and Shyam Ratan, editors, Proceedings of the 16th International Workshop on Semantic Evaluation, SemEval@NAACL 2022, Seattle, Washington, United States, July 14-15, 2022, pages 581–584 [5].
- SHREC 2021: Retrieval of cultural heritage objects. Ivan Sipiran, Patrick Lazo, Cristian Lopez, Milagritos Jimenez, Nihar Bagewadi, Benjamin Bustos, Hieu Dao, Shankar Gangisetty, Martin Hanik, Ngoc-Phuong Ho-Thi, Mike Holenderski, Dmitri Jarnikov, Arniel Labrada, Stefan Lengauer, Roxane Licandro, Dinh Huan Nguyen, Thang-Long Nguyen-Ho, Luis A. Pérez Rey, Bang-Dang Pham, Reinhold Preiner, Tobias Schreck, Quoc-Huy Trinh, Loek Tonnaer, Christoph von Tycowicz, and The-Anh Vu-Le. Comput. Graph.,100:1–20, 2021 [88].
- A sketch-aided retrieval approach for incomplete 3D objects. Stefan Lengauer, Alexander Komar, Arniel Labrada, Stephan Karl, Elisabeth Trinkl, Reinhold Preiner, Benjamin Bustos, and Tobias Schreck. Comput. Graph., 87:111–122, 2020 [49].
- Sketch-Aided Retrieval of Incomplete 3D Cultural Heritage Objects. Stefan Lengauer, Alexander Komar, Arniel Labrada, Stephan Karl, Elisabeth Trinkl, Reinhold Preiner, Benjamin Bustos, and Tobias Schreck. In Silvia Biasotti, Guillaume Lavoué, and Remco C. Veltkamp, editors, 12th Eurographics Workshop on 3D Object Retrieval, 3DOR@Eurographics 2019, Genoa, Italy, May 5-6, 2019, pages 17–24. Eurographics Association, 2019 [48].
- Motif-driven Retrieval of Greek Painted Pottery. Stefan Lengauer, Alexander Komar, Arniel Labrada, Stephan Karl, Elisabeth Trinkl, Reinhold Preiner, Benjamin Bustos, and Tobias Schreck. In Selma Rizvic and Karina Rodriguez-Echavarria, editors, GCH 2019 - Eurographics Workshop on Graphics and Cultural Heritage, GCH 2019, Sarajevo, Bosnia and Herzegovina, November 6-9, 2019, pages 89–98. Eurographics Association, 2019. [47].

1.6 Thesis organization

This thesis is organized in 10 chapters as follows:

- Chapter 1 presents the introduction, motivation, and goals of this thesis.
- Chapter 2 describes the basic concepts to comprehend better the problems tackled in this thesis.
- Chapter 3 contains a review of studies performed in the 3D shape research field. We explain the state of the art in different tasks like 3D shape retrieval, 3D shape embedding, and 3D shape representation.
- Chapter 4 describes an image-based 3D model retrieval method (SC-GALIF) using suggestive contours and Gabor filters to extract features from images and 3D models represented as a set of image views.
- Chapter 5 describes a new mid-level representation used for 3D shape retrieval.
- Chapter 6 proposes a 3D shape descriptor using deep features applied to image-based 3D shape retrieval. Besides, we compare the results using the proposed deep features extraction technique against the engineered features extracted with our proposal from Chapter 4 (SC-GALIF).
- Chapter 7 describes two proposals for Cultural Heritage Retrieval. First, a sketch-aided retrieval system, and second, a Motif-driven Retrieval framework.
- Chapter 8 proposes a new convolutional architecture for computing 3D model embedding applied to 3D shape retrieval.
- Chapter 9 describes an extension of our network for computing 3D shape embedding used in the SHREC (3D Shape Retrieval Challenge) 2021.
- Chapter 10 proposes a new deep learning architecture for image-based 3D shape retrieval using our network for computing embeddings. We also compare the results using the proposed artificial neural network architecture against the learned features extracted using our proposal from Chapter 6.

Chapter 2

Background

In this section, we discuss some background concepts for the best understanding of our Ph.D. research.

2.1 Hand-engineered features for multimedia objects

The Bag of Features (BOF), a widely employed low-level representation in the field of multimedia, finds versatile application in tasks like multimedia retrieval and multimedia classification. This technique revolves around characterizing multimedia objects as a collection of distinctive features. To achieve this, one or more descriptors are carefully selected based on the nature of the multimedia object under consideration. These descriptors are then applied to predefined local regions within the original multimedia object to extract relevant features from those regions. Subsequently, the ensemble of all extracted features from these local regions collectively constitutes the BOF representation of the multimedia object.

Visual multimedia data encompassing images and 3D models, (BOF) representations hold prominence. The literature has seen a plethora of descriptor choices for feature extraction in this domain, including but not limited to gradient [45], Gabor filters [27], shape descriptors, and DSIFT (dense scale-invariant feature transform) [29], each tailored to capture unique aspects of visual content.

2.1.1 Bag of Words (BOW): a mid-level representation for multimedia objects

The problem with BOF is that a multimedia object is represented as a set of features, which can be in the order of the thousands of features, and each feature is usually a vector of

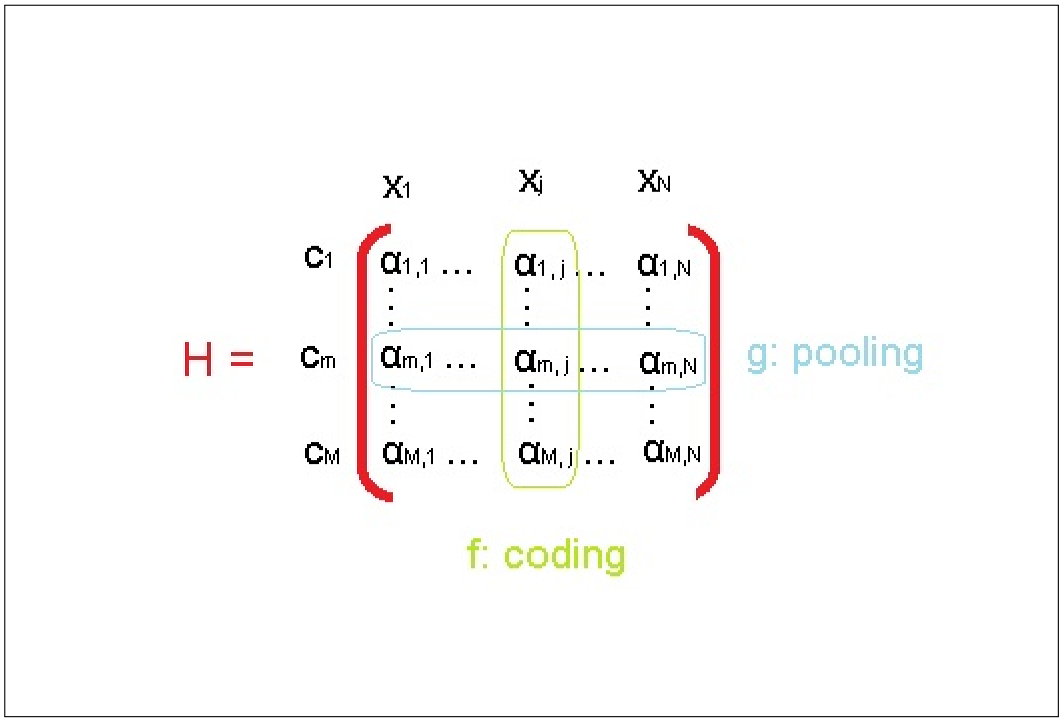

Figure 2.1: BOW: H matrix representing coding and pooling functions.

more than one hundred dimensions, so this representation is very ineffective. A frequently addressed solution to this problem is the mid-level representation. This kind of representation tries to capture in a single vector as much information as possible from the set of features that represent a multimedia object.

A representation of medium level widely used in the state of the art is the Bag of Words (BOW). Su et al. [10] formalize BOW by following a three-step algorithm coding, pooling, and concatenating. Given the set of local descriptors by $X = \{x_j\}, j \in \{1; N\}$, where each local feature $x_j \in R^d$ and N is the number of local regions of interests on the image, let us denote the visual dictionary as $C = \{C_m\}, m \in \{1; M\}$, where M is the number of visual words and $Z \in R^M$ is the final vectorial representation of the image. The final goal of the three-step methodology is a mapping from X to Z.

The coding step is where each local descriptor is projected to the visual dictionary. This step is modeled by the function f:

$f : R^d \to R^M$

$x_j \to f(x_j) = \alpha_j = \{\alpha_{m,j}\}, m \in \{1; M\}$ (1)

After the coding, the pooling step is modeled by the following function g:

$g : I^N \to R$

$\alpha_m = \{\alpha_{m,j}\}, j \in 1; N \to g(\alpha_m) = z_m$ (2)

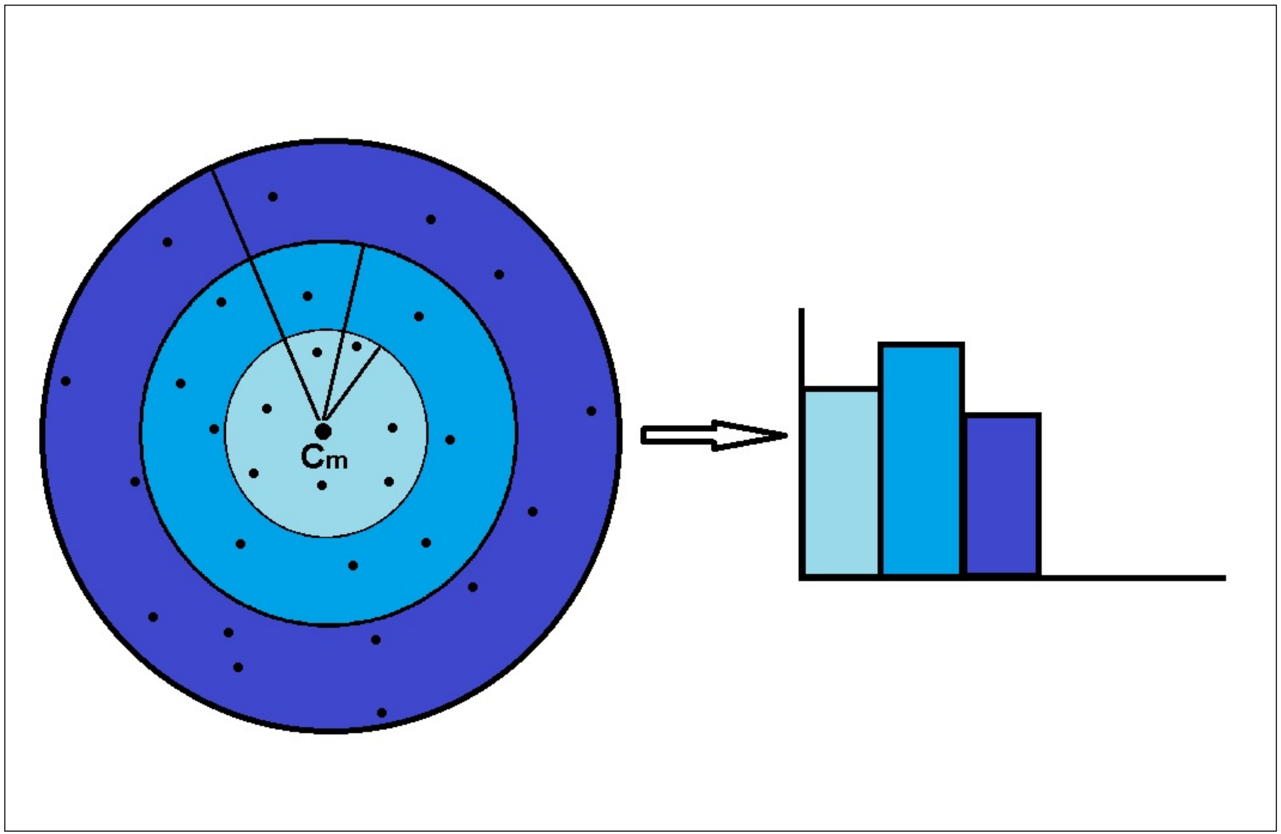

Figure 2.2: Illustration of a local histogram z_m.

As illustrated in Figure 2.1, the coding function f for a given descriptor x_j corresponds to the j^{st} column and the pooling function g for a given visual word c_m corresponds to the m^{st} row. For example, to compute the basic BOW representation, first f assigns a constant weight to its closest center:

$$f(x_j) = \begin{cases} 1 & \text{if } m = argmin \left\| x_j - c_k \right\|^2 ; k \in \{1, M\} \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

g computes the sum over the pooling region.

$z_m = \sum_{j=1}^{N} \alpha_{m,j}$ (4)

The final image representation, Z, is given by sequential coding, pooling, and concatenating: $Z = [z_1, z_2, ..., z_M]$.

2.1.2 Other mid-level representations (BOH and BossaNova)

Although BOW is the most used mid-level representation, it has the problem that all features belonging to the same visual word provide the same amount of information, regardless of the distribution of these features within that visual word. Different solutions have been proposed to address this problem like BossaNova [23] and BOH [4].

BossaNova proposes an extension of BOW, called BOSSA (Bag Of Statistical Sampling Analysis) [23]. The idea here is to keep more information during the pooling step. For instance, in BOW, the pooling function summarizes the information contained in $\alpha_{m,j}$ into a single scalar value. On the other hand, BossaNova computes a histogram of the distribution of the features within the $\alpha_{m,j}$. To calculate this histogram, BossaNova divides each visual word into B bins. With this, each Z_m is now a vector of size B, which makes the final representation Z of size $M \cdot B$. Figure 2.2 shows an illustration of a local histogram Z_m, and the formula below shows the new pooling function.

$g : R^N \to R^B$

$\alpha_m \to g(\alpha_{m,j}) = z_m$

$z_{m,k} = card(x_j \mid \alpha_{m,j} \in \alpha_m^{max} \cdot [\frac{k}{B}; \frac{k+1}{B}])$ (5)A where B denotes the number of bins of each histogram z_m, and α_{max} m is the maximum radius within visual word. We obtain the z_m by concatenating the $z_{m,k}$ and the final image representation Z, same as in BOW, is given by sequential coding, pooling, and concatenating: $Z = [z_1, z_2, ..., z_M]$.

We can make two observations of this new pooling function first when $B = 1$ BossaNova is equal to BOW. The second observation deals with how this B bins divides the d dimensional hyperspace. As we can see from the formula, each bin has the same radius. In other words, the distance between bins is the same. However, all the bins of a specific histogram have different hyper-volumes. For example, let us say that $d = 2$ and $B = 2$, the second bin of any histogram is a circumference of radius twice the first one, which means that the second bin has three times more area than the first. This difference grows exponentially with the number of dimensions. For example, for $d = 3$, the volume of the second bin is seven times the volume of the first, and in general, for $d = n$ and $B = 2$, the proportion between the volumes is $2^n - 1$.

Due to this exponential difference between the bins of a same histogram, the distribution of features x_j in a z_m histogram will not be uniform since it is more likely that if one bin has much more volume than another, then the probability that the features accumulate in that bin is greater.

To solve this problem, another mid-level representation based on BOW is used, Bag of local distribution of descriptors on concentric Hyperspheres (BOH) [4]. The idea in this proposal is that all bins in the same histogram have the same volume. Therefore, the formalization of the BOH method is very similar to that of BossaNova, with the difference of the pooling function g, which is defined as follows:

$g : R^N \to R^B$

$\alpha_m \to g(\alpha_{m,j}) = z_m$

$z_{m,k} = card(x_j \mid \alpha_{m,j} \in \alpha_m^{max} \cdot [\sqrt[d]{\frac{k}{B}}; \sqrt[d]{\frac{k+1}{B}}])$ (6)

2.2 Basic concepts of deep learning

Here, we discuss basic techniques and tools to build an artificial neural network.

2.2.1 Artificial Neural Network

Artificial Neural networks [44], rooted in the mathematical modeling of artificial intelligence, are computational models specifically designed for optimizing complex tasks by minimizing a downstream cost function. These networks are composed of interconnected processing nodes, referred to as neurons, organized into layers. Each neuron within a layer processes information from the preceding layer and produces an output, which is then propagated through subsequent layers. The final layer generates the network's ultimate output, representing the network's learned solution.

One of the notable attributes of artificial neural networks is their adaptability: each layer may encompass a different number of neurons, and the connections between neurons, governed by adjustable parameters, have varying strengths. These parameters are honed during the training process using optimization techniques like gradient descent, where the network iteratively adjusts its parameters to minimize the discrepancy between its predictions and the desired outcomes. This capacity for self-adjustment empowers artificial neural networks to tackle intricate tasks such as image recognition, speech synthesis, language translation, and more. They have revolutionized machine learning and artificial intelligence, enabling the development of sophisticated systems capable of handling real-world challenges with remarkable precision.

2.2.2 Activation Functions

In deep learning, an activation function is applied to the output of each neuron in an artificial neural network layer to obtain the final result of the said layer. This function is commonly used to allow the network to learn non-linear models since the rest of the operations made in a layer are linear. However, linear functions can be used when trying to learn linear models. Some commonly used activation functions are Sigmoid, Hyperbolic tangent (tanh), ReLU, Leaky ReLU, softmax, etc. Their use is tied to the task tackled by the network. For example, softmax is the standard on the output of the last layer in classification networks, and relu family functions are commonly used to activate convolutional layers. Figures 2.3, 2.4, 2.5, 2.6, and 2.7 show the formula of the mentioned activation functions and their plots.

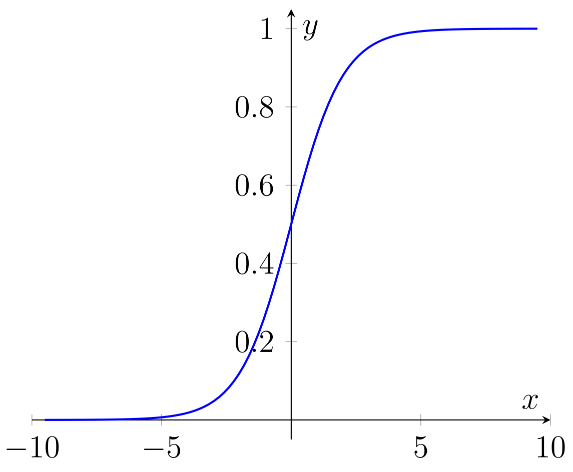

Figure 2.3: Sigmoid function: $y = \frac{1}{1+e^{x}}$

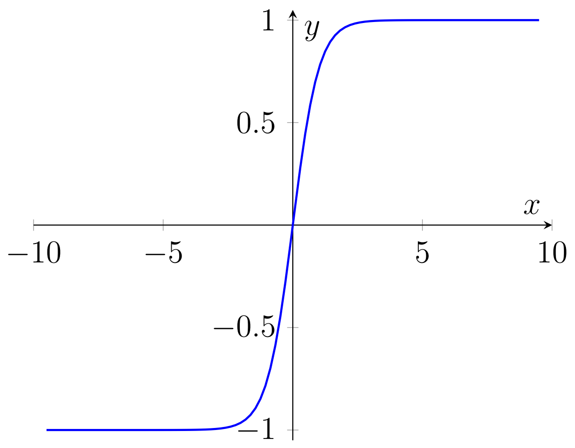

Figure 2.4: Hyperbolic tangent function: $y = \tanh(x)$

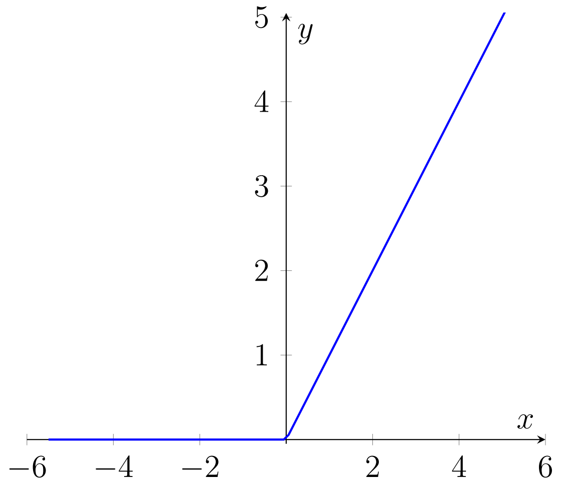

Figure 2.5: ReLU function: $y = max(0, x)$

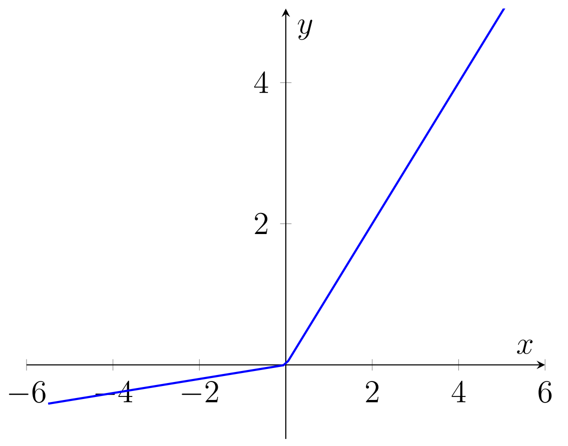

Figure 2.6: Leaky ReLU function: $y = max(0.01, x)$

2.2.3 Optimizers

In deep learning, an optimizer is a technique used to adjust an artificial neural network's weight and learning rate. In other words, the optimizer defines how the network will learn. Standard optimizers are Stochastic Gradient Descent (SGD), SGD with momentum, RMSprop, and Adam. Though Adam optimizer is the most common, like other params in an artificial neural network, choosing the optimizer is also a task-dependent process.

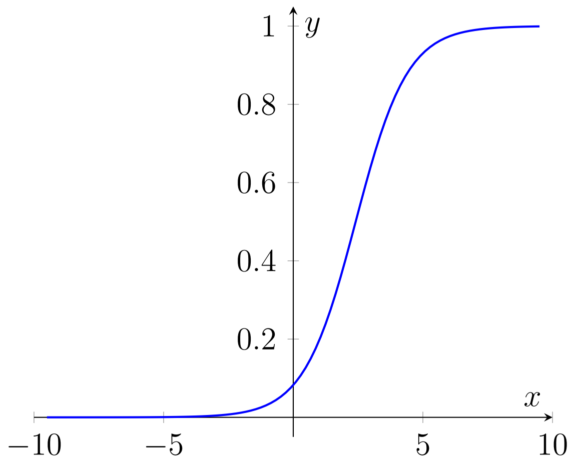

Figure 2.7: Softmax function: $y_i = \frac{e^{y_i}}{\sum_{j=1}^{k} e^{y_j}}$

2.3 Important Artificial Neural Networks Architectures

Here, we describe three artificial neural network architectures used throughout our thesis proposal. These architectures are Convolutional Neural Networks, Autoencoders, and Siamese Networks.

2.3.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) [45] represent a class of artificial neural networks renowned for their exceptional performance across diverse tasks. Particularly dominant in image processing, CNNs gained significant recognition after the groundbreaking success of AlexNet [42] in 2012. These networks have since been extensively employed in various applications, including image recognition, image classification, object detection, and face recognition.

The standard architecture of a CNNs is distinctive, featuring two key types of layers: convolutional layers and pooling layers. Convolutional layers revolutionize traditional artificial neural networks by employing a convolution operation, effectively transforming the input data through a set of learnable filters or kernels. This process generates feature maps that capture localized patterns and structures within the input data, effectively automating the feature engineering process that was previously manual and labor-intensive.

Pooling layers, on the other hand, play a crucial role in spatial dimension reduction. Various pooling methods, such as max pooling, average pooling, and sum pooling, extract the most salient information from the feature maps while reducing their size. This reduction enhances computational efficiency and helps focus on the most relevant features.

CNNs exhibit a hierarchical feature learning process. Initial layers detect simple features like edges and gradients, while deeper layers progressively recognize more complex and abstract features, including shapes, textures, and objects. The adaptability of CNNs is a key strength, allowing their architecture to be tailored to specific tasks. For instance, in image classification tasks, fully connected layers follow the convolutional and pooling layers, culminating in a final connection to a classifier, typically employing a softmax activation function and cross-entropy loss.

In summary, CNNs have revolutionized the field of deep learning and computer vision by automating feature extraction and demonstrating remarkable capabilities in various applications, making them an indispensable tool for image-related tasks and beyond.

2.3.2 Autoencoders

Autoencoders [73] are a versatile class of artificial neural networks primarily employed in unsupervised learning tasks, serving various purposes, including data representation, feature embedding, and data compression. These networks are designed with a distinctive architecture that includes four essential components: the encoder, bottleneck, decoder, and reconstruction loss.

The encoder is responsible for learning a reduced-dimensional representation of the input data, effectively capturing essential features and patterns. The bottleneck, often a layer with a lower dimensionality than the input, serves as a compressed knowledge representation of the original data. The decoder's role is to reconstruct the input data from the bottleneck's representation, aiming to minimize the information loss during the compression process. The reconstruction loss quantifies the dissimilarity between the reconstructed output and the original input, measuring how well the autoencoder is performing. Figure 2.8 shows a standard Autoencoder architecture.

Autoencoders find widespread use in image processing, particularly for image compression and representation. One of the most common variations is the convolutional autoencoder, which adapts the encoder-decoder architecture to images. The encoder portion typically consists of convolutional layers and pooling layers, facilitating dimension reduction by downsampling. Subsequently, the encoder connects to a fully connected layer serving as the bottleneck. The decoder, responsible for image reconstruction, utilizes de-convolutional layers and un-pooling layers. De-convolutional layers perform the inverse operation of their corresponding convolutional layers, while un-pooling layers are used to increase resolution, effectively reversing the effects of pooling layers. Notably, addressing the non-invertibility of traditional pooling operations, recent years have seen the development of various un-pooling techniques [15, 95] to facilitate the autoencoder's image reconstruction capabilities.

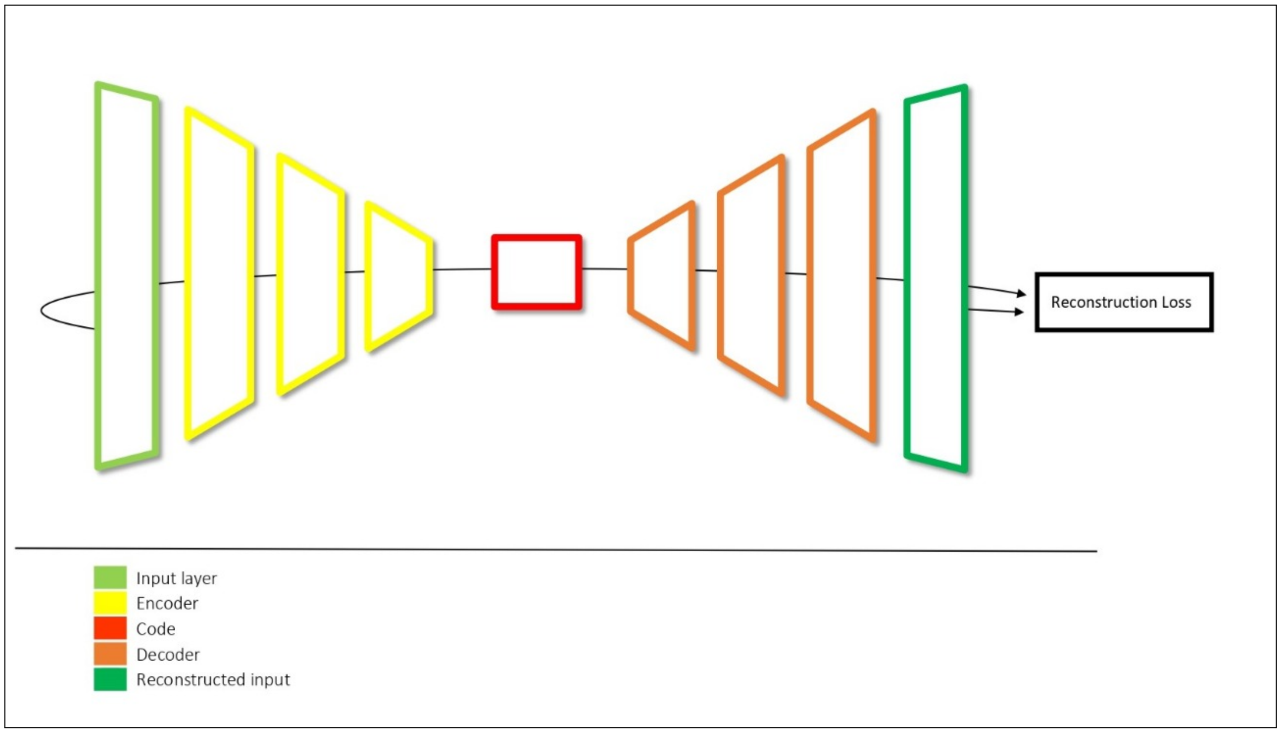

Figure 2.8: A standard Autoencoder architecture

In summary, autoencoders offer a powerful framework for tasks such as dimensionality reduction, data compression, and image processing, with convolutional autoencoders being particularly well-suited for image analysis and representation applications. These networks have found applications in various fields, from computer vision to data compression, facilitating meaningful feature extraction and efficient data representation.

2.3.3 Siamese Networks

Siamese Networks [12] are artificial neural networks, usually composed of two subnetworks, but can be more than just two. These subnetworks not only have identical architecture, but the weights have to be shared among them as well. In other words, all the subnetworks are copies of the first subnetwork.

To train a Siamese network, we usually use a tuple of input (x_1, x_2) with a binary label ℓ. This label indicates if x_1 and x_2 belong to the same class. The idea is to pass x_1 and x_2 through the subnetworks to compute a fixed-length feature vector for each $(h(x_1), h(x_2))$. After that, we define a loss function that penalizes the similarities or dissimilarities between $h(x_1)$ and $h(x_2)$ according to ℓ. This loss function is often a distance between vectors like the euclidean distance or the cosine distance, with binary cross-entropy using the binary label.

The inputs of a Siamese network can be anything from numerical data, image data, or

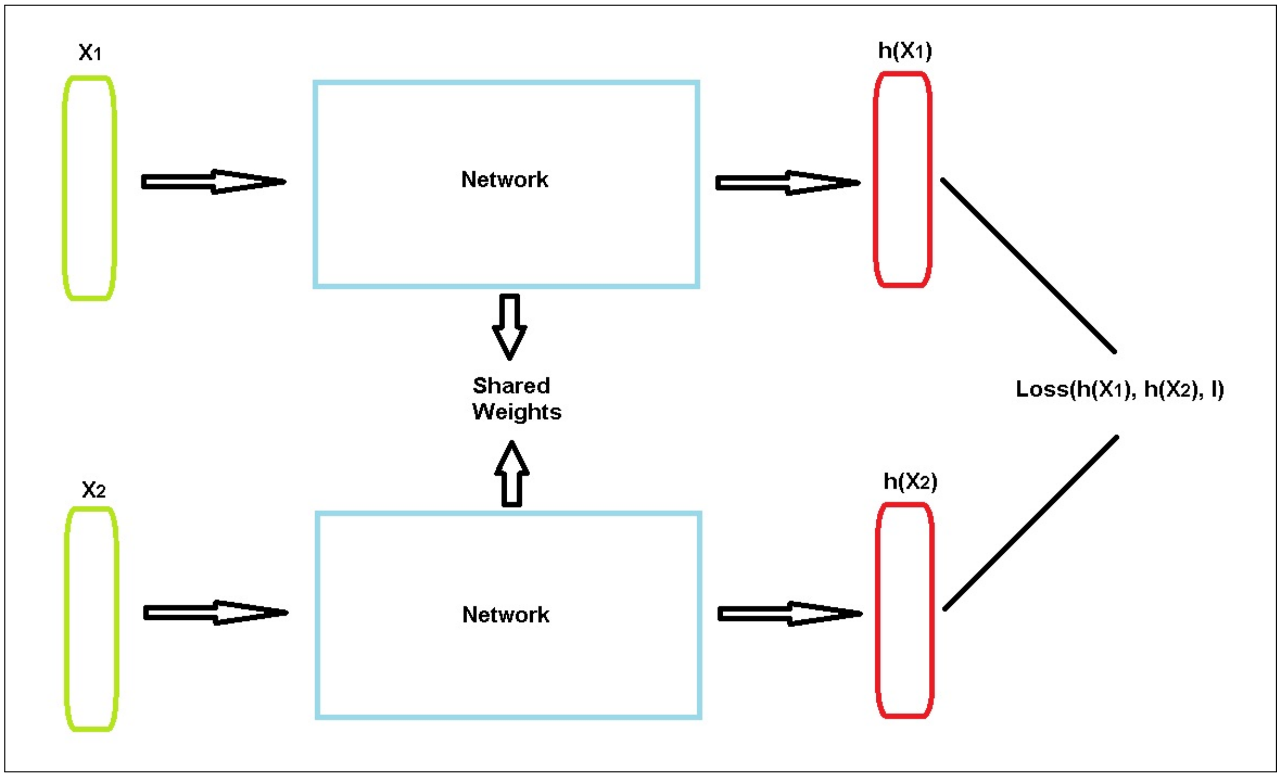

Figure 2.9: Siamese Networks

even sequential data such as sentences or time signals. As for the subnetworks, we can use any deep learning architecture such as a Recurrent neural network (RNNs), fully connected layers, CNNs, etc. The goal is to transform the data using this subnetwork before applying the loss function between the transformations of the inputs. The primary purpose of these kinds of architectures is metric learning. In other words, these networks learn whether inputs are similar or not. However, given the nature of the subnetworks to compute feature vectors of the inputs, we can use them for data representation. This allows us to use siamese networks to solve a wide variety of tasks like retrieval.

2.4 Transfer Learning

Transfer learning is a machine learning technique where a model developed for a task is reused for another model on a second task. More specifically, the idea is to use what was learned for a particular task (source task) to solve a different task (the destination task). To that end, the source and the destination task must be "sufficiently similar."

In the deep learning field, the transfer learning paradigm is especially useful. We can use pre-trained models for our specific tasks. For example, we can train a model for image classification using Imagenet, a very standard and extensive image dataset. After that, we can re-use this trained model for another task like segmentation, object detection, or even

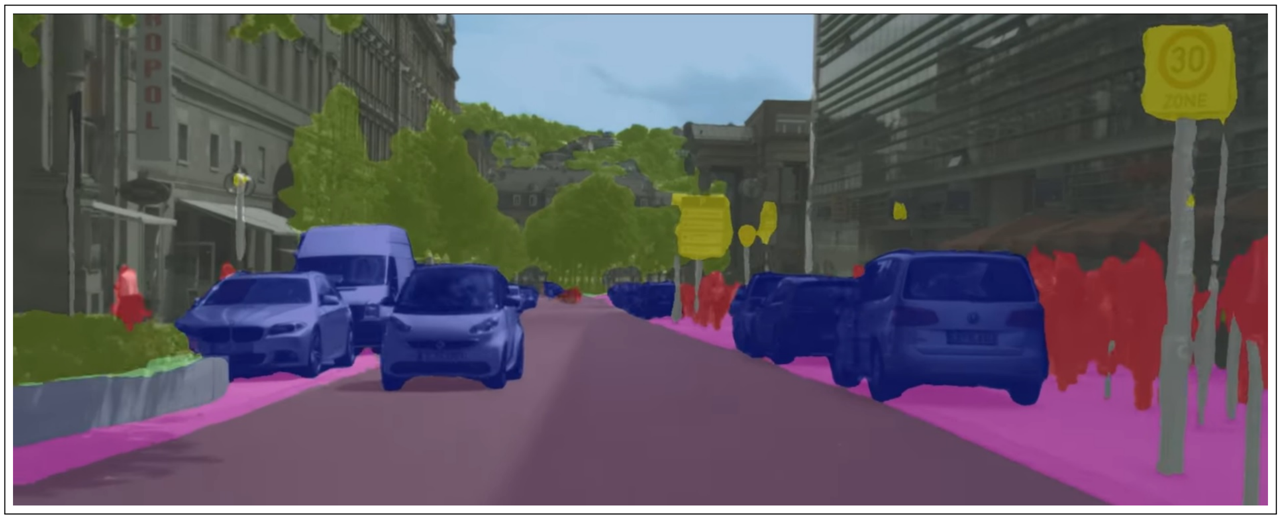
Figure 2.10: Semantic Image Segmentation

image classification as well in another dataset. Furthermore, we can also use a model trained by other researchers and adapt it to our necessities. With this technique, we can save training time and computational resources. But more relevant, we can use deep learning even when we have little data for training.

2.5 Semantic Image Segmentation

A recent application of the CNNs is semantic image segmentation. This task consists of classifying each pixel in an image into a class. In other words, the goal is to assign a label to each pixel of a given image. A basic approach to solve this task would be an artificial neural network architecture with a stack of convolutional layers with the same padding to preserve dimension and output a final segmentation map. This way, we learn a mapping directly from the image to its corresponding segmentation. However, since we are preserving the dimension from layer to layer, this would be quite computationally expensive, especially if we keep increasing the number of feature maps (channels) per layer. To solve this, researchers tend to follow an encoder/decoder structure similar to an Autoencoder architecture. The encoder solves the dimensionality preserving problem by downsampling the spatial resolution of the input. Then, the decoder is used for upsampling the feature representations into a full-resolution segmentation map. Figure 2.10 shows the result of the semantic segmentation of an image. To visualize this result, different colors are used for each different label of pixels.

Currently, the best approaches for semantic segmentation use end-to-end deep neural networks. Examples of these approaches are Mask R-CNN [36], FPN [59], FCN [63], U-NET [72], and deeplabv3+ [16]. This last approach reported the best results for the 2012 PASCAL VOC segmentation challenge. Implementation of these previous models can be found in the open-source machine learning library TensorFlow. This library allows us to train these models from scratch using our own data sets or to use one of several pre-trained deeplab models for semantic segmentation.

2.6 Evaluation metrics

To evaluate our proposals, we used six common performance metrics for retrieval techniques. These metrics are based on a ranked list, which consists of all 3D models ordered according to their similarity with a specific query. The metrics are Precision-Recall plot (PR), Mean Average Precision MAP, Nearest Neighbor NN, First Tier FT, Second Tier ST, and Discounted Cumulative Gain DCG. Their meanings and definitions are explained below.

2.6.1 Precision-Recall plot

A precision-recall plot is a graphical representation used to assess the performance of a ranking or retrieval system, typically in information retrieval or machine learning tasks. It aims to illustrate the trade-off between precision and recall for a given retrieval model and a specified number n of the top-ranked items.

Precision measures how many of the top n items retrieved by the model are relevant to the class C or the target category. It is calculated as the ratio of relevant items retrieved to the total number of items retrieved.

Recall measures how many of the total relevant items in the dataset were successfully retrieved within the top n items by the model. It is calculated as the ratio of relevant items retrieved to the total number of relevant items in the dataset.

In the precision-recall plot, recall is typically plotted on the horizontal axis, while precision is plotted on the vertical axis. Each point on the plot corresponds to a different value of n, representing the top n ranked items from the retrieval model.

A perfect retrieval system would achieve a horizontal line in the plot at a precision equal to one, indicating that all the retrieved items at that rank are relevant. In contrast, a shifted curve suggests a trade-off between precision and recall, with the system retrieving more relevant items as n increases and potentially introducing some irrelevant items.

2.6.2 Mean average precision

Mean Average Precision (MAP) is a widely used evaluation metric in information retrieval and ranking tasks. It measures the average precision score across multiple queries, providing a single aggregate value to assess the overall performance of a ranking system.

To compute the MAP first, we must compute the Average Precision AP for each query or data point. The AP quantifies how well the ranking system retrieves relevant items at different retrieval depths. Below, we show the formula for the Average Precision:

$$AP = \frac{1}{R}\sum_{k=1}^{R}\left(\frac{k}{rank_k}\cdot rel_k\right) \tag{2.1}$$

Where R is the total number of relevant items in the ground truth for the query, $rank_k$ is the position (rank) at which the k^{th} relevant item is retrieved, and rel_k is an indicator function that equals 1 if the k^{th} item is relevant, and 0 otherwise.

In other words, to calculate AP for a single query, you sum up the precision values at each relevant item's position and divide by the total number of relevant items. This yields the Average Precision for that query.

With the AP formula, the MAP of a set of N queries is calculated as follows: for each query, q, we compute its corresponding Average Precision AP, and then, the mean of all these scores (Formula 2.2). The resultant value measures the quality of models at carrying out queries and is approximately equal to the area under the precision-recall curve [104].

$$MAP = \frac{\sum_{q=1}^{N} AP(q)}{N} \tag{2.2}$$

2.6.3 Nearest Neighbor (NN)

NN is a simple metric to measure the percentage of times the element closest to a query (first element of the ranked list) belongs to the same class as the query. The goal with this metric is to obtain a result as close to 100% as possible since this would mean that for every query, its closest element belongs to its same class.

First tier and second tier

On the other hand, FT and ST go deeper than NN and measure the percentage of elements that belong to the query class C that appear within the first n elements matches of the ranked list, where n depends on the number of elements in C. Specifically, for the First Tier metric $n = |C| - 1$, and for the Second Tier metric $n = 2 * (|C| - 1)$. Similarly to NN, the optimum value of these scores is 100%.

2.6.4 Discounted cumulative gain (DCG)

Finally, the DCG metric uses the fact that search engine users are more interested in the top results of the queries they execute. For this reason, to compute the final score, DCG uses all the elements of the ranked list that belong to the same class as the query. However, the closer to the beginning of the ranked list the elements appear, the more weight they provide to the final score. To do this, the metric follows three steps. First, the ranked list R is converted to a list G, where element G_i has value 1 if element R_i is in C and value 0 otherwise, where C is the class of the query image. Second, the discounted cumulative gain is calculated as follows:

$$DCG_i = \left\{ \begin{array}{cc} G_1, & i = 1 \\ DCG_{i-1} + G_i / \log_2 i, & otherwise \end{array} \right\} \tag{2.3}$$

Finally, the obtained result is divided by the maximum possible DCG:

$$DCG = \frac{DCG_n}{1 + \sum_{j=2}^{|C|} \frac{1}{\log_2 j}} \tag{2.4}$$

where n is the size of R.

2.6.5 Micro and macro average

Along with these two evaluation metrics, MAP and DCG, we also use two versions of each metric. These versions are the micro averaged and the macro averaged. On the one hand, the micro averaged version will aggregate the contributions of every object to calculate the average metric. On the other hand, the macro averaged version will compute the metric for each class of objects and average these results. In multiclass classification and retrieval, the micro averaged version is more desirable if one knows that there is an imbalance in the categories' sizes.

2.7 Datasets

We use three different datasets for the experiments, the ShapeNet benchmark [14], the ModelNet [99] and the SHREC'12 benchmark [55]. Below, we describe the three datasets and the techniques for their processing.

2.7.1 SHREC'12

SHREC'12 [55] is built based on the Watertight Model Benchmark (WMB) dataset. This benchmark has 400 models, divided into 20 classes, with 20 models each. This dataset is built for sketch-based 3D model retrieval. However, given that the 3D models are well annotated, the dataset can be used for different tasks related to 3D models, like 3D model retrieval and image-based 3D model retrieval.

2.7.2 ShapeNet

This dataset [14] comprises 51,300 3D models grouped into 55 categories, and they are provided in OBJ format. The dataset counts with two versions, consistently aligned(normalized dataset) and a more challenging dataset where random rotations perturb models. The dataset provides a split 70%/10%/20% for training, validation, and testing, respectively. We also use a point cloud version of the normalized ShapeNet as well. This point cloud version is public on Github [1].

We also use a partition of this dataset to perform some minor experiments and tune some of our proposals. To do this, from all the 55 categories, we chose 20 randomly, and from each selected category, we extracted 200 3D models. Finally, we divide each of the chosen 200 3D models of each chosen class into 100 3D models for training and 100 for testing.

2.7.3 ModelNet

This dataset [99] has two versions ModelNet10 where the models are categorized into 10 classes, and ModelNet40, where the models are categorized into 40 classes. Both versions are composed of CAD models manually aligned but not scaled. The dataset also counts with its own training/testing split.

[1]Point Cloud Datasets

2.8 Image views extraction

As mentioned, we represent a 3D model using image views rendered from the 3D models themselves. To accomplish this, we use three different methods. We built the first of these methods by ourselves from scratch, while in the second one, we helped ourselves using a pre-coded tool. Finally, we proposed a Convolutional Neural Network for the third method to refine a given image view set, extracting the best views. We use Python to code for all three methods.

2.8.1 Image views extraction from scratch

For our first method, we extract 100 image views from each 3D model. To accomplish this, we utilize two fundamental measures: azimuth, which specifies the viewpoint's direction, and elevation, which represents the height of the viewpoint, both measured in degrees. These measures are defined with respect to the XY plane. We initiate with an initial viewpoint set at 0° azimuth and 0° elevation, serving as our reference point. We systematically vary the azimuth measure by adding 36° to the current value. This step is repeated ten times, creating a series of viewpoints spanning different azimuth angles. Finally, we reset the azimuth to 0° to repeat the aforementioned process and rotate the 3D object 18° over the X-axis downward. We repeat this last process ten times and obtain the 100 viewpoints for each 3D model. We explain below the details of each of the experiments.

2.8.2 Image views extraction using Stanford-Shapenet-renderer

For our second method to extract the image views, we use the Stanford-Shapenet-renderer script. This script uses the API from the Blender software to render images from different angles of a given 3D model in OBJ format. This code is public on Github [2] and fully parameterizable for different tasks. In our case, we use the code to render images using two different approaches. The first approach starts with the camera pointing to the coordinates' origin. From there, we generate an image every 12° in circumference around the 3D model itself, which provides 30 image views per 3D shape. For the second approach, we render 48 images by sampling in the vertex of a regular icosahedron containing the 3D models.

[2]Stanford-Shapenet-renderer

2.8.3 Image views refining using Convolutional Neural Networks

We propose a method to select the "best image views" from a large set of image views representing a 3D model. To do this, we train a deep convolutional architecture for classification using the whole set of image views. After the training phase, we use the trained network to select the views that best classify the 3D model. In other words, we choose the views that produce the highest accuracy score for classification.

2.9 Summary

In this chapter, we explain some basic concepts for a better understanding of our thesis proposal. We first discuss multiple techniques for 3D model retrieval using engineered features to build representations of the 3D models. We start with a low-level representation of the bag of features technique. Then, we explain how to group those features using different techniques to build mid-level representations like Bag of Words, BossaNova and BOH.

We also discuss important deep learning concepts in the chapter. We first explain different standard artificial neural network architectures used in our proposals. These architectures are the convolutional neural network, Autoencoder, and siamese network. We also discuss semantic segmentation and transfer learning techniques.

Finally, we detail the evaluation metrics used to evaluate our results (MAP, NN, FT, ST, DCG) and two ways to compute those metrics the micro average and macro average. We also describe the datasets (ShapeNet, SHREC'12) used in the experiments and the technique to render image views from the 3D shapes.

Chapter 3

State of the Art of 3D Model Processing

Over the years, the 3D model research field has been highly relevant in Computer Science due to the steady increase in the number and use of 3D objects for many practical application domains. Surface segmentation and face recognition are some of the many important tasks in this field. However, two of the most addressed tasks are 3D shape retrieval and classification. Traditionally, the main approaches for fulfilling these tasks relied on hand-engineered feature extraction methods. Many kinds of features can be used for this purpose. Nevertheless, shape features are the most used because, in many cases, one only has the shape of the 3D model as input data. Surveys in this area can be found in Generic 3D shape retrieval [50] and textured 3D model retrieval [13].

In particular, image views are a very feasible and accepted representation of 3D models for its processing. Since 3D models are a very complex and space costly type of data, in many cases, it is necessary to transform these objects into a more manageable kind of data such as an image or, as in this case, a set of images. Several works that use deep learning architectures in the 3D model field use this image view representation [37]. The reason behind this is that training a deep learning model directly with any of the standard 3D model representations (voxel grid, polygon mesh) could be a very costly process both in time and resources.

Recently, some works have proposed using artificial neural network techniques instead of extracting features to solve the 3D model retrieval and classification tasks with excellent results [37]. This new approach shows promising results in the area. However, it is still a relatively under-researched topic, and some 3D model tasks such as computing 3D model embeddings have received little to no attention.

There are three main approaches for using deep learning techniques in the area of 3D shape retrieval and classification. These approaches are based on how 3D models are represented as input to an artificial neural network model (image views, voxel grids, point clouds).

The first two approaches represent 3D models as a set of image views and voxel grids, respectively [37, 34]. In the first case, authors often adapt artificial neural networks that have been successfully used to solve image processing tasks to the image view domain. This is fulfilled by using techniques like the assembly of artificial neural networks to combine the output of several networks into one.

In voxel grid representation, the approach adapts the 2D operators used in a convolutional neural network to 3D operators. For example, instead of using convolutional layers of (number of inputs) x (input height) x (input width) x (input channels), convolutional layers of (number of inputs) x (input height) x (input width) x (input depth) x (input channels) are used.

The representation of 3D models with image views yields better results than voxel grids in many applications. However, incorporating the voxel grid can lead to better results if one uses more complex artificial neural network models, which are more expensive to train and require more training data.

The third approach consists of using point cloud representations to train deep learning models. However, this could be challenging since point clouds are an irregular representation and standard convolutional neural networks work with regular representations. To address this problem, two approaches are used. The first one consists of transforming the original point cloud into a regular grid. The second one consists of building a convolutional operator that works directly over the point cloud using structures to store the points like k-d trees.

This last approach has recently increased in popularity with some outstanding works like PointNet [69], DensePoint [60], Dynamic Graph CNN for Learning on Point Clouds [97], RS-CNN [61], and LDGCNN [101]. A survey on this topic can be found in Deep Learning for 3D Point Clouds: A Survey [34].

3.1 Image views representation for 3D deep learning

View-based 3D shape recognition and retrieval approaches are typical in the research community. These approaches are commonly focused on two main aspects: the quality of the image views and how to aggregate those image views to obtain the best representation. In the first case, different techniques exist for extracting the image views, different types of image views, and even methods to add more information to the set o image views like view-graph. An example of this last one is proposed in View-GCN [98]. In the second case, view pooling layers to fuse multi-view features are standard in combination with assembles of convolutional neural networks. We detail below essential works in these fields.

One of the earliest proposals for 3D shape retrieval using Convolutional Neural Networks is DeepEm [33]. They first propose an architecture for image embedding using a triple input for the training. The input consists of a query image, a positive image, and a negative image. Then, they compute the classification loss of each one using VGG19 [85] and a triplet loss using the last fully-connected layer of each of the three networks. Finally, to learn the embeddings, they jointly use all of the classification loss and the triplet loss. Using this network for image embedding, they also propose a 3D shape retrieval framework using Image Views representation. The goal is to compute embeddings from the image views and then use a set-to-set distance metric to calculate the similarity between two 3D shapes represented as a set of embeddings.

Aktar and Al Mamum [3] further explore the concept of using triplet loss with a convolutional neural network to compute image embeddings and then use these embeddings for 3D object retrieval. They also extend their proposal to unsupervised learning by using a convolutional Autoencoder.

Qi et al. [70] proposed three steps methodology for their Multi-View Convolutional Neural Networks proposal. First, they render multiple images from the 3D shapes. Second, they extract image features from each view using standard CNNs like VGG or AlexNet. Finally, they combine the features across views through a pooling layer, followed by fully connected layers.

Similarly, Su et al. [91] proposed a standard CNN trained to recognize the image views independently of each other, obtaining a very high accuracy for the recognition of 3D model even from a single view. Furthermore, they present a CNNs architecture with an image view pooling layer that combines information from multiple views into a unique and compact shape descriptor, offering even better recognition performance.

This concept of using pooling for aggregations of view has been explored in some works with good results. Sfikas et al. [79] proposed an extension of the PANORAMA 3D shape representation [67]. They use this representation as input to an ensemble of CNNs with the goal of computing feature continuity of 3D models. They test their proposal for 3D model classification and retrieval against several other state-of-the-art techniques, achieving very competitive results.

DRCNN [93] is another example of using view pooling layers. In this work, the author builds a novel layer called Dynamic Routing Layer (DRL) by modifying the dynamic routing algorithm of the capsule network to fuse the features of each view more effectively. In addition, based on DRL, they present a Dynamic Routing Convolutional Neural Network (DRCNN) for multi-view 3D object recognition.

More recently, Han et al. [35] further explore this approach of using pooling of views aggregation. The conjecture is that the redundant information within the views and their spatial relationships are lost during the pooling process. To solve this, they present a deep learning model (3D2SeqViews) with a novel hierarchical attention aggregation. This model not only aggregates the content information within all sequential views but also the sequential spatiality among the views.

“FMVAC: view-filtering-based multi-view aggregating convolution for 3d shape recognition and retrieval” [62] and “Multi-view 3d object retrieval leveraging the aggregation of view and instance attentive features” [58] are recent works that also rely on view pooling layers for aggregations of deep features. In the case of “FMVAC: view-filtering-based multi-view aggregating convolution for 3d shape recognition and retrieval” [62], the authors propose a voting-based view filtering module to select the top-k representative views before using the filtered top-k views to feed a multi-view aggregating module. As for the work presented in “Multi-view 3d object retrieval leveraging the aggregation of view and instance attentive features” [58], the authors point out that the existing multi-view convolutional neural network employs view pooling for feature aggregation, which ignores the local view-relevant discriminative information within each view image and the global correlative information across all view images. To address these problems, they propose two self-attention modules, View Attention Module and Instance Attention Module, to learn view and instance attentive features, respectively. Finally, they use these two modules to build the representation of a 3D object by aggregating three features: original, view-attentive, and instance-attentive.

We also propose an approach for computing 3D model embedding based on image views. However, we do not use view pooling layers or assembles of convolutional neural networks since we represent the 3D shapes as single multichannel objects. This makes the computational graph of our network smaller and more straightforward than the computational graph in a standard assembly of CNNs since this last approach needs to keep the information of a whole network for each view.

3.2 Deep Features

In the realm of machine learning and artificial intelligence, deep learning models have emerged as powerful tools, not only for predictive tasks but also for feature extraction. This chapter explores the pivotal role played by deep learning models in feature extraction, particularly within the domain of Multimedia Information. Traditionally, hand-crafted features have been employed extensively to represent multimedia objects for various tasks. However, recent advancements in deep learning have introduced ”learned features,” revolutionizing how we represent and extract meaningful information from data.

Learned features refer to representations or characteristics of data that machine learning models automatically acquire during training. These features are derived from raw input data through a data-driven approach, allowing models to uncover meaningful patterns and representations within the data. On the other hand, deep features are a category of ”learned features” that emerge from processing data through deep neural networks, typically deep feed-forward neural networks (like convolutional neural networks or recurrent neural networks). These features are extracted from intermediate layers of the network and are characterized by their hierarchical and abstract nature. The term ”deep” in deep features reflects the depth of the artificial neural network architecture, where each layer learns increasingly complex representations of the input data. Deep features capture intricate patterns, textures, and structures within the data, making them highly expressive and informative.

One notable application of deep learning for feature extraction is in the field of text processing. Here, the Word2Vec [71] algorithm stands as a prominent example. Word2Vec employs a deep learning model trained to predict words based on contextual usage. Following the training phase, a specific hidden layer of the model serves as a vector representation for each word. The key objective is to position the vectors representing words so that words with similar contextual usages are closer to each other in this vector space.

Deep learning models have also made significant inroads into the domain of image processing, where they are instrumental in feature extraction. A common architecture employed for this purpose is the Siamese network. The Siamese network consists of two identical sub-networks and a final module that leverages the outputs of both sub-networks to produce a final decision or representation.

In the training phase, pairs of objects are fed into the Siamese network, accompanied by a label indicating whether the objects belong to the same class. Once the training concludes, the model becomes proficient at discerning whether arbitrary objects share the same class. Noteworthy examples of works using Siamese networks for learned feature extraction from images include works like LIFT [100] and “Learning global representations for image search” [31].

The realm of 3D models is not exempt from the influence of learned features. In this context, we encounter the application of Convolutional Neural Networks (CNNs) and Siamese networks, as exemplified in the work “Sketch-based 3d shape retrieval using convolutional neural networks” [96] for sketch-based 3D shape retrieval.

In this cross-modal retrieval task, two Siamese networks are employed—one for the query sketch domain and another for the 3D models domain, represented as a set of 2D images. The outputs of these Siamese networks are connected to a module that generates the final output. During training, quadruple entries comprising two sketches and two 2D images are utilized, each labeled to indicate whether the objects belong to the same class. Through this process, the network learns vector representations for the sketch and the 2D images, ensuring that if a sketch and a 2D image belong to the same class, their respective vectors are close to each other in the feature space.

In summary, deep learning models have significantly reshaped the landscape of feature extraction across various domains. They enable the automatic extraction of meaningful features from data, improving performance in various applications, from text processing to image analysis. These learned features have enhanced our ability to understand and interpret data and propelled the development of more advanced and effective machine learning models.

3.3 Data Embedding

One notably successful use of deep learning is obtaining data embedding representations. This technique aims to represent the data as vectors with enough information to substitute the data for different tasks. In the text processing field, several outstanding works apply this technique for words (e.g., Word2Vec [71], Glove [68]) and sentence embedding (e.g., InferSent [18], BERT [25]).

There is a plethora of research available on image embedding, as well. Many researchers have proposed new architectures of CNNs to compute embedding for images. Examples of these works are LIFT [100], "Learning Image Embeddings using Convolutional Neural Networks for Improved Multi-Modal Semantics" [41], and "Deep Image Retrieval: Learning Global Representations for Image Searc" [31]. However, even when deep learning models compute embeddings implicitly, there is not much work focused on computing 3D shape embedding. Moreover, the few works that focus on compute embedding for 3D shapes are mainly based on point cloud representations [1, 17] or voxel grids. There is also a proposal for computing 3D shapes embedding using engineered features and the bag-of-words framework [57]. We propose a deep neural network architecture for computing 3D Shape embeddings using image view representation and CNNs. We aim to compute good embeddings by imposing restrictions on our network to achieve a compact representation of the 3D shapes while preserving its information.

3.4 Image-based 3D Shape Retrieval

Image-based 3D shape retrieval is challenging due to the significant differences between 2D images and 3D objects. So far, little research has been done in this field, and all the works we have found rely on deep learning techniques, especially convolutional neural networks. Another impediment in this field is that, to our knowledge, there is no standard benchmark for this task, and the authors need to build their own datasets to experiment.

A common practice to solve image-based 3D shape retrieval is to learn a joint space for images and 3D shapes. The earliest work we can find to use this approach is "Cross-Domain Image-Based 3D Shape Retrieval by View Sequence Learning" [46]. In this work, the authors propose a cross-domain triplet neural network with a feature aggregation method to represent the 3D shapes.

In 2019 Zhu et al. [103] proposed a pair of discriminative neural networks to learn a domain-invariant representation between 3D shape and depth image. They connect the two networks by a loss function composed of two terms: the inter-class margin and the intra-class margin. If the two samples come from the same category, then the variance between networks' outputs is considered the intra-class margin. Otherwise, the variance can be seen as the inter-class margin.

We can also find two very recent works in this field "Joint Intermediate Domain Generation and Distribution Alignment for 2D Image-Based 3D Objects Retrieval" [92] and M-GCN [65]. In the first one, the authors construct an intermediate domain module based on maximum mean discrepancy to reduce the 2D and 3D distribution discrepancy. Then, they use source domain labels as semantic information to guide distribution alignment dynamically. In the second work, the author proposes a multi-branch graph convolution network. The idea is that they use visual information to construct one cross-modalities graph model to compute the similarity between the image and the 3D model. Then, they apply a multi-head attention mechanism further to predict the hidden relationship between 2D image and 3D model.

Chapter 4

SC-GALIF: A new method for image-based 3D model retrieval

We employ an approach that fuses the capabilities of line drawing algorithms, suggestive contours, and mid-level representation techniques to tackle the challenging task of image-based 3D shape retrieval. This task involves identifying objects within a 3D model dataset that bear the closest resemblance to a given query image.

Our method transforms the initial problem into a sketch-based 3D shape retrieval challenge. We achieve this transformation by converting our query image into a sketch-like representation, leveraging the suggestive contours derived from the original image.

We employ Gabor filters to extract meaningful features from the suggestive contours. These filters help us capture essential details and characteristics embedded within the contours, enhancing the discriminative power of our retrieval system.

We adopt the Bag of Words framework to construct a structured representation for each suggestive contour. This framework allows us to aggregate the Gabor filter-derived features into a coherent and informative descriptor for each contour.

Finally, we use a similarity metric to compare and rank the representations of suggestive contours. This step is the foundation for building our retrieval system, which identifies and retrieves 3D shapes that closely match the query image.

We will first provide detailed explanations of suggestive contours and Gabor filters to facilitate a comprehensive understanding of our proposal. Following this, we will present the core components of our approach, illustrating how these techniques synergize to address the image-based 3D shape retrieval task effectively.

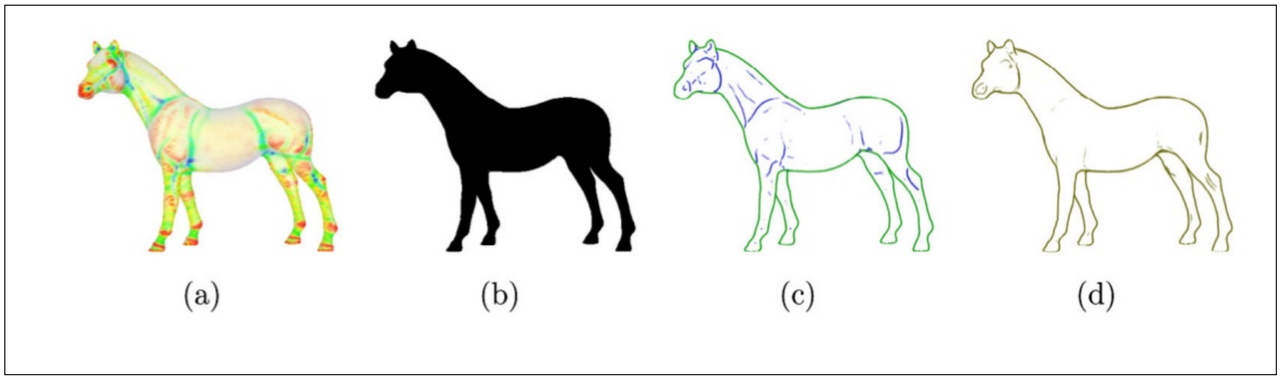

Figure 4.1: (a) A horse model (in curvature view); (b) Silhouette; (c) Contours (green) and Suggestive contours (blue); (d) Apparent ridges

4.1 Suggestive contours

The line drawing algorithms are techniques of image processing. These techniques are used to extract from a given image I the lines that compose the objects within it. Many line drawing algorithms have been proposed in the literature over the years, such as silhouette, contour or outline, suggestive contours, and apparent ridges. A revision of those algorithms can be found in the work "A comparison of methods for sketch-based 3d shape retrieval" [53]. An example of each line drawing algorithm is shown in Figure 4.1[1].

Suggestive contours are one of the most addressed line drawing algorithms in the literature over recent years. Visually, it is defined as lines drawn on clearly visible parts of the surfaces of an image, where a true contour would first appear with a minimal change in viewpoint. To extract the suggestive contours from a given image I, we first define a mask m. Then, we perform a convolution between m and each pixel of I. Finally, the suggestive contour of I is the subtraction of I minus the convolution performed on I with the mask m.

4.2 Sketch-based 3D shape retrieval using Gabor local line-based feature (GALIF)

Many techniques in the sketch-based 3D shape retrieval field have been proposed in recent years. The majority of these techniques follow a two major steps methodology. The first step is to preprocess the 3D models to transform them into a set of images (rendered views from different angles of the 3D model itself) and a line drawing algorithm is then used over the images to obtain a common domain for 3D models and sketches. The second step consists of using the bag-of-features search framework, which has become the method of choice for affine invariant image retrieval during the last few years. However, to enhance the performance of

[1]Source: [53]

the methods for sketch-based 3D shape retrieval, this methodology is often combined with some filtering preprocessing techniques to reduce the size of the image view space before the retrieval stage.

The bag-of-features framework consists of representing each one of the images of a dataset by a large set of small local features. These features are quantized in a training stage to form a “visual vocabulary,” and each image is then represented by a histogram of its specific distribution of “visual words.” In the query stage, with a given image, the distribution of visual words of the query is computed, and the images with the most similar distributions are returned.

To be able to apply the bag-of-features framework, an image descriptor has to be used for the local feature extraction. Many image descriptors have been proposed in the literature, such as gradient, Gabor filters, Fourier descriptors, Zernike moments, shape descriptors, DSIFT, etc. However many authors in the sketch-based 3D shape retrieval field agree that the Gabor filters are the ones that achieve the best results. For this reason, we use the GALIF method for the sketch-based 3D shape retrieval stage in our proposal.

4.3 Gabor Local Line-Based Feature (GALIF)

Eitz et al. [27] proposed the Gabor Local Line-based Feature (GALIF) a method for sketch-based 3D shape retrieval. In this work, we represent each 3D model in a database as a set of 102 rendered views from the 3D model itself. We select these views uniformly distributed around the model and then transform each into a Suggestive Contour to apply the Bag-of-features framework. This process requires several steps: select the location and size of features in the images, transform the pixel set of the feature into a smaller dimensional feature vector, find the closest match of this vector in a group of predetermined clusters and for the whole image, count the occurrences of cluster matches. The set of clusters is obtained by clustering the feature vectors found in the images of the database. A signature for each image is generated as a histogram of cluster matches.

An essential aspect of this method is the selection of the views for the representation of the 3D models. Since there is no a priori knowledge about which viewpoint a user chooses when sketching an object, the underlying retrieval system must encode all potential viewpoints. To this end, the viewpoints consider view directions towards the barycenter of the 3D model itself. Consequently, nearby viewpoints have different orientations assigned to them, approximating a globally rotation-invariant indexing of the shapes. We evaluate two strategies for selecting the viewpoints for the GALIF method: uniformly distributed views and perceptually best views, and the best results were obtained with uniformly distributed views. The work “Sketch-based shape retrieval” [27] explains this strategy as follows:

“We generate d uniformly distributed directions on the unit sphere using k-means clustering. Starting from a highly tessellated triangle mesh of a unit sphere $M = V, T$, with V a set of vertices and T a set of triangles, we select a set S of d random seed vertices among V and perform Lloyd relaxations iteratively. After convergence, we return the resulting Voronoi cell centers as the view directions v_i. We use $d \in 7, 22, 52, 102, 202$”

Finally, after several experiments, they propose the optimal value for the number of viewpoints as $d = 102$

To sample local features, we generate $32 \times 32 = 1,024$ key points evenly distributed over the image views by sampling on a regular grid. For each key point, we define a local image patch as $n \times n$ cells around the key point; then, we apply a Gabor Filter over this patch to obtain a features vector. Finally, we generate a visual vocabulary for the representation using k-means clustering over the set of feature vectors with about 1000 visual words (centroids). Finally, we use a Tf-idf weighting function to build the histograms of visual word occurrences.

The main difference between the GALIF method for sketch-based 3D shape retrieval and other methods that use the Bag of Features framework is the use of the Gabor filters as descriptors for the feature extraction from the image views and the query sketches. A Gabor filter in the frequency domain is defined as:

$$g(u, v) = exp(-2\pi^2((u_\Theta - w_0)^2\sigma_x^2 + v_\Theta^2\sigma_y^2))$$

where $(u_\Theta, v_\Theta) = R_\Theta(u, v)^T$ is the standard coordinate system rotated by angle Θ, and the other parameters can be changed so the Gabor filter can be tuned:

- w_0 : peak response frequency.
- Θ : filter orientation.
- σ_x : frequency bandwidth.
- σ_y : angular bandwidth

Finally, to compute the feature space transform, a filter bank of Gabor functions g_i with k different orientations is defined. The sketch is then convolved with the Gabor functions from the filter bank to yield a set of filter response images.

$$R_i = \|idft(g_i * dft(I))\|$$

where I is the input sketch, denotes point-wise multiplication and $idft$ and dft denote the inverse/forward discrete Fourier transform. Given the number of orientations k we define the Θ's used for the filter bank as $\Theta \in \{0, \pi/k, ..., (k-1)\pi/k\}$. The other parameters were adjusted to make evaluation results invariant to the size of a sketch. To this end, two new variables were added: $\lambda = \sigma_x/\sigma_y$ and $linewidth = \sigma_x/w$ where w denotes the side-length of a sketch. The optimal values for the parameters are: $linewidth = 0.02$, $\lambda = 0.3$, $k = 4$ and $w_0 = 0.13$ [27].

To perform a query for a given user draw sketch, first, we extract the local descriptors using the same methodology, then we quantize the descriptors against the visual vocabulary. Finally, we represent the sketch as a (sparse) histogram of visual word occurrences. Then, we define a similarity metric to determine the similarity between two histograms so that we consider two images similar if their histograms point in the same direction. Then, with the histogram of the query sketch and the similarity metric, we can perform a match against the database of views represented as histograms.

4.4 Our approach: SC-GALIF

The main idea of the SC-GALIF method for image-based 3D shape retrieval is to automatically extract an image sketch s from a given query image to apply methods of sketch-based 3D shape retrieval over the extracted sketch s. To this end, we use line drawing algorithms over the query images that allow us to extract from them feature images similar to sketches.

In the SC-GALIF approach, we use suggestive contours [24] for the feature image extraction from the image queries because it is one of the line drawing algorithms mostly used in the literature to this end. We use different masks m for our suggestive contours implementation and the one with the best results was:

$$m = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & -8 & -8 & 1 & 1 \\ 1 & 1 & -8 & -8 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Figure 4.2 shows an image view of a 3D object and its respective suggestive contour extracted using the mask m defined before.

A second stage in our proposed approach for image-based 3D shape retrieval consists of using sketch-based 3D shape retrieval techniques on the suggestive contours extracted from the images. For this stage, we use the GALIF method with some variations. For the view extraction, we use 100 viewpoints for each 3D model evenly distributed around a sphere containing the 3D model itself. Each of the image views and query images are rendered to a resolution of $256x256$. Finally, we set the size of the image patch to $8x8$. We use the rest of the parameters with the same values proposed in [27].

For the experiment, we use the SHREC'12 benchmark [56]. For the query images, we

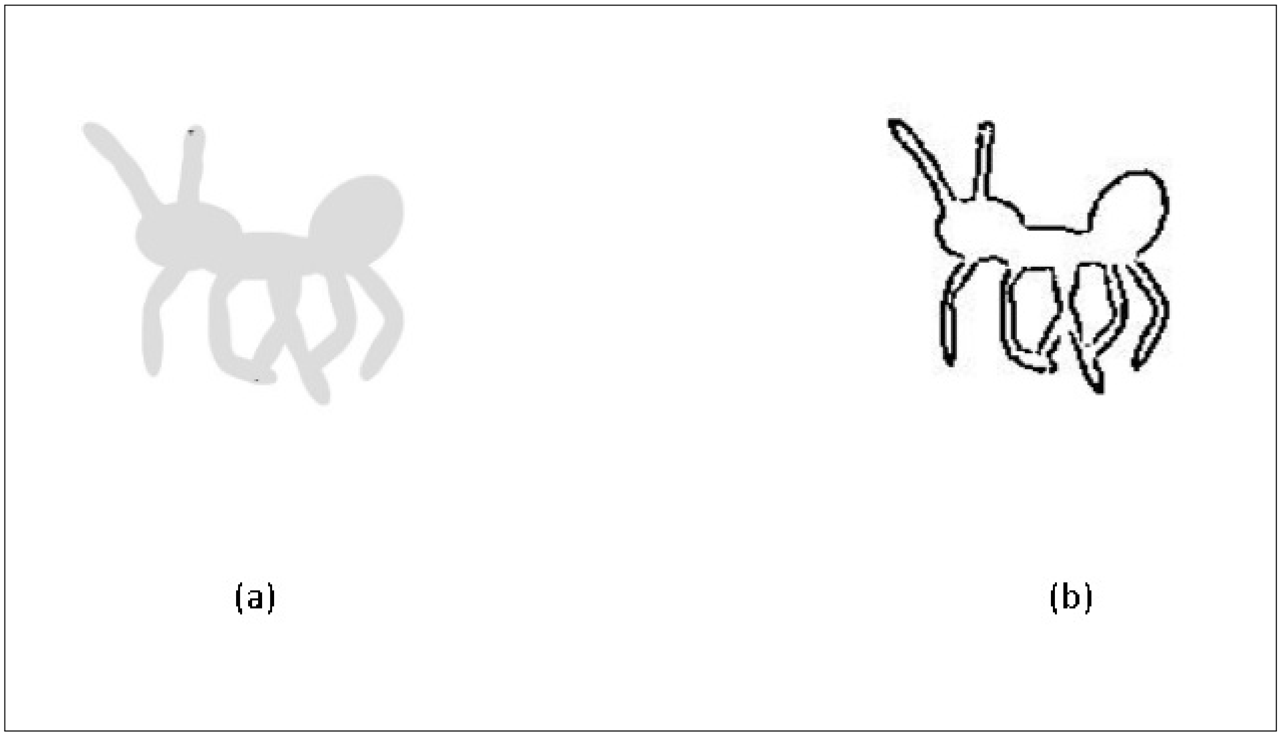

Figure 4.2: (a) A view from a 3D ant; (b) Suggestive contour extracted from the view

extract one view from a random viewpoint for each of the 400 models. We evaluate the proposed SC-GALIF method against four other methods (BOF-SBR, SBR-2D-3D, Orig-DG1SIFT, Dilated-DG1SIFT) for sketch-based 3D shape retrieval evaluated in [56] against the same benchmark. Figure 4.3 shows the results of the precision-Recall plot (PR) and the results for the metrics NN, FT, ST, and DCG are shown in Table 4.1.

As we expected, the SC-GALIF method outperforms three of the four methods for sketch-based 3D shape retrieval because the images and the 3D shape views are much more similar than the sketches and the 3D shape views. However the method SBR-2D-3D behaves similarly to SC-GALIF, even in several metrics has slightly better results.

The SBR-2D-3D [52] method is a sketch-based 3D shape retrieval algorithm that uses the same bag of features framework as GALIF, with the difference that a filtering preprocessing stage is added, in which a 2D-3D alignment between the query sketches and the 3D models is performed, before the retrieval stage. However, the SC-GALIF method applies only the bag of feature framework for the retrieval, which makes it simpler, more efficient, and faster in computation.

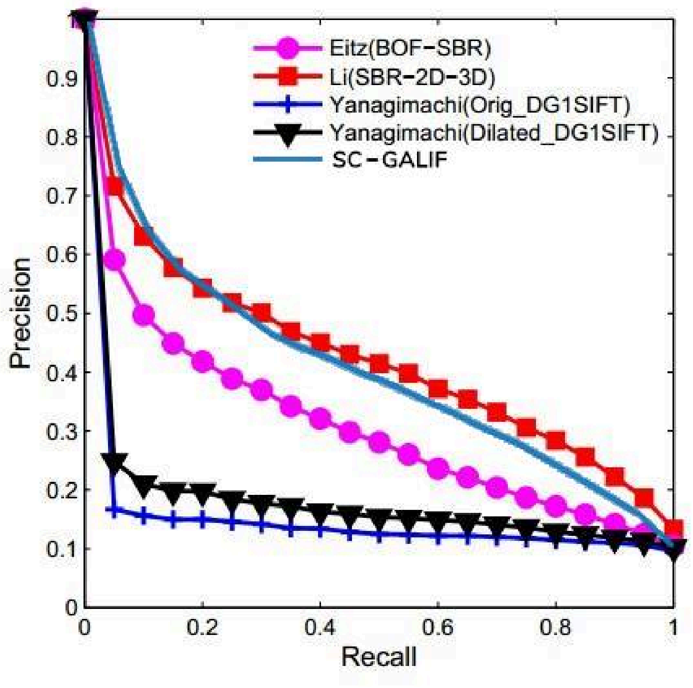

Figure 4.3: Precision-Recall plot for the methods: SC-GALIF, BOF-SBR, SBR-2D-3D, Orig-DG1SIFT, Dilated-DG1SIFT.

4.5 Summary

We propose an approach for image-based 3D shape retrieval using the advances in the sketch-based 3D shape retrieval field combined with line drawing algorithms. The general idea in the proposed SC-GALIF method is to use suggestive contours, one of the most efficient line drawing algorithms in the literature, to transform a given query image into a sketch image s. Then, to perform the retrieval of the 3D objects, s is used as input for the sketch-based 3D shape retrieval algorithm GALIF. For the evaluation of the SC-GALIF method we use the SHREC'12 benchmark [56] and we compare the obtained results by our proposal against the results of four other methods tested with the same SHREC'12 benchmark.

We can conclude that the results obtained by the SC-GALIF method for image-based 3D shape retrieval using the SHREC'12 benchmark are good since the algorithm behaves similarly to SBR-2D-3D, which obtained the best results of all the algorithms tested with the same benchmark. However, our proposed method directly uses the bag of features framework to perform the sketch-3D view matching for retrieval. In contrast to the method SBR-2D-3D, which adds a very costly filtering preprocessing step before using the bag of features framework.

Table 4.1: Other metric values for the methods: SC-GALIF, BOF-SBR, SBR-2D-3D, Orig-DG1SIFT, Dilated-DG1SIFT.

Method	NN	FT	ST	DCG
SC-GALIF	0.635	0.358	0.497	0.715
BOF-SBR	0.460	0.278	0.412	0.614
SBR-2D-3D	0.628	0.371	0.520	0.692
Orig-DG1SIFT	0.100	0.092	0.158	0.426
Dilated-DG1SIFT	0.168	0.120	0.212	0.462

Chapter 5

Slider: A new mid-level representation

The BOW formalism is a mid-level representation for multimedia objects that tries to capture the distribution of a set of features with respect to a set of visual words C. The set C is previously computed and is what we know as a visual dictionary. In other words, it is a histogram of the frequency with which the features belong to each visual word. However, BOW has the problem that all features belonging to the same visual word provide the same amount of information, regardless of how the features are distributed within that visual word.

5.1 Others mid-level representations

Some work has been proposed to use the information of the distribution of the features within a specific visual word, like BossaNova and BOH, both methods described in Chapter 2, Section 2.1.2. According to different criteria, these methods try to solve this problem by distributing the features belonging to the same visual word in bins. In the case of BossaNova the bins grouped the features according to their distance to the visual word so that the distance of the features belonging to the second bin is double the distance of those belonging to the first bin, the distance of those belonging to the third bin is triple of those belonging to the first bin and so on. On the other hand, the BOH formalism constructs bins so that all regions of hyperspace that represent these bins have the same hyper-volume. With this, what is sought is that all bins are equally probable.

Both methods have a greater degree of granularity in the distribution of features that belong to the same visual word. However, which is better depends on the task for which these methods will be used. In some cases it will be better BossaNova, in others BOH, and in others maybe a mid-level representation that is somewhere between BossaNova and BOH. This suggests that a whole spectrum of useful mid-level representations exists between BossaNova and BOH.

5.2 Proposal description

We propose a new mid-level representation, so-called Slider, that exploits the spectrum of representation between BossaNova and BOH. In this new representation, the features belonging to the same visual word are grouped in B bins with the particularity that these are constructed from linear combinations of BossaNova and BOH. This linear combination is governed by an adjustable $\beta \in [0, 1]$ parameter, which indicates how much each method contributes to the final formula so that the closer is beta to one, the closer is the slider to BOH, and the closer the β is to zero, the closer is the slider to BossaNova.

More in detail, the slider proposal is an extension of BOW described in Chapter 2, Section 2.2.1. This proposal aims to keep more information during the pooling step. For instance, in BOW, the pooling function summarizes the information in a pooling region m, the $\alpha_{m,j}$, into a single scalar value Z_m. In contrast, the slider computes a histogram of the distribution of the features within the pooling region. To do this, we divide each visual word into B bins. With this, each Z_m is now a vector of size B, which makes the final representation Z of size $M \cdot B$ where M is the number of visual words. With this, the pooling function g of our proposal is as follows:

$$g : R^N \to R^B$$

$$\alpha_m \to g(\alpha_{m,j}) = z_m$$

$$z_{m,k} = card(x_j \mid \alpha_{m,j} \in \alpha_m^{max} \cdot [\beta \cdot \sqrt[d]{\frac{k}{B}} + (1-\beta) \cdot \frac{k}{B}; \beta \cdot \sqrt[d]{\frac{k+1}{B}} + (1-\beta) \cdot \frac{k+1}{B}]) \quad (5.1)$$

Where B denotes the number of bins of each histogram z_m. Note that when $\beta = 1$ slider is exactly equal to BOH and when $\beta = 0$ then the slider is exactly equal to BossaNova. Another important change in our proposal is with respect to α_m^{max}. In our case α_m^{max} is the distance of the visual word to the farthest feature that belongs to said visual word. We obtain the z_m by concatenating the $z_{m,k}$ and the final image representation Z, same as in BOW is given by sequential coding, pooling, and concatenating: $Z = [z_1, z_2, ..., z_M]$.

We use this mid-level representation technique in the field of 3D shape retrieval. Specifically, to solve the image-based 3D shapes retrieval task, which consists of retrieving from a 3D models dataset those objects that are more similar to a given query image. To carry out this task, several stages of prepossessing the 3D shapes are needed. To explain these stages, we will divide them into three: the 3D shapes representation, the features extraction, and the building of the visual dictionary.

5.3 3D shapes Representation

To retrieve 3D shapes using images as a query object, we need to build a common domain for query images and 3D models. For this, we will use the image views representation, which are pictures taken from a 3D model from different angles. However, many authors propose different approaches to choosing the angles and the amount of image views to obtain the best representation of a 3D shape. For our representation of the 3D models, we use 100 image views uniformly distributed around a sphere containing the 3D model itself.

5.4 Features extraction

Once all the 3D models are represented as a set of image views, we need to use image descriptors for the feature extraction stage. Several image descriptors are used in the 3D models research field to this end: gradient, Gabor filters, Fourier descriptors, Zernike moments, shape descriptors, DSIFT, etc. However many authors agree that the Gabor filters are the ones that achieve the best results in similar tasks [54]. For this reason, we use these filters for our feature extraction stage.

To extract a set of features from a given image, we need to apply the Gabor filters on a set of predetermined samples extracted from that image. To this end, $32 \times 32 = 1,024$ key points evenly distributed over the image are generated. For each key point, a local image patch is defined as an $n \times n$ cells around the key point. Then, a Gabor filter is applied over this patch to obtain a features vector. For the experiments, we use image patch of size 8×8 and the features obtained from applying Gabor filters over those image patches are vectors of 64 dimensions.

5.5 Building of the visual dictionary.

As a final step, we need to build the visual dictionary for all the extracted features to apply the proposed mid-level representation techniques. To do that, we use a k-mean clustering with 1000 visual words (centroids) over all the features extracted. Once the visual dictionary is built, we are ready to obtain the mid-level representation of each image view and query image using a specific mid-level representation method and we need to define a similarity metric between those representations to perform the retrieval. In our case, we use the cosine distance.

5.6 Experiment and results

We carry out several experiments using the proposed methods, as well as the traditional methods of the area. For these experiments, we use two benchmarks the SHREC'12 benchmark [55] and ShapeNet [14]. The SHREC'12 benchmark is built based on the Watertight Model Benchmark (WMB) dataset which has 400 models divided into 20 classes, with 20 models each. For the query images, we extract one view from a random viewpoint for each of the 400 models. With this benchmark, we evaluate the mid-level representation methods BOW, BossaNova, BOH, and Slider. For the BossaNova and BOH methods, we use two different values for the number of bins: two and four. As for the Slider method, we use the number of bins equal to four and three different values for the β parameter $(0.1, 0.5, 0.9)$. The results of the experiments with the SHREC'12 benchmark are shown in Table 5.1.

The ShapeNet benchmark is much larger than the SHREC'12 benchmark. It is composed of more than 50,000 3D shapes divided into 55 classes. However, for us is very time-consuming to compute Gabor filters for this whole dataset. To address this problem, we selected 20 categories randomly for our experiments, and from each class, we extracted the first 200 3D shapes. We extract one view from a random viewpoint for each of the 8000 models for the query images. With this benchmark, we evaluate the mid-level representation methods BOW, BossaNova, BOH, and Slider. For all the methods, we use the number of bins equal to four. As for the Slider method, we use the same three different values for the β parameter as before. Table 5.2. shows the results of the experiments with the ShapeNet benchmark.

5.7 Summary

We develop a mid-level representation method to exploit the spectrum of representations between BossaNova and BOH. We perform several experiments using the proposal and other mid-level representation methods. For all the experiments, we use Gabor filters for the features extraction task and K-means to build the visual dictionary.

The results of the experiments show that BOH performs the best as a mid-level representation to solve the image-based 3D shape retrieval task. However, BOW performs very closely to BOH. From this find, we can conclude that having a greater degree of granularity in the distribution of features that belong to the same visual word can improve the quality of the representation.

On the other hand, the method with the worst results is the BossaNova, which worsens with the increase in the number of bins, as seen in Table 5.1. However, although both BOH and BossaNova methods propose having a greater degree of granularity in the distribution of features that belong to the same visual word, the distribution of those features makes one method perform much better.

Finally, we can see that the slider method obtains better results than BossaNova and worse than BOH. However, as the parameters that govern the slider method cause it to behave close to BOH, the results improve significantly.

Table 5.1: Experiment results for the SHREC'12 benchmark

Method	Number of Bins	param	NN	FT	ST	DCG
BOW			0.62	0.3648	0.4937	0.7183
BossaNova	4		0.59	0.3553	0.4911	0.7092
BOH	4		0.6225	0.3656	0.4981	0.7178
BossaNova	2		0.605	0.3681	0.5072	0.7175
BOH	2		0.6225	0.3660	0.4991	0.7182
Slider	4	0.5	0.61	0.3502	0.4799	0.7073
Slider	4	0.9	0.6225	0.3645	0.4981	0.7178
Slider	4	0.1	0.5875	0.3508	0.4952	0.7079

Table 5.2: Experiment results for the ShapeNet benchmark

Method	Number of Bins	Param	NN	FT	ST	DCG
BOW			0.5125	0.2097	0.3245	0.6996
BossaNova	4		0.441	0.1683	0.2651	0.6706
BOH	4		0.5175	0.2097	0.3245	0.6997
Slider	4	0.5	0.4822	0.1959	0.3058	0.6889
Slider	4	0.9	0.5105	0.2095	0.3242	0.6994
Slider	4	0.1	0.444	0.1685	0.2670	0.6707

Chapter 6

Deep features for image-based 3D model retrieval

In recent years, deep learning techniques for extracting learned features have been used in several fields from multimedia information for classification and retrieval, with excellent results. For instance, in areas such as text processing and image processing, several network architectures specialized in extracting learned features have been proposed such as Autoencoders and Siamese networks. However, a more direct approach for extracting learned features is to train a standard artificial neural network architecture for the classification. Then, when the network finishes training, a hidden layer of this is used to represent the objects. This last approach, despite not being the standard approach for learned features, is used in different works since it is often easier to train and produce faster results.

It is in our best interest to use the learned features in the cross-modal 3D shape retrieval area. In this paper, we propose an initial approach for cross-modal 3D shape retrieval, specifically for image-based 3D shape retrieval using learned features. For the extraction of the latter, we use a standard CNN, which we train for classification tasks. Once the training is finished, we use a hidden layer of the CNN already trained as a vector representation of the images.

6.1 Proposal description

As mentioned earlier, we propose solving the image-based 3D shape retrieval problem using learned features. The image-based 3D shape retrieval problem consists in retrieving the 3D models of a database that are most similar to a given query image; for this objective, it is necessary to build a common domain for 3D models and query images. To solve this problem, we used image views that are 2D projections of a 3D object from different angles. In our case, each 3D model was represented as a set of 100 image views with resolution 128×128 evenly distributed around the sphere containing the 3D model.

Once extracted the image views of all the 3D models, we trained a CNN architecture to classify the image views according to the classes to which they belonged. This network comprises four blocks of two convolutional layers and one pooling layer with stride two to reduce the dimensionality by half. The last block output is the input to a fully connected layer, which will be the deep feature representation of the image after the training phase. Finally, this layer is linked to another fully connected layer with size C, where C is the number of classes for classification. We use the Relu function to activate the layers except for the last layer, in which we use the standard softmax function for classification. We also use cross-entropy as the lost function and Adam optimizer. Figure 6.1 shows the architecture of the network.

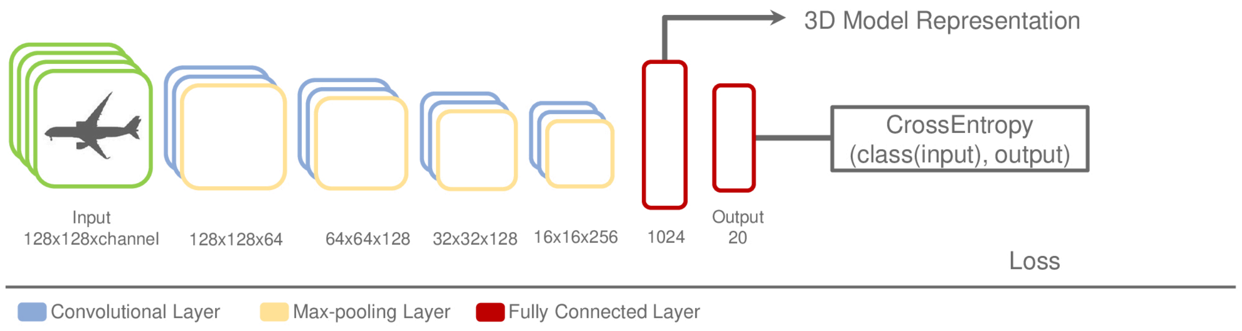

Figure 6.1: Convolutional Neural Network architecture for computing learned features, using a classification architecture composed of four blocks of two convolutional layers and one max pooling layer. Each block reduces the dimensionality of the input from 128×128 to 16×16 in the last block before connecting it to two consecutive fully connected layers and applying the soft-max activation function over the last layer.

Using the mentioned $CNNs$ architecture and all the image views, we train the model for image classification until the network achieves good accuracy (ideally above 90 percent). Once the training is over, we use the first fully connected layer of the network as a learned feature so that the vectors will be of size 1024, and we represent all the image views by their respective learned feature.

In the query stage, given a specific query image, we first used the trained network to extract the vector representation of the query. Then, we matched the vector of the query image and the vectors of all the image views to obtain the retrieval results. To perform this last step we needed to use a similarity metric between vectors; in our case, we used the cosine distance.

6.2 Experiments and Results

An essential aspect of our work is to compare our approach using learned features to solve the image-based 3D shape retrieval task and the traditional approach of the field using engineered features. To achieve this comparison, we implemented a second solution to the problem using our SC-Galif proposal from Chapter 4.

We conducted several experiments using our new proposal for learned features and our implementation of the SC-GALIF method for hand-engineered features. In our experiments, we used the ShapeNet benchmark [14], composed of 51300 3D models grouped into 55 categories. However, for us is very time-consuming to compute Gabor filters for this whole dataset. To address this problem, we first use a dataset sample to compare against SC-GALIF. To build this sample, we select 20 random categories, and from each category, we choose 200 3D models. Finally, we divide each of the 20 categories into 100 3D models for training and 100 for testing. We chose a random image view of each of the 2000 3D test models for the query images. Table 6.1 shows the results of the experiments using the sampled dataset. We also compute the retrieval results of this proposal using the entire dataset for future comparison. Table 6.2 shows these last results for the MAP and DCG metrics.

The experiments show that the results obtained with our proposal for image-based 3D shape retrieval were significantly better than the results obtained with Gabor filter histograms for all the metrics. Considering this is an initial proposal for using learned features to solve this task, we think we can obtain even better results with a more complex architecture designed for this specific task.

6.3 Summary

We proposed solving the image-based 3D shape retrieval problem using $CNNs$ to extract learned features. For this solution, we first represent the 3D shapes as a set of 100 image views. Then, we use these image views to train a $CNNs$ for classification until we achieve 90 percent accuracy for the classification task. After the training, we use a fully connected layer of the network to represent the images. Finally, we use cosine distance to measure the distance between the representations and compute the retrieval results.

Our solution was tested against a hand-engineered features solution using Gabor filter histograms. According to the results of the experiments, we could see that the learned features were better than the Gabor filter histograms in solving the image-based 3D shape retrieval task. We also compute the results of our proposal against the entire dataset to compare with future proposals for image-based 3D shape retrieval.

Table 6.1: Evaluation of the performance of the learned features and the engineered features using a sample of the ShapeNet benchmark.

Feature Type	NN	FT	ST	DCG
Engineered	0.5032	0.2230	0.3401	0.7038
Learned	0.7522	0.3995	0.5455	0.8074

Table 6.2: Evaluation of the performance of the learned features using the full ShapeNet benchmark.

Feature Type	MAP	DCG
Learned	0.2801	0.6413

Chapter 7

Retrieval techniques for Cultural Heritage objects

We present two approaches for Greek pottery retrieval, "A sketch-aided retrieval approach for incomplete 3D objects" and "Motif-driven Retrieval of Greek Painted Pottery." Both proposals have small datasets, so we need to test the feasibility of using deep learning techniques to solve both approaches. We detail below both proposals as well as our contributions. This work was in collaboration with the CGV group at TU Graz.

7.1 A sketch-aided retrieval approach for incomplete 3D objects

A major challenge for a computer-aided search is that only a fraction of the excavated Cultural Heritage objects are complete. Still, most of them are present in various degrees of fragmentation or erosion, making it difficult to use them directly as input for shape comparison and search.

This proposal addresses this issue by defining an appropriate workflow for content-based retrieval of 2D image data from incomplete 3D objects. The workflow is built around a human-in-the-loop approach, allowing experts to provide sketch-aids for adding missing shape information, query weighting, and visual result comparison.

Missing parts of Cultural Heritage objects can be estimated with high precision by domain experts. The basic idea of sketch-aids is to allow users to create additional object structure, which is filled by a texture inpainting step, which serves as input for content-based retrieval. Fig. 7.1 shows the complete pipeline of the proposal.

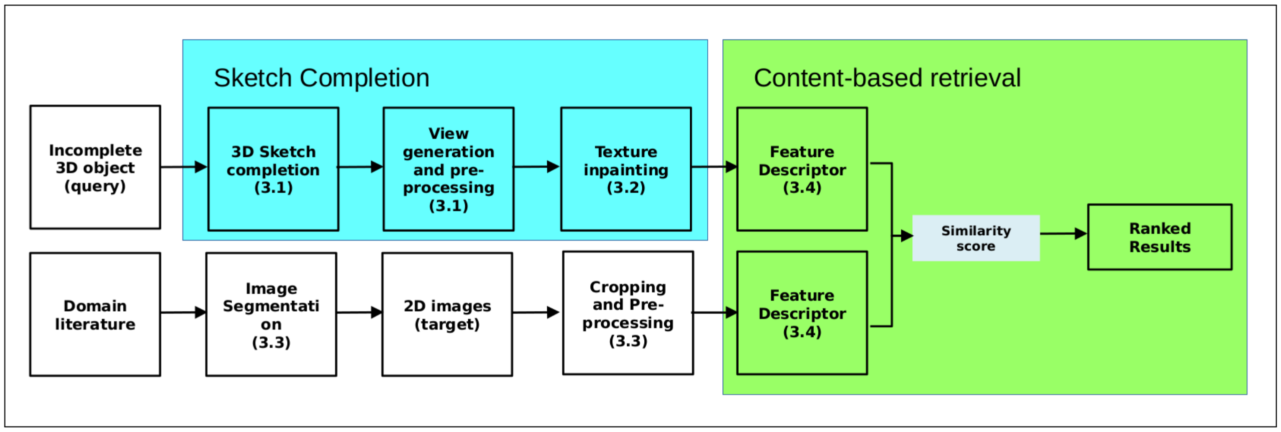

Figure 7.1: Complete Pipeline for sketch-aided retrieval approach for incomplete 3D objects (top left side of the figure are the preprocessing steps for the incomplete 3D object query to extract the feature descriptor, the bottom left side of the figure are the preprocessing steps for 2D image data to extract the feature descriptors, and finally the right side of the image is the content-based retrieval using the query descriptor and the 2D image descriptor)

7.1.1 Content-Based-Retrieval

After the sketch-completion, texture inpainting and preprocessing steps, the query and target collection are available, allowing for a conventional image feature-based similarity search. There exists a wide range of established 2D image features, incorporating global features like Color Histogram, Edge Histogram, Tamura, Color and Edge Directivity Descriptor (CEDD), Histogram of Oriented Gradients, and local features like Scale Invariant Feature Transform (SIFT). Global features are computed from the whole image and can be further divided into features based on color, texture, and shape. The group of local features relies on "significant" points in the image, which noticeably differ from their neighborhood.

We found that color-based features performed poorly for our specific use case of pottery images, as they exhibit mostly undiscriminating color distributions, and many target images are available only in grayscale in the first place. In preliminary experiments, global shape-based feature descriptors yielded the most promising results due to the query and the search space images being depicted with the whole object in view and with a characteristic orientation. Two descriptors, yielding appropriate results, are evaluated in more depth: The HOG [20] feature descriptor and the shape contour descriptor (SCD) [6].

7.1.2 Deep Features for Content-Based-Retrieval

One of the strengths of artificial neural networks is deep feature extraction. In this work, the research was made using traditional engineering features given the fact that the database was too small. However, this research needed to perform a comparison between the results obtained using engineered features and deep features. To that end, we implemented two different types of deep features.

For the first one, we use transfer learning with the mobilenetv2 [74] for the source task to train a standard CNNs for image classification. Then, we use the last layer of our trained network as the image representation.

Our first approach consists of using standard CNNs for image classification. In our case, we use two different networks: ResNet and MobileNetV2. These networks are well known in the field, with outstanding results for different image processing tasks.

ResNet, learns residual functions with reference to the layer inputs. This architecture stacks 50 residual blocks on top of each other to form the network. The Residual Blocks are skip-connection blocks composed of multiple convolutional layers and a pooling layer at the end.

The architecture of MobileNetV2 contains the initial fully convolution layer with 32 filters, followed by 19 residual bottleneck layers. It is based on an inverted residual structure where the residual connections are between the bottleneck layers. In addition, the intermediate expansion layer uses lightweight depthwise convolutions to filter features as a source of non-linearity.

For both ResNet and MobileNetV2, we use pre-trained versions with high accuracy for image classification. Then we apply transfer learning techniques and adapt them to our domain. Finally, we use the layer's output before the last fully connected layer to represent the image.

For the second approach for deep features, we use an auto-encoder architecture. We train this network over the whole dataset for 10000 iterations. After the training, we use the code layer of the network as the image representation.

7.2 Motif-driven Retrieval of Greek Painted Pottery

This approach for retrieval of Greek painted pottery presents an integrated retrieval system that combines the interactive specification of a query motif by a user with a suitable unsupervised motif extraction pipeline of a search space to allow for a search of specific scenery depicted on painted pottery. By motif, we refer to an ornament or a figure, which is itself not a composition or part of another motif.

It is crucial that this segmentation step is performed in a robust and reliable automatic way to be useful for a content-based search engine. In the case of Greek ancient painted pottery, the motifs of interest are mostly painted in two major styles: red figures on a black background (red-figure pottery) and vice versa (black-figure pottery), yielding objects with a binarized colorization.

7.2.1 Segmentation and Feature Extraction for Pottery Motifs

The approach poses two major challenges. First, generating the search space by discovering and extracting image segments, possibly corresponding to an individual motif, from domain-specific images. Second, the similarity retrieval is based on the contour of the extracted segments.

For the unsupervised motif extraction from Greek painted pottery images, we use several standard image segmentation algorithms and deep learning techniques for semantic image segmentation. For the standard image segmentation algorithms, we obtained the best results using Graph-cut [11] based Segmentation, but still, these algorithms could not get proper segmentation for the entire data set. As for the deep learning techniques, we use the deeplabv3+ [16] model from the TensorFlow library.

For similarity matching, we rely on shape as well as color. In terms of shape, the shape context feature descriptor has been found to be a good choice for this application. This local shape feature descriptor takes a representation of a shape by a number of points and defines a feature vector for each point based on the relative location, directivity, and distance of all the other points in the form of a distribution histogram with b_r radial bins and b_θ angular bins.

7.3 Experiment and results for sketch-aided retrieval approach for incomplete 3D objects

We encounter two major problems in our research in CGV using deep features for image retrieval and Image-based 3D model retrieval. First, the database was small for the use of artificial neural network techniques. Second, given the lack of annotated data, we had to use unsupervised techniques. We use two approaches for deep feature extraction to address these problems: CNNs for image classification with transfer learning and auto-encoder architectures.

However, a manual evaluation with many queries is not feasible because several steps of the proposed pipeline require user interaction. For this reason, we produce qualitative results by visualizing the query results. Two feature extraction methods were the most promising: Histogram of Oriented Gradients (HOG) and shape contour descriptor (SCD).

So far, the deep features approaches have not surpassed the engineered features' effectiveness due to the lack of annotated data. However, we believe this result could improve considerably by increasing the annotated data.

7.4 Experiment and results for Motif-driven Retrieval of Greek Painted Pottery

We select two published datasets exhibiting a representative range of various motifs on different vessel shapes. Those publications are the Corpus Vasorum Antiquorum Berlin 13 [ZE13] and Corpus Vasorum Antiquorum Dresden 3 [Esc18]. Almost all of the depicted vessels exhibit motifs on their surfaces. All images depicting whole vessels with motifs were manually selected, resulting in a data basis of 57 images from CVA Berlin 13 and 42 images from CVA Dresden 3.

For the experiments, we adapt a pre-trained deeplabv3+ model implemented in the TensorFlow library with mobilenetv2 as the backbone of the network. We use this architecture and several other standard image segmentation algorithms for the segmentation step of the proposal. Then, we visually select the best result of each segmentation algorithm used previously over the whole dataset.

Once again, given the lack of anatoted data and data in general, we could not obtain quantitative results for the proposal, and we produced qualitative results by visualizing the query results. The deeplabv3+ model outperforms every other segmentation algorithm since we obtained proper segmentation for most of the images in the entire dataset. Figure. 7.2 shows visual results.

Once again, given the lack of anatoted data and data in general, we could not obtain quantitative results for the proposal, and we produced qualitative results by visualizing the query results. The deeplabv3+ model outperforms every other segmentation algorithm since we obtained proper segmentation for most of the images in the entire dataset. Figure. 7.2 shows visual results.

7.5 Summary

We work in collaboration with the CGV group at TU Graz. In this collaboration, we apply deep learning techniques in practical scenarios of 3D shape retrieval. Specifically, we collaborate on two works, "A sketch-aided retrieval approach for incomplete 3D objects" and "Motif-driven Retrieval of Greek Painted Pottery".

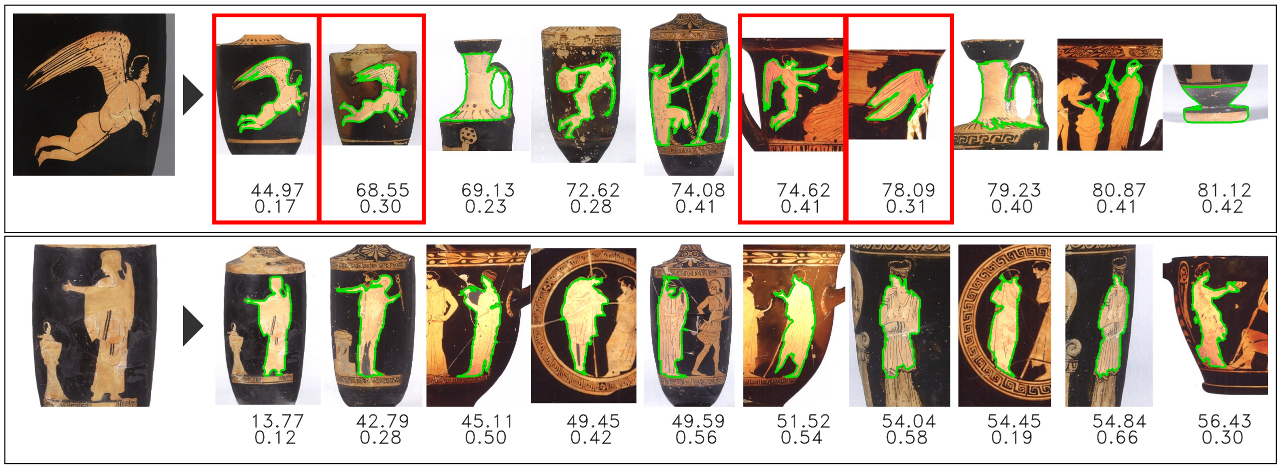

Figure 7.2: Results for Motif-driven Retrieval of Greek Painted Pottery

The first work “A sketch-aided retrieval approach for incomplete 3D objects” defines an appropriate workflow for content-based retrieval of 2D image data from incomplete 3D objects. We need to extract features from the image for their representation and further processing to fulfill this. We explore engineered and learned features using artificial neural network techniques in this step. However, due to the lack of annotated data, engineered features proved to be a more efficient technique in this scenario.

In the second work “Motif-driven Retrieval of Greek Painted Pottery.”, we propose an approach for retrieval of Greek painted pottery present. This approach consists of an integrated retrieval system that combines a user’s interactive specification of a query motif with a suitable unsupervised motif extraction pipeline of search space to allow for a search of specific scenery depicted on painted pottery. We use a neural network-based approach to extract the motif from the pottery. Specifically, we adapt a pre-trained deeplabv3+ model for semantic segmentation. In this case, the deep learning approach outperforms every other segmentation algorithm since we could obtain proper segmentation for most images in the entire dataset.

Chapter 8

A New Convolutional Architecture for 3D Model Embedding and Retrieval (SE3D)

We propose a convolutional architecture for 3D model embedding (SE3D) 8.1 “Reproduced with permission from Springer Nature” [43]. The idea is to obtain a deep 3D model representation for effective 3D model retrieval. To this end, first, we represent the 3D models as a set of image views. Then, this representation is used as input to train our proposed convolutional model. The problem is that a standard convolutional network uses an image as input; in our case, we have a set of images as input. To solve this, we model the information as a multichannel image where the number of channels is the number of image views that we use to represent a 3D model. In other words, we combine all the image views in a single tensor with no specific order. Then, we use this tensor as the input of our network.

With this representation to handle the image view sets as input, we develop our convolutional architectures for 3D model embedding. We start with a standard Autoencoder architecture, and we iterate, adding new components to our architecture to refine and enhance the performance of our proposal. We combine our architecture for 3D model embedding with a technique to refine the image views. This last step aims to find better image views from a set of image views instead of using fixed viewpoints to render image views for each 3D model.

8.1 Autoencoder Network

As mentioned before, our proposal starts from a base convolutional Autoencoder architecture. In this model, we connect our proposed multichannel input to a standard convolutional neural network, which serves as the model's encoder. Then, we connect the encoder to a fully connected layer (the bottleneck), which, at the same time, we link to the decoder of the network.

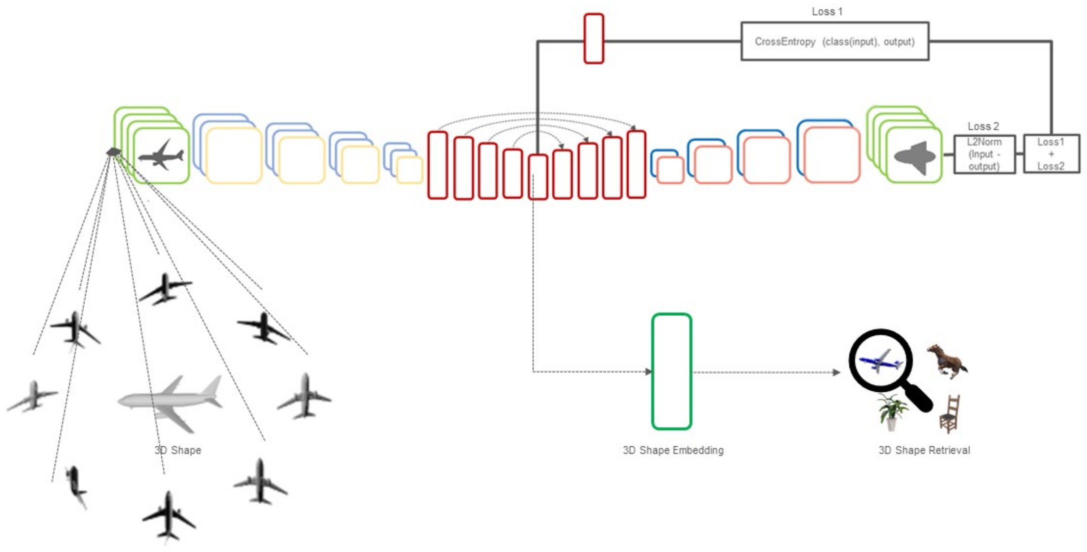

Figure 8.1: Proposal overview: A deep architecture for computing embeddings. It uses an Autoencoder combined with a classification network and various constraints to improve the quality of the embeddings that are later used in 3D shape retrieval tasks.

The network's encoder has four blocks of two convolutional layers and one max pooling layer with stride two, which means that each block reduces the dimensionality by half. Each convolutional layer has a kernel of 5×5 and channels equal to $64 * 2^k$, where k is the respective block $(1, 2, 3, 4)$.

The network's decoder also has four blocks composed of one transposed convolutional layer, also known as a deconvolutional layer. One un-pooling layer with stride two doubles the dimensionality of each block. We select a zero-padding technique for the un-pooling layer. We implement the un-pooling layer to copy the values in the input feature map to their respective position in the output feature map and set all the other positions to zero. Many works have used zero padding techniques to increase a feature map dimensionality with excellent results [95, 22, 26]. Figure 8.2 "Reproduced with permission from Springer Nature" [43] shows the un-pooling layer used in the model. Each deconvolutional layer has a kernel of 5×5 and channels equal to $64 * 2^k$, where k is the respective block $(4, 3, 2, 1)$.

Finally, we use the L_2 norm of the difference between the output and the model's input as reconstruction loss. Equation 8.1 shows the formula of the loss function, where the I_i are each of the pixels in the input, and the O_i are the respective pixels in the network's output (the reconstruction of the input). Figure 8.3 "Reproduced with permission from Springer Nature" [43] shows our convolutional Autoencoder model.

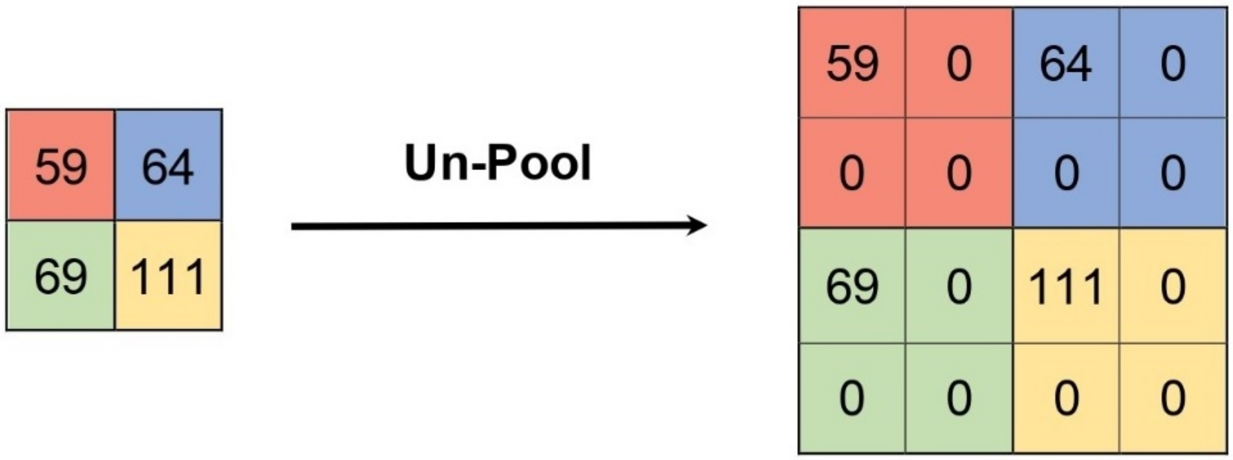

Figure 8.2: Un-pooling layer used for the decoder of the model

$$Loss = \sqrt{\sum_{i}^{n}(I_i - O_i)^2} \tag{8.1}$$

8.2 Classification Network

Autoencoder models are dominant in compressing and computing representations of the data. However, these models do not learn relations between objects because they only use the input information to train. We address this problem with another network consisting of a CNN for classification. This network uses the same architecture as the Autoencoder network until the bottleneck component. After that, we connect the bottleneck to a fully connected layer of size C, where C is the number of classes for classification. Then, we apply a standard soft-max activation function over this layer. Finally, we use cross-entropy as the loss function. We train this model until we obtain very high accuracy, and then, we use the bottleneck layer as the embedding. Equation 8.2 shows the formula of the loss function, where the N is the number of classes, the y_i are the truth labels, and the p_i are the predicted probability for each class. Figure 8.4 "Reproduced with permission from Springer Nature" [43] shows the convolutional model for classification.

$$Loss = -\sum_{i}^{n} y_i \times Log(p_i) \tag{8.2}$$

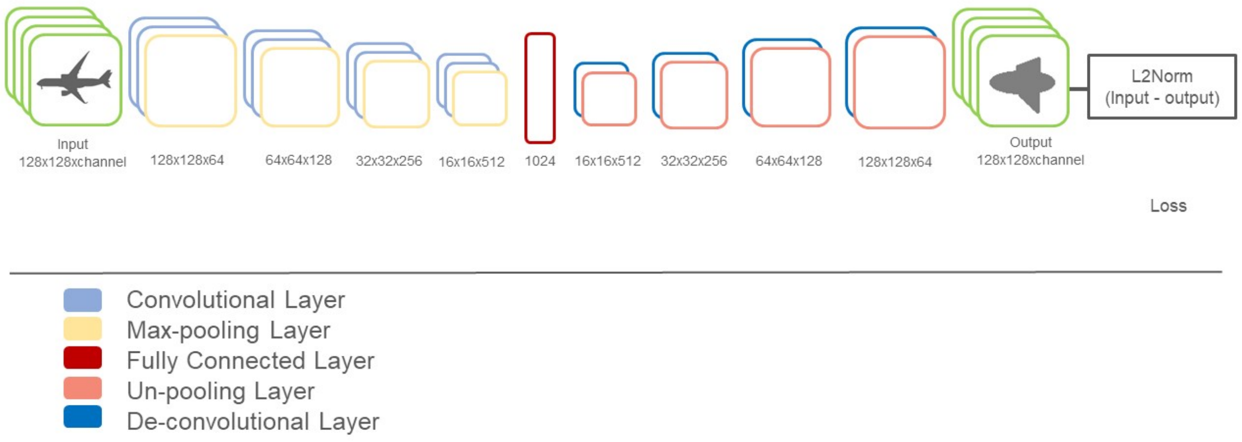

Figure 8.3: Network for computing 3D shape embeddings using an Autoencoder architecture composed of its three main components: encoder (four blocks of two convolutional layers and one max pooling layer), bottleneck (one fully connected layer), and decoder (four blocks of one deconvolutional layer and one un-pooling layer).

8.3 Combination of the Autoencoder and the classification network

We want our 3D model embedding network to have high compression power and learn the relation between the 3D models. To address this, we use a combination of both already described models. This model uses the Autoencoder and the classification network architecture simultaneously. In other words, for a given input, we compute two losses ($Loss1$ and $Loss2$). $Loss1$ is the L_2 norm of the difference between the output of the Autoencoder and input, and $Loss2$ is the loss computed with the cross-entropy function in the classification network. Finally, the complete loss of this model is the sum of $Loss1$ and $Loss2$. After the training, we again use the bottleneck layer as the embedding of the 3D models. Figure 8.5 "Reproduced with permission from Springer Nature" [43] shows this combined model.

8.4 Improving the embedding

With the combination of the classification network and the Autoencoder, we achieve high compression power, and we also manage that our model learns the relation between the 3D models. However, we still need to solve two major problems. First, the number of images in our input must be small because our model needs to learn the input after the encoded phase

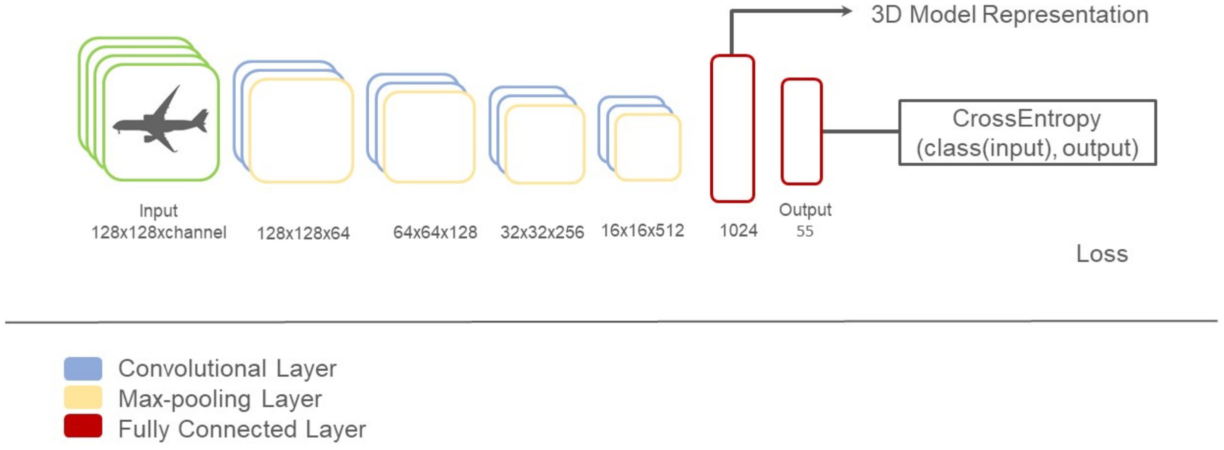

Figure 8.4: Network for computing 3D shape embeddings, using a classification architecture composed of four blocks of two convolutional layers and one max pooling layer. Each block reduces the dimensionality of the input from 128×128 to 16×16 in the last block before connecting it to two consecutive fully connected layers and applying the soft-max activation function over the last layer.

from 1024 sized vector. Second, our embedding representation is also a 1024 sized vector, and we would like to have a more compressed representation.

To solve the second problem, we add two more fully connected layers, the encoder component of the network. The first one reduces de dimensionality of the vector to 512, and the second reduces the dimensionality further to 256. We also add the respective two fully connected layers to the decoder of the model. However, these changes increase our first problem since now our model needs to learn the input from a 256 sized vector.

To address this aggravated second problem, we incorporate skip connections between the encoder and decoder of the model. This architecture aims to learn the input using the bottleneck of the network and the information before the bottleneck as well. Figure 8.6 "Reproduced with permission from Springer Nature" [43] shows our proposed architecture for computing the 3D model embeddings. As shown in this figure, we add the output of the fully connected layers in the encoder to their respective fully connected layers in the decoder. By doing this, we reinforce the amount of information passed to the decoder of the network, which allows a better reconstruction of the input. This improvement of the embedding over the last architecture constitutes our full proposal named SE3D.

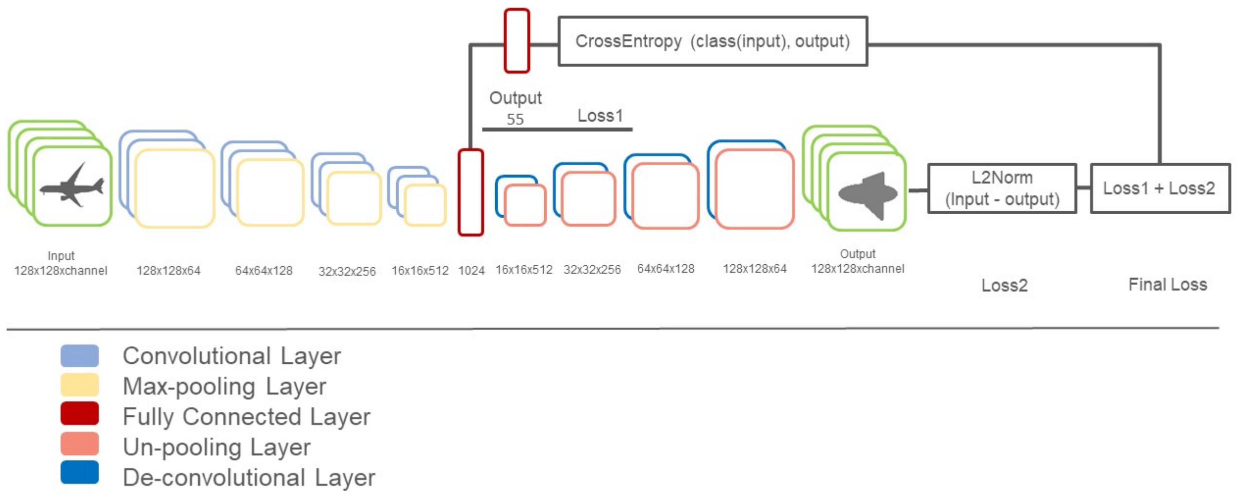

Figure 8.5: Network for computing 3D shape embeddings using a combination of Autoencoder and classification architectures. These architectures share the first half of the network and then compute two losses for the Autoencoder and classification separately. Finally, the losses are combined into a single final loss function.

8.5 Image view extraction

A recurrent problem in the 3D model processing field using image views is precisely how to extract the image views. Many methods to extract the "best image views" from a given 3D model have been proposed over the years. However, these methods are very task-dependent, and there is no wide-accepted agreement on the best way to extract image views [37, 35, 2].

We propose a method to select the "best image views" from a large set of image views representing a 3D model. To do this, we train a deep convolutional architecture for classification using the whole set of image views. After the training phase, we use the trained network to select the views that best classify the 3D model. We describe this process below.

8.5.1 Network for classification of the image views

The first stage of our proposal for selecting the best image views is to train a CNN for 3D model classification using the whole set of image views representing the 3D models. We require that our network classify the 3D model using the entire collection of image views. We use a standard convolutional neural network for image classification as the core of this proposed architecture to fulfill this. Then, we use this core network for two tasks for each

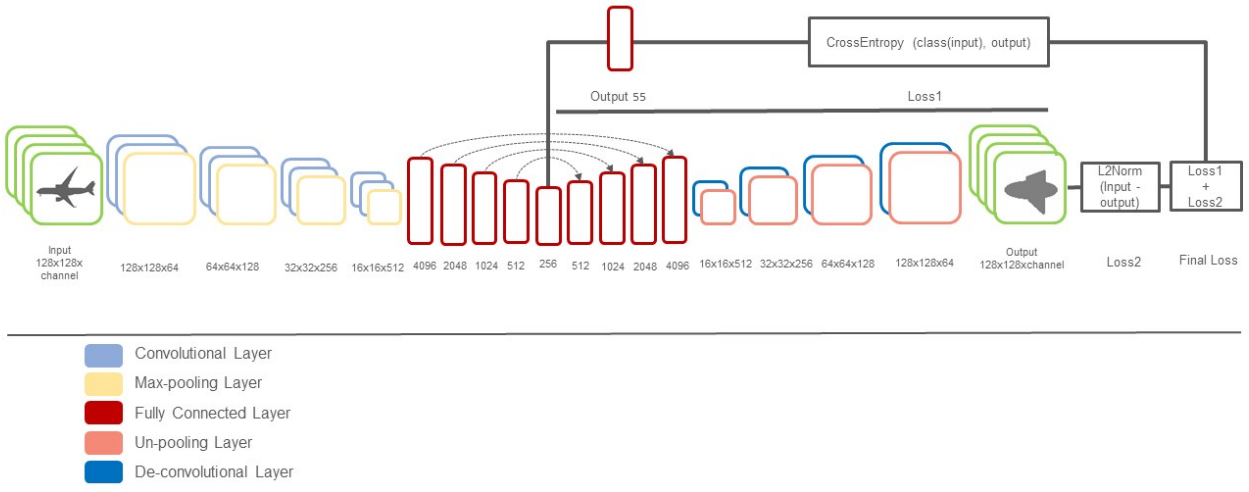

Figure 8.6: Full network proposal for computing 3D shape embeddings using a combination of an Autoencoder architecture and a classification architecture, incorporating skip connections and increasing the bottleneck capacity by adding fully connected layers.

image view. The first task is to compute classification loss for each image view, and the second task is to obtain a deep representation for each image view. We accumulate both these losses and deep representations by adding them. Finally, we use the accumulation of deep representations as the input to another fully connected network to produce a classification loss. The final loss of our model is the sum of this last loss with the accumulation of classification losses for each image view. Figure 8.7 "Reproduced with permission from Springer Nature" [43] shows the architecture of this network.

8.5.2 Selection of the image views

We use the trained network to compute the class of the 3D models and the network's output for each image view (the probability vector without applying the softmax function). Then, we sort the image views according to the probability of the views to belong to the same class as the 3D model. For example, if the class of a 3D model is one, we sort all the image views according to the value in the position one of their probability vectors. Finally, we use this sorted list to select the potential "best image views" according to two different criteria: the number of views we want for the final representation of the model and the proximity between the views. We use this last criterion to ensure that all the views selected are not from the same perspective of the 3D model. For example, we could choose the first n views of the

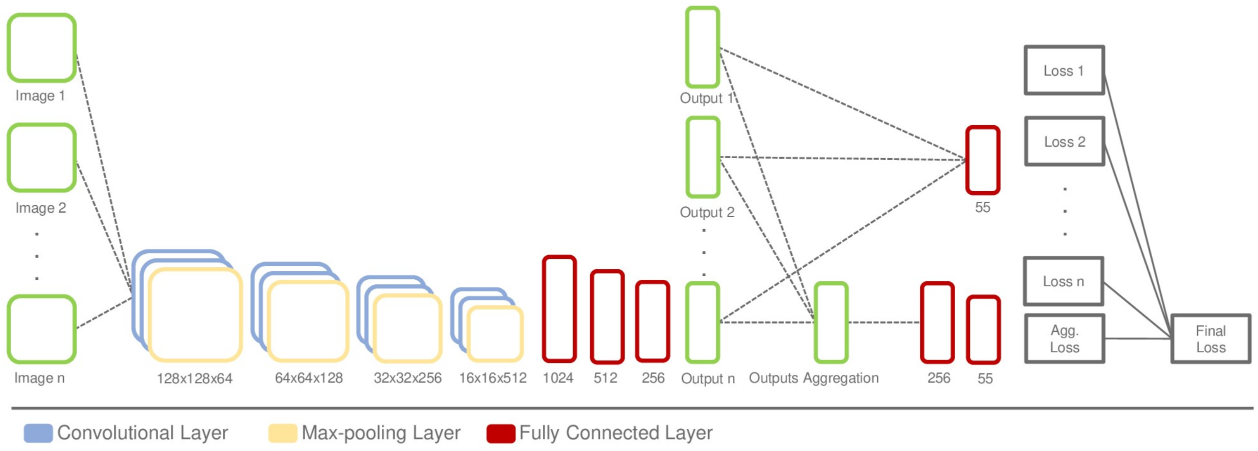

Figure 8.7: Network architecture for image view classification optimized to refine a view set representing a 3D shape. The goal is to identify and select the image views best suited for accurately classifying the 3D shape in question.

sorted list, where n is the final number of image views that we will use to represent the 3D model.

8.6 Overview

We propose a pipeline for 3D shape retrieval using image view representations. We begin the retrieval process by extracting the image views from the 3D shapes. We use the Stanford-Shapenet-renderer script, but any technique or tool can be used for this end. After this, we are ready to feed the image to the neural networks to train our models. However, we add an intermediate step for filtering the image views when dealing with noisy data.

After we train our artificial neural network models with the sets of image views, we select the output of a specific layer as the representation of the 3D shape. After that, we use cosine similarity to measure the distance between these representations. Finally, we retrieve similar 3D models according to the distance between them. Figure 8.8 “Reproduced with permission from Springer Nature” [43] shows the complete pipeline for the retrieval process.

8.7 Experiments and Results

In this section, we discuss all the experimental setups, describe the benchmark and the preprocessing of the data, and give the technical aspects of our three proposed architectures.

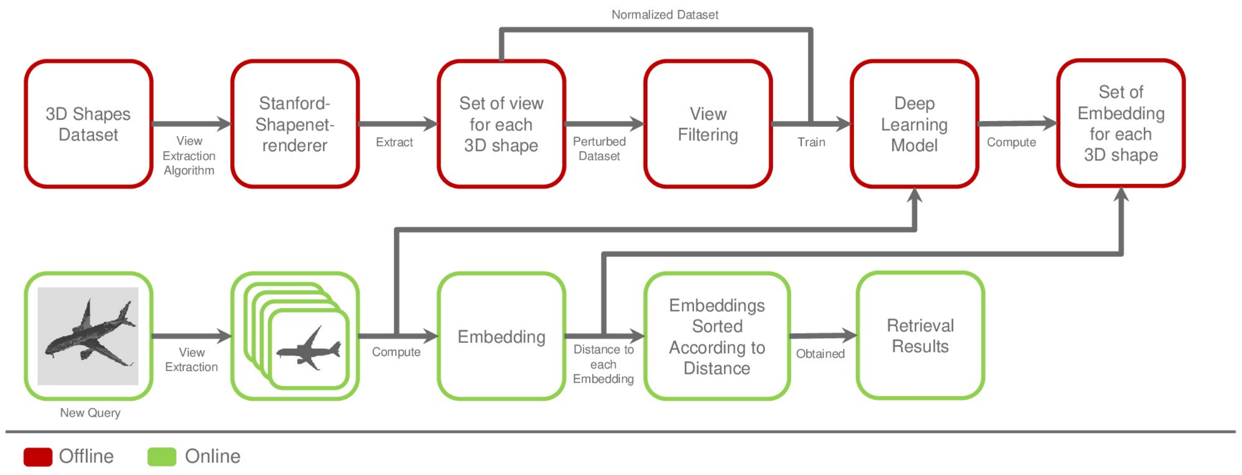

Figure 8.8: Entire pipeline for the retrieval process. We start from the offline stage, where we train an artificial neural network to compute embedding representations for every 3D shape. In the online stage, we compute the embedding for a given 3D shape and measure the distance from this embedding to every previously computed embedding to obtain the retrieval results.

8.7.1 Data configurations

We use two different datasets for the experiments, the ShapeNet benchmark [14] and the ModelNet [99]. On the one hand, the ShapeNet benchmark comprises 51,300 3D models grouped into 55 categories, and they are provided in OBJ format. The dataset count with two versions, consistently aligned (normalized dataset), and a more challenging dataset where random rotations perturb models. The dataset provides a split 70%/10%/20% for training, validation, and testing, respectively. We conducted several experiments using both versions of the dataset. On the other hand, ModelNet [99] has two versions: ModelNet10 where the models are categorized into ten classes, and ModelNet40, where the models are categorized into 40 classes. Both versions are composed of CAD models manually aligned but not scaled. This dataset also provides its own training/testing split.

As mentioned before, our proposed models consume sets of image views as input. An image view is a picture of a 3D model from a predefined viewpoint. So, after choosing our dataset of 3D models for the experiment, we have to render the image views from each 3D model.

In the case of the ShapeNet benchmark, to extract the image views, we use the Stanford-Shapenet-renderer script. This script uses the API from the Blender software to render images from different angles of a given 3D model in OBJ format. This code is public on Github[1] and fully parameterizable for various tasks. In our case, we use the code to render 48 images by sampling in the vertex of a regular icosahedron containing the 3D models.

[1]https://github.com/panmari/stanford-shapenet-renderer

However, our models are not powerful enough to handle sets of 48 image views as input. To solve this problem, we use two different approaches to reduce images per 3D model. The first approach is to define a fixed number of viewpoints and use the image views extracted from those viewpoints for all the 3D models. The second approach is to use our proposal to select the possible best image views from the whole set of 48 image views for each 3D model.

In the case of the ModelNet benchmark, we use a prerender dataset of image views used in the RotationNet [38] experiments. In this case 20 image views are extracted per 3D model with a resolution of 224×224. This dataset is public on Github[2].

8.7.2 Experimental setup

For all the experiments, we use adam for the optimization, ReLU as the activation function, and a batch of size 100 for the training. In the case of the Autoencoder, the Combination Network, and the full proposal, we train for 50,000 iterations since the Autoencoder component takes more time to converge. On the other hand, for the Classification Model (our second base model), we only use 20,000 iterations since the classification accuracy quickly achieves values above 98 percent.

We aim to use the embeddings computed with all the architectures for 3D model retrieval. To do that, we use the cosine distance to measure the similarity between the embeddings and build a ranked list for each 3D model. After that, we apply the two mentioned metrics to measure retrieval performance over the ranked lists.

8.7.3 Ablation study

Given the multi-stage nature of our proposal, it is necessary to perform multiple experiments to verify how much each of these stages contributes to the overall performance of the full proposal. We perform an ablation study of our proposal for computing 3D model embedding. Below, we detail each step of this study.

Comparison of architectures

Our first stage of experimentation consists of computing 3D model embedding using each of the three base models of our architecture. This means to compute embeddings using the Autoencoder, the classification network and the combination of the Autoencoder and the classification network. The goal of these experiments is to prove that each stage improves over the last one.

[2]https://github.com/CabinfeverB/RotationNet-TensorFlow

We use four image views per 3D model for these experiments using fixed viewpoints. We also use one fully connected layer after the encoder of the network with an embedding size of 1024. Table 8.1 shows the results of all three models using the normalized version of the dataset.

The experiments show that the network with the worse results was the Autoencoder since, as we already mentioned, this model does not learn relations between the 3D models. Instead, to compute the embeddings for a given 3D model, it uses the 3D model itself. On the other hand, our second proposal highly outperforms the first one. Since the second proposal uses a classification network, this model intrinsically learns relations between the 3D models, specifically, if two 3D models belong to the same class or not. However, we obtain the best result using the third proposal, which combines the first two models.

Comparison of the number of fully connected layers

Our second stage of experimentation is to find out how adding multiple layers after the encoder of the network affects the performance. We use the combination of the Autoencoder and the classification network for these experiments since it was the best architecture from stage one of experimentation. We select the number of fully connected layers in the set $\{1, 3, 5\}$ and we use again embedding size of 1024. Table 8.2 shows these results.

These experiments show that varying the amount of fully connected layers does not constitute a significant change in the performance. This is a desirable feature in our case since we need to add multiple fully connected layers in our complete proposal.

Comparison of embedding sizes

Our third stage of experimentation is to obtain the optimum size of our embedding. To fulfill this task, we use five fully connected layers from the core of our architecture, and we reduce the dimensionality from one layer to another. We perform our experiments by choosing the embedding size in the set $\{1024, 512, 256, 128\}$.

We perform these experiments using the network that combines the Autoencoder with the classification network. We also use fixed viewpoints for these experiments with four image views per 3D model. Table 8.3 shows the results of our second stage of experimentation.

For the experiments, we can see that as the embedding size reduces, the network's performance increases until the embedding size is 256. From there, the performance heavily deteriorates with an embedding size of 128.

Comparison for the full proposal

Our fourth stage of experimentation compares the best model so far against our complete proposal. In other words, we compare the network that combines the Autoencoder with the classification network using five fully connected layers and embedding size 256 against our full proposal with the same parameters. We also use fixed viewpoints for these experiments with four image views per 3D model. Table 8.4 shows the results of these experiments.

The result shows that our complete proposal outperforms the best network so far. We can conclude that adding skip connections can improve the network's performance.

Comparison of the number of image views per 3D model

Our fifth experimentation stage consists of a study of how the number of image views representing a 3D model affects the proposal's performance. We use our full proposal to fulfill this requirement, selecting the number of image views per 3D model in the set $\{4, 8, 16\}$. When increasing this number, we expect an improvement in performance and an increase in the use of computation resources. We want to study this trade-off. Table 8.5 shows these results.

From these experiments, we can see that as the number of images increases, the performance of the model only shows a slight increase. We believe that these small performance gains do not justify increasing the number of image views.

Comparison of the image views selection techniques

Our last stage of experimentation is to find out how our proposal for selecting the potential best image views affects the model's performance. First, we experiment using image views extracted from fixed viewpoints. Then we use our proposal for the image view selection, and we use those selected images for the experiments. Finally, we compare both results. Table 8.6 and Table 8.7 show the result for these experiments using the normalized version of the dataset and the perturbed version of the dataset, respectively.

We can arrive at several conclusions from these experiments. First, we can see that using fixed viewpoints to extract the image views in the normalized version of the dataset achieves better results. We hypothesize that by selecting image views, we lose the information of the viewpoint since, in the normalized version of the dataset, all the 3D models are consistently aligned. However, in the perturbed dataset, using fixed viewpoints can lead to a lot of variation in the image view extraction process because of the random rotation of the 3D models. This variation leads to a bad performance of our model when used in the perturbed dataset. In this case, we increase the model's performance using the image view selection technique.

8.7.4 Inference time Study

As mentioned, our proposal learns features from the whole set of views simultaneously instead of using the standard approach in the field to combine the features from each image view by incorporating view pooling layers. Our goal for avoiding this type of architecture is to propose a network for computing 3D shape embedding using image views, fast and low resource demanding while being competitive with state-of-the-art techniques.

To support this claim, we compare the inference time of our network against several state-of-the-art techniques. We generate a 2000 image view set to conduct these experiments, and each set is composed of 50 views with a resolution of 128×128. Finally, we make inferences with each selected network using the described dataset and batch of size one. The state of the art methods are PANORAMA-ENN [79], MVCNN [91], 3D2SeqViews [35], and RotationNet [38]. Table 8.8 shows the average inference time for each network. The experiment results show that our proposal inference time is much lower than the other methods, being between 8 times and 50 times faster.

8.7.5 Proposal results against the state of the art

We compare the best results obtained by our proposed model (SE3D) against several other works (PANORAMA-ENN [79], RotationNet [38], GIFT [104], ReVGG [104], MVFusionNet, CM-CNN [104], VoxelNet [104], DLAN [104], ZFDR [104], PointNet [69], DensePoint [60], FMVAC [62], MVLA [58]) using both versions of the ShapeNet dataset. We chose these works because, to our knowledge, they report the best results over the ShapeNet dataset. Also, we include works for different types of 3D shape representation (voxel, image views, and point cloud).

The methods that use image view representation, except for ZFDR, which uses handcrafted features, also use some form of image view aggregation technique. This technique means these methods must compute features from each image view separately, in contrast to our proposal that extracts features from a tensor containing all the image views, making the training and the inference of the network faster.

For the works PointNet and DensePoint, we could not find results for retrieval over the ShapeNet dataset. In these cases, we run the experiment ourselves. To accomplish this, we first train both models over their original datasets to reproduce the results reported by the author. Then, we use a point cloud version of the normalized version of ShapeNet, and we train both models for classification using the default configuration proposed by the authors.

After the training, we use the layer's output before the last fully connected layer to represent the 3D shapes. Finally, we use the cosine distance between these representations to measure the similarity and obtain the retrieval results. The point cloud version of the ShapeNet dataset is public on Github [3]. The implementation of PointNet and DensePoint are also public on Github [4], respectively.

Table 8.9 and Table 8.10 show the results using the normalized and the perturbed version of the dataset. Note that the perturbed version of the dataset does not include point clouds. Thus, we present results for PointNet and DenseNet only for the normalized version. We also compute the micro and macro averages of both metrics used in our experiments.

In addition, Table 8.11 shows the available results over the ModelNet benchmark for the methods presented in Table 8.9. These results include the classification accuracy for all techniques and some techniques' retrieval MAP. The DCG metric is not reported for the ModelNet Benchmark.

We compare our proposal against several state-of-the-art works, including some of the more recent ones to our knowledge. The results show that our proposed model achieves very competitive effectiveness, especially in the normalized version of ShapeNet, where we obtain the highest DCG among all methods.

However, the effectiveness of our proposed method is affected when we use the perturbed version of the ShapeNet dataset. In this dataset, we use a different technique to select the image views representing the 3D models, improving the effectiveness of the retrieval. However we still fall behind the best state-of-the-art results, which means that our proposal still needs improvement to capture the information in rotated objects. An alternative solution is the pose normalization of the 3D models in the dataset.

In the case of the ModelNet dataset, the DCG results are not reported, so we can only make a comparison using the MAP for retrieval and the accuracy for classification. Regarding this last metric, our results fell behind the best results. However, our proposal is optimized for classification, so we were not expecting the best results for this particular task.

Finally, we get these results while avoiding using image view aggregation techniques or view pooling layers. Instead, we learn features from the complete set of image views as a single object, making our proposal faster and easier for training and inference computing.

[3]https://github.com/AnTao97/PointCloudDatasets
[4]https://github.com/charlesq34/pointnet, https://github.com/Yochengliu/DensePoint

Table 8.1: Evaluation of the performance of the embedding for 3D models Retrieval using the normalized version of the ShapeNet.

Model	Views	FC layers	Embedding size	DCG	MAP
Autoencoder	4	1	1024	0.655	0.290
Classification	4	1	1024	0.857	0.567
Autoe.+Class.	4	1	1024	0.896	0.680

Table 8.2: Evaluation of the performance changing the amount of fully connected layers for 3D models Retrieval using the normalized version of the ShapeNet.

Model	Views	FC layers	Embedding size	DCG	MAP
Autoe.+Class.	4	1	1024	0.896	0.680
Autoe.+Class.	4	3	1024	0.898	0.673
Autoe.+Class.	4	5	1024	0.897	0.679

8.8 Summary

We propose an artificial neural network architecture for computing 3D model embeddings using sets of image views extracted from them. The core of this proposal consists of two different networks: a convolutional Autoencoder and a convolutional neural network for classification. We impose restrictions on these networks and add new parameters to improve the quality of the embedding computed by the network. We use the same technique for modeling the input for all the networks. This technique transforms a set of image views representing a 3D model into a multichannel image where each channel is an image view.

We conducted several experiments using our proposals, and we can conclude that our work has three main contributions. First, we propose a convolutional architecture that can handle the 3D models represented as sets of image views faster and easier for training and inference computing. Second, we present an analysis of the performance of different convolutional architectures for computing 3D model embedding. Finally, we present a study of how the different parameters affect the quality of the computed embedding.

The most important conclusion is that our proposed models can successfully compute embedding representations for 3D models. We obtain good results for the two evaluation metrics, especially with our last model, which shows results for the MAP above 73 percent and DCG above 91 percent. These results are very competitive within the ShapeNet Dataset results, especially in the normalized version of the dataset. We will continue this research by using our computed embedding in other 3D model processing tasks, like cross-modal 3D shape retrieval.

Table 8.3: Evaluation of the performance changing the embedding size for 3D models Retrieval using the normalized version of the ShapeNet.

Model	Views	FC layers	Embedding size	DCG	MAP
Autoe.+Class.	4	5	1024	0.897	0.679
Autoe.+Class.	4	5	512	0.902	0.694
Autoe.+Class.	4	5	256	0.907	0.710
Autoe.+Class.	4	5	128	0.828	0.600

Table 8.4: Evaluation of the performance adding skip connection for 3D models Retrieval using the normalized version of the ShapeNet.

Model	Views	FC layers	Embedding size	DCG	MAP
Autoe.+Class.	4	5	256	0.907	0.710
Autoe.+Class+Skip	4	5	256	0.911	0.721

Table 8.5: Evaluation of the performance for different number of image views per 3D model using normalized version of the ShapeNet.

Model	Views	FC layers	Embedding size	DCG	MAP
Autoe.+Class+Skip	4	5	256	0.911	0.721
Autoe.+Class+Skip	8	5	256	0.916	0.728
Autoe.+Class+Skip	16	5	256	0.919	0.732

Table 8.6: Evaluation of the performance for the two different techniques to select image views using normalized version of the ShapeNet with 16 views, five fully connected layers and embedding size 256.

Model	Views selec.	DCG	MAP
Autoe.+Class+Skip	Fixed	0.919	0.732
Autoe.+Class+Skip	Network	0.892	0.693

Table 8.7: Evaluation of the performance for the two different techniques to select image views using perturbed version of the ShapeNet with 16 views, five fully connected layers and embedding size 256.

Model	Views selec.	DCG	MAP
Autoe.+Class+Skip	Fixed	0.646	0.354
Autoe.+Class+Skip	Network	0.745	0.532

Table 8.8: Comparison of the Average Inference Times

Network	Average Inference Time
Autoe.+Class+Skip	0.187
PANORAMA-ENN	0.980
MVCNN	1.043
SeqViews2Seq	2.167
RotationNet	11.501

Table 8.9: Comparison of our computed embedding against other 3D model retrieval methods using the normalized version of the ShapeNet.

Metric Calculation	Micro Average		Macro Average	
Model	DCG	MAP	DCG	MAP
SE3D	0.919	0.732	0.774	0.513
RotationNet	0.865	0.772	0.656	0.583
GIFT	0.827	0.722	0.657	0.575
ReVGG	0.828	0.749	0.559	0.496
DLAN	0.762	0.663	0.563	0.477
PANORAMA-ENN	0.845	0.739	0.656	0.588
MVFusionNet	0.732	0.622	0.502	0.418
CM-CNN	0.654	0.540	0.404	0.339
ZFDR	0.330	0.199	0.377	0.255
VoxelNet	0.277	0.192	0.337	0.232
PointNet	0.769	0.580	0.416	0.639
DensePoint	0.807	0.679	0.700	0.515
VFMVAC	0.862	0.778	0.677	0.607
MVLA	0.867	0.809	0.667	0.630

Table 8.10: Comparison of our computed embedding against other 3D model retrieval methods using the perturbed version of the ShapeNet.

Metric Calculation	Micro Average		Macro Average	
Model	DCG	MAP	DCG	MAP
SE3D	0.745	0.532	0.546	0.371
RotationNet	0.702	0.606	0.407	0.327
GIFT	0.701	0.567	0.513	0.406
ReVGG	0.783	0.696	0.479	0.418
DLAN	0.754	0.656	0.560	0.476
PANORAMA-ENN	0.759	0.703	0.554	0.462
CM-CNN	0.642	0.524	0.395	0.329
ZFDR	0.303	0.172	0.336	0.215
VoxelNet	0.043	0.009	0.109	0.047

Table 8.11: Comparison of our computed embedding against other 3D model retrieval methods using ModelNet.

Model	Classification(Accuracy)	Retrieval(MAP)
SE3D	0.883	0.847
RotationNet	0.973	-
GIFT	0.831	0.819
PANORAMA-ENN	0.955	0.863
FusionNet	0.908	-
VoxelNet	0.830	-
PointNet	0.892	-

Chapter 9

SHREC 2021: A practical application of our proposal for the 3D shape retrieval

SHREC (3D Shape Retrieval Challenge) is a yearly initiative with the general objective of evaluating the effectiveness of 3D shape retrieval algorithms. We participate in SHREC2021, the sixteenth edition of the contest, specifically in the Retrieval of Cultural Heritage object track [87].

9.1 SHREC 2021: Retrieval of Cultural Heritage Objects

The cultural heritage domain has benefited substantially from 3D shape processing. This track presents an initiative to promote the research of 3D shape analysis methods in cultural heritage domains. To this end, two retrieval challenges are presented considering two aspects: the shape and the culture. Regarding the shape, archaeologists in the Josefina Ramos de Cox (JRC) museum classified the scanned objects by shape using specific taxonomies for Peruvian pre-Colombian artifacts. Regarding culture, the JRC museum keeps records of the pre-Colombian cultures to which the artifact belongs. This metadata is collected for the scanned models, which serves as input for our retrieval tasks.

The proposed challenges have different degrees of complexity. Retrieval-by-shape is probably the more affordable challenge given that there exist suitable methods in the 3D shape retrieval literature to deal with geometric characterization. Nevertheless, there are cases where the distinction between objects in different classes is barely perceived. On the other hand, retrieval-by-culture is a more difficult challenge. Models from the same culture can have varied shapes, and probably the most distinguishable characteristic is the combination of geometry and painting style.

9.1.1 Dataset

The dataset for this track consists of 3D scanned models from cultural heritage objects captured in the Josefina Ramos de Cox museum in Lima, Perú. The technology used to acquire the 3D models was a structured-light desktop scanner, which produces high-resolution 3D textured models. A post-processing step is applied to normalize the position by translating the objects' center to the origin of 3D space. The orientation is also changed manually, such that shapes are oriented up on the Y-axis. However, the original scale of models is kept because the scale can be a distinctive feature that differentiates objects. Finally, the triangular mesh of each shape is simplified to have nearly 40,000 triangle faces.

9.1.2 Challenge description

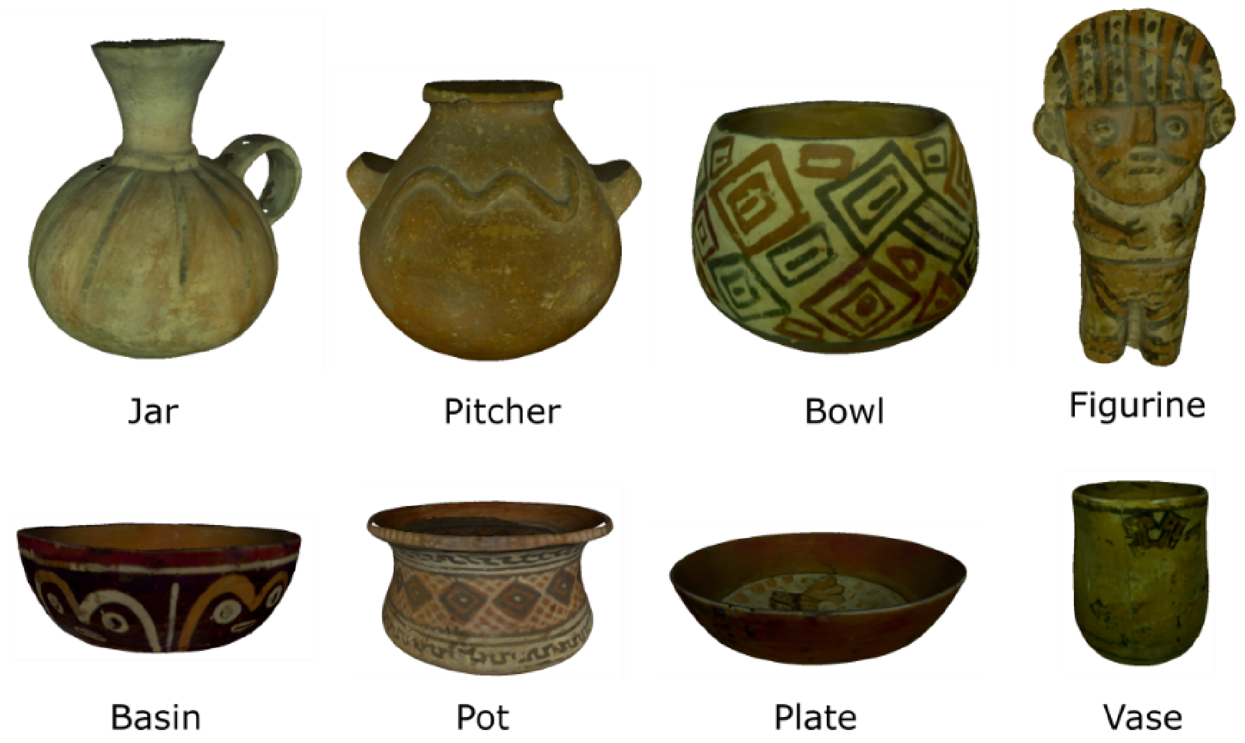

Figure 9.1: Example of each class for the shape challenge

The dataset for the shape challenge consists of 938 objects classified into eight categories: jar, pitcher, bowl, figurine, basin, pot, plate, and vase. The dataset is split into a collection set (70% of the dataset) and a query set (30% of the dataset). The collection set contains 661 objects, and the query set has 277 objects. The bowl class contains the highest number of models (with 221 models in total), and the vase class contains the lowest number of models is (with 34 objects in total). Figure 9.1 shows examples of models in each class.

The dataset for the cultural challenge consists of 637 objects classified into six categories:

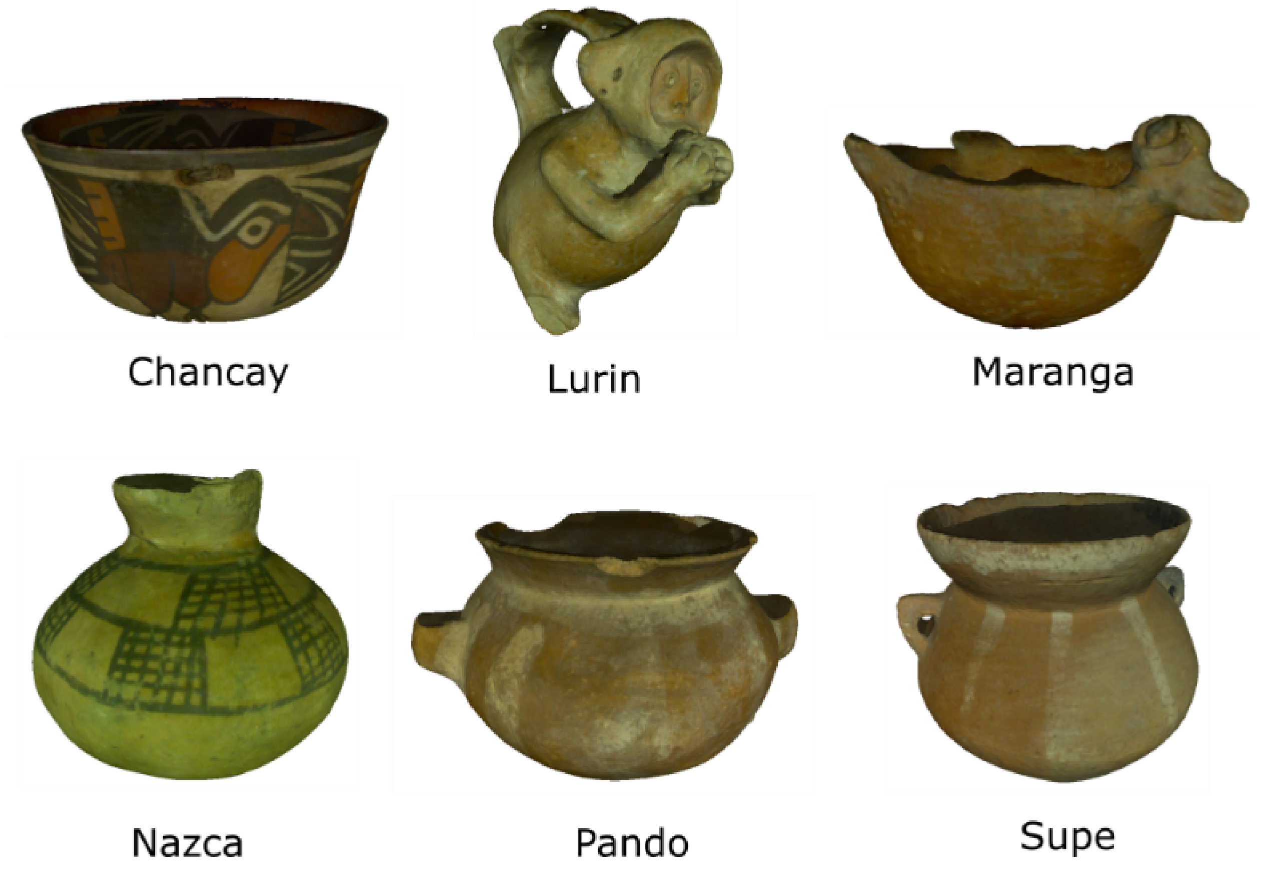

Figure 9.2: Example of each class for the cultural challenge

Chancay, Lurin, Maranga, Nazca, Pando, and Supe. It is split as before into a collection set (70% of the dataset) and a test set (30% of the dataset). The collection set contains 448 objects, and the test set has 189 objects. The class with the highest number of models is Lurin (with 455 shapes in total), and the class with the lowest number of models is Nazca (with seven shapes total). Figure 9.2 shows examples of objects in each class.

It is worth noting that, in both challenges, the classes are not balanced. This phenomenon is even more noticeable in the retrieval-by-culture challenge. Nevertheless, these challenges propose a real application for 3D retrieval algorithms; hence, one of the goals is to evaluate the robustness of methods against unbalanced data.

9.2 Proposal

We propose a convolutional architecture to compute 3D shape embedding based on our previous model described in Chapter 8. Our goal is to transform the 3D shapes into vectors with enough information to substitute the 3D shape itself in further processing. In this case, we use the computed embeddings for 3D shape retrieval. To better understand the proposal, we divide the description into the 3D shape representation and the network architecture.

For the 3D shape representation, we first extract a set of image views from each 3D shape. Then, we represent the 3D shapes as multichannel objects, where each channel is an image view. Finally, we conduct different experiments using different amounts and types depending on the task we are tackling (shape challenge, cultural challenge). In the shape challenge, the image views are single channel (grayscale), and the number of image views per 3D shape is selected in the set [4, 8]. We use three channels (RGB) in the cultural challenge, and the number of image views per 3D shape is selected in the set [2, 4].

We use this multichannel representation of the 3D shapes to train our convolutional network, which is composed of a combination of a convolutional Autoencoder and a convolutional neural network for classification. For the Autoencoder part of the network, we use the standard encoder-bottleneck-decoder architecture with reconstruction loss L_2 norm of the difference between the output and the input. As for the classification part, we use the same Autoencoder network until the bottleneck component. Then, we connect the bottleneck to a fully connected layer of size C, where C is the number of classes for classification. After we train our model, we use the bottleneck layer to represent the 3D shapes and cosine distance to compute the similarity between embeddings.

9.3 Experiment and results

Along with our method for 3D Shape Embeddings (SE3D), another nine proposals competed in the contest. Deep Node Embedding (DNE), Normalized Profile Curve (NPC), Deep Representative Learning and Feature Learning (DRF-L), Triplet Loss and Autoencoder Architectures (TL), Ring View Net (RVN), Multi-view CNN on Silhouettes (MVCNN), Mesh-based Neural Networks (MeshNN), Multi-view ResNet (MVResNet), and CNN on Principal Component Images (PCI). All the methods were evaluated using five retrieval metrics (NN, FT, ST, mAP, nDCG) and the precision-recall curve. Finally, there was a maximum of five runs per submission on each of the two challenges. We submit all the five-run in each challenge with the next configuration:

- Run 1: Embeddings created with 10,000 epochs in training and four views.
- Run 2: Embeddings created with 30,000 epochs in training and two views.
- Run 3: Embeddings created with 30,000 epochs in training and four views.
- Run 4: Embeddings created with 60,000 epochs in training and two views.
- Run 5: Embeddings created with 60,000 epochs in training and four views.

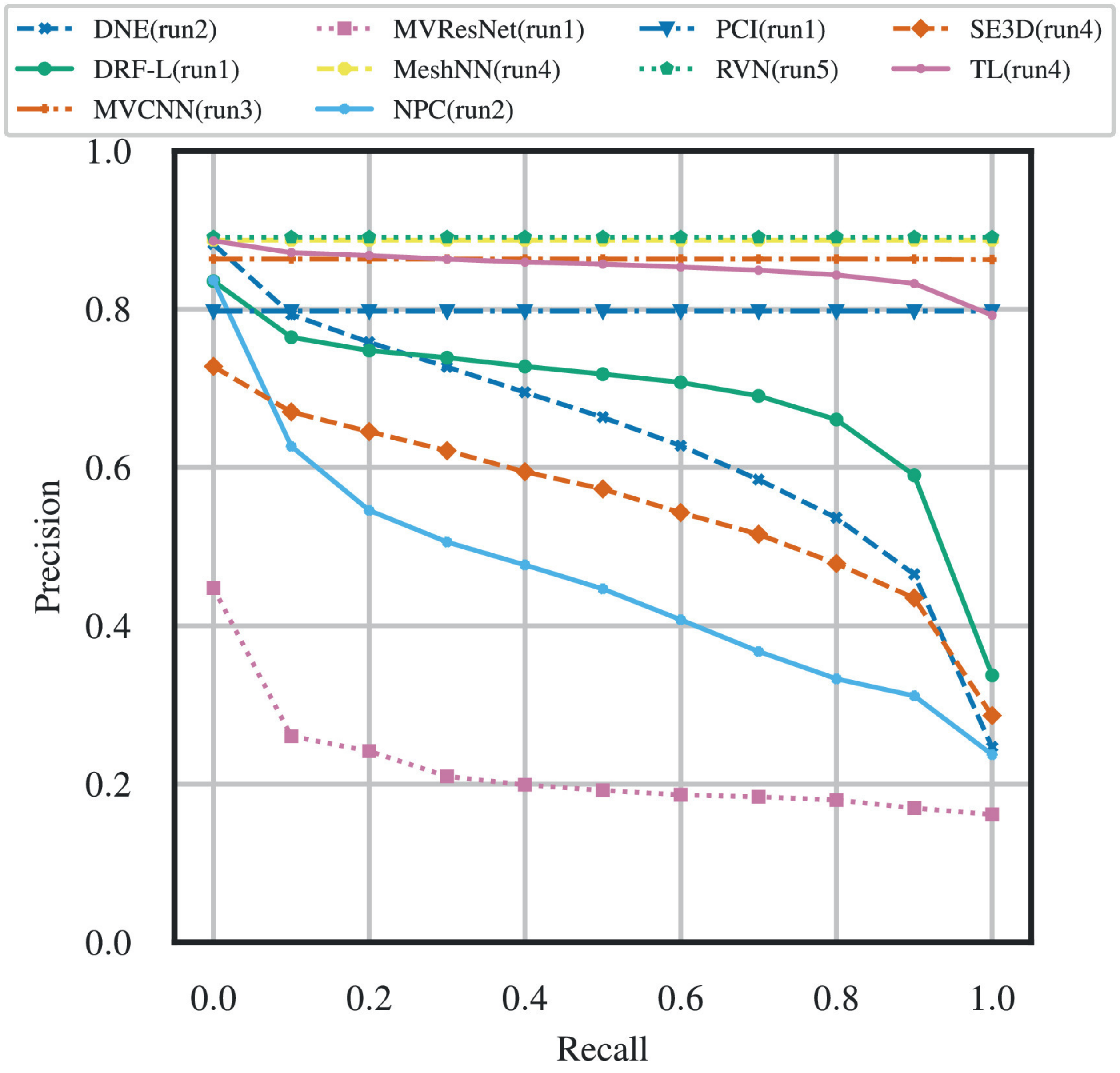

Figure 9.3: Precision-recall plot for the retrieval-by-shape challenge.

We show the results of the precision-recall curve for both challenges in Figure 9.3 and Figure 9.4, respectively. The retrieval metrics are shown in table 9.1 and table 9.2 for the two challenges. As the results show, we achieved good results, ranking around the middle of all proposals.

9.4 Summary

We participated in the SHREC 2021 contest, specifically the Retrieval of Cultural Heritage Objects track. We submit a version of our proposal for 3D shape embedding adapted for the two specific challenges of the contest. Our proposal was evaluated against the other nine methods, achieving results around the middle of the performances.

We created our method to be trained using the ChapeNet dataset, which consists of around 50000 3D shapes in contrast with the two challenges of the contest, which count with less than 1000 objects. This small amount of training data compared to the amount of data in the ShapeNet dataset could hurt the performance of our proposal. Another observation is that most methods that perform better than our proposal use machine learning techniques to render/select the best views of the 3D shapes. These encourage us to use similar techniques in our process for the image views extraction in future works.

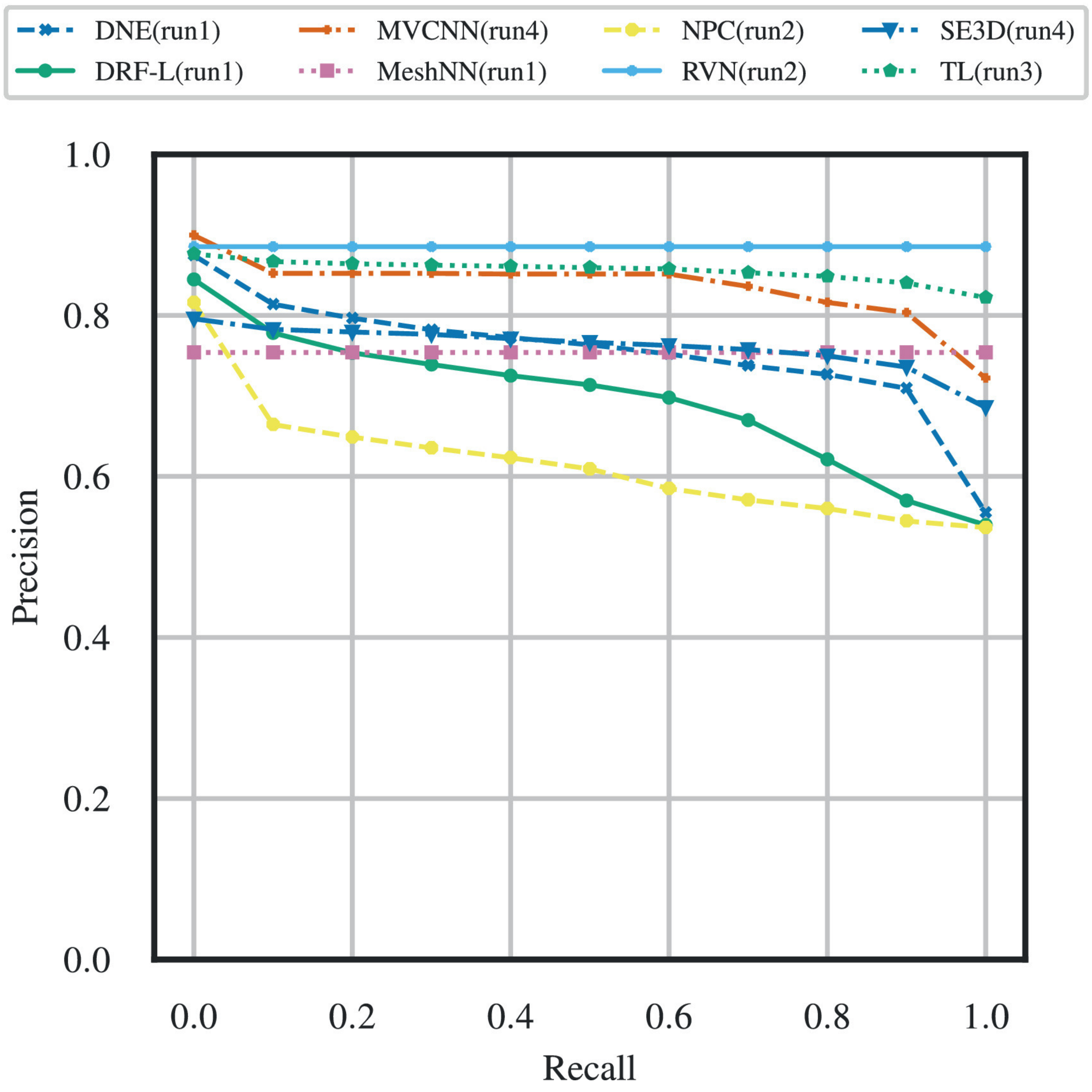

Figure 9.4: Precision-recall plot for the retrieval-by-culture challenge

Table 9.1: Evaluation measures for the retrieval-by-shape challenge

Methods	NN	FT	ST	mAP	nDCG
DNE (run1)	0.7509	0.5813	0.4385	0.6273	0.8704
DNE (run2)	0.8014	0.5933	0.4037	0.6303	0.8746
DRF-L (run1)	0.7256	0.6494	0.5162	0.6780	0.8755
DRF-L (run2)	0.7220	0.4744	0.3493	0.4841	0.8220
MVCNN (run1)	0.7509	0.6164	0.4927	0.6489	0.8794
MVCNN (run2)	0.7545	0.6006	0.4995	0.6350	0.8756
MVCNN (run3)	0.7545	0.8010	0.8860	0.8235	0.9122
MVResNet (run1)	0.1841	0.1430	0.1409	0.1761	0.6281
MeshNN (run1)	0.5054	0.3678	0.2941	0.3673	0.7587
MeshNN (run2)	0.5993	0.6388	0.7358	0.6915	0.8428
MeshNN (run3)	0.7220	0.7871	0.8788	0.8054	0.9013
MeshNN (run4)	0.7906	0.8356	0.9174	0.8518	0.9256
NPC (run1)	0.6751	0.3766	0.3106	0.4019	0.7874
NPC (run2)	0.6787	0.4151	0.3237	0.4274	0.7996
PCI (run1)	0.6751	0.7100	0.7827	0.7496	0.8695
RVN (run1)	0.7942	0.6675	0.5863	0.7084	0.8995
RVN (run2)	0.7726	0.6715	0.5763	0.7074	0.8989
RVN (run3)	0.7762	0.6694	0.6168	0.7086	0.8992
RVN (run4)	0.7798	0.6680	0.6095	0.7069	0.8994
RVN (run5)	0.8087	0.8471	0.9065	0.8604	0.9286
SE3D (run1)	0.7112	0.4380	0.3340	0.4514	0.8068
SE3D (run2)	0.6029	0.4903	0.4126	0.5081	0.8119
SE3D (run3)	0.6354	0.4757	0.4035	0.5065	0.8149
SE3D (run4)	0.5993	0.5090	0.4147	0.5431	0.8205
SE3D (run5)	0.5848	0.5018	0.4358	0.5395	0.8169
TL (run1)	0.8159	0.8230	0.8551	0.8409	0.9262
TL (run2)	0.6751	0.4727	0.3578	0.4897	0.8247
TL (run3)	0.7870	0.7252	0.6895	0.7693	0.9106
TL (run4)	0.8412	0.8297	0.8357	0.8428	0.9293
TL (run5)	0.7437	0.5385	0.4667	0.5717	0.8482

Table 9.2: Evaluation measures for the retrieval-by-culture challenge

Methods	NN	FT	ST	mAP	nDCG
DNE (run1)	0.7831	0.7466	0.7767	0.7457	0.8799
DNE (run2)	0.7566	0.7074	0.7673	0.7190	0.8715
DRF-L (run1)	0.7302	0.6302	0.7831	0.6729	0.8648
DRF-L (run2)	0.7249	0.5516	0.7670	0.5713	0.8383
MVCNN (run1)	0.1852	0.8088	0.7989	0.7410	0.8532
MVCNN (run2)	0.2063	0.8292	0.8130	0.7626	0.8675
MVCNN (run3)	0.1534	0.7932	0.7925	0.7882	0.8748
MVCNN (run4)	0.1799	0.8130	0.8077	0.8081	0.8889
MVCNN (run5)	0.7566	0.6354	0.7783	0.6772	0.8686
MeshNN (run1)	0.7249	0.7418	0.7354	0.7415	0.8353
NPC (run1)	0.7037	0.5489	0.7539	0.5736	0.8267
NPC (run2)	0.6772	0.5523	0.7565	0.5776	0.8273
RVN (run1)	0.8254	0.8497	0.8536	0.8484	0.9043
RVN (run2)	0.8360	0.8738	0.8783	0.8698	0.9186
RVN (run3)	0.8307	0.8080	0.8257	0.8207	0.9138
RVN (run4)	0.8201	0.8252	0.8420	0.8320	0.9176
SE3D (run1)	0.6772	0.6029	0.7605	0.6091	0.8374
SE3D (run2)	0.7619	0.6916	0.7656	0.7133	0.8663
SE3D (run3)	0.7778	0.6957	0.7612	0.7181	0.8663
SE3D (run4)	0.7090	0.7444	0.7727	0.7487	0.8611
SE3D (run5)	0.7037	0.7180	0.7582	0.7321	0.8542
TL (run1)	0.8254	0.8356	0.8229	0.8291	0.8989
TL (run2)	0.7619	0.6481	0.7661	0.6510	0.8542
TL (run3)	0.8360	0.8477	0.8503	0.8450	0.9112
TL (run4)	0.7989	0.8092	0.8034	0.8031	0.8827
TL (run5)	0.7937	0.5960	0.7639	0.6210	0.8543

Chapter 10

Deep learning architectures for Image-based 3D Shape Retrieval (CrossSE3D)

3D model retrieval is the problem of searching for an object in a 3D model collection. Instead of using a textual query that describes what the user needs, we are interested in the case where the query is an object similar to the 3D models in the collection. For example, the search in traditional 3D model retrieval starts with a 3D shape as a query object. While this is a common approach in multimedia information retrieval, known as query-by-example, there could be a contradiction in the assumption that the user has available a 3D shape for making the query. Indeed, the query-by-example approach supposes that there is a mechanism to obtain or search the 3D query object before using the retrieval algorithm. From the user perspective, it would be easier for many practical applications of 3D object retrieval if the search algorithm received an image as input. Thus, a relevant problem is how to find a 3D model given an image that depicts it, which we call "image-based 3D shape retrieval".

Applying deep learning techniques for solving the 3D model retrieval problem is a current interesting trend, as this type of technique has shown promising results in several computer vision tasks during the last decade [37, 34]. So, the question is how to train a deep neural network capable of finding relevant 3D objects given an image as a query. In a deep learning context, one challenge is to have good representations for both the query image and relevant 3D models such that these representations share a common place in the feature space (also known as the latent space). To solve this, we need to find an appropriate artificial neural network architecture that allows us to train it so that it learns to match images with 3D objects effectively.

Another challenge related to this problem is the data. Images and 3D objects are complex data types. In addition, these data may be subject to several types of transformations (scale, translation, orientation, mirroring, noise, etc.), so there is a crucial need to develop retrieval algorithms that are robust to these transformations. Moreover, in our case the retrieval algorithms must tackle the invariance problem in both domains (image and 3D shape) simultaneously, making the search problem even harder than when the user uses the same type of object as in the collection for processing a query.

We study and propose deep learning models for image-based 3D shape retrieval. The first model is based on computing embeddings for both the 3D shapes and the images, which can be used to implement the retrieval phase. In this model, we use an image-view approach for representing the 3D shapes. However, the experimental evaluation of this model shows that it has poor performance in solving the image-based 3D shape retrieval problem. This result indicates us that metric learning struggles to find a suitable joint space for image embeddings and 3D model embeddings. Therefore, we need to explore novel architectures for improving the effectiveness of the retrieval.

The insights obtained during the evaluation of our first model led us to propose a second model, which is an end-to-end architecture that directly learns how to match images and 3D shapes. We show that the development of this second model requires us to present solutions to a number of technical difficulties, as each data object is represented as a multichannel data source but with (most probably) a different number of channels. We explain how to tackle all these difficulties. Finally, we conduct an experimental evaluation with this second model and show that the end-to-end architecture is far more effective than the embedding-based architecture for the image-based 3D shape retrieval task.

10.1 A deep learning architecture for Image-based 3D Shape Retrieval

We propose a deep learning model for solving the task of image-based 3D shape retrieval (CrossSE3D). The idea is to train our network to learn whether an image and a 3D shape are similar. To accomplish this, our network comprises two major components: compute embeddings from 3D shapes and images and compute the similarity between those embeddings.

10.1.1 Architecture for computing embeddings

To compute embeddings from the 3D shapes and the image, we use a variation of the network architecture in Chapter 8 (A New Convolutional Architecture for 3D Model Embedding and Retrieval (SE3D)). This model uses a convolutional Autoencoder architecture with a standard convolutional neural network for classification to compute embeddings of 3D shapes.

To represent the 3D shapes, we use an image views approach. We first extract a set of image views from each 3D shape, and then we transform this set into a multichannel object similar to an RGB image where each channel is an image view.

This representation is the input to our network architecture for embedding computation. As mentioned before, this network is composed of a convolutional Autoencoder and a convolutional neural network for classification. The Autoencoder uses the standard approach of encoder-bottleneck-decoder. The encoder is composed of four blocks combining convolutional layers and max-pooling layers to reduce the blocks' dimensionality. We connect the encoder to the network's bottleneck, a fully connected layer of size 1024. Finally, we connect the bottleneck to the network's decoder composed of four blocks combining transposed convolutional layers, also known as deconvolutional layers and un-pooling layers, to increase the dimensionality of the blocks. Figure 10.1 shows the un-pooling technique used in the model. We want the decoder output to reconstruct the input. To this end, we use the L_2 norm as the loss function between the input and the decoder output.

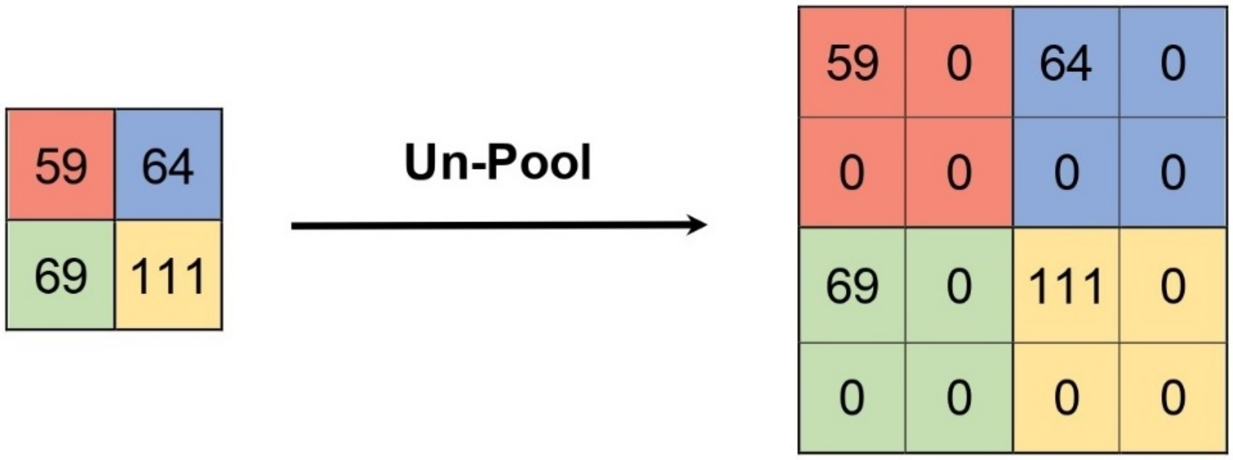

Figure 10.1: Un-pooling layer used for the decoder of the model

We use the same convolutional Autoencoder architectures for the model's convolutional neural network component until the bottleneck layer (a fully connected layer of size 1024). From there, we connect this layer to another fully connected layer of size C, where C is the number of classes for classification. Then, we apply the standard softmax activation function over this last layer and use cross-entropy as the classification loss. Finally, the whole model's loss is the sum of the classification and reconstruction losses. After the training phase, we use the model's bottleneck as embeddings for the 3D shapes.

Finally, we add parameters and restrictions to the network to improve the quality of the embeddings. First, we increase the number of fully connected layers in two on the encoder component of the network. The first of these layers reduces the embedding size to 512, and the second reduces the size again to 256. We also add the respective two fully connected layers to the decoder of the model. Second, we incorporate skip connections between the encoder and decoder of the model. By doing this, we add the output of the fully connected layers in the encoder to their respective fully connected layers in the decoder, reinforcing the amount of information passed to the decoder of the network, which allows a better

This architecture aims to compute embeddings with the high compression power of an Autoencoder while preserving information about the class of the original 3D shape. Figure 10.2 shows the architecture of the model.

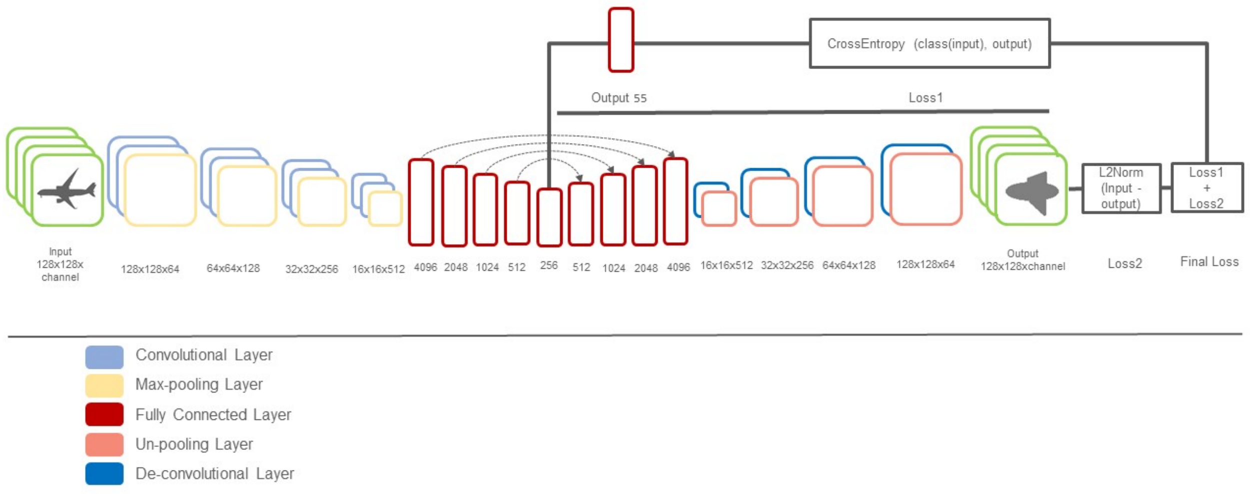

Figure 10.2: Network architecture for computing embeddings

10.1.2 Compute similarity between embbeddings

Once we have the embeddings, we need a network to learn whether the two embeddings representing a 3D model and an image respectively are similar or not. For that particular task, we use a Siamese network architecture.

This network consists of five fully connected layers with dimensions 1024, 2048, 4096, 2048, and 1024. We train this architecture using three elements tuples, an embedding of a 3D shape, an embedding of an image, and a binary label with the information of whether those embeddings belong to the same class or not. Finally, we use the cosine distance combined with binary cross-entropy as the loss function using the binary label.

After we train the model, we use the last fully connected layer's output as the embedding of the 3D shapes and the images as well. Finally, we use a similarity metric between the embeddings for the image-based 3D shape retrieval task. In our case, we experiment using the cosine distance.

10.2 Experiments and Results

In this section, we discuss all the experimental setups, describe the benchmark and the preprocessing of the data, and give the technical aspects of our proposed architecture.

10.2.1 Data configurations

We use the ShapeNet [14] benchmark dataset for the experiments. This dataset comprises 51,300 3D models grouped into 55 categories, and they are provided in OBJ format. The dataset counts with two versions: the normalized and the perturbed datasets. The dataset provides a split 70%/10%/20% for training, validation, and testing, respectively. We experiment using the normalized versions of the dataset.

10.2.2 Image views represenation

As mentioned before, our proposed models need 3D shapes represented as sets of image views and images as input for training and testing. To solve this, we use the Stanford-Shapenet-renderer script for the image views extraction. The script uses the Blender software's API to render images from different angles sampling uniformly around a circumference surrounding a given 3D model in OBJ format. This code is public on Github [1] . To represent the 3D shapes, we use the code to render image views from the 3D shapes starting with the camera on coordinates origin sampling one image every 90 degrees, which provides 4 image views per 3D shape. On the other hand, we use the same Stanford-Shapenet-renderer script to render one image view from the 3D shapes from a random viewpoint to obtain the images from training and testing.

10.2.3 Proposal setups and results

To train our proposal, we first use our model for computing embeddings from 3D shapes and images. We train this model using a batch with size 50 with Adam optimizer and 40000 iterations. We first train with the 3D shapes to obtain the 3D shapes embedding, and then we train with the image to obtain the image embedding. After that, we use our siamese network to learn whether 3D shape embeddings and image embeddings are similar. We use the three-element input, a 3D shape embedding, an image embedding, and the binary label to train this network. However, we need our training dataset to be balanced according to the label. In other words, if we build this three-element input randomly since there are 55

[1] Stanford-Shapenet-renderer

categories in the dataset, the odds of the label being one will be $\frac{1}{55}$. To handle this, we manually built the training dataset by iterating for each 3D shape and adding 50 images that belong to the same class and 100 other images randomly. With this, we assure that the odds of the label being one is at least $\frac{1}{3}$. Since the training data is considerably bigger now, we train the siamese network for 100000 iterations with a batch size of 1000 and Adam optimizer. Table 10.1 shows the results of our proposed model.

This model shows poor performance for solving the image-based 3D shape retrieval problem with a DCG of 59 and a MAP below one. This result indicates that metric learning fails to find a suitable joint space for images and 3D shapes. We hypothesize that this problem occurs because we divide the learning into two processing stages, each stage with a different neural network trained for very different tasks.

10.3 An end-to-end neural network for image-based 3D shape retrieval

To address the problem of the first model, we design an end-to-end network for image-based 3D shape retrieval. To fulfill this requirement, we need our network to compute both embedding from the 3D shape and the image and penalize the similarity between them at the same time. This proves to be a very challenging task since our network's input layer needs to process multichannel objects with different numbers of channels in the same iteration. For instance, the 3D shape representation could have as many channels as image views used for its representation. In contrast, we use greyscale images, which are single-channel objects. To solve this, we use an approach similar to a recurrent neural network for our model's input. The idea here is that we iterate over the number of channels of a given object, and we pass each channel to the input layer to produce an output per channel. We compute the average of these outputs, and this average is the input for the rest of the network.

We train this network using tuples of five elements as inputs. A 3D shape with the respective class, an image with the respective class, and a binary label with value one if the 3D shape and the image belong to the same class and value zero otherwise. With this input, we first compute the embedding of the 3D shape using its multichannel representation and the information of its class. Then, we compute the embedding of the image in the same way. Finally, we use a loss function between the two embeddings to penalize de model according to the distance between the two embeddings and the label representing whether or not they belong to the same class. We experiment using the same loss function as before, the cosine distance between the embeddings, with binary cross-entropy using the binary label.

We can compute both embedding losses and the similarity loss between the embeddings in each iteration with our proposed architecture and the five elements input. This is necessary to compute the whole loss of our end-to-end model by adding all three losses.

After we train the model, we use the encoded layer output as the embedding of the 3D shapes and the images. The rest of the process is the same as the first model for image-based 3D shape retrieval.

10.3.1 Experiments and Results for the end-to-end proposal

We use the five-element input (3D shape, category of the shape, image, category of the image, binary label) to train this last proposal. We need the network to learn the embeddings and the similarities between them in the same pass. We built the training dataset in the same way as before to balance the categories. We train the model for 100000 iterations and a batch size of 15 because that is the maximum our hardware allows us. Finally, we use Adam optimizer for the training. Table 10.2 shows the results of this last model, along with the results of the previous model and our first proposal for image-based 3D shape retrieval described in Chapter 6 (Deep features for image-based 3D model retrieval).

These results show that the end-to-end network obtains the best result of all proposals, heavily outperforming the first model. Especially in the MAP metric, in which the result is one magnitude above. We confirm that the end-to-end model successfully learns a joined space for image and 3D shapes. We complement these findings by extending the experiments to the modelNet dataset. Specifically, to ModelNet40, where the models are categorized into 40 classes manually aligned but not scaled, this dataset also provides its training/testing split. Table 10.3 shows the results of these last experiments for our proposed Image-based 3D shape retrieval methods.

10.4 Summary

We explore the problem of image-based 3D model retrieval using a learning approach. We compare the use of pre-trained embeddings with embeddings obtained through an end-to-end learning approach. From our experiments, we observe that learning embeddings jointly for images and 3D objects is beneficial to increase the retrieval performance compared to embeddings learned separately. The challenge is to provide a clever way to let the network receive inputs from different modalities. We solve this problem in an effective method that processes image representation with an arbitrary number of channels.

However, although our end-to-end approach shows promising retrieval results, the problem is far from being solved. We believe that the study and analysis of complementary representations for 3D objects (point clouds, meshes, or implicit surfaces) could enrich the input data for a learning approach. There is also an imperative need for a suitable dataset to evaluate the image-based 3D model retrieval problem.

Table 10.1: Evaluation results for the embedding-based proposal

Model	Views	DCG	MAP
embedding-based	4	0.59	0.092

Table 10.2: Evaluation results for both proposal

Model	Views	DCG	MAP
embedding-based	16	0.590	0.092
Learned Features	-	0.641	0.280
end-to-end	16	0.707	0.379

Table 10.3: Evaluation results over the ModelNet dataset

Model	Views	DCG	MAP
embedding-based	16	0.617	0.184
end-to-end	16	0.747	0.454

Chapter 11

Conclusions and future work

This work exposes and details our research and contributions to several 3D model retrieval tasks. This task includes engineered features and deep features for 3D model representation and different mid-level representation techniques, and 3D model embedding techniques.

11.1 Summary of Contributions

Our first approach in this research field consists of using engineered features. These features were the standard approach in the area for many years and maintain their relevance in scenarios where deep learning techniques are not suitable, like, for example, lack of training data. In our case, we compute the engineered features using Gabor filters, which have proven to be very efficient in image representation. After that, we use the extracted features to build bag-of-words to represent the 3D models and the images. Finally, we use these representations for 3D model retrieval and image-based 3D model retrieval achieving competitive results against similar methods.

We further extend this research in engineered features by proposing a new mid-level representation technique called slider to substitute the previous bag-of-words methods. This technique allows us to explore a full spectrum of mid-level representation for multimedia data.

We then expand our research to deep representations of 3D shapes using deep learning techniques. First, we experiment with a naive approach using a standard Convolutional network for classification tasks. The idea is to train this network to classify the image views until we achieve high accuracy. Then we use the trained network to produce image view representations. Finally, we use these learned representations for 3D shape retrieval by measuring the similarities between image views. With this proposal, we obtained the best result for 3D retrieval so far.

Then we further increase our retrieval performance by proposing a sophisticated new artificial neural network to compute 3D model embeddings. In this case, we combine different network architectures while imposing additional restrictions on the whole model to improve the quality of the calculated embeddings. With this proposal, we obtain results competitive with state-of-the-art with an easier-to-train and less resource-demanding network. We also integrated our proposed network for computing 3D model embedding into another end-to-end network to measure the similarity between an image and a 3D model. This last proposal aims to build a framework for image-based 3D model retrieval.

With this last work, we successfully computed embedding representations for 3D models using image views and applied those embeddings to 3D shape retrieval, obtaining competitive results with state-of-the-art methods. However, our proposal avoids the standard approach in the field to combine the features from each view by incorporating view pooling layers. Instead, our approach learns features directly from a single tensor with all the views. Besides, we do not need to specify an order for the image views. Which makes our architecture significantly more time and resource-efficient than the standard approach.

We also use our research on more practical applications in the works: "Motif-driven Retrieval of Greek Painted Pottery" and "A sketch-aided retrieval approach for incomplete 3D objects". Finally, we test our best model against the state-of-the-art proposals with our participation in the SHREC (3D Shape Retrieval Challenge), achieving competitive results.

Throughout our research, we believe that we have accomplished our proposed objectives. Table 11.1 summarizes all proposed goals and the chapters where we tackled them.

11.2 Future Work

3D model retrieval involves a wide variety of issues yet to be explored, especially in fields like cross-modal retrieval. Although this thesis contributes with new techniques in this direction, we can still expand or add to our research. We detail below some of these possible future works.

Improvement of the computed embedding for 3D model: Our network to compute embeddings from 3D models was very competitive when used over the normalized 3D models. However, we see a decrease in the performance of our model when used with rotated 3D models. In this case, we increase the proposal's performance using a different image view selection technique. We could enhance this performance even further if we manage to reduce the impact of the rotation in the learning process of our network. Some work has proposed before rotation-invariant features extraction from 3D models. So this is an achievable goal.

Application of the computed embeddings in different tasks: One of the most relevant contributions of our research is precisely our technique to compute 3D model embeddings. We obtain good results for the two evaluation metrics, especially with our last model, which shows results for the MAP above 73 percent and DCG above 91 percent. These results are very competitive with the Shapenet Dataset results, especially in the normalized version of the dataset. However, we could extend this research to other tasks related to 3D model processing like recognition, classification, segmentation, compression, or even another cross-modal 3D model retrieval like sketch-based retrieval. In these cases, we need to adapt the network to impose specific learning restrictions according to the tackled task.

Improvement of our proposal for image-based 3D model retrieval: We also explore the problem of image-based 3D model retrieval using a learning approach. We propose two techniques: the use of pre-trained embeddings and embeddings obtained through an end-to-end learning approach. We observe that learning embeddings jointly for images and 3D objects is beneficial to increase the retrieval performance compared to embeddings learned separately. However, although our end-to-end approach shows promising retrieval results, the problem is far from being solved. The study and analysis of complementary representations for 3D objects could enhance the input data for a learning approach. There is also an imperative need for a suitable dataset to evaluate the image-based 3D model retrieval problem. Also, we tackled the issue of learning similarities between images and 3D models with a siamese network. We could use another metric learning approach like Triplet Networks. Finally, since this proposal depends on the computed embeddings for 3D models and images, we could improve the overall performance by improving the quality of the embeddings.

Table 11.1: Objectives by Chapter

Objective	Chapters
Develop methods to solve the 3D shape retrieval (cross-modality 3D shape retrieval, partial 3D shape retrieval) and classification tasks using neural networks	4,5,6,8,10
Select and implement the current methods that perform best in the area of 3D shape retrieval (cross-modality 3D shape retrieval, partial 3D shape retrieval) and classification	4,8
Establish standard benchmarks for experimentation	4,5,6,8,10
Establish metrics to measure the performance of the methods	4,5,6,8,10
Make comparisons between the proposed methods and the most important methods in the area	4,8,9
Test the proposed methods in real scenarios and refine them if necessary to improve the results	7,9

Bibliography

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas J. Guibas. Learning representations and generative models for 3d point clouds. In 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Workshop Track Proceedings. OpenReview.net, 2018.

[2] Eman Ahmed, Alexandre Saint, Abd El Rahman Shabayek, Kseniya Cherenkova, Rig Das, Gleb Gusev, Djamila Aouada, and Björn E. Ottersten. Deep learning advances on different 3d data representations: A survey. CoRR, abs/1808.01462, 2018.

[3] Sakifa Aktar and Md Al Mamun. Multi-view 3d object retrieval using autoencoder & deep embedding network. In 2019 International Conference on Electrical, Computer and Communication Engineering (ECCE), pages 1–6. IEEE, 2019.

[4] Raquel Almeida, Benjamin Bustos, Zenilton Kleber G. do Patrocínio Jr., and Silvio Jamil Ferzoli Guimarães. Human action classification using an extended bow formalism. In Image Analysis and Processing - ICIAP 2017 - 19th International Conference, Catania, Italy, September 11-15, 2017, Proceedings, Part I, pages 185–196, 2017.

[5] Aymé Arango, Jesus Perez-Martin, and Arniel Labrada. Hateu at semeval-2022 task 5: Multimedia automatic misogyny identification. In Guy Emerson, Natalie Schluter, Gabriel Stanovsky, Ritesh Kumar, Alexis Palmer, Nathan Schneider, Siddharth Singh, and Shyam Ratan, editors, Proceedings of the 16th International Workshop on Semantic Evaluation, SemEval@NAACL 2022, Seattle, Washington, United States, July 14-15, 2022, pages 581–584. Association for Computational Linguistics, 2022.

[6] Emad Attalla and Pepe Siy. Robust shape similarity retrieval based on contour segmentation polygonal multiresolution and elastic matching. Pattern Recognit., 38(12):2229–2241, 2005.

[7] Authors. A brief survey on 3d semantic segmentation of lidar point cloud with deep learning. In 3rd Novel Intelligent and Leading Emerging Sciences Conference, NILES 2021, Giza, Egypt, October 23-25, 2021, pages 405–408, 2021.

[8] Dana H. Ballard. Modular learning in neural networks. In Kenneth D. Forbus and Howard E. Shrobe, editors, Proceedings of the 6th National Conference on Artificial Intelligence. Seattle, WA, USA, July 1987, pages 279–284. Morgan Kaufmann, 1987.

[9] Silvia Biasotti, Andrea Cerri, Mostafa Abdelrahman, Masaki Aono, Abdessamad Ben Hamza, Moumen T. El-Melegy, Aly A. Farag, Valeria Garro, Andrea Giachetti, Daniela Giorgi, Afzal Godil, C. Li, Y.-J. Liu, H. Y. Martono, Chika Sanada, Atsushi Tatsuma, Santiago Velasco-Forero, and C.-X. Xu. Retrieval and classification on textured 3d models. In Eurographics Workshop on 3D Object Retrieval, Strasbourg, France, 2014. Proceedings, pages 111–120, 2014.

[10] Y-Lan Boureau, Francis R. Bach, Yann LeCun, and Jean Ponce. Learning mid-level features for recognition. In The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2010, San Francisco, CA, USA, 13-18 June 2010, pages 2559–2566, 2010.

[11] Yuri Boykov and Gareth Funka-Lea. Graph cuts and efficient N-D image segmentation. Int. J. Comput. Vis., 70(2):109–131, 2006.

[12] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a siamese time delay neural network. In Advances in Neural Information Processing Systems 6, [7th NIPS Conference, Denver, Colorado, USA, 1993], pages 737–744, 1993.

[13] Andrea Cerri, Silvia Biasotti, Mostafa Abdelrahman, Jesús Angulo, K. Berger, Louis Chevallier, Moumen T. El-Melegy, Aly A. Farag, F. Lefebvre, Andrea Giachetti, H. Guermoud, Y.-J. Liu, Santiago Velasco-Forero, Jean-Ronan Vigouroux, C.-X. Xu, and J.-B. Zhang. Shrec'13 track: Retrieval on textured 3d models. In Eurographics Workshop on 3D Object Retrieval, Girona, Spain, 2013. Proceedings, pages 73–80, 2013.

[14] Angel X. Chang, Thomas A. Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qi-Xing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. Shapenet: An information-rich 3d model repository. CoRR, abs/1512.03012, 2015.

[15] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE Trans. Pattern Anal. Mach. Intell., 40(4):834–848, 2018.

[16] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In Proceedings of the European conference on computer vision (ECCV), pages 801–818, 2018.

[17] Xuelin Chen, Baoquan Chen, and Niloy J. Mitra. Unpaired point cloud completion on real scans using adversarial training. In 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020. OpenReview.net, 2020.

[18] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. Supervised learning of universal sentence representations from natural language inference data. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017, pages 670–680, 2017.

[19] Nicu D. Cornea, M. Fatih Demirci, Deborah Silver, Ali Shokoufandeh, Sven J. Dickinson, and Paul B. Kantor. 3d object retrieval using many-to-many matching of curve skeletons. In 2005 International Conference on Shape Modeling and Applications (SMI 2005), 15-17 June 2005, Cambridge, MA, USA, pages 368–373, 2005.

[20] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005), 20-26 June 2005, San Diego, CA, USA, pages 886–893. IEEE Computer Society, 2005.

[21] Eli David and Nathan S. Netanyahu. Deeppainter: Painter classification using deep convolutional autoencoders. CoRR, abs/1711.08763, 2017.

[22] Omid E. David and Nathan S. Netanyahu. Deeppainter: Painter classification using deep convolutional autoencoders. In Alessandro E. P. Villa, Paolo Masulli, and Antonio Javier Pons Rivero, editors, Artificial Neural Networks and Machine Learning - ICANN 2016 - 25th International Conference on Artificial Neural Networks, Barcelona, Spain, September 6-9, 2016, Proceedings, Part II, volume 9887 of Lecture Notes in Computer Science, pages 20–28. Springer, 2016.

[23] Sandra Eliza Fontes de Avila, Nicolas Thome, Matthieu Cord, Eduardo Valle, and Arnaldo de Albuquerque Araújo. BOSSA: extended bow formalism for image classification. In 18th IEEE International Conference on Image Processing, ICIP 2011, Brussels, Belgium, September 11-14, 2011, pages 2909–2912, 2011.

[24] Doug DeCarlo, Adam Finkelstein, Szymon Rusinkiewicz, and Anthony Santella. Suggestive contours for conveying shape. ACM Transactions on Graphics (TOG), 22(3):848–855, 2003.

[25] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers), pages 4171–4186, 2019.

[26] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. CoRR, abs/1603.07285, 2016.

[27] Mathias Eitz, Ronald Richter, Tamy Boubekeur, Kristian Hildebrand, and Marc Alexa. Sketch-based shape retrieval. ACM Trans. Graph., 31(4):31–1, 2012.

[28] Mathias Eitz, Ronald Richter, Tamy Boubekeur, Kristian Hildebrand, and Marc Alexa. Sketch-based shape retrieval. ACM Trans. Graph., 31(4):31:1–31:10, 2012.

[29] Takahiko Furuya and Ryutarou Ohbuchi. Dense sampling and fast encoding for 3d model retrieval using bag-of-visual features. In Stéphane Marchand-Maillet and Yiannis Kompatsiaris, editors, Proceedings of the 8th ACM International Conference on Image and Video Retrieval, CIVR 2009, Santorini Island, Greece, July 8-10, 2009. ACM, 2009.

[30] Andrea Giachetti, Francesco Farina, Francesco Fornasa, Atsushi Tatsuma, Chika Sanada, Masaki Aono, Silvia Biasotti, Andrea Cerri, and Sungbin Choi. Retrieval of non-rigid (textured) shapes using low quality 3d models. In Eurographics Workshop on 3D Object Retrieval, Zurich, Switzerland, May 2-3, 2015., pages 137–144, 2015.

[31] Albert Gordo, Jon Almazán, Jérôme Revaud, and Diane Larlus. Deep image retrieval: Learning global representations for image search. In Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VI, pages 241–257, 2016.

[32] Robert Gregor, Danny Bauer, Ivan Sipiran, Panagiotis Perakis, and Tobias Schreck. Automatic 3d object fracturing for evaluation of partial retrieval and object restoration tasks - benchmark and application to 3d cultural heritage data. In Eurographics Workshop on 3D Object Retrieval, Zurich, Switzerland, May 2-3, 2015., pages 7–14, 2015.

[33] Haiyun Guo, Jinqiao Wang, Yue Gao, Jianqiang Li, and Hanqing Lu. Multi-view 3d object retrieval with deep embedding network. IEEE Trans. Image Process., 25(12):5526–5537, 2016.

[34] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep learning for 3d point clouds: A survey. CoRR, abs/1912.12033, 2019.

[35] Zhizhong Han, Honglei Lu, Zhenbao Liu, Chi-Man Vong, Yu-Shen Liu, Matthias Zwicker, Junwei Han, and C. L. Philip Chen. 3d2seqviews: Aggregating sequential views for 3d global feature learning by CNN with hierarchical attention aggregation. IEEE Trans. Image Process., 28(8):3986–3999, 2019.

[36] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In Proceedings of the IEEE international conference on computer vision, pages 2961–2969, 2017.

[37] Anastasia Ioannidou, Elisavet Chatzilari, Spiros Nikolopoulos, and Ioannis Kompatsiaris. Deep learning advances in computer vision with 3d data: A survey. ACM Comput. Surv., 50(2):20:1–20:38, 2017.

[38] Asako Kanezaki, Yasuyuki Matsushita, and Yoshifumi Nishida. Rotationnet for joint object categorization and unsupervised pose estimation from multi-view images. IEEE Trans. Pattern Anal. Mach. Intell., 43(1):269–283, 2021.

[39] Shun Kawamura, Kazuya Usui, Takahiko Furuya, and Ryutarou Ohbuchi. Local goemetrical feature with spatial context for shape-based 3d model retrieval. In Eurographics Workshop on 3D Object Retrieval 2012, Cagliari, Italy, May 13, 2012. Proceedings, pages 55–58, 2012.

[40] Michael M. Kazhdan, Thomas A. Funkhouser, and Szymon Rusinkiewicz. Rotation invariant spherical harmonic representation of 3d shape descriptors. In First Eurographics Symposium on Geometry Processing, Aachen, Germany, June 23-25, 2003, pages 156–164, 2003.

[41] Douwe Kiela and Léon Bottou. Learning image embeddings using convolutional neural networks for improved multi-modal semantics. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL, pages 36–45, 2014.

[42] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States., pages 1106–1114, 2012.

[43] Arniel Labrada, Benjamin Bustos, and Ivan Sipiran. A convolutional architecture for 3d model embedding using image views. The Visual Computer, pages 1–15, 2023.

[44] Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. Backpropagation applied to handwritten zip code recognition. Neural Comput., 1(4):541–551, 1989.

[45] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324, 1998.

[46] Tang Lee, Yen-Liang Lin, HungYueh Chiang, Ming-Wei Chiu, Winston H. Hsu, and Polly Huang. Cross-domain image-based 3d shape retrieval by view sequence learning. In 2018 International Conference on 3D Vision, 3DV 2018, Verona, Italy, September 5-8, 2018, pages 258–266. IEEE Computer Society, 2018.

[47] Stefan Lengauer, Alexander Komar, Arniel Labrada, Stephan Karl, Elisabeth Trinkl, Reinhold Preiner, Benjamin Bustos, and Tobias Schreck. Motif-driven retrieval of greek painted pottery. In Selma Rizvic and Karina Rodriguez-Echavarria, editors, GCH 2019 - Eurographics Workshop on Graphics and Cultural Heritage, GCH 2019, Sarajevo, Bosnia and Herzegovina, November 6-9, 2019, pages 89–98. Eurographics Association, 2019.

[48] Stefan Lengauer, Alexander Komar, Arniel Labrada, Stephan Karl, Elisabeth Trinkl, Reinhold Preiner, Benjamin Bustos, and Tobias Schreck. Sketch-aided retrieval of incomplete 3d cultural heritage objects. In Silvia Biasotti, Guillaume Lavoué, and Remco C. Veltkamp, editors, 12th Eurographics Workshop on 3D Object Retrieval, 3DOR@Eurographics 2019, Genoa, Italy, May 5-6, 2019, pages 17–24. Eurographics Association, 2019.

[49] Stefan Lengauer, Alexander Komar, Arniel Labrada, Stephan Karl, Elisabeth Trinkl, Reinhold Preiner, Benjamin Bustos, and Tobias Schreck. A sketch-aided retrieval approach for incomplete 3d objects. Comput. Graph., 87:111–122, 2020.

[50] Bo Li, Afzal Godil, Masaki Aono, X. Bai, Takahiko Furuya, L. Li, Roberto Javier López-Sastre, Henry Johan, Ryutarou Ohbuchi, Carolina Redondo-Cabrera, Atsushi Tatsuma, Tomohiro Yanagimachi, and S. Zhang. Shrec'12 track: Generic 3d shape retrieval. In Eurographics Workshop on 3D Object Retrieval 2012, Cagliari, Italy, May 13, 2012. Proceedings, pages 119–126, 2012.

[51] Bo Li and Henry Johan. View context: A 3d model feature for retrieval. In Advances in Multimedia Modeling, 16th International Multimedia Modeling Conference, MMM 2010, Chongqing, China, January 6-8, 2010. Proceedings, pages 185–195, 2010.

[52] Bo Li and Henry Johan. Sketch-based 3d model retrieval by incorporating 2d-3d alignment. Multimedia Tools and Applications, 65(3):363–385, 2013.

[53] Bo Li, Yijuan Lu, Afzal Godil, Tobias Schreck, Benjamin Bustos, Alfredo Ferreira, Takahiko Furuya, Manuel J Fonseca, Henry Johan, Takahiro Matsuda, et al. A comparison of methods for sketch-based 3d shape retrieval. Computer Vision and Image Understanding, 119:57–80, 2014.

[54] Bo Li, Yijuan Lu, Afzal Godil, Tobias Schreck, Benjamin Bustos, Alfredo Ferreira, Takahiko Furuya, Manuel J. Fonseca, Henry Johan, Takahiro Matsuda, Ryutarou Ohbuchi, Pedro B. Pascoal, and Jose M. Saavedra. A comparison of methods for sketch-based 3d shape retrieval. Computer Vision and Image Understanding, 119:57–80, 2014.

[55] Bo Li, Tobias Schreck, Afzal Godil, Marc Alexa, Tamy Boubekeur, Benjamin Bustos, J. Chen, Mathias Eitz, Takahiko Furuya, Kristian Hildebrand, S. Huang, Henry Johan,

Arjan Kuijper, Ryutarou Ohbuchi, Ronald Richter, Jose M. Saavedra, Maximilian Scherer, Tomohiro Yanagimachi, Gang-Joon Yoon, and Sang Min Yoon. Shrec'12 track: Sketch-based 3d shape retrieval. In Eurographics Workshop on 3D Object Retrieval 2012, Cagliari, Italy, May 13, 2012. Proceedings, pages 109–118, 2012.

[56] Bo Li, Tobias Schreck, Afzal Godil, Marc Alexa, Tamy Boubekeur, Benjamin Bustos, Jipeng Chen, Mathias Eitz, Takahiko Furuya, Kristian Hildebrand, et al. Shrec'12 track: Sketch-based 3d shape retrieval. In 3DOR, pages 109–118, 2012.

[57] Haisheng Li, Li Sun, Shuilong Dong, Xiaobin Zhu, Qiang Cai, and Junping Du. Efficient 3d object retrieval based on compact views and hamming embedding. IEEE Access, 6:31854–31861, 2018.

[58] Dongyun Lin, Yiqun Li, Yi Cheng, Shitala Prasad, Tin Lay Nwe, Sheng Dong, and Aiyuan Guo. Multi-view 3d object retrieval leveraging the aggregation of view and instance attentive features. Knowl. Based Syst., 247:108754, 2022.

[59] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 2117–2125, 2017.

[60] Yongcheng Liu, Bin Fan, Gaofeng Meng, Jiwen Lu, Shiming Xiang, and Chunhong Pan. Densepoint: Learning densely contextual representation for efficient point cloud processing. In 2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019, pages 5238–5247. IEEE, 2019.

[61] Yongcheng Liu, Bin Fan, Shiming Xiang, and Chunhong Pan. Relation-shape convolutional neural network for point cloud analysis. In IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019, pages 8895–8904. Computer Vision Foundation / IEEE, 2019.

[62] Zehua Liu, Yuhe Zhang, Jian Gao, and Shurui Wang. VFMVAC: view-filtering-based multi-view aggregating convolution for 3d shape recognition and retrieval. Pattern Recognit., 129:108774, 2022.

[63] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 3431–3440, 2015.

[64] Anh Nguyen and Bac Le. 3d point cloud segmentation: A survey. In IEEE 6th International Conference on Robotics, Automation and Mechatronics, RAM 2013, Manila, Philippines, November 12-15, 2013, pages 225–230, 2013.

[65] Weizhi Nie, Minjie Ren, Anan Liu, Zhendong Mao, and Jie Nie. M-gcn: Multi-branch graph convolution network for 2d image-based on 3d model retrieval. IEEE Transactions on Multimedia, 2020.

[66] Panagiotis Papadakis, Ioannis Pratikakis, Theoharis Theoharis, Georgios Passalis, and Stavros J. Perantonis. 3d object retrieval using an efficient and compact hybrid shape descriptor. In Eurographics Workshop on 3D Object Retrieval, 3DOR 2008, Crete, Greece, 2008. Proceedings, pages 9–16, 2008.

[67] Panagiotis Papadakis, Ioannis Pratikakis, Theoharis Theoharis, and Stavros J. Perantonis. PANORAMA: A 3d shape descriptor based on panoramic views for unsupervised 3d object retrieval. International Journal of Computer Vision, 89(2-3):177–192, 2010.

[68] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL, pages 1532–1543, 2014.

[69] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017, pages 77–85, 2017.

[70] Charles Ruizhongtai Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas J. Guibas. Volumetric and multi-view cnns for object classification on 3d data. In 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016, pages 5648–5656. IEEE Computer Society, 2016.

[71] Xin Rong. word2vec parameter learning explained. CoRR, abs/1411.2738, 2014.

[72] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In International Conference on Medical image computing and computer-assisted intervention, pages 234–241. Springer, 2015.

[73] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning internal representations by error propagation, 1985.

[74] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 4510–4520, 2018.

[75] Michalis A. Savelonas, Ioannis Pratikakis, and Konstantinos Sfikas. Partial 3d object retrieval combining local shape descriptors with global fisher vectors. In Eurographics

Workshop on 3D Object Retrieval, Zurich, Switzerland, May 2-3, 2015., pages 23–30, 2015.

[76] Manolis Savva, Fisher Yu, Hao Su, M Aono, B Chen, D Cohen-Or, W Deng, Hang Su, Song Bai, Xiang Bai, et al. Shrec16 track: largescale 3d shape retrieval from shapenet core55. In Proceedings of the eurographics workshop on 3D object retrieval, volume 10, 2016.

[77] Alize E. H. Scheenstra, Arnout C. Ruifrok, and Remco C. Veltkamp. A survey of 3d face recognition methods. In Takeo Kanade, Anil K. Jain, and Nalini K. Ratha, editors, Audio- and Video-Based Biometric Person Authentication, 5th International Conference, AVBPA 2005, Hilton Rye Town, NY, USA, July 20-22, 2005, Proceedings, volume 3546 of Lecture Notes in Computer Science, pages 891–899. Springer, 2005.

[78] Konstantinos Sfikas, Ioannis Pratikakis, Anestis Koutsoudis, Michalis A. Savelonas, and Theoharis Theoharis. Partial matching of 3d cultural heritage objects using panoramic views. Multimedia Tools Appl., 75(7):3693–3707, 2016.

[79] Konstantinos Sfikas, Ioannis Pratikakis, and Theoharis Theoharis. Ensemble of panorama-based convolutional neural networks for 3d model classification and retrieval. Computers & Graphics, 71:208–218, 2018.

[80] Konstantinos Sfikas, Theoharis Theoharis, and Ioannis Pratikakis. Rosy+: 3d object pose normalization based on PCA and reflective object symmetry with application in 3d object retrieval. International Journal of Computer Vision, 91(3):262–279, 2011.

[81] Konstantinos Sfikas, Theoharis Theoharis, and Ioannis Pratikakis. Pose normalization of 3d models via reflective symmetry on panoramic views. The Visual Computer, 30(11):1261–1274, 2014.

[82] Konstantinos Sfikas, Theoharis Theoharis, and Ioannis Pratikakis. Exploiting the PANORAMA representation for convolutional neural network classification and retrieval. In Eurographics Workshop on 3D Object Retrieval, 3DOR 2017, Lyon, France, April 23-24, 2017, 2017.

[83] Ariel Shamir. A survey on mesh segmentation techniques. Comput. Graph. Forum, 27(6):1539–1556, 2008.

[84] Philip Shilane, Patrick Min, Michael M. Kazhdan, and Thomas A. Funkhouser. The princeton shape benchmark. In 2004 International Conference on Shape Modeling and Applications (SMI 2004), 7-9 June 2004, Genova, Italy, pages 167–178, 2004.

[85] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In Yoshua Bengio and Yann LeCun, editors, 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015.

[86] Ayan Sinha, Jing Bai, and Karthik Ramani. Deep learning 3d shape surfaces using geometry images. In Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VI, pages 223–240, 2016.

[87] Ivan Sipiran, Patrick Lazo, Cristian Lopez, Milagritos Jimenez, Nihar Bagewadi, Benjamin Bustos, Hieu Dao, Shankar Gangisetty, Martin Hanik, Ngoc-Phuong Ho-Thi, et al. Shrec 2021: Retrieval of cultural heritage objects. Computers & Graphics, 2021.

[88] Ivan Sipiran, Patrick Lazo, Cristian Lopez, Milagritos Jimenez, Nihar Bagewadi, Benjamin Bustos, Hieu Dao, Shankar Gangisetty, Martin Hanik, Ngoc-Phuong Ho-Thi, Mike Holenderski, Dmitri Jarnikov, Arniel Labrada, Stefan Lengauer, Roxane Licandro, Dinh-Huan Nguyen, Thang-Long Nguyen-Ho, Luis A. Pérez Rey, Bang-Dang Pham, Reinhold Preiner, Tobias Schreck, Quoc-Huy Trinh, Loek Tonnaer, Christoph von Tycowicz, and The-Anh Vu-Le. SHREC 2021: Retrieval of cultural heritage objects. Comput. Graph., 100:1–20, 2021.

[89] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhöfer. Deepvoxels: Learning persistent 3d feature embeddings. In IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019, pages 2437–2446, 2019.

[90] Michela Spagnuolo and Bianca Falcidieno. 3d media and the semantic web. IEEE Intelligent Systems, 24(2):90–96, 2009.

[91] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik G. Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In 2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015, pages 945–953, 2015.

[92] Yu Ting Su, Yu Qian Li, Dan Song, An-An Liu, and Jie Nie. Joint intermediate domain generation and distribution alignment for 2d image-based 3d objects retrieval. IEEE Transactions on Multimedia, 2020.

[93] Kai Sun, Jiangshe Zhang, Junmin Liu, Ruixuan Yu, and Zengjie Song. DRCNN: dynamic routing convolutional neural network for multi-view 3d object recognition. IEEE Trans. Image Process., 30:868–877, 2021.

[94] H. Sundar, Deborah Silver, Nikhil Gagvani, and Sven J. Dickinson. Skeleton based shape matching and retrieval. In 2003 International Conference on Shape Modeling and Applications (SMI 2003), 12-16 May 2003, Seoul, Korea, pages 130–142, 290, 2003.

[95] Volodymyr Turchenko, Eric Chalmers, and Artur Luczak. A deep convolutional auto-encoder with pooling - unpooling layers in caffe. CoRR, abs/1701.04949, 2017.

[96] Fang Wang, Le Kang, and Yi Li. Sketch-based 3d shape retrieval using convolutional neural networks. In IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015, pages 1875–1883, 2015.

[97] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph CNN for learning on point clouds. CoRR, abs/1801.07829, 2018.

[98] Xin Wei, Ruixuan Yu, and Jian Sun. View-gcn: View-based graph convolutional network for 3d shape analysis. In 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020, pages 1847–1856. Computer Vision Foundation / IEEE, 2020.

[99] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015, pages 1912–1920. IEEE Computer Society, 2015.

[100] Kwang Moo Yi, Eduard Trulls, Vincent Lepetit, and Pascal Fua. LIFT: learned invariant feature transform. In Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VI, pages 467–483, 2016.

[101] Kuangen Zhang, Ming Hao, Jing Wang, Clarence W. de Silva, and Chenglong Fu. Linked dynamic graph CNN: learning on point cloud via linking hierarchical features. CoRR, abs/1904.10014, 2019.

[102] Long Zhao, Shuang Liang, Jinyuan Jia, and Yichen Wei. Learning best views of 3d shapes from sketch contour. Vis. Comput., 31(6-8):765–774, 2015.

[103] Jing Zhu, John-Ross Rizzo, and Yi Fang. Learning domain-invariant feature for robust depth-image-based 3d shape retrieval. Pattern Recognit. Lett., 119:24–33, 2019.

[104] Keneilwe Zuva and Tranos Zuva. Evaluation of information retrieval systems. International journal of computer science & information technology, 4(3):35, 2012.