ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DEPARTMENT OF COMPUTER SCIENCE
AND ENGINEERING

ARTIFICIAL INTELLIGENCE

MASTER THESIS

in

Machine Learning for Computer Vision

AN EXPERIMENTAL STUDY ON CONNECTING NEURAL RADIANCE FIELDS TO IMAGES AND TEXT

CANDIDATE

Luca Torzi

SUPERVISOR

Prof. Samuele Salti

CO-SUPERVISORS

Dr. Francesco Ballerini

Dr. Pierluigi Zama Ramirez

Academic year 2023-2024

Session 4th

Abstract

Being able to process 3D data is an important skill of a deep learning architecture, because it can unlock deeper understanding of the world around us, but it is a complex goal to achieve due to the fragmentation of the methods employed to store and save 3D data, that leads to the development of different techniques specialized on each of them. In the last years, the computer vision research field focused on learning 3D data implicitly, such that a neural network could learn a continuous function that describes the properties of the object of interest: this representation is called Implicit Neural Representation or Neural Field. Between them, NeRF (Neural Radiance Field) was one of the most promising methodologies for learning functions representing 3D objects and scenes. Consequently, recent work proposed methodologies to process neural network weights directly, in order to generate a compact embedding that can be used to perform deep learning tasks efficiently. On top of this, how to link this embedding space to spaces embedding images and texts has been explored. The goal of this thesis is to expand the analysis performed in the latter, investigating the use of contrastive losses during the training phase. Moreover, we study which embedding space (the NeRF one or the image/text one) is more effective to perform retrieval of NeRFs. Finally, a fine-tuning of the NeRF embedding model is accomplished to explore the behavior of the whole architecture after receiving influence from the joint image/text embeddings during training.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The world around us has three dimensions, and we sense it in this way. In the computer science field, different possible ways to depict 3D objects and scenes have been presented (voxels, polygon meshes, point clouds), but a unique way to depict 3D objects and scenes has still not emerged nonetheless.
Recently, a new promising methodology was developed, called Implicit Neural Representation (INR), with the aim of learning a function that can implicitly represent shapes or scenes using a neural network as a medium. Unlike the explicit representations, whose detail richness (i.e. the resolution) is proportional to the memory footprint, it allows for decoupling the latter and the resolution, because the continuous function can be queried at any level of detail during the explicit reconstruction of the object of interest. Therefore, this data type is also called Neural Field, since a *field* is a quantity defined for all spatial and/or temporal coordinates [1]. NeRF [2] is one of the most important works in this field, in which a multilayer perceptron is trained to learn the volume density and the directional emitted radiance at any point in the space. Recent research, such as [3] and [4], focuses on extracting information directly by processing the weights of the MLP in order to perform deep learning tasks on the learned function. The first work presents `inr2vec`, an architecture that focuses on process implicit neural representations of 3D shapes, and, as a step

forward, the successive publication focuses on NeRFs, presenting the architecture `nf2vec`. An embedding is extracted from the neural network weights, and it is used to perform a variety of downstream tasks, such as classification, retrieval and generation.

Concurrently, another line of research directed its attention toward developing Multi-Modal Models, i.e. deep learning architectures able to digest different types of input data. A key contribution to this field is found in CLIP [5], an architecture that connects images and text in a shared embedding space. The authors demonstrated that the contrastive learning training methodology, employed to align the image embedding to the one that belongs to its own caption, along with the huge amount of data used, generated a highly expressive latent embedding space that can be used to perform a variety of downstream tasks. Due to this discovery, follow-up research has tried to connect other types of input data to this embedding space, in order to leverage its properties. Between them, a framework for connecting NeRFs with it was proposed in the work of Ballerini et alii [6], where they take advantage of the `nf2vec` architecture to generate a compact embedding of the neural network weights and then link it to the CLIP embedding space developing the twin architectures `nerf2clip` and `clip2nerf`.

This thesis aims to expand the experiments performed in the last mentioned work, in particular by focusing on the use of a different loss function (the *SigLIP* loss [7]) to learn this mapping and compare its effectiveness in the zero-shot classification and retrieval task. An intense effort is put into the latter, analyzing the efficacy of performing it in the NeRF embedding space (as proposed in [6]) or in the CLIP embedding space, which could guarantee a more robust search space. Moreover, further analyses investigate this methodology on augmented datasets and on changing the batch size. In addition, in the classification task, the use of different labels is tested to increase

the accuracy of the models. Finally, after a fine-tuning of the `nf2vec` architecture through the supervision of the CLIP model, the whole framework is again analyzed to understand its behavior after receiving information from this expressive embedding space.

This thesis is organized as follows:

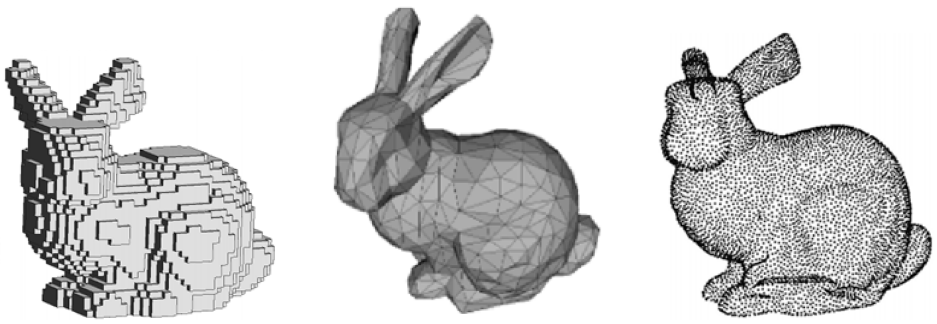
- Chapter 2 introduces all the related works, in order to give to the reader the knowledge necessary to understand this work
- Chapter 3 explains the methodology employed to perform the two tasks analyzed
- Chapter 4 lists and explains all the experiments done, along with the results obtained
- in Chapter 5 the most important outcomes are wrapped up with the concluding remarks.

Chapter 2

Related works

2.1 Explicit representations of 3D data

Finding a way to represent and save information without loss of detail and quality has always been an important goal of the computer science field; in particular, this operation is notably challenging if we take into account 3D objects. In previous years, three main methodologies to store them emerged, depicted in Figure 2.1, and they will be discussed in the following paragraphs, along with the deep learning architectures employed to process them.



(a) (b) (c)

Figure 2.1: Explicit representations of 3D data: (a) voxel representation, (b) polygon mesh, (c) point cloud. The depicted object is the Stanford bunny model [8].

Voxel-based methods

The input object processed by these networks is made of voxels (Figure 2.1a), the 3D counterpart of the pixels, used to represent 3D data. Due to the analogy with images, these data representations can be processed using convolutional neural networks (*CNNs*), a type of architecture that became famous for the processing of 2D data taking advantage of their regular structure [9, 10]. A first work that proposed to process a scene using a 3D CNN is *VoxNet* [11]: the authors first "voxelize" the space in a volumetric occupancy grid and then they apply a deep learning pipeline with the goal of performing shape classification. The downside of these architectures is that they use a large amount of memory, so in the following years researchers proposed a type of architecture (sparse convolutional neural network [12]) that was able to process only non-empty voxels.

Polygon meshes

The objects are represented by collections of vertices, edges and faces (Figure 2.1b) that define them, so different from the above-mentioned methodology, they have an irregular structure. They could be seen as a particular type of graph data [13], so different types of deployed models make use of graph neural networks (*GNNs*) to process them, an architecture that processes and connects information in the neighborhood of a vertex.

Point clouds

It is a methodology that has been deployed to represent 3D data and that has gained attention in the last decade. The object is represented by sparse points in the space (Figure 2.1c), and since it is an irregular structure, it was initially used only as an intermediate representation. Therefore, before proceeding with the processing, the point clouds were transformed in a regular form (such as a voxel grid or a collection of images). The pioneering works of Qi et

alii [14, 15] were the first to propose two architectures, named *PointNet* and *PointNet++*, that are able to perform deep learning tasks directly on the point clouds themselves. They proposed the *max pooling* as the principal aggregation operator capable of respecting the permutation invariance of the input. Successive works mainly focused their efforts on designing architectures in order to process this structure, making use of different neural primitives such as graph neural networks [16, 17] or the attention operator [18].

2.2 Implicit neural representations

The above-mentioned representation methodologies have their pros and cons, but all of them tend to represent objects with a fixed size (the number of voxels, the number of points or the number of vertices and faces); moreover, the increase of these quantities leads to the increase of the disk space needed to store the file itself and, consequently, to an increment of the memory needed while processing them with deep learning pipelines.

In the last few years, a new methodology emerged, named *implicit neural representation* (INR). The concept behind it is grounded in the universal approximation theorem stating that multilayer perceptrons can approximate any continuous function to any desired precision [19, 20]. Therefore, the objective is to parameterize a neural network such that it learns a function containing *implicitly* the information of interest, belonging to any type of complex and high-dimensional data, for example audio, images, videos and, in particular, 3D shapes. After the training process, the neural network can be queried (giving in input the coordinates) to obtain in output the desired information and reconstruct the object of interest by performing multiple queries. Moreover, since the learned function is continuous, it is possible to acquire data at varying levels of precision, overcoming the fragmentation of the discrete representations used so far, thereby guaranteeing a decoupling between the resolution of

the reconstruction and the memory space occupied, that scales with the number of parameters of the neural network (the so-called network complexity) [1].

The publication of Mescheder et alii [21] is one of the first research works that proposed to learn 3D shapes through this methodology, in which an architecture called *Occupancy Network* is defined, and it is able to learn 3D shapes leveraging the so-called occupancy function o (Equation 2.1), which defines if a point belongs or not to the object.

$$o : \mathbb{R}^3 \rightarrow \{0, 1\} \tag{2.1}$$

Simultaneously, Park at alii [22] proposed a different methodology that aims to learn a neural field leveraging the Signed Distance Function (SDF): the values of a point in the space is defined by its distance to the surface boundaries, as shown in Figure 2.2.

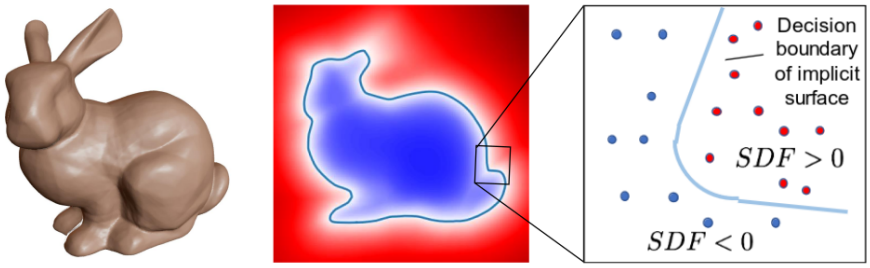


Figure 2.2: DeepSDF data representation on the Stanford bunny model [22, 1].

2.2.1 NeRF

NeRF [2] is a method that achieves great results in novel view synthesis of complex scenes: it represents a scene as the volume density and the directional emitted radiance at any point in the space. A NeRF learns an implicit neural representation by fitting an MLP using multiple views of the scene as

training data. Therefore, both the spatial (x, y, z) coordinates and the camera view directions (θ, ϕ) are used as continuous input coordinates. The model returns the volume density and the view-dependent RGB colors as output. The model is optimized such that the output of the MLP corresponds to one of the ground truth views and, to render the color of any ray passing through the scene, they use principles from classical volume rendering. Figure 2.3 shows a visual representation of the training methodology employed.

Since neural networks are biased toward learning lower frequency functions [23], the authors proposed to encode the input using a positional encoding, to enhance the performance of the model. Moreover, the use of hierarchical volume sampling during the training process allowed efficient and faster convergence.

Despite the fact that this methodology achieved state-of-the-art results, it has downsides in the training time needed to fit the MLPs. So, in successive years, researchers analyzed the feasibility of speed-up it focusing in particular on the two innovations above-mentioned.

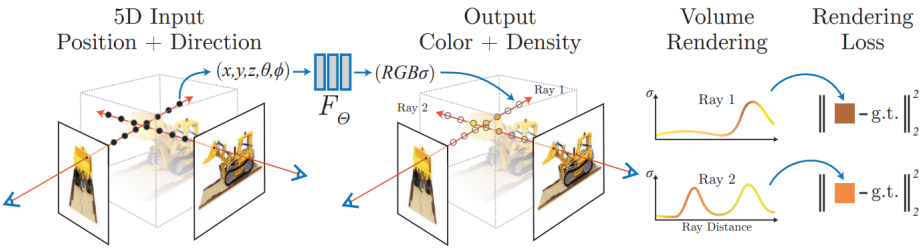


Figure 2.3: NeRF training methodology [2].

instant-ngp

This work [24] proposes to improve the training time speed of neural implicit representations by reducing the number of floating point operations needed to encode the input. In [2], the scalar positions $x \in \mathbb{R}$ are encoded as a sequence of $L \in \mathbb{N}$ sine and cosine functions, with the same formulation proposed in

[25] (Equation 2.2).

$$enc(x) = \Big(sin(2^0 x), ..., sin(2^{L-1} x), cos(2^0 x), ..., cos(2^{L-1} x)\Big) \quad (2.2)$$

Therefore the encoding is a continuous function of the input. Instead, Muller et alii deployed a multi-resolution hash encoding: the positional encoding is composed as a concatenation of the features obtained from each resolution encoding, computed as the interpolation of the features corresponding to the corners around the exact point to be encoded. The features of each corner are parameters that are learned through gradient descent optimization, but the overhead added by this modification does not enlarge the computational complexity; instead, the hash table lookup to obtain the encoding values guarantees a reduced training time. In addition, due to the structure of the hash encoding, there will be colliding gradients during training, leading to an optimization process that will prioritize sparse areas with fine scale details.

NerfAcc

To perform a fast rendering, it is important to define a sampling technique that prioritizes regions based on their importance. Since most of the works related to this field develop systems using highly optimized implementations that are tightly connected with the underline architecture, `nerfAcc` [26] tries to create a Python framework that makes them interchangeable. They unified importance sampling under the mathematical formulation of transmittance and tested various NeRF implementations using their framework and different sampling techniques.

2.3 Deep learning on INRs

In the last few years, a new branch of study emerged, whose goal is to understand if it is possible to perform directly deep learning tasks on INRs themselves. Since they encode the data using a single continuous function, performing downstream tasks on them could enhance the quality of information provided as input to the neural networks. Mostly, two different types of approach were presented, that differ on how the implicit neural representation is encoded:

- define a shared base network, that can efficiently encode the shared characteristics of a dataset, and then represent each datapoint as modulation
- fit each object independently and perform deep learning tasks directly on the multilayer perceptron weights.

Functa

Functa [27] is one of the first works attempting to carry out it on this type of data representation. The goal of the authors is to be able to fit each object in a few gradient steps and have a simple latent vector that could be used for downstream tasks. In order to do so, they defined a base network, performing a training process on the entire dataset, that can efficiently encode its shared characteristics, and afterwards fit a latent modulation that is used to condition the base network to represent each datapoint (called in this work functa). This framework was tested on different types of data, such as images, voxels, NeRFs and manifolds.

nf2vec

In the previous approach, to define an embedding representing an object, the entire dataset must be fit to create the shared base network. An alternative is

presented in [3], where the authors' goal is to define a framework able to process INRs that are fitted independently, without having to work on the entire dataset. They define an encoder-decoder architecture that is able to generate a compact latent representation of the neural network weights, containing all the information needed for the reconstruction of the object itself. With the objective of having a simple encoder (Figure 2.4) with a small parameter footprint, they stack the INR weights in a single matrix and process each row separately with a model composed of a series of linear layers, batch normalizations [28] and ReLU non-linearities; at the end, they perform a global max pooling column-wise to obtain the final embedding. Instead, the decoder is designed to take in input this latent representation and the coordinates, with the purpose of outputting the information needed to reconstruct the original object. This framework was tested on INRs representing 3D shapes and, in a successive extension [4], on NeRFs, showing that the embedding could be used for tasks such as classification, part segmentation, retrieval and generation.

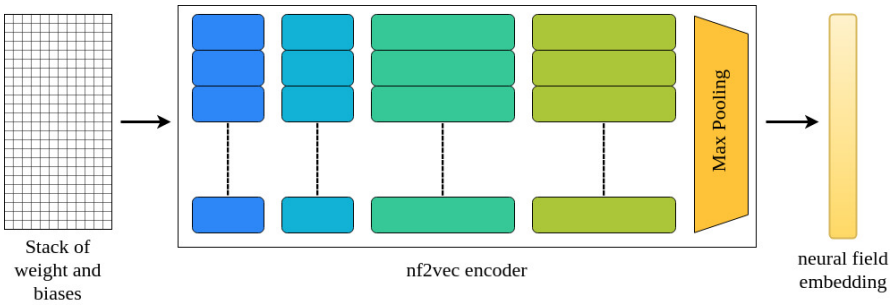


Figure 2.4: `nf2vec` encoder architcture.

2.4 Multi-Modal Models

Multi-Modal Models are neural architectures that aim to connect different input data modalities, ensuring the opportunity to obtain information from multiple sources and take advantage of a shared embedding space.

CLIP

CLIP [5] is a large scale vision-language model and represents one of the most important works in this field. The goal of the authors was to build an architecture capable of processing both images and text data, in order to direct the learning process through text-guided supervision. To do so, they trained a text encoder (composed of a stack of transformers blocks [25]) and an image encoder (Res-Net50 [29] or ViT [30]), and the objective of the training process was to align the two embeddings related to the images and their own captions, through the use of a contrastive loss function [31] (the pipeline is shown in Figure 2.5).

$$-\frac{1}{2|\mathcal{B}|}\sum_{i=1}^{|\mathcal{B}|}\left(\overbrace{\log\frac{e^{t\mathbf{x}_i\cdot\mathbf{y}_i}}{\sum_{j=1}^{|\mathcal{B}|}e^{t\mathbf{x}_i\cdot\mathbf{y}_j}}}^{\text{image}\rightarrow\text{text softmax}}+\overbrace{\log\frac{e^{t\mathbf{x}_i\cdot\mathbf{y}_i}}{\sum_{j=1}^{|\mathcal{B}|}e^{t\mathbf{x}_j\cdot\mathbf{y}_i}}}^{\text{text}\rightarrow\text{image softmax}}\right) \tag{2.3}$$

The loss objective is shown in Equation 2.3, where $\mathcal{B}$ is a mini-batch, x_i is the normalized image embedding, g_i is the normalized text embedding and t is the temperature. Radford et alii trained the CLIP model using a huge amount of data (equal to 400 million of text-image pairs), that jointly with the effective learning process, allowed this pre-trained model to learn a highly expressive latent space, enabling zero-shot capabilities in downstream tasks, in particular classification where the model showed to be robust to distribution shift.

SigLIP

SigLIP [7] is an extension to the CLIP model that proposes the use of a simple pairwise Sigmoid loss, instead of the original contrastive loss function.

$$-\frac{1}{|\mathcal{B}|}\sum_{i=1}^{|\mathcal{B}|}\sum_{j=1}^{|\mathcal{B}|}\log\underbrace{\frac{1}{1+e^{z_{ij}(-t\mathbf{x}_i\cdot\mathbf{y}_j+b)}}}_{\mathcal{L}_{ij}} \tag{2.4}$$

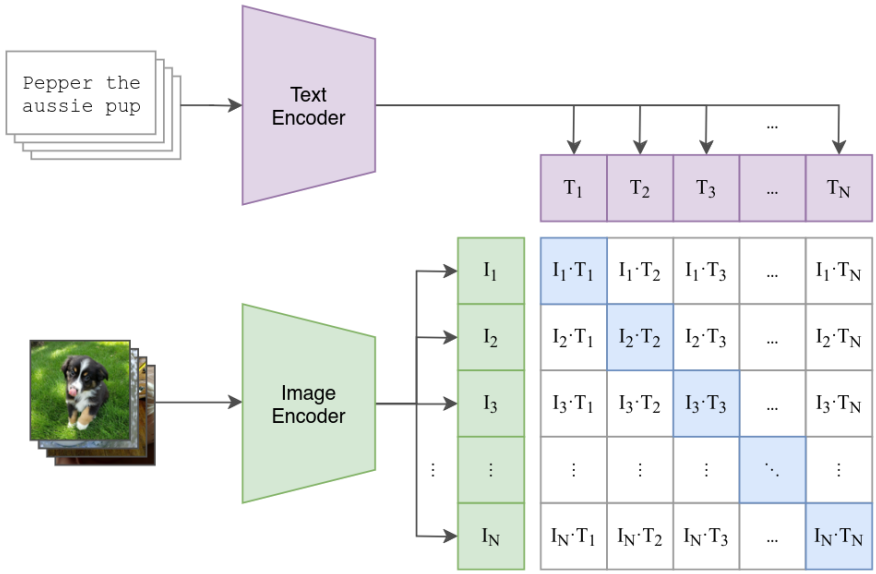


Figure 2.5: CLIP architecture and training methodology [5].

The training objective is defined in Equation 2.4, where $\mathcal{B}$, x_i, g_i and t have the same meaning of Equation 2.3, z_{ij} is the label for an input text-image couple (equals to 1 if they are paired and -1 otherwise) and b is a learnable bias term. This small yet effective modification is memory efficient and allows for better scaling up of the performance, and, in addition, it even enhances the results when the training is performed using smaller batch sizes.

2.4.1 Link other modalities to CLIP

AudioCLIP

AudioCLIP [32] is one of the first works which attempted to add another modality on top of the CLIP model. The authors pre-trained a model in order to process audible data and then they linked it to the CLIP embedding space using a contrastive learning methodology. Afterward, a fine-tuning of the whole architecture was done to slightly increase the performance. Finally, the model is tested in the classification and retrieval tasks on audio signals, achieving state-of-the-art results.

ImageBind

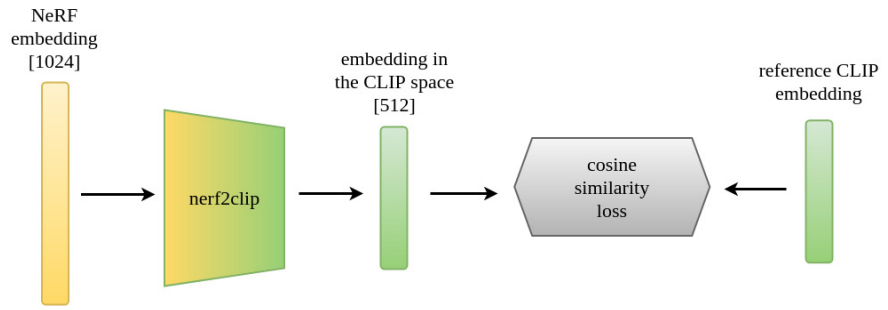
It is a work [33] that has its own root in the CLIP model, and it aims to expand the type of input that, after a processing, can lie in its embedding space. The authors aligned the embeddings of each modality (audio, depth map, thermal map, and IMU data) to the one of the images and they observed emergent behavior on tasks involving pairs on which the model is not directly trained, such as zero-shot text-audio classification.

PointCLIP

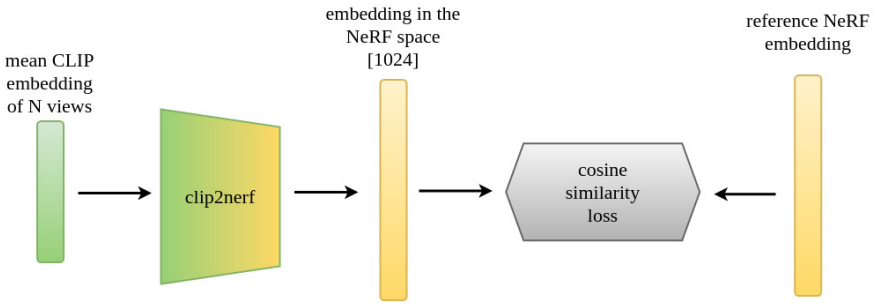
In this work [34], for the first time, the CLIP model is used to perform tasks on 3D data. Since a variety of applications relies on real-time processing of them, Zhang at alii proposed to process directly views of the point cloud, without additional pre-processing. These views are given in input to the CLIP image encoder to obtain high level features, that, jointly with hand-crafted labels processed by the CLIP text encoder, can be used to perform zero-shot classification. Since the performance of this model was far from the one obtained by supervised classifiers (such as PointNet++ [15]), the authors developed an inter-view adapter, that extracts global representations and generates view-wise features. In addition, they find out that the feature learned by their model is complementary with the one learned by supervised trained models, so they concluded that multi-knowledge ensembling can enhance the performance.

2.4.2 Connecting NeRFs to CLIP

In a recent work [6], it is proposed to link the NeRF modality to images and text, leveraging the CLIP model, in order to create a connection between them and being able to perform different types of downstream tasks, taking advantage of the pre-trained model and its highly expressive embedding space. To do so, the authors used the `nf2vec` framework in order to obtain a NeRF embedding and afterwards they created a bidirectional mapping using two twin



(a) `nerf2clip` architecture.



(b) `clip2nerf` architecture.

Figure 2.6: `nerf2clip` and `clip2nerf` architectures.

architectures composed by a multilayer perceptron (called `clip2nerf` and `nerf2clip`) to link a CLIP embedding to the NeRF one and vice versa, maximizing the cosine similarity between the two latent representations during the training phase. The `nerf2clip` architecture (Figure 2.6a) maps the NeRF embeddings to the CLIP embedding space, so it is possible to use these expressive latent features to perform the task of NeRF zero-shot classification. Instead the `clip2nerf` architecture (Figure 2.6b) learns to map the CLIP embeddings to the NeRF embedding space, therefore this model is used to perform NeRF retrieval from images or text and NeRF generation.

Chapter 3

Methodology

This thesis builds on top of the work by Ballerini at alii [6] and browses options to enhance the framework. It explores the feasibility of using the *SigLIP* loss [7] instead of the cosine similarity to train the architectures and tests the results of these models comparing them to the original ones. Furthermore, it inspects the benefits of unfreezing the `nf2vec` model, such that it could be possible to learn embeddings that are directly mapped in the clip embedding space.

3.1 Dataset

To train the architecture, a set of NeRFs is needed, which will be the input of the `nf2vec` model, along with a set of reference CLIP embeddings. The NeRFs dataset, used to train the architecture and constructed by Ballerini et alii, is build upon *ShapenetRenders* [35], which contains 36 image views of 33296 different objects. These objects come from the famous *ShapeNetCore* dataset, a subset of the larger *ShapeNet* dataset [36] (a large-scale repository of shapes represented by 3D CAD models of objects), but with single clean 3D models and manually verified categories and alignment annotations. The ground truth views images, used to fit the NeRF multilayer perceptrons, are also used to generate the reference CLIP embeddings that will be used during the training phase to align the two latent representations. Alongside them, the

latent representations are also aligned to embeddings created from the images rendered from the NeRF itself, in order to analyze the performance of the model in the scenario where the ground truth images are not available.
Using a well established training methodology, the dataset is divided into three different sets:

- the training set, employed to update the model weights
- the validation set, employed to test the performance of the network during the training process
- the test set, employed to execute the final tests on unseen data.

Moreover, to avoid the presence of data leakage, that could bring in incorrect results, these splits are equal to the one used to train and test the `nf2vec` architecture, since the latter was built using the same dataset.

3.2 SigLIP loss to connect the embeddings

In [6], the two above-mentioned frameworks (Figure 2.6) were trained with the objective of maximizing the cosine similarity, shown in Equation 3.1, where x_i and y_i are the two vectors that must be aligned (i.e. the NeRF embedding and the reference CLIP embedding).

$$\frac{1}{|\mathcal{B}|}\sum_{i=1}^{|\mathcal{B}|}\left(1-\frac{x_i\cdot y_i}{\|x_i\|\|y_i\|}\right) \tag{3.1}$$

Instead, I leverage the *SigLIP* loss (Equation 2.4) defined in [7], to try to enhance the performance of the model using a contrastive objective that allows the model to learn embeddings that align to the CLIP embedding space, maximizing the log-sigmoid values of similar pairs, and in addition minimizing the one of dissimilar pairs in order to keep them distinct. Similarly to what was done in [6], I tested the trained models in the zero-shot classification and in the retrieval tasks.

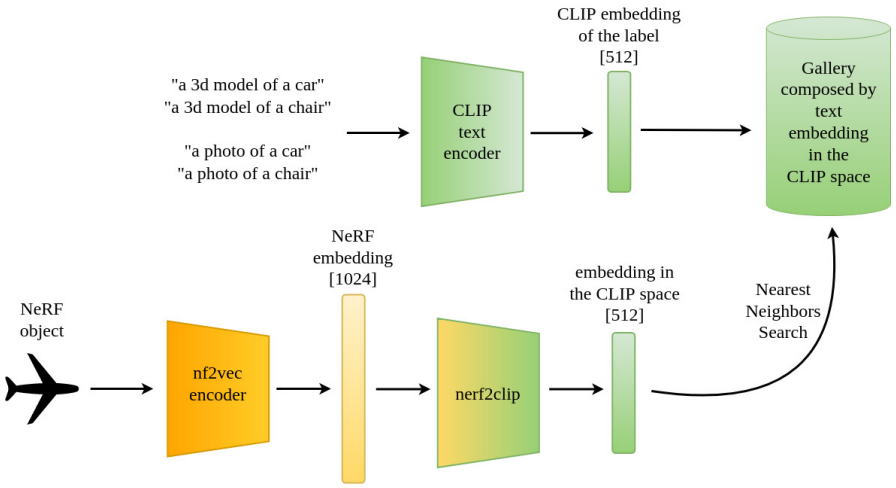


Figure 3.1: Overview of the methodology used to perform the zero-shot classification task.

3.3 Zero-shot classification task

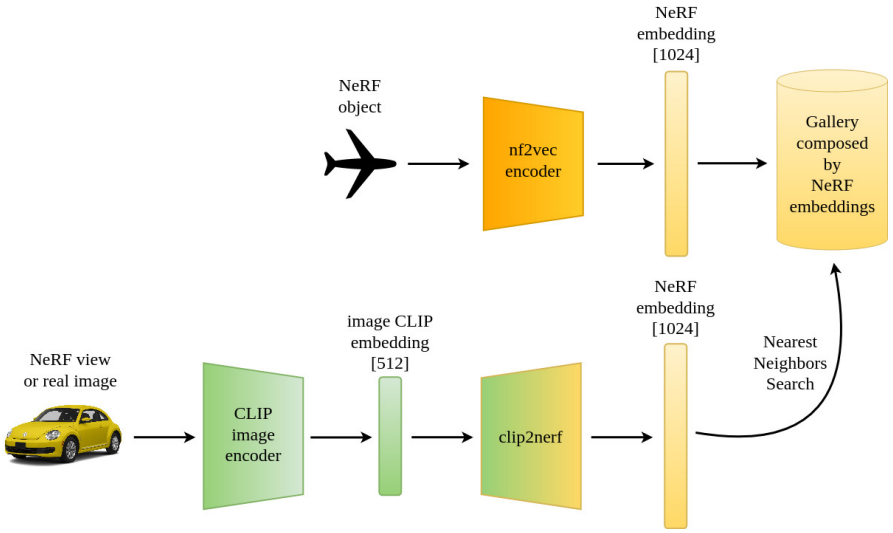
It is a task to understand the performance of the model in classifying unseen NeRF data, without being trained explicitly for the classification task, but only leveraging the features of the CLIP embedding space. To do it, I used a methodology similar to the one defined in [5] (Figure 3.1), which consists of generating the embeddings related to the text labels using the CLIP text encoder and then finding the closest label in the CLIP embedding space using a nearest neighbors search [37], querying it with a NeRF embedding projected in the CLIP embedding space utilizing the `nerf2clip` model.

3.4 Retrieval methodology

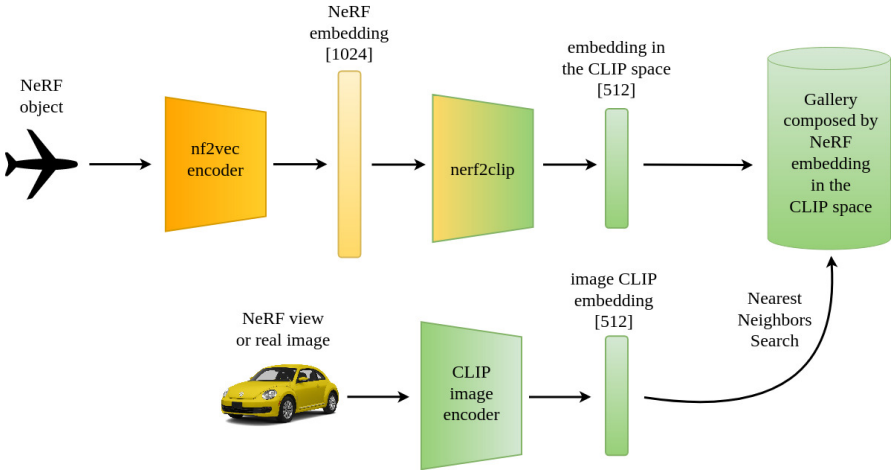
The goal is to retrieve relevant NeRF data with respect to the image query; two different types of images are used:

- synthetic images that come from ground truth images used to train NeRFs that belong to the test set
- real images from the DomainNet dataset [38].

To perform this task, a gallery of embeddings must be constructed: in [6] it is built using NeRF embeddings generated from the NeRFs in the test set. Therefore, the image query is embedded using the CLIP image encoder and then projected into the NeRF embedding space leveraging the `clip2nerf` model (Figure 3.2a). Here also the inverse modality is tested: the one in which the gallery is built using NeRF embeddings projected in the CLIP embedding space using the `nerf2clip` architecture and the query images are simply encoded with the CLIP image encoder (Figure 3.2b). This new methodology is developed in order to study the effectivity of these two embedding spaces in the retrieval task, and to analyze which one is more suited and for which category of input query.



(a) Overview of the methodology used to perform the retrieval leveraging the `clip2nerf` model. This is the methodology used in [6].



(b) Overview of the methodology used to perform the retrieval leveraging the `nerf2clip` model. This is the inverse method, analyzed here for the first time.

Figure 3.2: Overview of the methodologies used to perform the retrieval.

Chapter 4

Experiments and results

4.1 Experiments in the zero-shot classification case study

During the training of the `nerf2clip` architecture, used to perform the zero-shot classification task, every NeRF embedding, generated with the `nf2vec` encoder, is aligned to a reference CLIP embedding. The latter is generated starting from two different types of images: either the ground truth one used to train the NeRF itself or the rendering form the learned NeRF. It is computed as the mean of the N different image embeddings: the number of images used to generate it goes from 1 to 36, duplicating at every step except the last one, such that it could be possible to verify the performance curve slope changing this parameter. Finally, it is tested in the zero-shot classification task: Table 4.1 reports the results and we can see that, in this particular task, the use of the SigLIP loss function is not beneficial.

In addition, differently from the original paper, I also tested the zero-shot capabilities using a different type of label ("A photo of `object_name`", instead of "A 3d model of `object_name`"), showed in Table 4.2, because the original CLIP model was trained using this textual reference (to connect images to text), so under my hypothesis the results might be better. Moreover, I analyzed

Accuracy using "A 3d model of"			
Method	**Views**	**Cosine similarity**	**SigLIP loss**
Nerf2Clip rendered	1	81.0	79.6
	2	81.7	78.5
	4	82.6	78.4
	8	81.8	80.1
	16	**83.1**	80.3
	N	82.5	**80.4**
Nerf2Clip GT	1	80.9	77.6
	2	83.4	79.6
	4	**<u>84.7</u>**	**80.8**
	8	83.3	80.4
	16	83.9	80.7
	N	84.2	80.8

Table 4.1: Results about the zero-shot classification task using as label "A 3d model of `object_name`". The best results for each methodology and loss function are highlighted in bold, the best result overall is underlined.

the performance of the CLIP model, used as baseline in [6], to understand its performances using this new type of label, since in the work of Ballerini et alii the main label used is the one mentioned above. To obtain the results, the model is queried with N different views of the rendered NeRF and, at the end, the average embedding is computed and used to perform the classification. In Table 4.2 we can see how the results increase due to the use of the new labels; moreover, similarly in Table 4.3 the column on the right shows that the performance of the CLIP model grows, due to the motivation above-mentioned; furthermore, it obtains the highest result overall when using all the N views in input. The disadvantage of this last methodology is that, to perform the classification of a NeRF, its views must be rendered, and it is time consuming with respect to doing the classification directly on the NeRF weights.

Accuracy using "A photo of"			
Method	**Views**	**Cosine similarity**	**SigLIP loss**
Nerf2Clip rendered	1	83.1	82.3
	2	84.8	81.9
	4	83.4	81.9
	8	**84.9**	82.7
	16	83.8	**83.3**
	N	84.6	83.1
Nerf2Clip GT	1	84.7	82.1
	2	84.8	82.6
	4	84.8	83.8
	8	**85.0**	**84.0**
	16	84.4	83.3
	N	84.7	84.0

Table 4.2: Results about the zero-shot classification task using as label "A photo of `object_name`". The best results for each methodology and loss function are highlighted in bold, the best result overall is underlined.

Method	**Views**	**Accuracy using "A 3d model of"**	**Accuracy using "A photo of"**
Baseline CLIP	1	74.9	78.3
	2	78.9	82.3
	4	81.9	84.9
	8	82.9	85.7
	16	83.2	86.3
	N	**83.6**	**86.7**

Table 4.3: Results of the CLIP baseline on the zero-shot classification task. The best result for each column are highlighted in bold.

4.2 Experiments in the retrieval case study

Having in mind the retrieval methodology employed, explained in Section 3.4, now I can illustrate how the training is performed and the results obtained in the experiments. Two versions of the architectures are developed, one using the cosine similarity and the other leveraging the *SigLIP* loss, in order to analyze the differences in the results obtained. The architectures are trained to create a mapping between a NeRF embedding and the 36 CLIP embeddings of the different views obtained from the ground truth images used to train the NeRF itself, so it is a N to 1 mapping in the case of the `clip2nerf` model and vice versa in the other one. The model used as the baseline is still CLIP: while performing the retrieval with it, the gallery is built using two different methodologies, that differ in how the embeddings of a single NeRF are added to the gallery itself: the mean CLIP embedding or all the 36 image embeddings independently.
Using this task as a point of reference, two other analysis are done on the architectures. The first one is centered on understanding the performance of the models after a training with an augmented dataset, with a size that is three times larger than the original one, and the second analysis is performed changing the batch size used during the training while using the *SigLIP* loss, to be able to state how this hyper parameter can influence the results.

The outcomes about the retrieval done on synthetic images are reported in Table 4.4. Table 4.4a shows that the methodology presented in [6] is still the best one in this case, because both the use of the *SigLIP* loss and the inversion of the modality do not improve the results, except when considering the *Recall@10* metrics, but the improvement is marginal. Instead, in the case of the retrieval done on real images (Table 4.5a), oppositely to what happens in the previous experiment, these two novelties produce better results, exceeding the CLIP baseline (Table 4.5b). This outcome may be due to the fact that the CLIP

Method	Recall @1	Recall @5	Recall @10
clip2nerf cosine [6]	**87.02**	**94.60**	96.44
clip2nerf siglip	86.61	94.42	96.42
nerf2clip cosine	80.46	92.63	95.38
nerf2clip siglip	76.95	94.06	**96.81**

(a) Results of the `clip2nerf` and `nerf2clip` models.

Method	Recall @1	Recall @5	Recall @10
CLIP mean [6]	82.51	94.08	96.34
CLIP all [6]	84.74	93.17	95.64

(b) Results of the baseline models.

Table 4.4: Results about the retrieval task on synthetic images. The best results are highlighted in bold.

embedding space was generated via a training on a large amount of data, so it could be more expressive than the NeRF embedding space, especially when the retrieval is performed on real images, that are a type of element never seen in the training phase of the feature mapping networks.

To try to verify this hypothesis, I created a t-SNE plot [37, 39] of the gallery projected in the two embedding spaces. Figure 4.1a shows the plot of the NeRF embedding space obtained using the `clip2nerf` model trained with the cosine similarity, instead Figure 4.1b shows the one of the `nerf2clip` model trained with the SigLIP loss. However, the two plots are quite similar and do not show high differences, except that in the CLIP embedding space the clusters seem slightly better formed.

4.2.1 Analysis of the embedding spaces

To analyze the embedding spaces more in detail, to understand if their features are similar or complementary, I decided to perform the same experiment described in [34]. The authors that developed the PointCLIP architecture trained

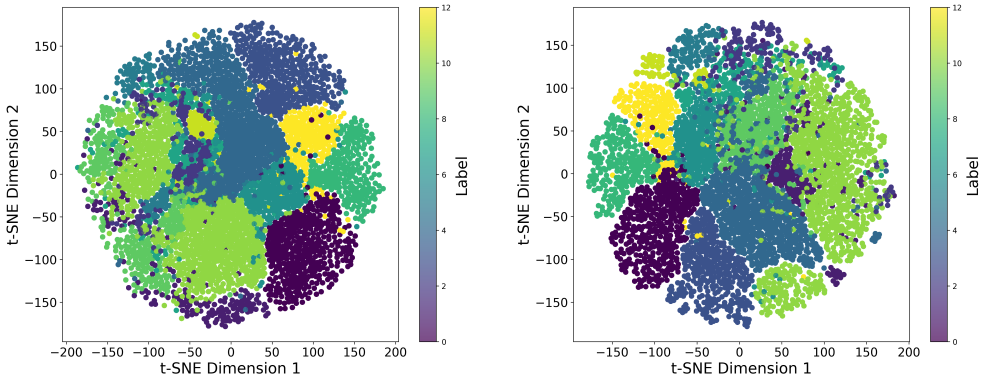
Method	Recall @1	Recall @5	Recall @10
clip2nerf cosine [6]	69.91	81.86	86.58
clip2nerf siglip	70.19	82.51	86.55
nerf2clip cosine	70.75	86.25	89.66
nerf2clip siglip	**76.49**	**90.76**	**95.02**

(a) Results of the `clip2nerf` and `nerf2clip` models.

Method	Recall @1	Recall @5	Recall @10
CLIP all [6]	73.87	89.36	93.27

(b) Result of the baseline model.

Table 4.5: Results about the retrieval task on real images. The best results are highlighted in bold.



(a) t-SNE plot of the NeRF embedding space obtained using the `clip2nerf` model trained with the cosine similarity.

(b) t-SNE plot of the CLIP embedding space obtained using the `nerf2clip` model trained with the SigLIP loss.

Figure 4.1: t-SNE plot of the retrieval gallery projected in the NeRF and in the CLIP embedding spaces.

Method	Accuracy
MLP trained on the NeRF embedding space	92.55
MLP trained on the CLIP embedding space	91.10
Ensembling	93.23

Table 4.6: Outcomes of the classification head trained on the two embedding spaces, along with the one obtained by the ensemble model.

a classifier on top of the features learned by their model, and then summed the final logits with the one belonging to a supervised trained model (PointNet++ [15]), in order to create an ensemble model. Their hypothesis (which is shown to be true) was that the features learned by a vision-language self-supervised model can be different from those learned by a model trained in a supervised way.

To analyze the NeRF and the CLIP embedding spaces, I trained two multilayer perceptrons using the original NeRF embeddings and the CLIP embeddings obtained projecting the NeRF ones, leveraging the `nerf2clip` architecture. The outcomes are reported in the first two rows of Table 4.6 and the result of the ensembling in the last one. The latter shows a higher result than the base models (and +0.68% than the best performing one), highlighting that the features of the two embedding spaces can be complementary.

4.2.2 Leveraging both the embedding spaces

From what emerged in the retrieval experiments, the NeRF embedding space is more suited to perform retrieval on synthetic images, and the CLIP embedding space can be more effective when using real images as query; therefore, I tried to combine these two different embeddings and generate vectors as a concatenation of them (living in a dimension having 1536 features). To generate the gallery, I concatenated the test set NeRF embeddings with their projection into the CLIP embedding space (done with the `nerf2clip` model trained

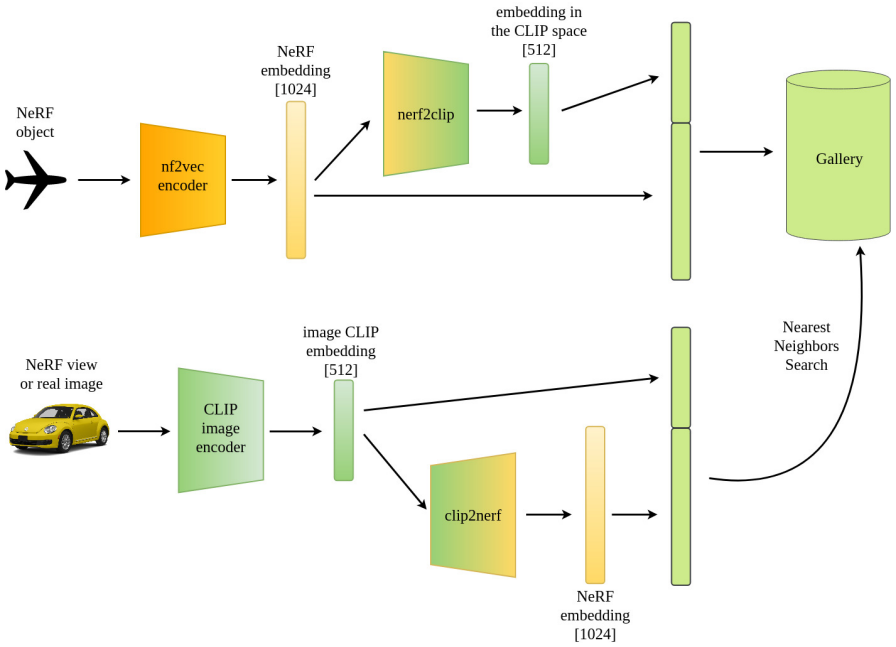


Figure 4.2: Overview of the methodology used to perform the retrieval leveraging both the embedding spaces.

Query image type	Recall @1	Recall @5	Recall @10
synthetic	77.16	94.08	96.78
real	76.49	90.86	95.05

Table 4.7: Results about the retrieval task performed concatenating the embedding spaces.

with the SigLIP loss); instead, to generate the queries, I concatenate the CLIP image embeddings with their mapping into the NeRF embedding space (done using the `clip2nerf` model trained with the cosine similarity): Figure 4.2 shows a visual picture. Although this methodology can exploit the features of both the embedding spaces, Table 4.7 shows us that the outcomes obtained are comparable to the one related to the only CLIP embedding space, indicating that a pure concatenation of the embedding spaces is not beneficial to enhance the performances.

4.2.3 Using the `syn2real` dataset

In the work of Ballerini et alii, after the testing of their models in the real image retrieval scenario, they highlighted a performance drop due to the domain-shift problem (still present here too, even if with a smaller magnitude). So, they generated a new *synthetic to real* dataset using ControlNet [40], and afterwards they created an augmented dataset containing 7 synthetic random views for each object and the same number of images generated using ControlNet. A training of the `clip2nerf` architecture on this new dataset allowed it to learn features that could be applied to real images: it is performed using all of the 14 different embeddings and the objective was to align them to the reference NeRF embedding. For the sake of completeness, I re-trained this model using the SigLIP loss in order to highlight any differences in the results, and, in addition, I analyzed the performance of the inverse retrieval modality training the `nerf2clip` architecture using the same dataset.
Table 4.8 presents the results, where we can notice that the use of the `nerf2clip` architecture and the inverse modality during the retrieval are beneficial, leading to improved results while using both the cosine similarity and the SigLIP loss. With respect to Table 4.5, here the results are enhanced due to the use of a larger dataset that can cover examples of real images, that allows the model to learn a more precise feature mapping.

4.2.4 Using an augmented dataset

To try to improve the performances of the models trained with only synthetic data, I tried to leverage the augmented dataset used to train the `nf2vec` architecture. It is made up of three times the original size and it is built by performing data augmentation on the objects from ShapeNetRenders [35]. I trained the best performing architectures so far, i.e. the `clip2nerf` model with the cosine similarity and the `nerf2clip` model with the SigLIP loss, and

Method	Recall @1	Recall @5	Recall @10
syn2real cosine (clip2nerf architecture) [6]	78.92	85.84	88.41
syn2real siglip (clip2nerf architecture)	78.68	85.95	89.05
syn2real cosine (nerf2clip architecture)	**80.42**	89.77	92.62
syn2real siglip (nerf2clip architecture)	79.82	**90.39**	**93.10**

Table 4.8: Results about the retrieval task on real images for the *syn2real* models. The best results are highlighted in bold.

Tables 4.9a and 4.9b show the results about the retrieval performed on synthetic and real images respectively. Here we can see that the performances of the models do not increase, but instead slightly decrease overall (with respect to Tables 4.4a and 4.5a). A possible motivation for why this happens is that the two feature spaces are already defined by the `nf2vec` and the CLIP models, indeed the feature mapping networks are only creating a link between them. Therefore, a data augmentation technique can only strengthen and make this link more robust, but the final outcomes are influenced by the features that belong to the embedding space used during the retrieval. Thus, in this case, different from what happens in Section 4.2.3, the use of this larger dataset to train the architectures could have added information that may be redundant or create noise.

4.2.5 Analysis of the batch size increment

In the work that presented the *SigLIP* loss [7], the authors performed an analysis of the performance of their model by changing the batch size used for training. Thanks to the high computational power that the authors had at their disposal, the batch sizes tested ranged from 16 thousands elements to 240 thousands. Zhai at alii performed two tasks (zero-shot image classification

Method	Recall @1	Recall @5	Recall @10
clip2nerf cosine	86.95	94.81	96.50
nerf2clip siglip	79.65	93.30	95.80

(a) Results about the retrieval performed on synthetic images.

Method	Recall @1	Recall @5	Recall @10
clip2nerf cosine	62.23	75.89	80.61
nerf2clip siglip	73.32	86.95	91.55

(b) Results about the retrieval performed on real images.

Table 4.9: Results about the retrieval task of the model trained with the augmented dataset.

and text-to-image retrieval) and they observed that the best perfomances are obtained while using a batch size of 32000 elements. They also highlighted that the use of the *SigLIP* loss can scale better with the batch size increment and that it performed well, if compared to the standard contrastive learning losses that use the *softmax*, even when this hyper-parameter has a smaller magnitude. Due to the limits of the machine that I used, I reproduce this analysis with smaller numbers. The results are reported in Table 4.10, where it is possible to notice that a particular trend does not emerge. In the retrieval done on synthetic images (Table 4.10a) the outcomes are all more or less similar, indicating that a change in the batch size does not influence it. Instead, in the retrieval on real images (Table 4.10b) the outcomes are more subject to a change when this hyper-parameter is modified, but there is no evident pattern, since the best result on the *Recall@1* metric is reached with a batch size of 128, but the best ones on the *Recall@5* and *Recall@10* are obtained with 512. In general, the best results are achieved while using a batch size of 128, which guarantees the best *Recall@1* (that is the most important benchmark), but also high results on the other two metrics.

Method	Recall @1	Recall @5	Recall @10
nerf2clip siglip 64	79.65	93.30	95.80
nerf2clip siglip 128	79.29	**94.03**	96.42
nerf2clip siglip 256	78.56	93.75	96.55
nerf2clip siglip 512	78.25	93.38	96.39
nerf2clip siglip 1024	78.33	93.72	**96.78**
nerf2clip siglip 2048	**79.73**	93.93	**96.78**

(a) Results about the retrieval performed on synthetic images.

Method	Recall @1	Recall @5	Recall @10
nerf2clip siglip 64	73.32	86.95	91.55
nerf2clip siglip 128	**77.34**	89.98	93.77
nerf2clip siglip 256	70.75	87.15	92.57
nerf2clip siglip 512	75.19	**90.67**	**94.73**
nerf2clip siglip 1024	76.72	88.36	92.60
nerf2clip siglip 2048	72.28	89.40	94.14

(b) Results about the retrieval performed on real images.

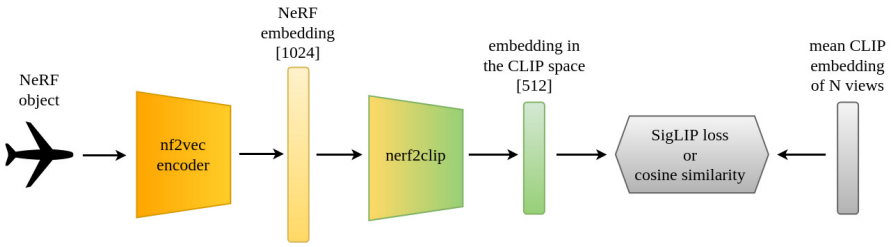
Table 4.10: Results obtained incrementing the batch size during the training with the *SigLIP* loss. The best results are highlighted in bold.

4.3 Fine-tuning nf2vec

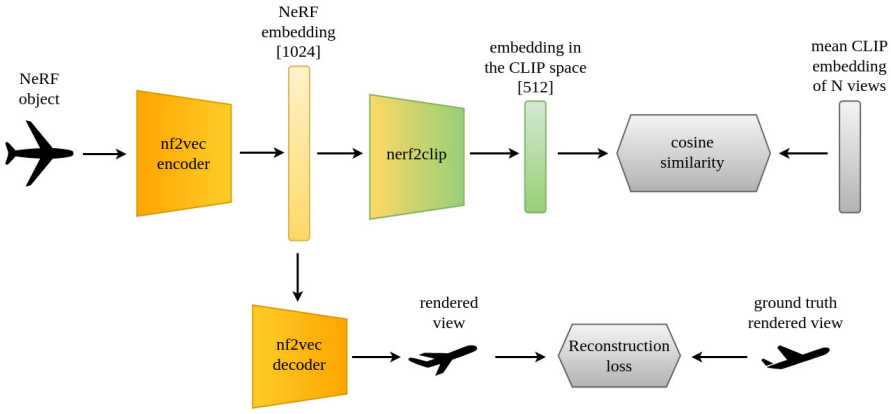
A second part of the study explores the feasibility of fine-tuning directly the `nf2vec` architecture in order to generate embeddings that align to the clip embedding space. In order to preserve the integrity of the model and to have results that can be comparable, the training process is done on an architecture composed by a stacking of the `nf2vec` architecture and the `nerf2clip` multi-layer perceptron.

A first experiment, whose setup is shown in Figure 4.3a, investigates the fine-tuning of the encoder only, using as training objective the one defined by the two types of losses analyzed above. Secondly, a further examination is

done leveraging the entire encoder-decoder architecture, as described in Figure 4.3b. In this case, the architecture will output two different vectors: one having a dimensionality of 1024 (related to the NeRF embedding space) and the other of 512 (to match the number of features belonging to the CLIP embedding space). These vectors will be used respectively by the cosine similarity and the reconstruction loss in order to update the weights of the architecture. The addition of this second loss function is to maintain the original purpose of the architecture, i.e. creating a compact embedding containing all the information needed to reconstruct the input object, and to avoid that the impact of the loss used to bring the latent representations closer could destroy this property. The fine-tuning done with the complete `nf2vec` architecture, is executed in two different modalities: with the decoder unfrozen and vice



(a) Architectures corresponding to the model trained with only the *SigLIP* loss or the cosine similarity.



(b) Architectures corresponding to the model trained with the addition of the reconstruction loss.

Figure 4.3: Overview of the fine-tuned architectures.

versa. The reference embedding is the mean of the CLIP embeddings obtained from the 36 images used to train the NeRF itself.
Finally, the trained models are tested both in the zero-shot classification task and in the retrieval task, and Table 4.11 reports the results. In the latter, the methodology applied is comparable to the one described in Figure 3.2b, since the trained feature mapping network is `nerf2clip`. We can notice how the fine-tuning of the `nf2vec` architecture does not enhance the performance in the first two downstream tasks analyzed (Tables 4.11a and 4.11b), but, similar to what was highlighted before in Section 4.2, there is increased accuracy in the retrieval with real images (Table 4.11c). However, if we consider the information reported in Table 4.5a, and in particular the fourth row, it is possible to conclude that the majority of the gain is due to the inversion of the retrieval methodology, and not to the fine-tuning of the `nf2vec` architecture, which overall does not enhance the performance of the model.

4.3.1 Analysis of the new NeRF embedding space

During the original training of the `nf2vec` architecture [4], the loss function used in the training phase was the reconstruction loss, in order to maximize the amount of information related to the object itself that can be used to minimize the error during the reconstruction while using the decoder. After fine-tuning the architecture, the supervision of the CLIP embeddings could give to the NeRF embedding space an increased amount of features that can be exploited to reorganize the NeRF embedding space. Therefore, I performed an analysis of this embedding space to perform the retrieval for a second time. The employed methodology is the one described in Figure 3.2a.
Table 4.12 shows the results. In the case of the retrieval done on synthetic images (Table 4.12a), the results are similar to those obtained by the original architecture; the same happens when the query image belongs to the "real" set, with a small increase in the accuracy when using the embedding space

Method	Accuracy using "A 3d model of"	Accuracy using "A photo of "
nerf2clip [6]	**84.7**	**85.0**
SigLIP loss	82.5	83.8
Cosine similarity	82.3	82.8
Cosine similarity + reconstruction loss (decoder unfrozen)	82.5	84.3
Cosine similarity + reconstruction loss (decoder frozen)	82.9	84.5

(a) Results about the zero-shot classification task.

Method	Recall @1	Recall @5	Recall @10
clip2nerf [6]	**87.02**	**94.60**	96.44
SigLIP loss	81.05	93.23	95.87
Cosine similarity	80.82	92.91	95.90
Cosine similarity + reconstruction loss (decoder unfrozen)	81.68	93.90	96.39
Cosine similarity + reconstruction loss (decoder frozen)	81.52	94.06	**96.50**

(b) Results about the retrieval task on synthetic images.

Method	Recall @1	Recall @5	Recall @10
clip2nerf [6]	69.91	81.86	86.58
SigLIP loss	68.52	83.20	89.10
Cosine similarity	**73.62**	87.50	91.18
Cosine similarity + reconstruction loss (decoder unfrozen)	69.34	86.18	91.53
Cosine similarity + reconstruction loss (decoder frozen)	72.58	**88.33**	**93.98**

(c) Results of the retrieval task on real images.

Table 4.11: Results of the fine-tuned models compared to the best one obtained with the original `clip2nerf` architecture. The best outcomes are highlighted in bold.

Method	Recall @1	Recall @5	Recall @10
clip2nerf [6]	87.02	**94.60**	**96.44**
SigLIP loss	86.17	94.16	96.31
Cosine similarity	86.04	94.26	96.18
Cosine similarity + reconstruction loss (decoder unfrozen)	86.84	94.55	96.42
Cosine similarity + reconstruction loss (decoder frozen)	**87.05**	94.50	96.06

(a) Results about the retrieval task on synthetic images.

Method	Recall @1	Recall @5	Recall @10
clip2nerf [6]	69.91	81.86	86.58
SigLIP loss	68.94	82.25	86.85
Cosine similarity	68.57	82.62	**87.06**
Cosine similarity + reconstruction loss (decoder unfrozen)	70.21	82.22	86.56
Cosine similarity + reconstruction loss (decoder frozen)	**70.22**	**82.94**	86.78

(b) Results about the retrieval task on real images.

Table 4.12: Results of the `clip2nerf` model when trained on the fine-tuned NeRF embedding space, compared to the best one obtained with the original architecture. The best outcomes are highlighted in bold.

obtained from the training with the reconstruction loss and the decoder frozen, but the performance is still inferior to those previously reported in Table 4.5. Even when there is no restriction to the supervision of the CLIP model from the reconstruction loss (second and third rows of the tables), the NeRF embedding space seems to not be able to receive the features needed to amplify its information richness, that is the key to increase the performance in this task accordingly.

Chapter 5

Conclusions

In this thesis I analyzed the use of the *SigLIP* loss to train the `nerf2clip` and `clip2nerf` architectures and I tested them in the zero-shot classification and retrieval of NeRF data, comparing the results with the models trained with the cosine similarity and analyzing the best configurations.
This research obtained two main achievements:

- one in the classification task, where the introduction of the new prompts, used to classify in the CLIP space, can unlock an increase in the accuracy of both `nerf2clip` and the CLIP baseline
- the other in the retrieval task, where the analyses performed highlighted how the robustness of the CLIP embedding space can be leveraged to strengthen the results when the query image belongs to a distribution different from the one used for training (i.e. real images).

In particular, it is possible to obtain benefits from the use of the *SigLIP* loss in this last mentioned case, instead in other case studies the differences with the cosine similarity are marginal or bring lower results.

This framework leverages the CLIP embedding space to perform the two above-mentioned tasks; therefore, the main limitation is that the performance is bounded by the expressiveness of its features. In order to be able to increase the results,

future works should consider the option of fine-tuning the CLIP architecture together with the `nf2vec` framework. This operation is computationally costly and should be done carefully in order to not destroy the robustness of the CLIP embedding space; nevertheless, it can allow this space to learn more specialized features. Another limitation of the framework was already cited in the work of Ballerini et alii., where they stated that the `nf2vec` model limits the processing to synthetic data, so future works can explore training this system on a larger dataset containing NeRF data from different sources.

In conclusion, this thesis was able to improve the outcomes obtained in the previous work, in both the tasks analyzed, marking a step forward in bridging the gap between NeRF representations and CLIP embeddings, contributing to the advancement of neural scene understanding and retrieval.

Bibliography

[1] Y. Xie, T. Takikawa, S. Saito, O. Litany, S. Yan, N. Khan, F. Tombari, J. Tompkin, V. Sitzmann, and S. Sridhar. Neural fields in visual computing and beyond, 2022. arXiv: 2111.11426 [cs.CV]. URL: https://arxiv.org/abs/2111.11426.

[2] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

[3] L. D. Luigi, A. Cardace, R. Spezialetti, P. Z. Ramirez, S. Salti, and L. D. Stefano. Deep learning on implicit neural representations of shapes, 2023. arXiv: 2302.05438 [cs.CV]. URL: https://arxiv.org/abs/2302.05438.

[4] P. Z. Ramirez, L. De Luigi, D. Sirocchi, A. Cardace, R. Spezialetti, F. Ballerini, S. Salti, and L. D. Stefano. Deep learning on object-centric 3d neural fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*:1–17, 2024. DOI: 10.1109/TPAMI.2024.3430101.

[5] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[6] F. Ballerini, P. Z. Ramirez, R. Mirabella, S. Salti, and L. Di Stefano. Connecting nerfs, images, and text. In *2024 IEEE/CVF Conference*

on Computer Vision and Pattern Recognition Workshops (CVPRW), pages 866–876, 2024. DOI: 10.1109/CVPRW63382.2024.00092.

[7] X. Zhai, B. Mustafa, A. Kolesnikov, and L. Beyer. Sigmoid loss for language image pre-training, 2023. arXiv: 2303.15343 [cs.CV]. URL: https://arxiv.org/abs/2303.15343.

[8] L. Hoang, S.-H. Lee, O.-H. Kwon, and K.-R. Kwon. A deep learning method for 3d object classification using the wave kernel signature and a center point of the 3d-triangle mesh. *Electronics*, 8(10), 2019. ISSN: 2079-9292. DOI: 10.3390/electronics8101196. URL: https://www.mdpi.com/2079-9292/8/10/1196.

[9] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. DOI: 10.1109/5.726791.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[11] D. Maturana and S. Scherer. Voxnet: a 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928, 2015. DOI: 10.1109/IROS.2015.7353481.

[12] B. Graham, M. Engelcke, and L. Van Der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9224–9232, 2018.

[13] H. Wang and J. Zhang. A survey of deep learning-based mesh processing. *Communications in Mathematics and Statistics*, 10(1):163–194, 2022.

[14] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: deep learning on point sets for 3d classification and segmentation, 2017. arXiv: 1612.00593 [cs.CV]. URL: https://arxiv.org/abs/1612.00593.

[15] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: deep hierarchical feature learning on point sets in a metric space, 2017. arXiv: 1706.02413 [cs.CV]. URL: https://arxiv.org/abs/1706.02413.

[16] G. Li, M. Müller, A. Thabet, and B. Ghanem. Deepgcns: can gcns go as deep as cnns? In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9266–9275, 2019. DOI: 10.1109/ICCV.2019.00936.

[17] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon. Dynamic graph cnn for learning on point clouds. *ACM Trans. Graph.*, 38(5), October 2019. ISSN: 0730-0301. DOI: 10.1145/3326362. URL: https://doi.org/10.1145/3326362.

[18] H. Zhao, L. Jiang, J. Jia, P. Torr, and V. Koltun. Point transformer. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 16239–16248, 2021. DOI: 10.1109/ICCV48922.2021.01595.

[19] T. Kim and T. Adalı. Approximation by fully complex multilayer perceptrons. *Neural Computation*, 15(7):1641–1666, 2003. DOI: 10.1162/089976603321891846.

[20] M. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015. Chapter 4. URL: https://books.google.it/books?id=STDBswEACAAJ.

[21] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger. Occupancy networks: learning 3d reconstruction in function space, 2019. arXiv: 1812.03828 [cs.CV]. URL: https://arxiv.org/abs/1812.03828.

[22] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove. Deepsdf: learning continuous signed distance functions for shape representation, 2019. arXiv: 1901.05103 [cs.CV]. URL: https://arxiv.org/abs/1901.05103.

[23] N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. Hamprecht, Y. Bengio, and A. Courville. On the spectral bias of neural networks. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5301–5310. PMLR, September 2019. URL: https://proceedings.mlr.press/v97/rahaman19a.html.

[24] T. Müller, A. Evans, C. Schied, and A. Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM transactions on graphics (TOG)*, 41(4):1–15, 2022.

[25] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need, 2023. arXiv: 1706.03762 [cs.CL]. URL: https://arxiv.org/abs/1706.03762.

[26] R. Li, H. Gao, M. Tancik, and A. Kanazawa. Nerfacc: efficient sampling accelerates nerfs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 18537–18546, 2023.

[27] E. Dupont, H. Kim, S. Eslami, D. Rezende, and D. Rosenbaum. From data to functa: your data point is a function and you can treat it like one. *arXiv preprint arXiv:2201.12204*, 2022.

[28] S. Ioffe and C. Szegedy. Batch normalization: accelerating deep network training by reducing internal covariate shift, 2015. arXiv: 1502.03167 [cs.LG]. URL: https://arxiv.org/abs/1502.03167.

[29] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. DOI: 10.1109/CVPR.2016.90.

[30] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: transformers for image recognition at scale, 2021. arXiv: 2010.11929 [cs.CV]. URL: https://arxiv.org/abs/2010.11929.

[31] A. van den Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding. *ArXiv*, abs/1807.03748, 2018. URL: https://api.semanticscholar.org/CorpusID:49670925.

[32] A. Guzhov, F. Raue, J. Hees, and A. Dengel. Audioclip: extending clip to image, text and audio. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 976–980. IEEE, 2022.

[33] R. Girdhar, A. El-Nouby, Z. Liu, M. Singh, K. V. Alwala, A. Joulin, and I. Misra. Imagebind: one embedding space to bind them all. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15180–15190, 2023.

[34] R. Zhang, Z. Guo, W. Zhang, K. Li, X. Miao, B. Cui, Y. Qiao, P. Gao, and H. Li. Pointclip: point cloud understanding by clip. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8552–8562, 2022.

[35] Q. Xu, W. Wang, D. Ceylan, R. Mech, and U. Neumann. Disn: deep implicit surface network for high-quality single-view 3d reconstruction. *Advances in neural information processing systems*, 32, 2019.

[36] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, et al. Shapenet: an information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[37] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[38] X. Peng, Q. Bai, X. Xia, Z. Huang, K. Saenko, and B. Wang. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1406–1415, 2019.

[39] L. Van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[40] L. Zhang, A. Rao, and M. Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3836–3847, 2023.

Acknowledgements

I am very grateful to my supervisor, Professor Samuele Salti, for the opportunity of working on this thesis, which allowed me to expand my knowledge about bleeding-edge topics. I am thankful to Doctor Francesco Ballerini, who has always been available to answer to my questions, and to Doctor Pierliugi Zama Ramirez.

I would like to thank all the friends that I met during these long five years, people with whom I have the opportunity to share passions and with which I spent a lot of funny moments that I will never forget.

I want to thank with all my heart my family for the support they gave me during all the years spent on the scholastic path.