

# A Convolutional Architecture for 3D Model Embedding

Arniel Labrada, Benjamin Bustos, Ivan Sipiran

March 2021

## Abstract

During the last years, many advances have been made in tasks like 3D model retrieval, 3D model classification, and 3D model segmentation. The typical 3D representations such as point clouds, voxels, and polygon meshes are mostly suitable for rendering purposes, while their use for cognitive processes (retrieval, classification, segmentation) is limited due to their high redundancy and complexity. We propose a deep learning architecture to handle 3D models as an input. We combine this architecture with other standard architectures like Convolutional Neural Networks and autoencoders for computing 3D model embeddings. Our goal is 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 which tries to capture the relevant information of a 3D model, we show that the embedding representation conveys semantic information that helps to deal with the similarity assessment of 3D objects. Our experiments show the benefit of computing the embeddings of a 3D model data set and use them for effective 3D Model Retrieval.

## 1 Introduction

Since the outstanding results obtained by AlexNet [16] for image classification in 2012, the architectures of neural networks, specifically Convolutional Neural Networks (*CNNs*) [17], have been continuously improving to solve visual computing tasks. Especially in the image processing field, *CNNs* have been used to solve the most relevant image processing 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 [14]. 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.

In this work, we propose three different neural network architectures for computing 3D model embeddings. For this purpose, we first render a set of image views from the 3D models. To process these image view sets, we propose a neural network module to consume them as input. Then, we combine this module with other standard neural network architectures like *CNNs* and autoencoders to obtain the 3D model embeddings.

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.

This paper presents the three main contributions:

- We propose a convolutional architecture that can handle the 3D models represented as sets of image views.
- We give an analysis of the performance of different convolutional architectures for 3D model embedding.
- We study how the number of image views affects the quality of the 3D model embedding, by experimenting with different amount of image views (2, 3, 4) to represent the models.

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 outstanding results for the 3D model retrieval task.

## 2 State of the Art

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 [18] and textured 3D model retrieval [4].

### 2.1 3D shape retrieval and classification using Neural networks

Recently, some works propose using neural network techniques instead of extracting features to solve the 3D model retrieval and classification tasks with

excellent results [14]. 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 a neural network model.

The first two approaches consist of representing 3D models as a set of image views and voxel grids, respectively [14, 12]. 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 neural network models, which are more expensive to train and need much more training data.

The third approach consists of the use of point cloud representations to train deep learning models. This approach has recently increased in popularity with some outstanding works like PointNet [23], Dynamic Graph CNN for Learning on Point Clouds [29], RS-CNN [20], and LDGCNN [31]. A survey on this topic can be found in Deep Learning for 3D Point Clouds: A Survey [12].

## 2.2 Image Views Representation

Image views is 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 [14]. The reason behind this is that training a deep learning model directly with any of the standards 3D models representation (voxel grid, polygon mesh) could be a very costly process both in time and resources.

One of the earliest proposals for 3D shape retrieval using Convolutional Neural Networks is DeepEm [11]. 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 [26] 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.

Su et al. [2] further explores the concept of using triplet loss with a convolutional neural network to compute image embedding and then using these embedding for 3D object retrieval. They also extend their proposal to unsupervised learning by using a convolutional autoencoder.

Su et al. [27] 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* ar-

chitecture 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. Su et al. [25] proposed an extension of the PANORAMA 3D shape representation [21]. 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 states of the art techniques achieving very competitive results.

More recently, Su et al. [13] 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.

### 2.3 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 [24], Glove [22]) and sentence embedding (e.g., InferSent [8], BERT [9]).

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 [30], “Learning Image Embeddings using Convolutional Neural Networks for Improved Multi-Modal Semantics” [15], and “Deep Image Retrieval: Learning Global Representations for Image Search” [10]. However, there is not much research on computing 3D shape embedding. Moreover, the few works that compute embedding for 3D shapes are mainly based on point cloud representations [1, 7] or voxel grids. We can also find a proposal for computing 3D shapes embedding using engineered features and the bag-of-words framework [19]. However, although some proposals indirectly use the concept, to our knowledge, no method has been proposed yet for 3D model embeddings using image view representation and *CNNs*.

## 3 Background Knowledge

In this section, we discuss some background concepts for a better understanding of our proposal.

### 3.1 Convolutional Neural Networks (*CNNs*)

Convolutional Neural Networks (*CNNs*) [17] are models of neural networks that have shown outstanding performance in different tasks. In the area of image processing, since the excellent results obtained in 2012 by AlexNet [16], these networks have been used to solve various tasks like image recognition, image classification, object detection, and face recognition.

The standard architecture of *CNNs* consists of a series of convolutional layers and pooling layers. The convolutional layers employ a convolution operation instead of simple matrix multiplication applied in traditional neural networks. The output of a convolutional layer is a feature map as the result of the convolution operation with a defined  $n \times n$  kernel. Thus, the network learns the filters that in traditional algorithms were hand-engineered. On the other hand, the pooling layers are used to reduce the dimensionality of the feature maps. There are several pooling layers, for example, max pooling, average pooling, and sum pooling.

For the image classification task, we use one or more fully connected layers on top of the convolutional layers and pooling layers. Then, we connect the last fully connected layer to a classifier that is usually a softmax function. Finally, we use the cross-entropy as the loss function.

### 3.2 Autoencoders

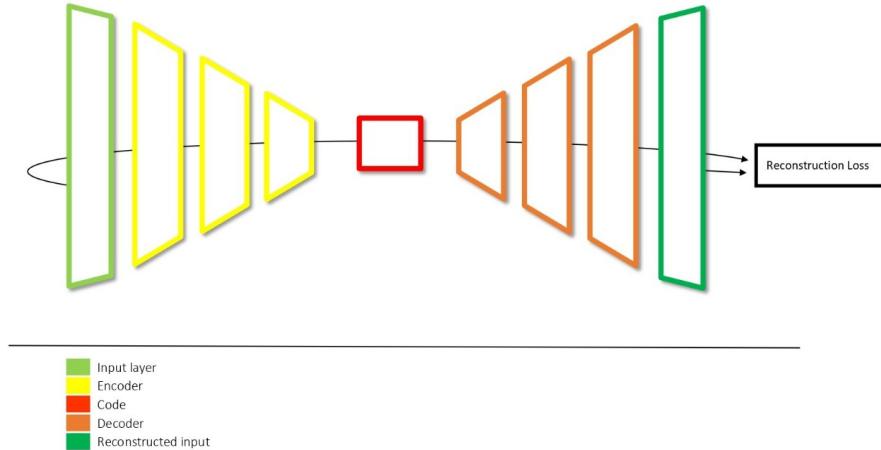


Figure 1: A standard autoencoder architecture

Autoencoders [3] are an unsupervised learning technique mainly used for data representation (embedding, deep features, etc.) and data compression.

We create these networks to impose a bottleneck in their design, which forces a compressed knowledge representation of the original input. In more detail, an autoencoder architecture consists of four parts: encoder, bottleneck, decoder, and reconstruction loss. The encoder learns how to reduce the input dimensions; the bottleneck is the layer that contains the compressed representation of the input data; the decoder reconstructs the data from the bottleneck; the reconstruction loss measures how well the decoder is performing and how close the output is to the original input. Figure 1 shows a standard autoencoder architecture.

In the image processing field, these architectures are used as well for image compression and image representation. The most common network used to this end is the convolutional autoencoder, which uses the mentioned encoder-decoder architecture with some adjustment. The encoder part of a convolutional autoencoder consists of a standard convolutional neural network built with convolutional layers and pooling layers to reduce the resolution. Then, the encoder is connected to a fully connected layer, which serves as the bottleneck. Finally, the bottleneck is the input to the network’s decoder, built with what is known as de-convolutional layers and un-pooling layers. A de-convolutional layer is just the transposed of its corresponding convolutional layer. For the un-pooling layers, it is a bit more complicated. These layers are used to increase the resolution, the inverse operation of its corresponding pooling layer. The problem is that the usual pooling layers used, like max-pooling, are non-invertible operations. For this reason, several un-pooling techniques have been defined in the last years [6, 28].

## 4 A New Convolutional Architecture for 3D Model Embedding and Retrieval

We propose a convolutional architecture for 3D model embedding. The idea is to obtain a 3D model deep 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 here is that a standard convolutional network uses an image as input, and in our case, we have a set of images as input. To solve this, we model the information as a multi-channel image where the number of channels is the number of image views that we use to represent a 3D model. With our proposed architecture to handle the image view sets as input, we develop three convolutional architectures for 3D model embedding.

### 4.1 Autoencoder Network

Our first proposal is a convolutional autoencoder architecture. In this model, we connect our proposed multi-channel input to a standard convolutional neural network, which serves as the model’s encoder. Then, we connect the encoder

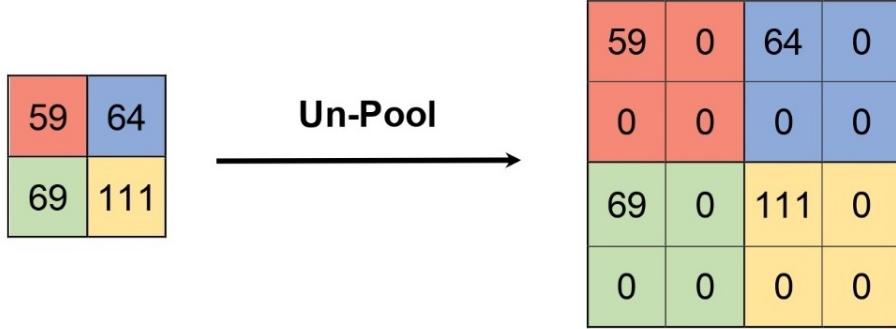


Figure 2: Un-pooling layer used for the decoder of the model

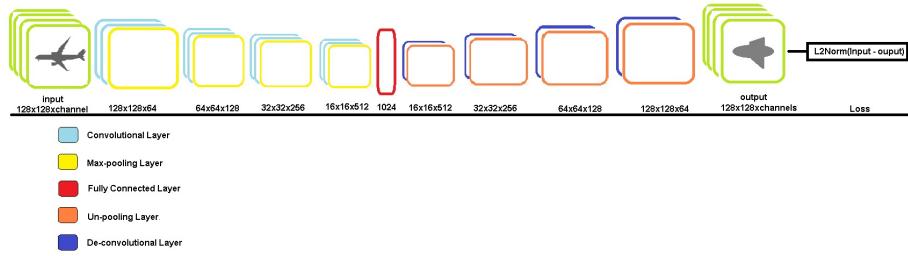


Figure 3: Convolutional autoencoder for 3D model embedding

to a fully connected layer (the bottleneck), which, at the same time, we link to the decoder of the network.

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 of the convolutional layers has a kernel of  $5 \times 5$  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. Figure 2 shows the un-pooling layer used in the model. Each of the deconvolutional layers has a kernel of  $5 \times 5$  and channels equal to  $64 * 2^k$ , where  $k$  is the respective block (4, 3, 2, 1).

Finally, we use as reconstruction loss the  $L_2$  norm of the difference between the output and the model's input. Figure 3 shows our convolutional autoencoder model.

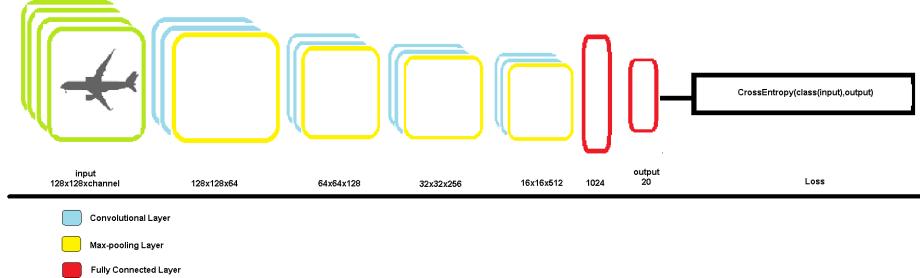


Figure 4: Classification network for 3D model embedding

## 4.2 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 our second proposal, which consists of a *CNN* for classification. This proposed network uses the same architecture as the autoencoder proposal 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. Figure 4 shows our convolutional model for classification.

## 4.3 Combination of the autoencoder and the classification network

Finally, we want our 3D model embedding network to have high compression power and learn the relation between the 3D models. With that in mind, our last proposal consists of a combination of both already described models. This new model uses the autoencoder and the classification network architecture at the same time. In other words, for a given input, the model computes the loss of the autoencoder and the loss of the classification network, and the final loss of this new network is the sum of both already calculated losses. After the training, we use once again the bottleneck layer as the embedding of the 3D models. Figure 5 shows this last proposed model.

## 5 Evaluation Metrics

We use two standard performance metrics for retrieval techniques to evaluate our proposal for 3D model embedding applied to 3D model retrieval. These metrics are based on a ranked list, consisting of all 3D models ordered according to their

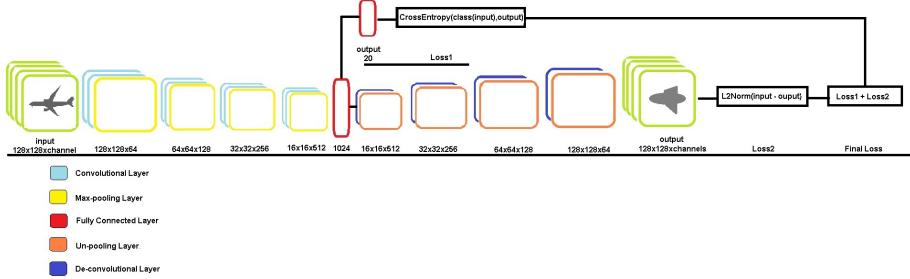


Figure 5: Combination of convolutional autoencoder and classification network for 3D model embedding

dissimilarity with a specific query. The metrics are Mean Average Precision *MAP*, and Discounted Cumulative Gain *DCG*. Their meanings and definitions are explained below.

The Mean Average Precision *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 (Eq. 1). The resultant value measures the quality of models at carrying out queries and is approximately equal to the area under the precision-recall curve [32].

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

The second metric, the *DCG* metric, uses the fact that users of a search engine 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. Thus, 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:

$$DCG_i = \begin{cases} G_1 & i = 1 \\ DCG_{i-1} + G_i / \log_2 i & i \neq 1 \end{cases} \quad (2)$$

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

$$DCG = \frac{DCG_n}{1 + \sum_{j=2}^{|C|} \frac{1}{\log_2 j}} \quad (3)$$

where  $n$  is the size of  $R$ .

## 5.1 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.

# 6 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.

## 6.1 Data configurations

We use the ShapeNet [5] 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 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 test, respectively. We conduct several experiments using both versions of the dataset.

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 our case, we first extract 30 image views per 3D model.

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<sup>1</sup> and fully parameterizable for different tasks. In our case, we use the code to render images from the 3D models starting with the camera, pointing to the coordinates origin. From there, we render an image every 12 degrees in circumference around the 3D model itself.

However, our models are not powerful enough to handle sets of 30 image views as input. To solve this problem, we add a new preprocessing step to the image view representation to reduce considerably the number of image views used to represent the 3D models. This new step consists of using a clustering algorithm over the set image views representing a 3D model to group them in  $k$  different clusters. After that, we choose a representative from each cluster, and

---

<sup>1</sup>[Stanford-Shapenet-renderer](#)

the final representation of the 3D model is the set of all  $k$  representatives. In our case, we use the k-means algorithm for the clustering of the image views with  $k$  centroids. As for the representatives, we choose the image view most similar to each centroid. To study the impact of the number of image views representing the 3D models, we use different parameter  $k$ . Specifically, we choose  $k$  in the set {2,3,4}.

## 6.2 Proposal setups and results

We propose three different *CNNs* architectures for computing 3D model embeddings. For all three proposals, 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 and Combination models (first and third proposed model, respectively), 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 proposal), we only use 20,000 iterations since the classification accuracy quickly achieves values above 98 percent.

Our goal is to use the embeddings computed with our proposals 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. Table 1 shows the micro averaged results of our proposed models using the normalized version of the dataset. We can see that the autoencoder model is the worst, and the combination of the autoencoder with the classification network achieves the best. We can also appreciate that the number of image views affects the result's quality: the more image views, the best performance of each model.

We also compare our proposed model against the four proposals that reported the best results in the ShapeNet dataset (RotationNet, GIFT, ReVGG, DLAN) [32] using both versions of the dataset. Table 2 and Table 3 show these comparisons using the normalized and the perturbed version of the dataset. We also compute the micro and macro averages of both metrics used in our experiments.

The results show that our proposed model achieves very competitive effectiveness, specially in the normalized version of the dataset where we obtain a very high *DCG*. We note that the embedding quality is affected by the number of image views representing the 3D models. For the three proposed methods, as the number of image views increases, the performance also increases. However, using a different number of image views makes little change in our last method's performance in the normalized version of the dataset.

We also note that the effectiveness of the combined method is affected when we use the perturbed version of the dataset. Nevertheless, when we increase the number of image views per 3D model in the dataset's perturbed version, the effectiveness increases notoriously. This phenomenon indicates that we need more images to capture each 3D model's information in the perturbed version. If each object has an arbitrary orientation, the chances of obtaining similar

Table 1: Evaluation of the performance of the embedding for 3D models Retrieval using the normalized version of the ShapeNet.

| Metric Calculation |       | Micro Average |              |
|--------------------|-------|---------------|--------------|
| Model              | Views | DCG           | MAP          |
| Autoencoder        | 2     | 0.649         | 0.276        |
| Autoencoder        | 3     | 0.655         | 0.288        |
| Autoencoder        | 4     | 0.655         | 0.290        |
| Classification     | 2     | 0.842         | 0.501        |
| Classification     | 3     | 0.846         | 0.527        |
| Classification     | 4     | 0.857         | 0.567        |
| Autoe.+Class.      | 2     | 0.895         | 0.663        |
| Autoe.+Class.      | 3     | 0.897         | 0.679        |
| Autoe.+Class.      | 4     | <b>0.901</b>  | <b>0.684</b> |

Table 2: 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              | Views | DCG           | MAP          | DCG           | MAP          |
| RotationNet        | -     | 0.865         | <b>0.772</b> | 0.656         | <b>0.583</b> |
| GIFT               | -     | 0.827         | <b>0.722</b> | 0.657         | 0.0575       |
| ReVGG              | -     | 0.828         | 0.749        | 0.559         | 0.496        |
| DLAN               | -     | <b>0.762</b>  | 0.663        | 0.563         | 0.477        |
| Autoe.+Class.      | 2     | 0.895         | 0.663        | 0.754         | 0.453        |
| Autoe.+Class.      | 3     | 0.897         | 0.679        | 0.765         | 0.461        |
| Autoe.+Class.      | 4     | <b>0.901</b>  | 0.684        | <b>0.768</b>  | 0.466        |

views in different objects is proportional to the number of extracted views. However, increasing the number of image views representing the 3D model could be very costly for our proposed model since our model’s autoencoder component is computationally expensive to train. Adding more layers and channels could lead to a better performance at the cost of increasing the computational load. An alternative solution is the pose normalization of the objects in the dataset.

Another important observation in our experiment results is that the two evaluation metrics, the *DCG* and the *MAP*, are more distant than in most other authors’ results. It happens because the relevant result obtained for a given query using our proposed model are accumulated at the beginning and the end of the ranked list. This result is a useful feature for a search engine that we can exploit in future works using our proposed model.

For reproducibility purposes, we made available the code <sup>2</sup> of the three proposed embedding models.

---

<sup>2</sup>Anonymized.

Table 3: 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              | Views | DCG           | MAP          | DCG           | MAP          |
| RotationNet        | -     | 0.702         | 0.606        | 0.407         | 0.327        |
| GIFT               | -     | 0.701         | 0.567        | 0.513         | 0.406        |
| ReVGG              | -     | <b>0.783</b>  | <b>0.696</b> | 0.479         | 0.418        |
| DLAN               | -     | 0.754         | 0.656        | <b>0.560</b>  | <b>0.476</b> |
| Autoe.+Class.      | 2     | 0.730         | 0.298        | 0.453         | 0.141        |
| Autoe.+Class.      | 3     | 0.743         | 0.320        | 0.467         | 0.160        |
| Autoe.+Class.      | 4     | 0.756         | 0.346        | 0.476         | 0.182        |

## 7 Conclusions

In this work, we propose three different neural network architectures for computing 3D model embeddings. The first one is based on convolutional autoencoders. The second one is based on a convolutional neural network for classification. The last one is a combination of the first two. For all of our three proposals, we use the same technique for modeling the input. This technique involves transforming a set of image views representing a 3D model into a multichannel image where each channel is an image view. We conduct several experiments using our proposals, and we can conclude that our work has three main contributions. A convolutional architecture that can handle the 3D models represented as sets of image views, an analysis of the performance of different convolutional architectures for computing 3D model embedding, and a study of how the number of image views affects the quality of the computed embedding by experimenting with different amount of image views (2, 3, 4) to represent the models.

The experiments show that the model 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 our last proposal, which combines the first two models.

The most important conclusion is that our proposed models can successfully compute embedding representations for 3D models. We obtain outstanding results for the two evaluation metrics, especially with our last model, which shows results for the *MAP* above 67 percent and *DCG* above 90 percent. These results are very competitive with the Shapenet Dataset results, especially in the normalized version of the dataset.

## References

- [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] 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.
- [3] 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.
- [4] 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.
- [5] 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.
- [6] 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.
- [7] 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.
- [8] 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.
- [9] 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.

- [10] 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.
- [11] 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.
- [12] 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.
- [13] 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.
- [14] 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.
- [15] 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.
- [16] 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.
- [17] 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.
- [18] 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.

- [19] 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.
- [20] 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.
- [21] 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.
- [22] 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.
- [23] 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.
- [24] Xin Rong. word2vec parameter learning explained. *CoRR*, abs/1411.2738, 2014.
- [25] 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.
- [26] 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.
- [27] 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.

- [28] Volodymyr Turchenko, Eric Chalmers, and Artur Luczak. A deep convolutional auto-encoder with pooling - unpooling layers in caffe. *CoRR*, abs/1701.04949, 2017.
- [29] 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.
- [30] 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.
- [31] 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.
- [32] Keneilwe Zuva and Tranos Zuva. Evaluation of information retrieval systems. *International journal of computer science & information technology*, 4(3):35, 2012.