Projet de Master - 2019 - Section de Microtechnique

Spherical Convolutional Neural Networks

Empirical analysis of SCNNs

Frédérick Matthieu Gusset

Professor	Pierre Vandergheynst
Supervisors	Michaël Defferrard Nathanaël Perraudin

LTS2

June 21, 2019

Master Thesis — Task Description

Spherical Convolutional Neural Networks

In the last 5 years, the field of Machine Learning has been revolutionized by the success of Deep Learning. Thanks to the increasing availability of data and computations, we now are able to train very complex and deep models to solve challenging tasks better than we ever did.

Nevertheless, Deep Learning is successful when the network architecture exploits properties of the data, allowing efficient and principled learning. For example, convolutional neural networks (CNNs) revolutionized computer vision because the network architecture has been specifically designed to deal with images. The main characteristic of CNNs is to be equivariant to translation: if the input is translated, so is the output. The translation equivariance property is extremely valuable in dense tasks such as segmentation. For global tasks such as object recognition, translation invariance is sought: a translation of the input image should not result in a change of class. This property of images and the adapted CNN architecture enables the spatial sharing of weights that dramatically reduces the number of parameters to be learned. By exploiting translation equivariance, CNNs exhibit lower computational and learning complexities on data that satisfies this property.

Beyond images, we need architectures adapted to other kinds of data, encoding both domain specific knowledge and data specific characteristics. For instance, spherical data is very common in (i) climate science with data on the Earth, (ii) in cosmology, where most observations are made from the Earth (see Figure 1), and (iii) in virtual reality, where one often works with user-centered 360° images. Spherical data is represented by pixels that live on the sphere. They are like curved images, but without borders and a potentially arbitrary orientation. Similarly to images, we'd like our architectures to exploit properties of the spherical domain. Instead of translation, the spherical domain naturally suggests equivariance to the rotation group SO(3): a rotation of the input implies the same rotation of the output.

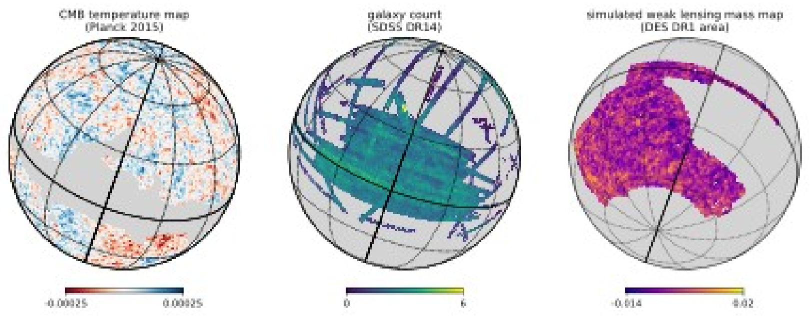

Figure 1: Example maps on the sphere: (left) the cosmic microwave background (CMB) temperature map from Plank, (middle) map of galaxy number counts, and (right) simulated weak lensing convergence map.

So far, two approaches have been followed. In the first, the data is transformed using a planar projection and a modified CNN is applied (see for example [1]). This strategy has the advantage to be built on top of a traditional CNN and hence to be efficient. Nevertheless, the distortions induced by the projection make the translation equivariance property of CNNs different from the desired rotation equivariance. In simple words, we are destroying the spherical structure of the data. The second approach [2, 3] leverages the convolution on the

SO(3) rotation group. This convolution is a generalization of the planar convolution for the sphere and similarly it can be performed by a multiplication in the spectral/Fourier domain. In this case, rotation equivariance is naturally obtained. However, the computational cost of the spectral projections (Fourier transforms) is important, limiting the size and the depth of these architectures.

In this project, you will work with an architecture that is almost rotation equivariant while remaining computationally inexpensive. [4, 5] The idea is to perform the convolution on a graph that approximates the sphere. The graph is a discrete model of the continuous 2D manifold. Similar to the traditional convolution, the graph convolution can be performed with a weighted average of neighboring pixels. Thanks to this property, we avoid computing Fourier transforms and obtain an operation with complexity linear in the data size.

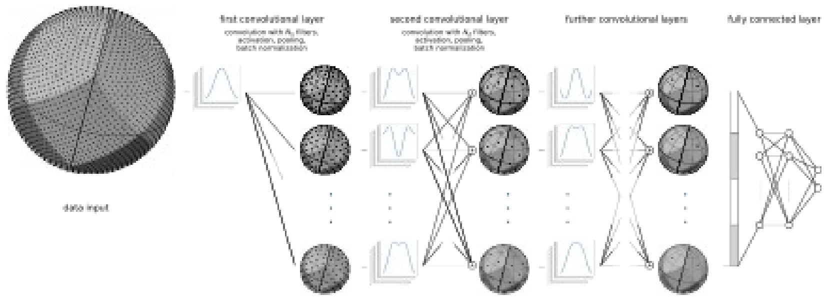
first convolutional layer
second convolutional layer
further convolutional layers
fully connected layer
data input

Figure 2: DeepSphere [5] overall architecture, showing here two convolutional layers acting as feature extractors followed by a fully connected layer with softmax acting as the classifier.

Project Goal. We aim here at making an extensive theoretical and empirical characterization of this new technique and benchmark it against the other spherical CNN architectures. In particular, the student will (i) collect and build some meaningful datasets, (ii) deal with the sphere pixelization as different architectures require different samplings, (iii) produce a reproducible research pipeline. This master thesis could potentially lead to a publication in a Machine Learning conference.

Prerequisites. Good knowledge of (Deep) Machine Learning and Python programming.

Contact

Nathanaël Perraudin nathanael.perraudin@sdsc.ethz.ch, Swiss Data Science Center, ETHZ
Michaël Defferrard michael.defferrard@epfl.ch, LTS2, EPFL

At their convenience, the student can work at EPFL or ETHZ.

References

[1] Boomsma, W., & Frellsen, J. (2017). Spherical convolutions and their application in molecular modelling.
[2] Cohen, T. S., Geiger, M., Köhler, J., & Welling, M. (2018). Spherical CNNs.
[3] Esteves, C., Allen-Blanchette, C., Makadia, A., & Daniilidis K. (2017). Learning SO(3) equivariant representations with spherical cnns.
[4] Khasanova, R., & Frossard, P. (2017). Graph-based classification of omnidirectional images.
[5] Perraudin, N., Defferrard, M., Kacprzakc, T., & Sgier, R. (2018). DeepSphere: Efficient spherical Convolutional Neural Network with HEALPix sampling for cosmological applications.

Abstract

Convolutional neural networks (CNNs) are powerful tools in Deep Learning mainly due to their ability to exploit the translational symmetry present in images, as they are equivariant to translations. Other datasets present different types of symmetries (e.g. rotations), or lie on the sphere S^2 (e.g. cosmological maps, omni-directional images, 3D models, ...). It is therefore of interest to design architectures that exploit the structure of the data and are equivariant to the 3D rotation group SO(3). Different architectures were designed to exploit these symmetries, such as 2D convolutions on planar projections, convolutions on the SO(3) group, or convolutions on graphs. The DeepSphere model approximates the sphere with a graph and performs graph convolutions.

In this study, DeepSphere is evaluated against other spherical CNNs on different tasks. While the SO(3) convolution is equivariant to all rotations in SO(3), the graph convolution is only equivariant to the rotations in S^2 and invariant to the third rotation. Our experiments on SHREC-17 (a 3D shape retrieval task) show that DeepSphere achieves the same performance while being 40 times faster to train than Cohen et al. [1] and 4 times faster than Esteves et al. [2]. Equivariance to the third rotation is an unnecessary price to pay.

In order to prove these results, DeepSphere was tested on the similar dataset ModelNet40 (a shape classification task), and similar results as obtained by Esteves et al. were achieved. The odd behaviour with rotations (the performance worsens in presence of rotation perturbations) may be inherent to the task and the classes, instead of the models or the choice of the sampling scheme.

Finally, regression tasks (both global and dense) were performed on GHCN-daily to prove the flexibility of DeepSphere with a non-hierarchical and irregular sampling of the sphere. The sCNN performed better than simply learning on the time series for each nodes.

Code and experiments are available at `https://github.com/Droxef/PDMdeepsphere`.

Contents

1 Introduction **1**
1.1 Motivation 1
1.2 Problem identification 1
1.3 DeepSphere 1
1.4 Related works 3
1.4.1 Sphere 3
1.5 Invariance and equivariance 6
1.5.1 Spherical CNNs 7
2 Methodology **11**
2.1 Different tasks 11
3 Benchmark **14**
3.1 Choice of tasks 14
3.1.1 Shape retrieval - SHREC17 14
3.1.2 Shape classification - ModelNet40 15
3.1.3 Dense regression - GHCN-daily 16
3.1.4 Climate event detection - ExtremeWeather 17
3.1.5 Climate pattern segmentation 17
3.2 Previous work 17
3.2.1 Cosmological maps classification 17
3.2.2 Comparison 18
3.3 Shape retrieval task - SHREC17 18
3.3.1 Task presentation 18
3.3.2 Other spherical models 18
3.3.3 Implementation 20
3.3.4 Analysis 23
3.3.5 Further improvements 26
3.4 In-depth investigations 27
3.4.1 Features influence 27
3.4.2 Sampling Density 28
3.4.3 Rotation Equivariance 28
3.4.4 Use of more equivariant graph 29
3.4.5 Preventing overfit 29
3.5 Classification task - ModelNet40 30
3.5.1 Task presentation 30
3.5.2 Other spherical models 31
3.5.3 Analysis 31
3.6 Dense regression - GHCN-daily 33
3.6.1 Dataset presentation 33
3.6.2 Implementation 36
3.6.3 Analysis 37
3.6.4 Further improvement 44
3.7 Climate event detection - ExtremeWeather 44

3.7.1 Task presentation 44
3.7.2 Implementation 44
3.8 Climate pattern segmentation 45
3.8.1 Task presentation 45
4 Conclusion **46**
4.1 Synthesis 46
4.2 Future Work 47
Bibliography **51**
A Miscellaneous **53**
B Pooling non-uniform graph **55**

1. Introduction

1.1 Motivation

The field of Machine Learning was changed radically with the apparition of Convolutional Neural Networks [3]. Their popularity comes from their ability to exploit the translational symmetry present in images. Indeed, the CNN is equivariant to translations due to the weight sharing of the kernel of the filters, meaning that all pixel will go through the same convolution and an element will react identically wherever it is in the image. Then the model does need to learn a single kernel for all the pixels instead of one for each. Furthermore, this invariance does not need to be learned, thus reducing the learning time and increasing the performance.

Symmetries are often present in data, but practically never exploited. Taking the CNN as an inspiration, other symmetries could be exploited, and new deep learning architectures and operations could be designed to be invariant or equivariant to them.

1.2 Problem identification

In 3D data, and more specifically for spherical signals on S^2, the symmetries of interest are rotations. An idea was to leverage the classical CNN to rotational symmetries on spherical signals.

The DeepSphere model [4] is one of the spherical CNNs created for this purpose. In order to reach an efficient spherical convolution equivariant to rotations and peak performance, hypotheses were made.

The goal of this thesis is to compare DeepSphere against different spherical convolutional neural networks (SCNNs) to see the influence of the hypotheses and differences with the SCNNs, and put forward the advantages of DeepSphere.

First, different tasks, with meaningful datasets, will be analyzed to filter the most interesting ones. These tasks should have sufficient results in other publications of spherical CNNs to have a reference point. Furthermore, these tasks should put forward the differences of DeepSphere with the other SCNNs to be able to show their impact on the performance.

Then DeepSphere will be adapted to run these tasks and its performance will be compared to existing results.

1.3 DeepSphere

DeepSphere is a graph convolutional network, which works on spherical data and leverages convolutions and hierarchical pooling on graphs, achieving rotation equivariance, computational efficiency and flexibility [4].

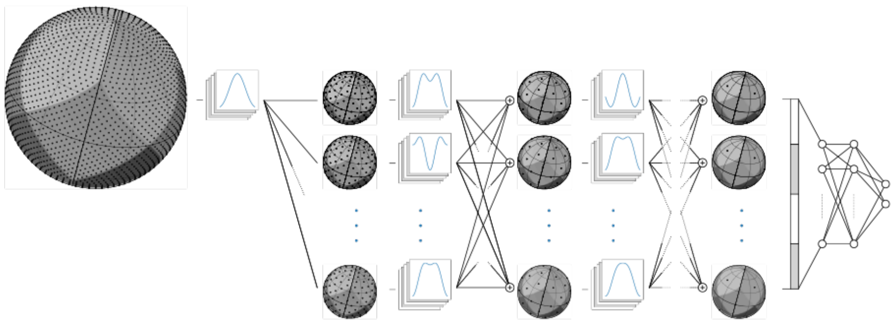

Figure 1.1: DeepSphere architecture

A graph is constructed with the idea of data lying on the sphere S^2, thus presenting the symmetry of this group. This data consisted originally of cosmological maps, but can be applied to any other data fitting this description.
An important point to precise is that DeepSphere deals with the signal on the sphere, and therefore on the graph. Thus it is not the graph itself that is equivariant to rotations but the operations defined on it, such as the convolution, that are rotation equivariant.

This graph approximating the sphere S^2 and using HEALPix [5] sampling scheme initially, has some interesting properties. The eigenvectors of its Laplacian approximate the spherical harmonics. Using these as an orthogonal basis for the graph filter leads to a structure equivariant to rotation. As it only approximates the spherical harmonics, the equivariance to rotation is not perfect either.
The eigenvectors only move towards spherical harmonics with the number of pixels $N \rightarrow \infty$ under the conditions studied by [6].

Initially, the graph convolution is performed on the spectral domain using the eigenvectors of the Laplacian [7]. To avoid the heavy computation of the graph Fourier transform, the eigendecomposition of the Laplacian, it is possible to use the Tchebyshev polynomial filter [8] for a local filter. Thus, the graph convolution is similar to a classical CNN as the convolution is performed in the spatial plane as a sum of weighted neighbors. The neighbors in the graph are regrouped in hops matching the order of the Tchebyshev polynomial [8].

Using such an isotropic filter leads to equivariance to rotation for S^2 group, but the model is invariant to the third rotation of the $SO(3)$ group. Indeed, the filter deals with neighbors in hops around the node of interest, and rotation of the signal around the pixel will not change the response. Another precision is that the graph approximates the S^2 manifold, therefore only two of the $SO(3)$ rotations can be represented on it.

In summary, the key points to take into account are (1) the computational efficiency, (2) the (approximative) rotation equivariance, (3) the flexibility of the model, and (4) the isotropic filters.

1.4 Related works

1.4.1 Sphere

The main reason to use spherical signals is to exploit the rotational symmetry of the sphere S^2 or group $SO(3)$. Indeed, some data present symmetry of rotation, thus even after undergoing a rotation transformation, the instance remains fundamentally the same, and its features captures its properties and its state (the orientation of the signal in the space).

1.4.1.1 S^2 and $SO(3)$

In this section, there is an explanation of these important notions to clarify them.

$SO(3)$ is the group gathering all the rotations from R^3. They correspond to yaw, pitch and roll. This group represents the rotational symmetries of an object, and thus all its potential orientations.

The sphere S^2 is the 2-dimensional manifold embedded in R^3 corresponding to the surface of the ordinary sphere. It is not a group in mathematical terms as there is no composition for points on the sphere. This manifold can be parameterized with only two rotations as it is simply a surface.

1.4.1.2 Fourier and Spherical harmonics

A spherical signal can be represented as a sum of spherical harmonics (S^2) or Wigner D-functions ($SO(3)$)[9]. These functions form an orthogonal basis and are similar to the Fourier mode [10].

The spherical signal has a similar operation to those of time series or euclidean space signals. The spherical harmonics correspond to the frequencies. And in the case of band-limited signals, these spherical harmonics are represented until l_{max} which is the maximum order found in the signal.

1.4.1.3 Discretization

In many fields such as computer analysis and Deep Learning, it is not possible to work with continuous signals. Therefore, the signals are discretized to be used correctly. Even if the 2D plane has an optimal uniform discretization, which is the regular grid, there is no such discretization for the sphere S^2 [4][1][10]. There exist different samplings, which can have the following properties:

- Iso-latitude: this allows a fast Fourier transform and fast computation of spherical harmonics.
- Hierarchical pixelization: useful for pooling operation.
- Same-area coverage: each pixel has the same importance as it covers the same area on the sphere.
- Uniformity: how well the sampling spans the sphere.

Several discretizations are analyzed, and described with their properties hereafter.

Equiangular (also called equirectangular):

Equiangular is the most intuitive and widespread discretization of the sphere. It corresponds to a regular grid in the polar coordinate system, the latitude (ϕ) and longitude (θ), equispaced in the coordinate system, therefore intuitive to understand. A bandwidth L is fixed, corresponding to the maximum frequency (spherical harmonic order) that can faithfully be represented. The grid is composed of 2L latitude rings and up to 2L longitude pixels per ring.

This sampling is iso-latitude and hierarchical, so pooling and SHFT can be done rather easily on them. Nowadays, it is widely used for any spherical signal representation such as omnidirectional imaging or cartography. But it has a main downside as the pixels do not have the same area coverage, meaning that some parts of the sphere (the poles) are more densely pixelated, leading to redundant information. Additionally, the planar projection of this grid has a lot of deformation in the regions of the poles for the same reasons.

Different sampling schemes based on this idea are presented below:

- Driscroll-Healy [10]
  It is the oldest and simplest sampling, and thus it is widely used for planar projection, e.g in cartography or omnidirectional imaging. One of the pole is represented by 2L pixels because the last ring has a radius of 0, aggravating the redundancy of the sampling. These 2L pixels are, in fact, only a single pixel.

$$\theta_i = \pi i/2L, \phi_j = \pi j/L$$

  for $i, j = 0, \ldots, 2L-1$

- SOFT [11]
  This version of the equiangular sampling is an upgrade of the previous and was created in order to use a faster theorem for $SO(3)$ Fourier transform. Here, the poles are no longer a part of the sampling, but the pixel density is still lower at the equator than near the poles.

$$\theta_i = \pi(2i+1)/4L, \phi_j = \pi j/L$$

  for $i, j = 0, \ldots, 2L-1$

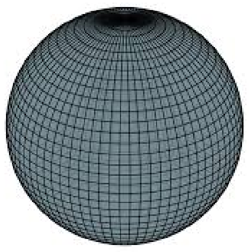
Figure 1.2: Equiangular sampling scheme [4]

- Gauss-Legendre [12]
  The difference with this sampling is that it uses the Gauss-Legendre quadrature of the sphere to find the L latitude rings, and the longitude pixels are equiangular spaced. It is used mainly to construct the exact Spherical Harmonics Transform from its samples. Apart from that, there is not much difference to the classical equiangular sampling scheme.

Geodesic polyhedron [13]:

To have the most uniform repartition and equally distributed pixels as possible, different platonic solids can be used: the tetrahedron, the cube, the octahedron, the dodecahedron and the icosahedron. To have a denser sampling, one has to subdivide each face and project the new points on the sphere. Therefore, by construction, this sampling scheme is hierarchical and the pixels have same-area coverage, although some distortion appear for border pixels.

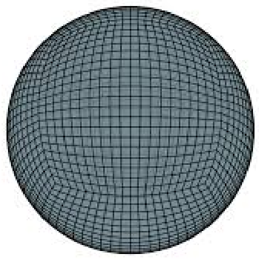

Figure 1.3: equiangular cubed-sphere grid [14]

HEALPix [5]:

"HEALPix is a Hierarchical, Equal Area, and iso-Latitude Pixelisation of the sphere designed to support efficiently (1) local operations on the pixel set, (2) a hierarchical tree structure for multi-resolution applications, and (3) the global Fast Spherical Harmonic transform" [1]

It is based on a rhombic dodecahedron. This sampling scheme is popular especially in cosmology and astrophysics. It presents all interesting features.

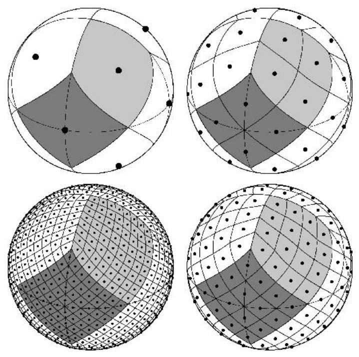

Figure 1.4: HEALPix tesselation

There are more sampling schemes that are not presented here. They were not considered for this study, as no spherical CNNs seems to have used any of them. An in-depth analysis of different samplings had been carried out by Elahi et al [15]. The geodesic polyhedron sampling is also to be mentioned of interest as a great number of studies focused on this

[1]`https://healpix.jpl.nasa.gov/html/intro.htm`

sampling scheme, but will not be investigated any further on this current report as it is not the goal of the thesis. In the end, only two sampling schemes will be used on this study: (1) HEALPix due to the variety of its properties and the fact that an interesting graph already exists, and (2) equiangular due to its wide use in most of the fields working with spherical representations (e.g. omnidirectional imaging) in order to demonstrate the flexibility of DeepSphere. An equivariant to rotation graph is planned to be built, specifically for this project.

1.5 Invariance and equivariance

An important aspect of this study is invariance and equivariance to transformation, more precisely rotation in the scope of this report. In simple terms, invariance is a property held by a group that is not affected by a specific transformation, as equivariance is a symmetry for functions. So, when affected by a specific transformation before the function, the result is the same as applying another transformation to the output of the function. More precisely, equivariance exists if the next equation applies.

$$T_1(f(X)) = f(T_2(X)) \tag{1.1}$$

where:

T_i a transformation operation (e.g. a rotation)

f a filter

X the data analyzed

Invariance is when T_2 is the identity. Thus the result of the filter will always be the same whatever the transformation is, and the descriptor of the data is still the same.

Same-equivariance is when $T_1 = T_2$, so the transformation of the input is the same as the transformation of the output. It is a special case of equivariance. Thus equivariance leads to descriptors that catch the properties of the data linked to the specific transformation. In the case of DeepSphere, the specific transformation is rotation and the properties will be the orientation of the signal.

To take a specific example in Deep Learning, classification tasks are invariant to transformations by definition. Indeed, the goal is to identify data (e.g. object in image, 3D shape, ...) and compute its output. If the data is intrinsically the same even after a transformation, the output should be the same. This invariance is often achieved by using global average pooling at the end of the network. But it is still desired to have an equivariant network before this last step in order to understand correctly the important features from the data.

In the case of a classical CNN, the structure is equivariant to translation by construction, making it a powerful tool for image analysis. Unlike a fully connected network, the power of a CNN comes from weight sharing. Each pixel undergoes an operation using the same learned filter. Thus, the pixels of interest will activate the network the same way wherever they are in the image.

For other transformation such as rotation, flip or dilation, data augmentation is used to learn the new invariance. Repeatedly showing the different transformations to the model and teaching it that it is the same data will force the model to learn invariant filters. Thus, the learning is faster using directly an equivariant architecture, as the filters are already equivariant to the desired transformations.

Another possibility to reach equivariance is to use a transformation of the input space to translate the new desired transformation into a useful one - translation in case of a CNN. For example, using a polar projection instead of the 2D grid will translate rotation to translation, even if the result is not easily understandable for the human eyes (the euclidean coordinates (x, y) change to polar coordinates (θ, ρ)). But it is not always possible to use this method. For example, the sphere, which has no perfect planar projection, cannot have a transformation where the translation on the projection does translate correctly to the desired rotations.

1.5.1 Spherical CNNs

Architectures that exploit structure and symmetries present in data are powerful because they do not have to learn them and can directly focus on the main features needed for the task. Thus being invariant or equivariant to some symmetries such as translation or rotation in the data domain can speed up the training time, and make the given model more robust to these perturbations.

More datasets present rotation symmetries or lie on the sphere S^2 (different examples are explained in section 2.1). In this section, architectures working on this kind of data are presented, along with their approach.

2D CNNs on planar projection

One of the most instinctive use of CNN on spherical data is to use a planar projection of the sphere S^2 and apply a standard CNN. However, as it was seen in sec. 1.4.1.3, there is no perfect uniform sampling of the sphere and thus no projection without deformation.

Then, it is not possible to translate the desired equivariance to rotation to an equivariance to translation, because deformation is induced on the projection (see Fig. 1.5).

One motivation to use a planar projection of the sphere is to be able to use the highly scalable and powerful standard CNN. The idea is to take advantage of the existing tool, and to transform the input data to faithfully represent it in a suitable format for it.

The first natural approach is to use the classical equirectangular projection and use a standard CNN on the resulting image. But the deformation is large at the poles and the idea of weight sharing does not make sense anymore. Furthermore, the square image with borders does not represent the continuity of the sphere. Thus, different ideas were found to get around this problem.

As a response to the problem of the deformation, a different kernel weight in function of the latitude ϕ is used in [16]. Unfortunately, this does not resolve the border problem. Others decided to apply the kernels on tangent planes rather that directly on the equirectangular projection (such as [17]), and represented the distorted kernel on the plane.

All these approaches focus more on the distortion free aspect rather than the equivariance to rotation. However, this kind of architecture is mostly used to analyze omnidirectional images and it is argued that rotation equivariance is not necessary, as the images are always oriented with the gravity. That means that the ground will be at the bottom of the image and ceilings and the sky at the top of the image, as images will then unlikely be upside-down.

Other projections are based on the platonic solids, such as the cubed sphere [18], or the icosahedron [19]. In this case, each face is a planar projection, and instead of using padding, the missing values are taken directly on the adjacent faces. Using this projection, the equivariance is not guaranteed, and needs a corresponding filter [20]. But depending on the task, this equivariance is not always necessary, as explained in [21].

Any other sampling scheme presented, and more specifically the polyhedron, can be used for this kind of spherical neural network. There is even a case of a HEALPix projection in [22].

These projections, at best, can only reach perfect discrete rotation equivariance, the rest is only approximation. However, this method still performs well, is scalable and fast, has local filters and operates in the spatial domain.

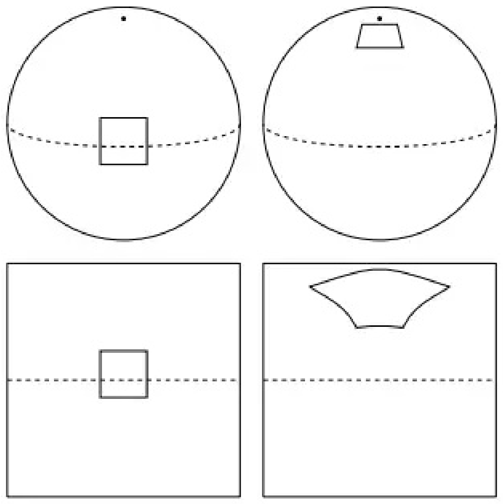

Figure 1.5: Deformation of planar projection [1]

As shown in the work of Cohen [23, 20, 24], the difference between isotropic and anisotropic filters is important. Anisotropic filters, also called regular feature fields, are filters that have different weights for each neighbor, opposed to DeepSphere filters, that have a weight for a hop of neighbors, and react the same way wherever an important neighbor is. They perform generally better than scalar feature fields (isotropic filters) on tasks where the direction is an important feature.

Furthermore, the gauge transformation used by Cohen let him have all kinds of equivariance (from Equ. 1.1, T_2 can be chosen to fit the needs), which permits flexibility superior to the same-equivariance generally proposed, and seen in DeepSphere.

Spherical Harmonics Fourier Transform

Similar to the case with images, in order to reach true equivariance and find correct convolution, one needs to use Fourier transform and go in the spectral domain. Depending on the group of symmetry one wants to use, the Fourier modes correspond to Spherical Harmonics for the group S^2 and to the Wigner D-functions for the group $SO(3)$.

Theoretically, convolutions in the spectral domain of $SO(3)$ are perfectly equivariant to rotation for continuous signals, using the properties of the symmetry group. Using a discretized signal leads to small errors, and is exact only for band limited functions, because the high frequencies (spherical harmonics or Wigner D-functions) are not represented correctly and disappear in the spectral domain.

Models using this method will be called SHFT-CNNs in this report.

Taking into account that the spherical signals lie on the sphere S^2, it seems normal to use the spherical harmonics to achieve a convolution equivariant to rotation [2]. New faster Spherical Fourier transform (SFT) were developed to reach the goal of having this convolution.

However, the space of moves on the sphere is more complex, and the whole $SO(3)$ group must be considered to be as accurate as possible according to [1]. Different models based on this idea emerged [25, 26]. As well as the S^2 counterpart, new SFT for $SO(3)$ were developed to obtain convolutions that are equivariant to rotation.

All these methods use an equirectangular grid to represent the data. This is probably due to SFT being mainly developed for a specific sampling scheme [11]. Indeed, the SFT algorithms depend on the chosen sampling and cannot easily be translated to another.

The main problem with this method is that the Fourier transform must be computed for each forward and backward pass during the training of the neural network. This is computationally expensive, even when using a Fast Fourier transform algorithm.

It is possible to expand this methodology to other groups of symmetry, and using translations as well, such as in [27] or [28]. In this case, the data is not spherical anymore. This goes beyond the scope of this study.

Graph CNNs

Another way to represent the data that seems more and more used, is to use a graph to approximate the manifold where the data lies. It is then possible to use a graph-CNN on this representation (e.g. in [7] or [29]). The graph convolution is designed in the Fourier domain as there is no meaningful translation operator in the vertex domain [8]. Thus it is the same as applying a graph filter, using the Laplacian eigenvectors, with the main downside being the expensive graph Fourier transform, to perform each time. Indeed, there is no FFT generalized for graph Fourier Transform.

The advantage of graphs is that they can use any coordinates and sampling, thus fitting any data and going beyond classical euclidean domain. This is the case for MoNet [30], which regroups different Graph Convolutional Neural Networks and devised a more general framework from the others, learning the weight function, which is fixed in other representations. A review on Deep Learning on graph can be found in [31].

To fit the spherical data, a graph approximating the sphere S^2 is a logical choice, since this same choice is made for cosmological map classification with DeepSphere [4] and omnidirectional images representation [32, 33]. Furthermore, to have more computationally efficient graph convolution, the expensive eigendecomposition can be avoided by replacing the filters with a polynomial function of the Laplacian [8, 33]. However, this applies only for localized filters. The order of the polynomial function K can be seen as the number of neighboring hops a localized kernel can interact with.
At the same time, as the kernel sees rings of neighbors, the hops from the localized pixel, it has no idea of orientation, resulting in an isotropic filter. Thus, some information may be lost, and in terms of image processing, these filters can only detect blobs of pixels, not edges or directional changes of intensity[2].

All these spherical CNNs are task-agnostic and can be adapted for any task, as long as the input can be explained in terms of a graph (e.g. a spherical signal for DeepSphere). Nonetheless, it can be domain specific if the model is dependant of a specific coordinate system.

[2]https://twitter.com/TacoCohen/status/1123736021546545156

2. Methodology

As the goal of this study is to compare DeepSphere [4] against several other spherical CNNs to show its advantages and the impact of the differences with the SCNNs and the hypothesis made, the first step to do is to find different tasks to work on and the datasets linked to them. These tasks must have results from different spherical CNNs in order to compare them with DeepSphere results, and challenge the different aspects of DeepSphere. Another set of tasks has to be chosen such that it puts forward DeepSphere's advantages, where few of the other models could compete on these properties.

As a reminder, the main advantages of DeepSphere are:

- The convolution is similar to a traditional CNN: A weighted sum of neighbors, leading to a fast computation.
- Flexibility: Any sampling can be used, as long as a graph can be built, even if the sampling covers only part of the sphere, or the sampling is irregular.

More importantly, the differences and hypothesis made that have a yet unkown influence:

- By construction, the convolution is only equivariant to rotation on S^2 (the graph approximates this manifold). The convolution is invariant to the 3$^{\text{rd}}$ rotation presents only in $SO(3)$.
- The filters are isotropic.
- The equivariance to rotation is theoretically not perfect, due to graph construction.

2.1 Different tasks

This study focuses only on supervised task, and a list of potential interesting ones is presented below.

- Cosmological models classification

  This is the original task from [4]. The goal of this task is to discriminate cosmological maps simulated with two different sets of parameters. The main problem with this task is the high-density of the data, and the fact that the necessary features to discriminate the models are small and will be lost if the maps are down-sampled too much. Thus SHFT-SCNNs (cf. section 1.5.1) cannot be trained in a reasonable time.

- Proof of concept classification or segmentation

  For a majority of SCNNs, mock datasets are standard 2D images that are projected on the sphere. They are called mock datasets because they are 2D images and are artificially transformed to a spherical signal. Unfortunately, this is often senseless, as no real signal could be similar to them. Therefore, it does not make sense to use this type of tasks to compare DeepSphere with the others as they are designed to prove the good functioning of the spherical models.

The most popular example of this type of task is the projected MNIST dataset, used by a great number of publications such as [1, 20, 32, 33, 34, 22, 17]. This tasks make the least sense as there is no character recognition on the sphere in real life.

Other mock datasets are PASCAL VOC [16], a dataset of planar images for object class recognition, or FlyingCars [17], equirectangular projection of panoramic images with additional 3D model of cars to find.

- Shape retrieval

  This task consists in finding descriptors of 3D CAD models in such a way similar models could be retrieved from a single one, minimizing the distance between the descriptors. It is used by [1, 2, 35], and the main interest is the fact that the models remain fundamentally the same, even rotated, thus presenting $SO(3)$ symmetry.

  ShapeNet and ModelNet are two datasets of CAD models used for this task.

- Shape classification

  This task is similar to the previous one, and uses the same datasets, but the goal is only to classify the models, which seems easier. Results for this task are in [2, 35, 21].

- Scene identification

  This task consists in classifying scenes where the configuration is inside-out, the point of view is on the center of the sphere, and the scene viewed is projected onto the outer sphere. The data has the form of panoramas images (equirectangular projections) or point clouds.

  Some datasets present in literature are Matterport3D [36] used by [35] or SUN360 used by [19].

- Omnidirectional segmentation

  The previous task can be transformed easily into a segmentation problem, to identify the objects within the scenes. The data often consist of omnidirectional images with RGBD channels, and are presented as equirectangular projection or point clouds.

  Datasets present in the literature are 2D3DS [37] used by [21, 20] or SYNTHIA used by [19].

  Even if omnidirectional images lie on the sphere, due to the optical properties of the lens, or the type sensor used to capture the images, they do not necessarily present rotation symmetry. Indeed, they are always oriented by gravity (with the ground at the bottom of the image).

- Climate pattern segmentation

  Planetarian data seems a logical approach, as they lie directly on the sphere - the Earth. Planetarian data are all signals evolving on the Earth such as weather, economics or demography. Again they do not necessarily present rotation symmetry, as the globe has a single axis of rotation, and the behaviour of the data is not the same on certain parts of the sphere.

  A simple signal is the weather, and a task born from this data is the classification of climate or the detection of extreme weather pattern from multiple weather signals. This task is performed by [21, 20].

Of course, many other different tasks that may be pertinent, but those presented were the most represented in the analyzed literature. Another point would be to create our own task from scratch, and it would still be possible to compare the DeepSphere results with the other public spherical CNN ones. But a task, or data, where DeepSphere could be put forward was not found.

3. Benchmark

3.1 Choice of tasks

First, a global task in common with the other sCNNs is chosen, in order to see how well DeepSphere performs against them despite having an invariance to the 3rd rotation. The SHREC17 competition[1], using a shape retrieval task, is selected for this purpose. Cohen [1] and Esteves [2] used this task to demonstrate their model efficiency. Their methodology can be reused as well as their results. Another similar dataset with a classification task is chosen as well: modelNet40[2]. The methodology is similar, and the task is used by Jiang [21]. These two tasks will demonstrate how well DeepSphere performs in comparison to other state of the art spherical CNNs in terms of speed and performance, as well as confirm the influence of the isotropic filter and the imprecision of the rotation equivariance.

The main other aspect that remains to be shown is the flexibility of DeepSphere. To demonstrate this, another sampling can be chosen, using only a part of the sphere, or any spherical signal that is sampled non uniformly over the sphere. Measures on the Earth such as weather, demography or economics is a great example because the Earth can be approximated by a sphere, and the density of measurements is greater on the continents than on the oceans. For this reason, the GHCN-daily [38] dataset is chosen, even if there is no underlying task and no comparison with the other spherical CNNs, as they cannot work with other sampling.

Finally, a segmentation task is chosen to demonstrate that DeepSphere can be adapted for any task. A climate event detection task with the Extreme Weather dataset [39] or the climate pattern segmentation task [40] can be chosen. The second one has the advantage to be performed by Jiang [21] and Cohen [20] as well. It is also more interesting as published results are available. Both these tasks are on hold due to a lack of time for the experimentations.

All the experiments code is available on Github[3] and they were run on NVIDIA GeForce GTX 1080 Ti GPU with 11GB of memory.

3.1.1 Shape retrieval - SHREC17

SHREC17 is the SHape REtrieval Contest of 2017. The goal of this contest is to develop a tool to manage and analyze 3D shapes, in the form of CAD models. The contest focuses more on the retrieval part than classification. Retrieval means to quantify how much a model is similar to another one. In other words the goal is to find good descriptors for these shapes, the retrieved models being the ones with a high similarity/a low distance between descriptors. More information can be found on their website [41].

[1]`https://shapenet.cs.stanford.edu/shrec17/`
[2]`http://modelnet.cs.princeton.edu/`
[3]`https://github.com/Droxef/PDMdeepsphere`

The dataset has 51'300 models, unevenly distributed in 55 classes and is a subset of the ShapeNet dataset. The subcategories will not be used for our study as Cohen and Esteves did not use them. It is separated in three sets: training (70%), validation (10%) and test (20%). The class distribution is similar between the sets. Two versions of this dataset exists: an aligned version where all the models have the same natural orientation, and a perturbed version where the models are randomly rotated. The second one is used by Cohen to test the rotation equivariant representation [1], and is also chosen in this work for the same reason.

Another advantage of this dataset is that it can be used as a classification task for another comparison as well. Further it seems that there is still room for improvement considering the results of the spherical CNNs and the winners of the contest, which are task-specific models.

Figure 3.1: Example of a shape from SHREC17 dataset

3.1.2 Shape classification - ModelNet40

The task is global classification of 3D shapes and the data is similar to the previous task as it is a collection of 3D CAD models [42]. The dataset is a collection of 12'311 models unevenly distributed in 40 classes. It is split in two parts, 80% for training and 20% for test. The class distribution is similar between the parts. Even though a new clean dataset exists with aligned models, the older version was chosen for the same reasons as for SHREC17 (cf. section 3.1.1).

The main advantage to choose a similar task is to reuse the preprocessing and the methodology, and to be able to compare with more available results. The challenge with this specific task is the relatively small size of the dataset, and the fact that some of the classes are not easily distinguishable. For example, it is easy to distinguish an airplane from a stair, but not a flower pot from a vase or a plant.

3.1.3 Dense regression - GHCN-daily

As said earlier, GHCN-daily is a dataset gathering different measurements from over 100'000 weather stations around the globe, with daily measures since 1891 [38]. The main features are meteorological measurements such as temperature, precipitation and snow fall and are represented daily, as suggested by the name of the dataset. More measurements exist, but they are not represented enough to be used correctly.

The main interest for choosing this dataset is that the data lies naturally on a sphere, the Earth, and the pixels, the weather stations, form a non-regular and non-uniform sampling. Indeed, there is a higher concentration in some countries such as USA, Germany or Australia, and almost nothing over the ocean.

There are no specific tasks defined for this dataset, and no spherical CNN or state-of-the-art architecture has been used on it at the time of this study. The only application found is the climate classification by Jessica Stringham[4]. The first thing to do is to define a task that makes sense. One type of tasks not performed yet with DeepSphere is regression problem, and as features with potential correlation is available, exploring regression is interesting.

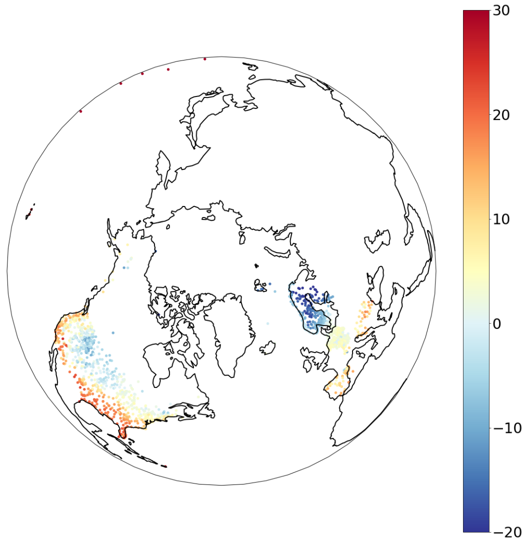

Figure 3.2: Minimal temperature over the Earth

[4]http://jessicastringham.net/2018/06/11/climate-classification-with-neural-nets.html

3.1.4 Climate event detection - ExtremeWeather

The ExtremeWeather dataset [39] is used for extreme weather events detection based on simulations. It was constructed for the use of CNN, for semi-supervised learning and provides only bounding boxes instead of groundtruth labels for the events.

The data com from CAM5 [43] that is a meteorological atmospheric model. It is presented on a equiangular projection with a 25-km resolution at equator (a grid of resolution 0.23°for longitude by 0.31°for latitude), and the different extreme weather events are shown with bounding boxes.

One possible idea is to create a supervised segmentation task, creating labels the same way as in Mudigonda work [40]. Once the groundtruth labels are found within the bounding boxes, it becomes a dense classification task.

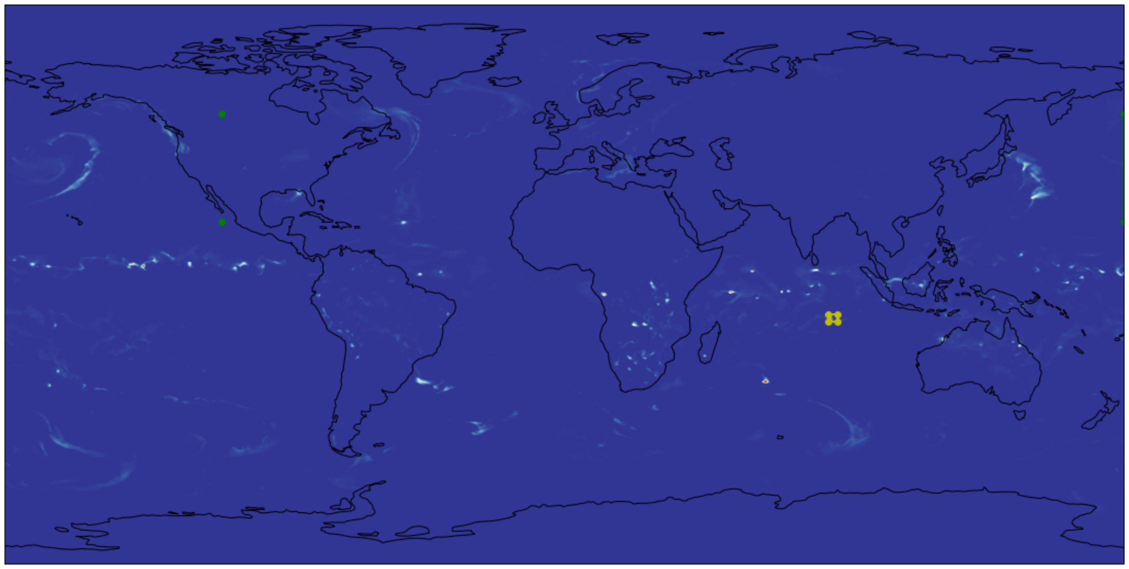

Figure 3.3: Equiangular projection of precipitation over the earth. Low values are blue and high values red.

3.1.5 Climate pattern segmentation

This section is still incomplete as there is not enough time to preprocess the data and do the experiments. But the goal is to perform the same climate pattern segmentation as Jiang and Cohen did, based on the first experiment of Mudigonda [40, 21, 20] with a dataset based on CAM5.

3.2 Previous work

3.2.1 Cosmological maps classification

The first problem where DeepSphere was demonstrated is a discrimination problem. The goal was to classify cosmological convergence maps in two models [4].

The maps used the standard cosmological sampling, HEALPix, with a density of $N_{side} = 1024$, corresponding to 12.5×10^6 pixels per map. As the distribution of the data in the maps is isotropic and homogeneous, the maps were split from 12 parts of sphere to 192 parts of sphere to increase the difficulty of the problem.

3.2.2 Comparison

It is difficult to compare with other spherical CNNs because there are too many pixels and the computation time is too high for Fourier transform, and others were demonstrated with much lower resolution (only 10×10^3 pixels).

It is not possible to reduce the density of the maps too much (less than $N_{side} = 512$) as too much information would be lost, and the task would become impossible.

An analysis of the inference time[5] is computed to show that SHFT-SCNNs cannot compute in a reasonable time the cosmological tasks. The results are seen in Fig. 3.4, and the missing values are due to (1) not enough memory on the GPU for Cohen's model and (2) to the naive method for computing spherical harmonics for Esteves' model taking too much time (>24h) for $bw = 220$. The time for computing the spherical harmonics in the case of Esteves' model is not taken in consideration in this analysis because it is done offline.

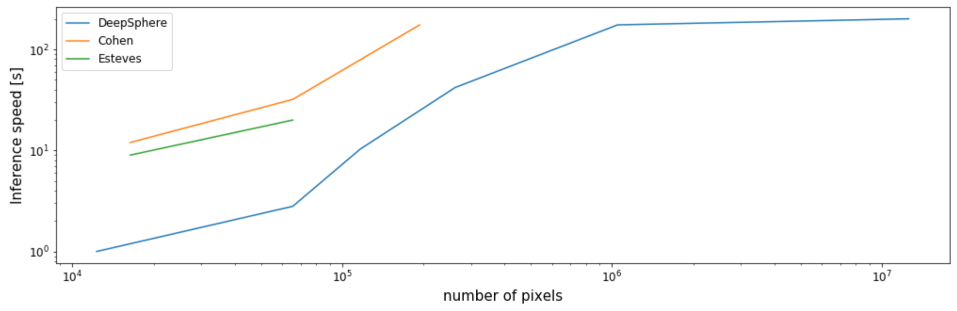

Figure 3.4: Analysis of the inference speed for different models and different map density on the cosmological maps classification task

3.3 Shape retrieval task - SHREC17

3.3.1 Task presentation

First, the approach of the other SCNNs are described in order to run the experiments in similar conditions. Indeed, in order to compare the SCNNs, the same basis much first be achieved.

3.3.2 Other spherical models

3.3.2.1 Cohen Model

Cohen used an equiangular grid following a SOFT sampling scheme [11] to project the data. The original model [1] uses a bandwidth bw of 128. For the comparison, we will use

[5]It is the time needed to compute a single pass of training for one instance

the version with $bw = 64$ required for the simple model provided in github[6].

The treatment used by Cohen with its s2cnn is a ray cast of all the pixels from the chosen sampling point in the direction of the center of the sphere. The mesh object is scaled to fit within the unit sphere, the center of mass of the shape being in the center of the sphere. Six features were computed from the rays: distance from the sampling point to the first intersection of the ray and the mesh object, cos (scalar product) and sin (cross product norm) between the ray and the normal of the mesh and the 3 same features using the convex hull of the mesh.

The model architecture can be summed up by:

$$\text{Cohen}(X) = FC \circ I_{SO3} \circ L_2 \circ L_1$$

where:
$L_1 = P_{16} \circ ReLU \circ BN \circ F_{S^2/100}$
$L_2 = P_{10} \circ ReLU \circ BN \circ F_{SO3/100}$
FC is a fully connected layer, I_{SO3} is an integration over $SO(3)$ layer, P_x is a pooling layer to a map with bandwidth x, BN is a batch normalization layer and $F_{X/Y}$ is a convolution layer over the domain X, where Y is the output features.

Thus, the first layer is always a convolution in S^2, as the spherical signal lies on the sphere. The further layers will always be convolutions over $SO(3)$ and the kernel used for the filter has non-local support.

A simpler model gives similar results, has one layer less than the original, uses data with a smaller bandwidth (64 instead of 128) and uses other data augmentation, like random rotations. Finally, the global max pooling from the initial model is replaced by an integration over $SO(3)$ in the simple model. It is observed that the resulting training time is smaller, as the number of parameters is reduced by 10^6.

We observe that random rotations are added as augmentation in the simpler model, despite being redundant as the structure is rotation equivariant. It may be because the sampling density is different at the poles and the equator, and high frequency information is better represented in high density zones.

3.3.2.2 Esteves Model

The model architecture of Esteves' model [2] is much more complex as it is deeper and narrower. Additionally, each feature is treated separately in a different branch, and the results are concatenated before the global average pooling. This original architecture leads to a more robust model.

Batch normalization is also used after each convolution, and a fully connected layer is added after the global average pooling. Additionally, random rotations are used for augmentation, likely for the same reasons as in Cohen's work. The much deeper model allows for a fast learning, and it needs only few epochs to reach peak performance.

[6]`https://github.com/jonas-koehler/s2cnn/tree/master/examples/shrec17`

The main original elements are:

- the pooling in spectral space, as it is easier to use, acts like a low-pass filter, and an inverse SFT is not needed;
- the convolution is in the spherical harmonics domain, and thus is only equivariant to the symmetries of S^2;
- the use of a localized filter where the spectral space is discretized with a few anchor points and linear interpolation is used;
- the global average pooling is weighted by the area of the pixel to take into account the deformation of the equirectangular projection.

The same projection and bandwidth as in Cohen's work is used, but only the two first features are kept (the distance to the model and the sin with the face normal).

3.3.3 Implementation

The data is first preprocessed in order to get a spherical signal. The same preprocessing is used for all models to have the same basis for the comparison.

The principle of the ray casting used by Cohen is sketched in Fig. 3.5. As Cohen did, six features are computed: distance of the ray to the first intersection, cos/sin with the normal of the mesh and the 3 same features using the convex hull. The resulting spherical signal of the first feature for the same object as the last figure is the Fig. 3.6.

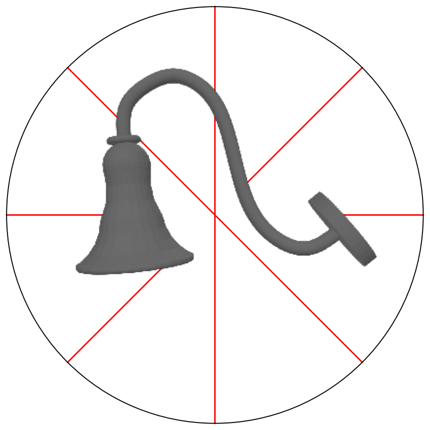

Figure 3.5: Sketch of the ray-casting on a sphere

The goal to use a spherical signal to represent a complex shape is to take advantage of the symmetries of the sphere. Indeed, 3D shapes have different possible orientations that spans entirely $SO(3)$. Whichever its orientation is, the internal representation of the shape is the same and should be recognized as the same object.

In a general way, the rotational symmetries of any task can be exploited by projecting the data on the sphere S^2.

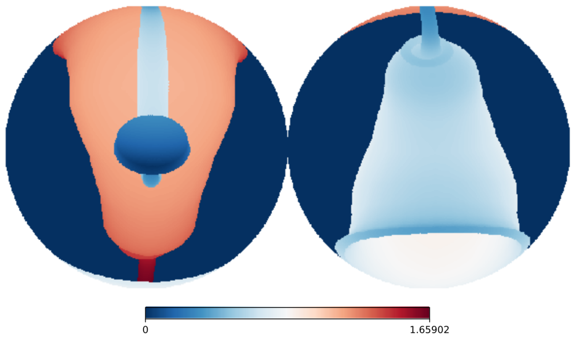

Figure 3.6: Back view (left) and front view (right) of the spherical signal obtained by ray-casting the distance from the ray to the first intersection on a sphere.

A sampling density compatible with the two comparison models is choosen first. Both models use an equiangular sampling with a bandwidth of 64 resulting in a total number of pixels of $(2 * bw)^2 \approx 16k$. The chosen sampling is HEALPix with an N_{side} of 32, in order to have a similar, but slightly lower, number of pixels.

The data was then normalized to have a normal distribution of mean 0 and standard deviation 1.

As a first step, a similar architecture as in Cohen's simple model is used, which is different than the one Cohen used in his publication. It is assured that the results are similar, and it was rerun to confirm it.

The DeepSphere model based on this is called *Cohen-like* later in the results. The use of a similar architecture is done to compare the influence of the different convolutions as truthfully as possible.

The same number of layers and feature maps as Cohen's simple model are chosen, and batch normalization is used. The only main differences are the convolution layers. Cohen does the convolution in the spectral space of $SO(3)$, while DeepSphere uses a graph Tchebyshev convolution.

To express the non-local filter used by Cohen, the Tchebyshev filter has an order high enough for the hops to span the whole graph. More specifically, the order of the filter should be equal to the diameter of the graph, so $K = \lceil N_{side} \cdot \sqrt{3} \rceil$. The integration over $SO(3)$ before the last fully-connected layer is replaced by a global average pooling.

A stochastic gradient descent optimizer is used with a fixed learning rate of 0.5, and a classical loss function for classification is used: the cross entropy loss.

Furthermore, because Tchebyshev filters with a high order are computationally more expensive, and non-local filters not really necessary, a lighter version than previously with a smaller order of $K = 5$ was performed, named *local filter* in the results. The cost of graph filters increases linearly with its order, but the performance decreases slightly with a local filter. Thus, having a non-local filter is not a necessary price to pay as a deeper model can be used.

In a next step, a deeper model is used, with more parameters, because DeepSphere is a faster and computationally less expensive method as SHFT-SCNNs, and deeper models tends to have better results. The resulting architecture is composed of 5 convolutional layers, following by a global average pooling and a fully-connected layer. The architecture NN is described below:

$$NN = \text{softmax} \circ FC \circ GAP \circ \text{ConvLayer}_{256} \circ P \circ \text{ConvLayer}_{128} \circ$$
$$P \circ \text{ConvLayer}_{64} \circ P \circ \text{ConvLayer}_{32} \circ P \circ \text{ConvLayer}_{16}$$

,

$$\text{ConvLayer}_i = ReLU \circ BN \circ GC_i$$

where $\circ$ is the composition operator, GC_i is the graph convolutional layer with i features using a Tchebyshev filter, BN is a batch normalization layer, P is a pooling layer, GAP a global average pooling layer and FC a fully-connected layer.

As a small local filter is sufficient and makes the training faster, an order of $K = 4$ is chosen for all the GC. The FC goes to 55 features, corresponding to the number of classes for this task as the output of the neural network is the probabilities to belong to each class. Each layer the pooling factor is 4 to reduce the sampling density N_{side} by 2. This architecture is named *Optimal* in the results, because it gives the best performance among the others.

3.3.3.1 Equiangular sampling

To test the flexibility of the model and to show that the performance of HEALPix is better than equiangular sampling, an equiangular graph is constructed and used in the same condition as the *Optimal* scenario. The bandwidth and sampling used are the same as in Cohen and Esteves models ($bw = 64$ and sampling is SOFT). This experiment is named *equiangular* in the results.

As the equiangular sampling is far from being uniform (the pixel density is higher at the poles than at the equator) and redundant [15], it is more challenging to have a graph with the same properties as the HEALPix graph. In order to have equivariance to rotation on the S^2 sphere, the eigenvectors of the Laplacian must at least approximate the spherical harmonics and have a spherical shape in the eigenspace. Martino Milani analyzed a full equiangular graph (all nodes are connected to all the others) in his study [6]. He showed that, even if this graph will never converge to the spherical harmonics, the first orders approximate well enough the spherical harmonics.
However, when using only the 4 first neighbors (north, east, south and west), the approximation of the spherical harmonics is to the full graph. The first three eigenvectors (except the eigenvector 0, which is constant) are plotted in Fig. 3.7.

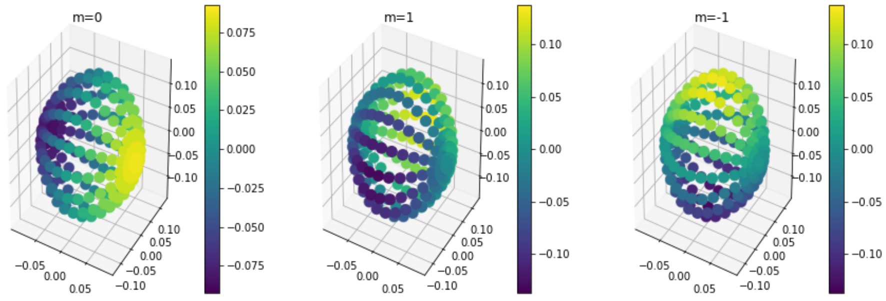

Figure 3.7: First three eigenvectors of the equiangular laplacian

Although the three eigenvectors seems to approximate well the first mode of the spherical harmonics, the embedding of the two first eigenvectors corresponding to Y_0^1 and Y_1^1 (Fig. 3.8) show that they cannot form a perfect sphere, thus it still is not a satisfactory approximation.

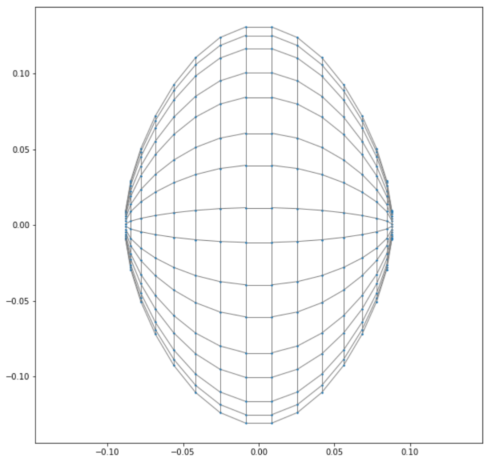

Figure 3.8: Embedding of the two first eigenvectors of the equiangular laplacian

The best case would be to use a full graph for learning. Unfortunately this Laplacian is not sparse and is too large for the GPU and Tensorflow (over 2GB limit). Furthermore, the filter will span the entire graph with only one hop, slowing the training. For these reasons, the test was done only with a 4-neighbors graph.

3.3.4 Analysis

Results

The SHREC17 contest is considered from two point of views. The only differences between them is the metric used to compute the performance of the models. The first metric is a shape classification task, using the accuracy and F1-score metrics from *scikit-learn*[7]. The

[7]`https://scikit-learn.org/`

second is the original shape retrieval task where the organizers provided a script to compute the necessary metrics (Precision, Recall, F1-score, mean Average Precision and Normalized Discount Cumulative Gain). Each metric is computed as an average per instance (macro) or per label (micro).

It is seen in Fig. 3.9, that the model learns really fast the general descriptors of the shapes, but overfits as the validation set predictions do not become better, and the model becomes more confident on the training set as the number of epochs increase. This explains why the loss on the training set keeps decreasing and the accuracy on the validation set is increasing, while the validation loss remains constant or increases steadily.

Even in this case, the results still seem to improve as the number of epochs increases, until the accuracy on the training set reaches a plateau near 100%. Many experiments were done in order to reduce the overfit. The summary can be seen in section 3.4.5.

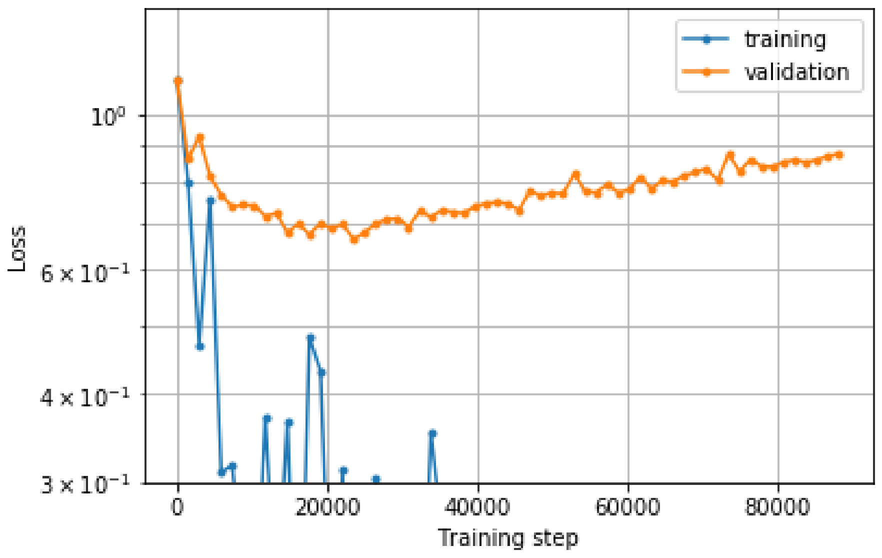

Figure 3.9: As the loss of the training set (blue) decreases, the loss of the validation set (orange) starts increasing, showing overfit of the model

However, it can be seen that using augmentation (either random translations or rotations of the models) increases slightly the performance, and the validation hits a lower plateau, with a smaller increase over time. Different augmentations were tested: 3 random translations with a maximum amplitude of 0.1, 3 random ZYZ rotations, and the combinations of them. The results of the first two is in Tab. 3.3. The combination, giving 6 augmented instances for the entire dataset, gives the best result, showing that the amount of data is insufficient without augmentation to learn perfectly the descriptors.

The result of the classification task is given in Tab. 3.1. The inference speed corresponds to the time for a single instance to do a single pass of the inference, and the training speed corresponds to the time for the model to reach peak performance. Cohen's simplified model and Esteves' model were rerun to confirm the classification metrics and the time.

Only Cohen's original model was not fully rerun, as the original script is not provided and the complicated learning rate schedule is not given. Thus, the experiment from Cohen et al could not be reproduced exactly, and the results seem off. Indeed, the training time should be 50h and with the given hyperparameters, the real time is near 200h. The results obtained seem lower too, compared to the results Cohen reported.

The *Cohen-like* model gives similar results as the other spherical CNNs to 2 points in accuracy. The lower performance can be explained by the difference between isotropic filters (DeepSphere) and anisotropic filters (Cohen and Esteves). It is observed that DeepSphere is faster with a local filter (up to 8 times faster for the inference speed), so the *Optimal* architecture can use more layers with a small impact the computational time, and reach better performance. The smaller inference time is due to a lower order for the filter ($K = 4$), and the better performance, to a deeper architecture. Indeed, a single pixel has a better vision of its surroundings after each convolutional layer and the final neighborhood is close to the whole graph.

The smaller inference time for Esteves is due to the use of a local filter, and the smaller training time due to the number of epochs optimized and a greater augmentation of the dataset. And the results for the *s2cnn* original seems off, due to the incorrect learning rate schedule and wrong optimizer. The publication states that the training time of the original model was 50h [9].

	performance		size	speed	
Method	Accuracy	F1-score	params	inference	training
Cohen *s2cnn*	(73.10)	(72.86)	1.4M	(38ms)	(65h)
Cohen *s2cnn_simple*	78.59	78.85	400k	12ms	32h
Esteves *sphericalcnn*	79.18	79.36	500k	9.8ms	2h52
Deepsphere *Cohen-like*	77.86	77.90	170k	17.5ms	9h43
Deepsphere *local filter*	76.83	76.66	60k	2.8ms	1h37
Deepsphere *equiangular*	73.36	73.67	190k	0.98ms	43m
Deepsphere *Optimal*	80.42	80.65	190k	1.0ms	48m

Table 3.1: Performance of different models (described in section 3.3.4 on SHREC17 dataset as classification task. The numbers in parentheses are got using incorrect parameters in contrast of the publication's results [1]

In the case of the shape retrieval task, an additional loss was added to improve the performance and reduce the overfit. Inspired by Esteves [2], the triplet loss promotes smaller distance between similar results and penalizes the distance between instances that are different. Thus, only the *Optimal* architecture had this extra loss.

The result of the retrieval task is given in table 3.2. These metrics are provided by the official script of the competition[8]. In addition to others SCNNs, the winners of the SHREC17 competition are added to compare with state-of-the-art task-specific models. The *s2cnn* original are the original publication results of Cohen's work [1], for the reasons explained earlier.

[8]https://shapenet.cs.stanford.edu/shrec17/code/evaluator.zip

Once again, Deepsphere reaches similar results as the others SCNNs, with lower performance for *Cohen-like* and *local filter*, and slightly better performance for the *Optimal* model.

Concerning the *equiangular* run, even using the *Optimal* architecture, the results are worse than the others. This shows that the chosen equiangular graph is not good enough, as expected from Fig. 3.8 and not as equivariant to rotation as the HEALPix graph. However, despite the disadvantages of the equiangular sampling, results are close to the other Deepsphere models. This may be explained by the fact that the low frequencies are the most important ones, in order to reach good results for this task, and this graph only approximates well the first order. Please refer to Martino Milani's work for further precisions [6]. If a better graph could be constructed for this model, similar performance as the HEALPix graph could be expected.

	micro (label average)				macro (instance average)			
Method	P@N	R@N	F1@N	mAP	P@N	R@N	F1@N	mAP
Furuya DLAN	0.814	0.683	0.706	0.656	0.607	0.539	0.503	0.476
Tatsuma ReVGG	0.705	0.769	0.719	0.696	0.424	0.563	0.434	0.418
Cohen *s2cnn*	0.701	0.711	0.699	0.676	-	-	-	-
Cohen *s2cnn_simple*	0.704	0.701	0.696	0.665	0.430	0.480	0.429	0.385
Esteves *sphericalcnn*	0.717	0.737	-	0.685	0.450	0.550	-	0.444
Deepsphere *Cohen-like*	0.695	0.688	0.684	0.654	0.424	0.478	0.423	0.389
Deepsphere *local filter*	0.684	0.682	0.676	0.643	0.410	0.452	0.398	0.354
Deepsphere *equiangular*	0.642	0.629	0.627	0.589	0.370	0.423	0.370	0.322
Deepsphere *Optimal*	0.725	0.717	0.715	0.686	0.475	0.508	0.468	0.428

Table 3.2: Performance of different models over SHREC17 perturbed dataset as a retrieval task, using the official script of the competition. The first two models are high-scorer of the competition.

To summarize, the equivariance to the 3$^{\text{rd}}$ rotation is an unnecessary price to pay, as the difference in speed is up to 40 times, and the difference in performance is small and can be counterbalanced by having a deeper model to reach similar results. The drawback of this choice is the increased computational cost, but DeepSphere is a scalable method instead of SHFT-SCNNs, thus it is still advantageous. Thus, an isotropic filter is sufficient for this task and this representation of the data.

3.3.5 Further improvements

The spherical projection used to represent the 3D shapes is a poor representation of it, mainly due to the fact that information is missing when there are occlusions within the object, or some surface are near parallel to the ray casted. In Fig. 3.10, there is a specific example of an object (a bookshelf in this case) with occlusions and near parallel surfaces and its representation as a spherical signal (Fig. 3.11).

A new possibility to represent the data is to use a multi-view CNN, as proposed by Esteves [35]. Each signal pixel is the resulting descriptor given by a classical CNN applied

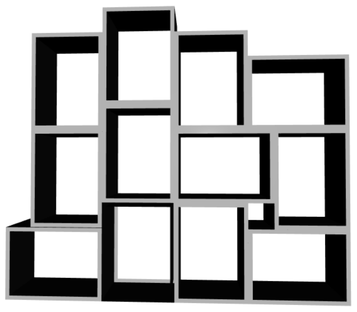
Figure 3.10: Bookshelf shape

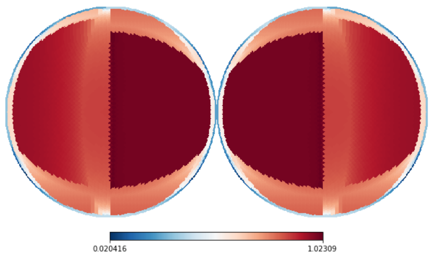

Figure 3.11: Back view (left) and front view (right) of the spherical signal of the bookshelf obtained by ray-casting the distance from the ray to the first intersection.

on the projected view from a certain perspective. These projected views can capture more information than a simple spherical projection.

Another possibility is to create a graph directly from the vertices of the mesh model. The problem is that the graph is not constant anymore and we do not know how the model will learn the filters. It would not be a signal on graph classification anymore, but more a graph classification task, where the goal is to identify the graph directly.

3.4 In-depth investigations

In this section, further investigations about the behavior of DeepSphere are done in order to understand better the previous results.

3.4.1 Features influence

As the model is clearly overfitting, many operations are tested in order to reduce the overfit. One of these operations is to reduce the number of features, keeping only the most important one, the distance feature. All the other features can be computed from this one only. One of the hypotheses is that too many features prevented the model to learn simpler correlation.

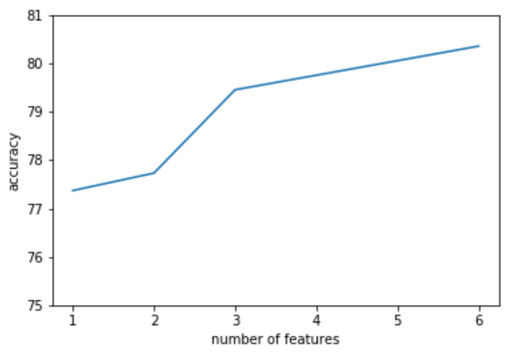

Figure 3.12: Accuracy (in percents) compared to the number of features

Keeping the fixed architecture *Optimal* and fixed number of epochs, the number of features changed, and the resulting accuracy is shown in Fig. 3.12. One can notice that most of the necessary information can be retrieved only by the first feature, the distance to the sphere. The influence of the remaining features is therefore small. However, this difference is non negligible, and the addition of the cosine information (feature 3) seems to help more than the others.

3.4.2 Sampling Density

To see where the important information needed for the neural network lies in the spectral domain, it is important to experiment with the sampling density. As the other models used only a low sampling density, it was assumed that it was due to their high computational cost. As DeepSphere is faster, sampling projections with higher density can more easily be used. The information given by a discrete sampling increases with its density, and similar to Shannon theorem, the maximum frequency that can be represented on the sampling scheme increases as well. But in the graph 3.13 one can see that the main information learned by the model lies in the low frequency. Indeed, even with a density of $N_{side} = 8$, the model performs well. It's only by going as low as $N_{side} = 2$ that it does not converge anymore. The maximum frequency (spherical harmonic order) l_{max} for HEALPix is given

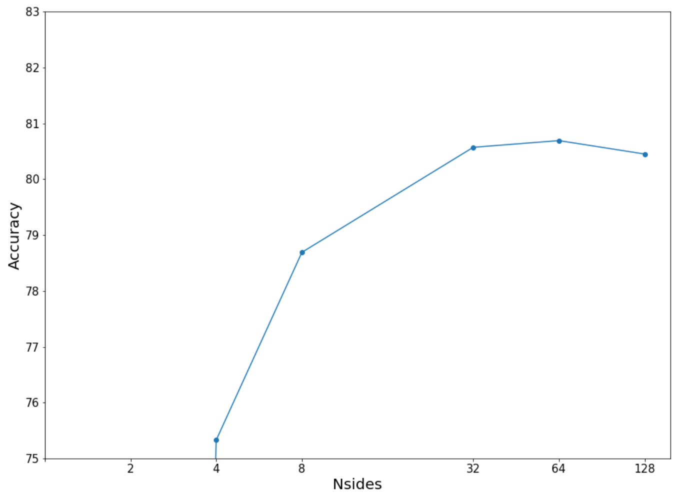

Figure 3.13: Accuracy (in percents) represented for different values of the sampling density N_{side}

by $3 * N_{side} - 1$, where discretized spherical harmonics are still linearly independent[9]. So the maximal frequency increases with the density N_{side}.

3.4.3 Rotation Equivariance

In order to see if the equivariance to rotation was sufficient for the given task, or if it was needed to learn the correct equivariance with data augmentation, another experimentation was done. For the same architecture, the model learned from two different datasets. The

[9] `https://healpix.sourceforge.io/pdf/facilities.pdf`

first was augmented with three random translations while the rotation is kept the same for every instance, resulting in the objects not being in the center of the sphere anymore. The second dataset was augmented with three random rotations. Thus, the two datasets have the same number of instances. It was then tested on the test sets processed with the same manner. The results can be seen in the Table 3.3.

A sampling density of $N_{side} = 32$ was chosen. It can be concluded that the model is

	NR/NR	R/NR	R/R	NR/R
Accuracy	79.57	79.26	79.25	79.82
mAP@N	67.1	67.0	67.5	67.4

Table 3.3: Performance of DeepSphere (accuracy and mean average precision in percent) in X/Y configuration, trained on X dataset while tested on Y dataset. R corresponds to rotation augmentation and NR corresponds to translation augmentation.

sufficiently equivariant to rotation for this task, as the results are similar, whichever the conditions.

3.4.4 Use of more equivariant graph

Following the work of Martino Milani, a more precise rotation equivariant graph has been constructed [6]. The experiments were rerun with the better HEALPix graphs and the results can be seen in Table 3.4. As seen in the section 3.4.3, the previous graph is sufficiently equivariant to rotation for this specific task, it is therefore no surprise to not to see any improvement with an improved graph. Indeed, in the first graph, the first orders of the spherical harmonics are well approximated, and higher orders seem to diverge more, so only high frequency signal would suffer from such a graph. In this task, it was shown experimentally earlier that the important frequencies are the lowest ones, and the model will not benefit from a better discrimination of the higher frequencies.

3.4.5 Preventing overfit

As shown earlier, the model overfits the data, and several methods are tested to reduce this effect. The resulting accuracy on the validation set is regrouped in Table 3.5. Without any method to prevent overfitting, the accuracy is 81.8%.

Regularization To prevent the apparition of a too complex model with overly large parameters, L2-regularization is used. In this specific case, a complex model is preferrable as the accuracy drops by 1 point.

Dropout Another classical method is to apply dropout on the fully-connected layer. This prevents this layer to become too specific. This method shows no clear improvement in the performance.

	old graph	new kernel size	new graph
accuracy	82.23	82.45	82.76

Table 3.4: Performance with better equivariance

DropFilter The same idea can be applied on the convolutional layers by dropping whole filters instead of neurons. Again, it is apparent that this method is not beneficial to the model. The layers are not trained long enough to become too specific.

Triplet Loss As explained earlier, an additional loss presented first by Esteves was added to increase the performance for the retrieval task. However, the gain is negligible.

Data augmentation Different data augmentation are tested. The first is the addition of a random Gaussian noise with $\sigma = 0.1$ to the spherical signal. This deviation is too big and the results remain stable. Another problem is that the data is too large too compute random noise on the fly, and only five noise configurations are computed before training.

Finally, the data was augmented using random translations and rotations. The rotation is not necessary, but reused from the section 3.4.3. After using 6 different configurations for each instance, the effects of overfitting are less present as the validation loss has a slower increase over time, and the final accuracy is slightly higher. It can be assumed that if more configurations were used, the performance would increase even more.

Summary No method prevents the overfitting in this case. Only the triplet loss and the data augmentation are able to reduce its effect. It may be due to the fact that the dataset is too small and a larger population is needed to find correct representations of shape descriptions.

	∅	Regularization	Dropout	DropFilter	Triplet Loss	Data aug.
Accuracy	81.8	80.7	82.4	81.5	82.5	83.4

Table 3.5: Different methods preventing overfit

3.5 Classification task - ModelNet40

3.5.1 Task presentation

The ModelNet40 dataset is very similar to the SHREC17 dataset, as it is a collection of 3D CAD models. The only differences are the number of classes and the total number of instances. There are only 40 common classes instead of 55, and about 80% less instances, making the classification task harder as it was shown that the SHREC17 dataset was not large enough for the model to grasp perfectly the shape descriptors.

Is it is a similar task as section 3.1.1, the preprocessing is kept the same, and there is no need to redo an intensive study of the task, and the decision was taken to reuse the same architecture. Additionally, publicated results on this tasks could be used for comparison with DeepSphere.

There are two different datasets, a manually aligned and a randomly rotated version. For the same reasons as for SHREC17 task, the perturbed dataset is chosen.

3.5.2 Other spherical models

The first spherical CNN to compare to is Esteves *sphericalcnn*, that is the same as for SHREC17 retrieval task. It is not specified which dataset is chosen, but it is more likely that the perturbed version has been used. However, in his latest study [35], the aligned version was used to compare more easily with other publications.

Again, equiangular is the sampling scheme chosen and augmentation was used to increase the performance and prove the equivariance to rotation. The model learns first on random rotation around z (euclidean coordinate), and is tested on random $SO(3)$ rotations. A drop in performance for this test can be seen. This was attributed to the problem of the equiangular sampling.

Jiang uses the aligned dataset because inherently it is not equivariant to rotation. Indeed, Jiang argued that, for classification problems, the orientation is important. For example with omnidirectional images with the gravity, and planetary signals with the axis of rotation [21]. Additionally, Cohen's *s2cnn* was used to have more comparison. But no augmentation was used and it was obviously not tested with random rotation disturbances.

3.5.3 Analysis

DeepSphere *Optimal* model is reused for the two configurations of the experiment: The first configuration being (x/x), with no augmentation during training, and the only difference being the non aligned dataset, explaining a little drop in performance. The second configuration includes augmentations only around the Z-axis (z) and with random $SO(3)$ rotations (SO3) for the test set. The notation (X/Y) simply shows that the model was trained on X and tested on Y. In this specific case, only 3 different random configurations are used for augmentation. The resulting accuracy can be observed in Table 3.6

Even if it was not tested, it is expected that Jiang would perform poorly with rotation perturbations, as it is not equivariant nor invariant to rotation. Otherwise, it outperforms other spherical CNNs with the aligned dataset.

	x/x	z/z	SO3/SO3	z/SO3
Cohen	85.0	-	-	-
Jiang	90.5	-	-	-
Esteves *scnn*	-	88.9	86.9	78.6
DeepSphere	87.8	86.8	86.7	76.9

Table 3.6: Accuracy (in percent) on test set, for different models. The x dataset has no augmentation, the z dataset is augmented with rotation around Z-axis, and SO3 with ZYZ rotations.

Again, similar results can be observed with only a drop of 1 point in accuracy, potentially due to the the isotropic filter. But the same drop in performance as Esteves when using SO3 test set can be observed, even using a more uniform sampling such as HEALPix. Thus the origin of this drop might be elsewhere than in the sampling. It is still possible that the sampling HEALPix, being iso-latitude, has a more faithful representation for rotation in Z (latitudes remain the same), but other possibilities must be explored.

Additionally, the behaviour is the same as SHREC17, where the model learns quickly (about 10 epochs) the shape descriptor and start overfitting.

A first hypothesis was that the equivariance to rotation is not good enough for this task. Thus, the evolution of the logits (probabilities to belong to each of the 40 classes) for different is observed. First, a discrete rotation of the sampling is made to prove that the theory is correct and the model is correctly implemented. A HEALPix map has discrete rotations around Z (ϕ) every 90°and around X or Y (θ) every 180°. Indeed, the empirical result show that these specific shifts of the map give the exact same result, so the model is exactly equivariant to discrete rotation and approximately equivariant for other rotation.

Next, random $SO(3)$ rotations are performed and the evolution of the logits are seen in Fig. 3.14. The x-axis represents the 40 classes, and the y-axis is the output of the softmax, the probability for the shape to belong to a specific class. A specific instance of a class with low accuracy is chosen (in this case, a 'desk'), and the model is trained on the dataset with Z-axis rotation augmentation. The blue dots correspond to the response of a model trained on original orientations, and the other three colors corresponds to models that learned with three $SO(3)$ random rotations.

When two classes have a high probability to be chosen, the small perturbations added by a non-discrete rotation are sufficient to change the final decision.

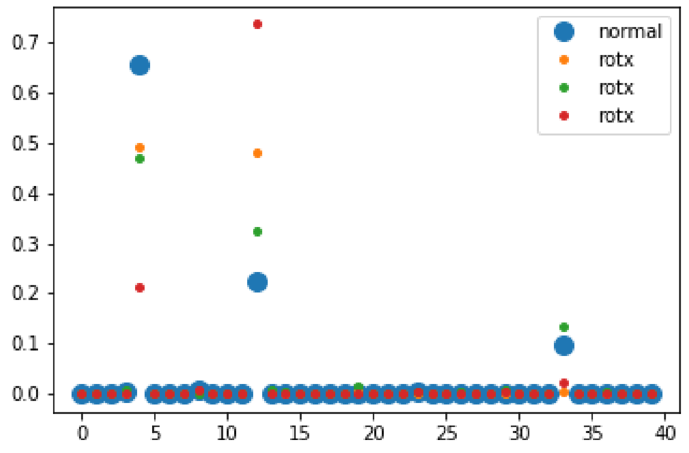

Figure 3.14: logits (probabilities to belong to each of the 40 classes) evolution for different rotation configuration for indistinguishable class

Upon further investigation, some 3D shapes are only surfaces and have no spherical projection. Adding the fact that some shapes have a poor spherical representation (see for example Fig. 3.11), the model cannot learn the correct descriptors to distinguish the classes.

Furthermore, some classes are indistinguishable with the level of information the model is able to grasp and accuracy will vary greatly between them. This can be seen in Fig. 3.15, where the confusion matrix of the Z test dataset is shown. Some very distinguishable classes such as 'airplane' and 'laptop' have a very high accuracy because they are very different from the rest of the classes. Even if the test is done with the SO3 dataset, the accuracy stays the same (see Fig. 3.16). But other classes such as 'plant', 'flower pot' and 'vase' are indistinguishable and have a poor accuracy (between 30 and 60 %) and testing on the SO3 dataset only worsens this accuracy (a decrease of 35 points can be observed). This shows that the model is still unsure how to classify correctly the shapes, and the very similar classes do not help in this case. Even trying to increase the training time for the model to be more confident does not change the behaviour (increasing the number of epochs by three).

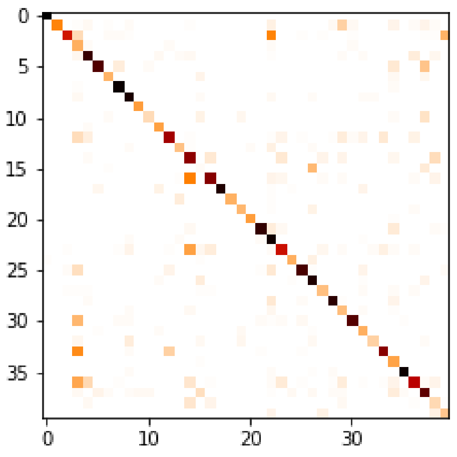

Figure 3.15: confusion matrix of test set

Thus, it is possible that there are not enough samples to learn the indistinguishable classes. Sampling density N_{side}, the number of features and the type of graph were changed to see the differences. However, there were no noteworthy differences observed. Even the more accurate graph in respect to the equivariance to rotation did not change the results.

The problem may be due to the spherical representation, where information is lost, and the dataset being too small for the model to comprehend the features correctly. Using augmentation (adding random $SO(3)$ rotations) during training, the drop in performance when testing on SO3 dataset could be reduced drastically, but the performance when testing on Z dataset is not increased.

3.6 Dense regression - GHCN-daily

3.6.1 Dataset presentation

The goal is to build an irregular, sparse and non-uniform graph around the world with every node corresponding to a weather station, and the graph signals to the different measurements. To be useful, the graph must be as large as possible. If the graph represents a too small region, the curvature of the Earth will not be visible and the resulting graph will not differ from a planar one, and it would make no sense to use a spherical CNN. An example of a graph using all the weather stations around the world is shown in Fig. 3.17.

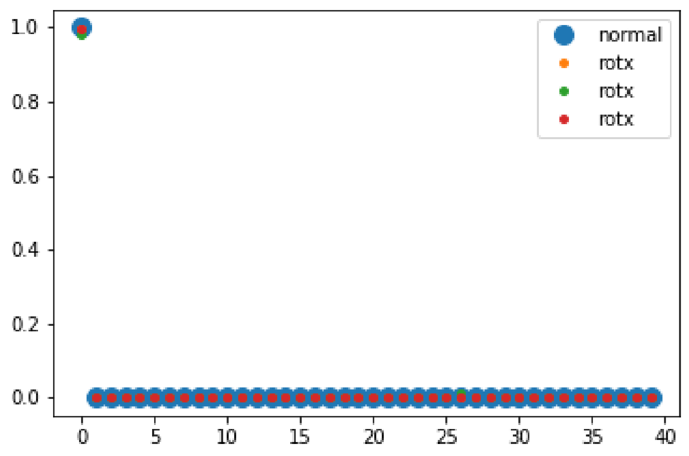

Figure 3.16: Logits evolution for different rotation configuration for distinguishable class. The invariance is perfect in this case

The data of interest for this study is chosen from 2010 to 2014, corresponding to 1826 days. The different measurements are described in the README of the dataset[10].

In all the cases, no irregular graph sampled densely over the continents can present the equivariance to rotation property. It is not the purpose of this part of the study, as were are interested in the flexibility of DeepSphere. In any case, the S^2 symmetry group does not seem present in this task. The weather is not the same between the poles and the equator, and there may be differences between oceans and continents. The only meaningful rotation might be Z due to the axis of rotation of the Earth.

Regarding the isotropic filter, it may not be a hindrance if one is only interested in finding different regions of specific weather (e.g. "blobs" of high precipitation).

For the current problem, only the most represented measurements (minimum and maximum temperatures, precipitation and snow fall) should be chosen to have the largest graph. Any other measurements are not represented enough over the globe (less than 10'000 weather stations during the interval of interest). On top of that, there are lots of abnormal data and impossible outliers, such as minimum temperature higher than maximum temperature or far outside possible ranges. Fortunately, the dataset provides a quality flag that lets one know if the data might be corrupted or has problems. Thus, any data that raised a quality flag was removed.

3.6.1.1 Find a task

This dataset was a perfect opportunity to prove the flexibility of DeepSphere, as the weather stations did not form a great regular sampling (higher concentration in some countries, and almost no sample points over oceans), and interpolation over the whole sphere does not make much sense because the data points are too far away from each other in less populated areas. At the time of the study, no one used this dataset and no specific task was

[10]`https://www1.ncdc.noaa.gov/pub/data/ghcn/daily/readme.txt`

obvious enough for a deep learning problem. So one of the first steps is to find a specific task to demonstrate the advantage of DeepSphere.

The main tasks considered are global classification (e.g. classify the weather on the whole globe such as find the season or era), global regression (e.g. prediction of a specific variable, such as time), segmentation or dense classification (e.g. find a class for each node, such as weather type) and dense regression (e.g. predict a specific variable from the other features).

Global tasks are not really useful nor complicated in this case therefore this possibility was immediately dropped. Dense classification and segmentation were also dropped because there were not enough labels and no information to apply ourselves the other labels, and a great coverage could not be constructed. Only dense regression tasks are doable realistically, using already known features as variables to predict.

As only few features are sufficiently present, only a few tasks were possible:

1. Find the maximum temperature from the minimum temperature
2. Find precipitation from temperature (max and min)
3. Find snowfall from temperature and precipitation
4. Find temperature at day T from N days before

The first task does not have a great meaning and is not really useful. For the second, although interesting, it does not yield compelling results as there does not seem to exist a great correlation between these features. The third yields similar results as the previous one. The fourth seems to be the most promising task, although we can only find statistical relations between different days, and not actually predict the future.

Different features were added in the hope of increasing the performance, such as the latitude and altitude of the weather station, and the time of the year (approximated by a sinus over the 366 days to represent the periodicity of the seasons). These additional features were added as there seems to be a correlation between the time of the year, the location of the place and the temperature. Indeed it is warmer in summer in the northern hemisphere for example.

To augment the data or have a larger graph, one idea was to take a superset of weather stations that had a majority of measurements (over 75% during the interval of interest) for the chosen features, and to inpaint (using regression Tikhonov[11] on the given graph) the missing data. These will be used for the learning and results will be masked for the computation of the loss and final response. The main problem is that the regression will give a smooth signal on the missing nodes, using the graph structure that is used for the learning afterwards. Basically, it is counterproductive and forces the model to learn a smooth response at the end.

Practically, a mask of the missing values is given as feature for the model so it knows where the real values are. The inpainted values are ignored for the back-propagation part of the learning (the loss of the masked values are forced to 0). Similarly, for the response of the model, the masked values return NaN.

[11] `https://github.com/epfl-lts2/pygsp/blob/master/pygsp/learning.py`

3.6.2 Implementation

A new graph must be constructed as the sampling is irregular and the HEALPix graph is useless in this case. The graph nodes correspond to the weather stations, represented by their latitude and longitude on Earth. To connect the nodes, a knn-graph[12] is used, with the number of neighbors for each node a hyperparameter. The first number of neighbours is fixed at 5 arbitrarily. Thus, the weight matrix is a standard heat diffusion kernel using the euclidean distance. There is an example of a graph using only USA weather stations in Figures 3.18 and 3.19. Figure 3.20 shows that the curvature of the Earth is significant even using only this country.

Each day corresponds to a new spherical signal represented by the chosen features.

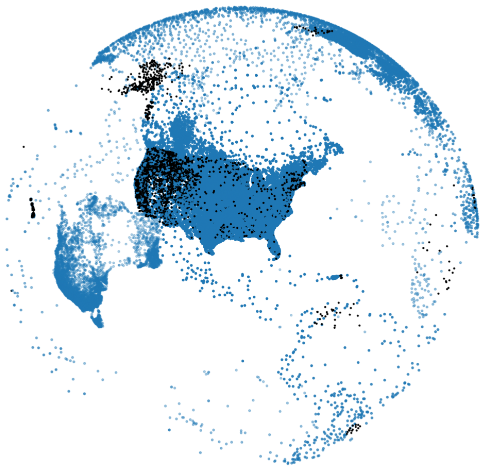

Figure 3.17: Graph of the world weather stations

For the architecture of DeepSphere, no pooling is used. Indeed, there is no rule for pooling a graph with an irregular, non-hierarchical sampling. And to keep the irregularity of the sampling, the combinatorial Laplacian of the graph is used.

An MSE loss is chosen for the regression task, with an RMSprop Optimizer (decay of 0.9 and momentum of 0) and a constant learning rate of 10^{-3}. The resulting architecture is:

$$NN = GC_1 \circ (ReLU \circ BN \circ GC_{100}) \circ (ReLU \circ BN \circ GC_{100}) \circ (ReLU \circ BN \circ GC_{50}).$$

The number of feature maps is arbitrarly chosen. The last one is the number of parameters to be found by the regression, so in this case just one. The order of the filters K is chosen to remain fixed for all layers, but considered as a hyperparameter to change too. A batch size of 64 is used due to the small graph and the model is run for 250 epochs.

The interval of time is separated in two parts. The training set corresponds to 70%, so three years and a half. The remaining 30% are the test set.

[12] `https://pygsp.readthedocs.io/en/stable/reference/graphs.html#pygsp.graphs.NNGraph`

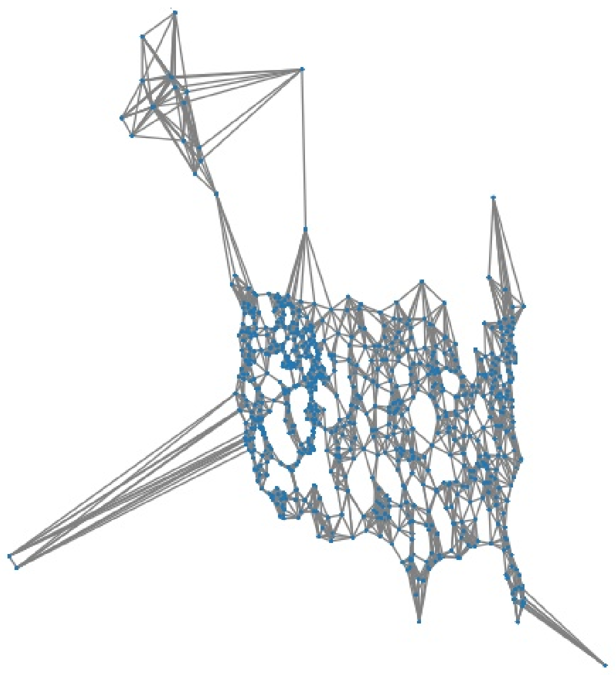

Figure 3.18: Front view of the graph for USA weather stations only

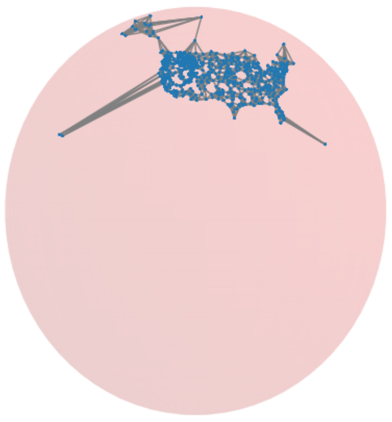

Figure 3.19: Graph of the USA weather stations represented on a spherical Earth

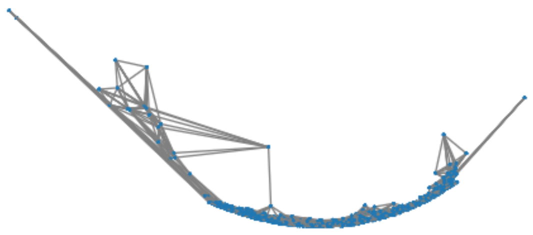

Figure 3.20: View showing the curvature of the Earth as derived from the graph for USA stations only

3.6.3 Analysis

The metrics chosen to estimate the performance of the model are again computed with *scikit-learn*. They are Mean Absolute Error, Mean Relative Error to observe if the predictions seems off, and R2 score to observe if the predictions are better than the mean.

A first thing to do is to see the evolution of the performance in respect to the number of neighbors in the knn-graph (N), and the order of the graph filter (K). All runs are trained for the same number of epochs. For a fixed K of 5 (and number of previous days of

5), the performance of future temperature prediction seems to increase with the number of neighbors (Tab. 3.7). A hypothesis is stated in the following section. The same operation was done with the precipitation task (Tab. 3.8).

neighbour	MSE	MAE	R2	batch time
1	9.55	2.27	0.90	0.33s
5	8.33	2.14	0.91	0.35s
10	7.97	2.09	0.92	0.46s

Table 3.7: Evolution of performance with different KNN-graph for the temperature task

neighbour	MSE	MAE	R2	batch time
1	41.58	2.87	0.11	0.22s
5	38.25	2.77	0.15	0.27s
10	43.86	3.07	0.12	0.32s

Table 3.8: Evolution of performance with different KNN-graph for the precipitation task

In conclusion, when the convolutions over a pixel use fare separated neighbors, the performance drops as there is no more significant influence outside of the direct neighborhood. Additionally, a denser graph will cause an increase in inference time for the same hyperparameters. Based on this, a N of 5 is chosen for the precipitation task and 10 for the future temperature task.

The filter order K will be further investigated in the future temperature prediction task.

3.6.3.1 Precipitation task

The range of temperatures is within [-53.3, 41.7] for the minimum and within [-47.2, 53.9] for the maximum. The range of precipitation is [0, 415] mm, with a mean of 2.3mm, showing that most of the data is close to 0mm.

The number of weather stations in the minSet, the set containing nodes having measurements for the whole interval of interest, is 1085 and span the USA and a part of Europe. The superSet, the set containing nodes that have at least 75% of measurements during the interval, has 9728 nodes.

But there is a problem with the inpainting problem. As all the signals are treated separately, some abnormal values appear. Indeed, as the different features such as minimum temperature and maximum temperature are separate and there is no temporal information during the Tikhonov regression, minimum temperature higher than maximum temperature, and changes over 30°in one day is observed. Thus such information must not be learned.

It can be seen in Fig. 3.21 that the model does not really perform better than random guessing, and that the outliers are generally ignored, resulting in predictions around the mean.

Without the additional features, so with only the temperature information, the model cannot find any relation, and the validation loss remains constant with some random spikes. So the model does not perform better than a random guess.

But even adding the new features does not improve the performance. The only change is that there are no more random spikes and the gradients are much smaller. Seeing the

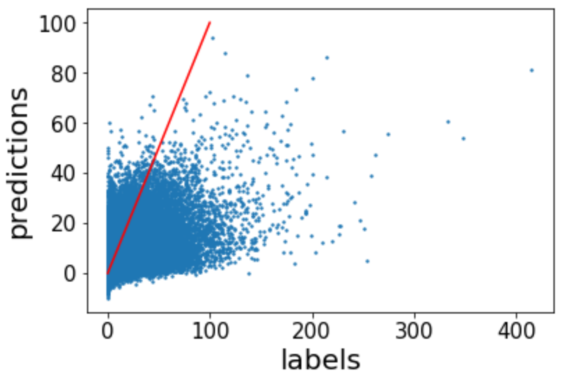

Figure 3.21: Predictions over true labels

results in Tab. 3.9, the mean absolute error is about 3 mm of precipitation and seems low in respect to the interval, but the mean relative error is too large, showing that there are only few big values that are ignored. And thus the model only predicts the mean, as specified by the r2 score near 0.

	MSE	MAE	MRE	R2
Base features	45.44	2.74	119	-0.04
Additional features	38.25	2.77	140	0.15

Table 3.9: Validation metrics for precipitation regression

The results for the snowfall task is also ignored as it is similar than the precipitation one, but with less data.

3.6.3.2 Future temperature task

The number of weather stations in the minSet, the set containing nodes having measurements for the whole interval of interest, is 1573 and span the USA, a part of Europe and few in Australia. The superSet, the set containing nodes that have at least 75% of measurements during the interval, has 11996 nodes.

As the goal of this task is to predict the temperature at day T, the chosen features for this task are n days [T-1, ..., T-n] before the day to predict, and the other additional features are the location and altitude of weather stations as well as the day of the year. No additional temporal information is added, but in future application it might be interesting to apply the CNN on the temporal axis as well. The minimum and maximum temperatures are considered to behave more or less the same way temporally, and are added as two different signals per day.

So the main hyperparameters are the number of days in the features (n), the number of neighbors (fixed at 10 for this experiment) and the order of the graph filter (the number of hops seen during the convolution).

A first hypothesis is that weather affects a whole region, so it is useful to take advantage of the locality of the data, and thus use the graph. But stations that are too much distanced won't influence each other. Thus there would be an optimal filter order where the model will not learn additional information and reduce its performance.

To prove it, a first baseline is built using the measurements from the weather stations as time series. Thus, no relation between the nodes can be constructed. Practically this corresponds to use a graph f of order 0.

It is observed that for a short time window, the temperature does not have a significant change. It is then interesting to observe if the model predicts a temperature closer to day T-1 instead of day T. This will be our second baseline.

Another interesting point is to fix the second hyperparameter, the number of previous days features for the model. The knn-graph is already constructed with the 5 nearest neighbors, and the order of the filter is still fixed at K=5.

ndays	MSE	MAE	R2	batch time
2	8.39	2.14	0.915	0.46s
5	8.13	2.11	0.917	0.47s
10	8.274	2.13	0.915	0.48s

Table 3.10: Evolution of performance for different number of days in feature

The performance seems to increase with the number of previous days, but the memory increases linearly and the change is not as significant as changing the graph or the order of the filter. Finally, a number of 5 days is chosen as a compromise.

order	MSE	MAE	MRE	R2
0	10.88	2.42	83.8	0.896
1	8.91	2.20	75.1	0.906
4	8.20	2.11	73.2	0.919
9	8.38	2.12	73.3	0.915

Table 3.11: Evolution of performance in respect of the order of the filter

order	MSE	MAE	MRE	R2
1	134.01	10.51	282.60	-0.319
4	129.48	10.31	283.29	-0.274
9	141.68	10.82	291.34	-0.389

Table 3.12: Predicting the present day (T-1) instead of the future day T

Now, the last interesting part is the influence of the order of the Tchebyshev filter. In this part, it is shown that the use of a graph is beneficial for the learning as the results (Tab. 3.11) are better than the baselines, and thus the structure of the data is helping.

The baseline, where the data are considered as simple time series during the training, is the order 0. The filter sees only the current node and no neighbouring nodes. The last test is to compute the metric with the present day instead of the future day (Table 3.12). This other baseline is used for proving that the model predicts effectively the next day and not a random similar day.

A limit can be seen where it is not beneficial anymore to use information from neighbors that are too far away. In our case, a drop of performance is observed when $K = 10$ and the peak performance is reached with $K = 5$. The final comparison shows that the model is able to predict more accurately the future than the present day, which proves that it has learned a statistical representation of the future day from previous ones.

One interesting result is that knowing how the rest of the world behaves helps the learning. In other words, our architecture is more successful than just using the time series from each weather stations separately.

Once all hyperparameters fixed, a quick ablation study of the additional features (location of nodes, and day in the year) is done, and shows that the only change is the rate of learning for the training set (1 point of MSE after 250 epochs), but the results for the testing set remains the same. Thus these features are useless and the correlation seen is grasped with the fundamental features.

Another test to do in order to know if the model grasped more or less the behaviour of the features is to only give it the first N days, feed only the predictions afterwards, and see if there is a divergence or a correct behaviour. Even if predicting the future is completely impossible and that a weather model is needed to predict anything correctly, it is possible to learn the statistical behaviour. And thus, the predictions should be oscillating between winter and summer, even without being near than correct. This is the case for the models learned.

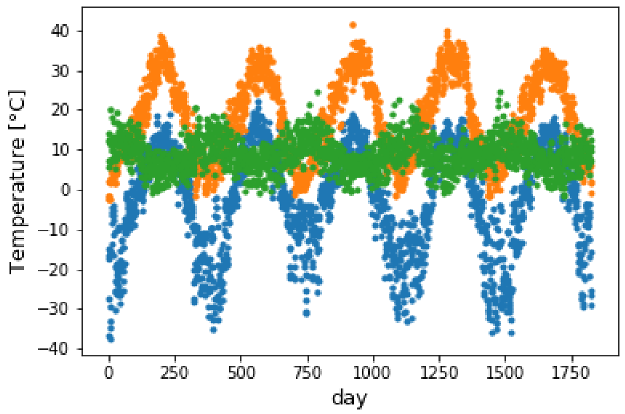

Figure 3.22: Evolution of temperature over time for three different stations

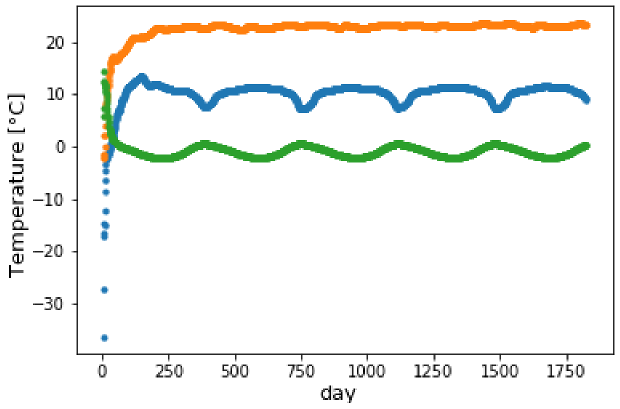

Figure 3.23: Complete prediction of temperature over time for three different stations

3.6.3.3 Global Regression - find time of year

Another possibility is to do a global regression and try to find the time, day, of the year. It is a nice addition to the other tasks, as it is different from the others and seems easier to do, and it is also a global regression task.

The input features are temperatures and precipitation, and the data is split the same way as before for training and test set. The same parameters and architecture as for the other tasks on this dataset are used, with the addition of a global average pooling at the end.

Using a linear representation for the day D of the year (from 0 to 365), the model poorly finds the correct result, with $\mathrm{MSE} = 1.57 \times 10^4, \mathrm{MAE} = 59.78, \mathrm{R2} = 0.4317$. The prediction can be observed in Fig. 3.24.

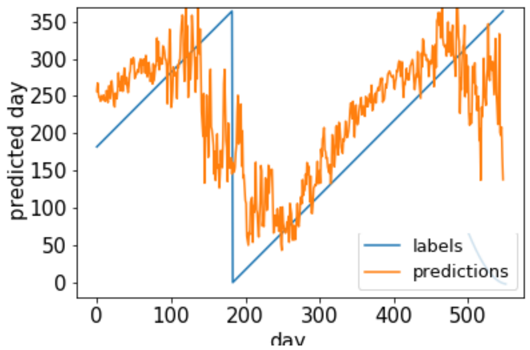

Figure 3.24: Global regression with linear representation

The predictions seem to have a smoother periodic response, so the representation may be at fault. Indeed, using Mean squared error loss and a non continuous function as label is not compatible. The parameters for day 0 and day 365 are very similar as they are both in winter, but the loss will be maximal if there is an error between them, while the true error is only very small.

Otherwise, using a cosine representation ($\text{day} = (\cos 2\pi \text{D}/366 + 1)/2$) produce better results, with $\text{MSE} = 0.05, \text{MAE} = 0.05, \text{R2} = 0.9686$. The prediction can be observed in Fig. 3.25. The problem with this representation is that it is impossible to differentiate Spring days from Autumn days.

Using only the polynomial of order 0, the results are $\text{MSE} = 0.10, \text{MAE} = 0.10, \text{R2} = 0.8817$. Again, the same trend is observed and the CNN performs better than learning only on the time series.

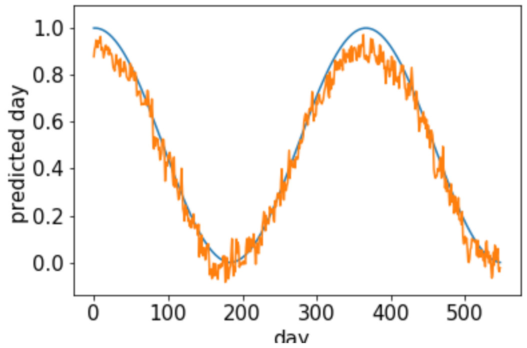

Figure 3.25: Global regression with cosine representation

As seen in Figure 3.22, the temperature is already a function of the time similar to a sine. To make the problem harder, only the precipitation feature is used for the next experiment, as its correlation with time is not easily understandable by a human (Fig. 3.26).

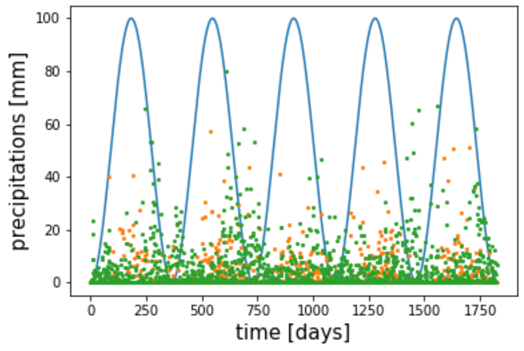

Figure 3.26: Precipitation for two stations
Blue cosine is the evolution of the years

The result is not as smooth as with the temperature as input feature, but the behaviour is still the same. With a filter of order 0, the model behaves poorly. The metrics are: $\text{MSE} = 0.58, \text{MAE} = 0.42, \text{R2} = -0.98$. Without any surprise at this point, having a convolution with an order K=5 increases the performance, and the metrics are better: $\text{MSE} = 0.50, \text{MAE} = 0.18, \text{R2} = 0.5971$ and the prediction is shown in Fig. 3.27. Even if the result is better than the performance of the filter of order 0, it is still far from ideal and cannot be called a complete success.

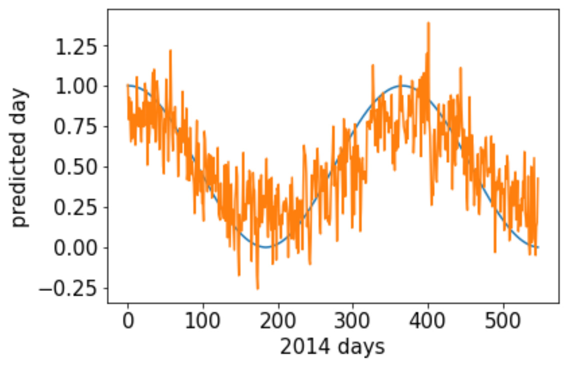

Figure 3.27: Global regression using only precipitation feature

3.6.4 Further improvement

One possibility to improve the results, even if there is little hope to do that, and an interesting thing to do, is to add pooling, and use an auto-encoder decoder architecture. An exploration of different methods for irregular graph pooling is in appendix B.

Another possibility is to perform convolution on the time axis. In practice, the graphs can be multiplied by a Kronecker product for each day.

Another interesting approach that has not yet been tried, is to use only station related and time related features, and do the regression to find the measurements. The model will then learn descriptors for the weather stations and find relations between them such as climate type.

3.7 Climate event detection - ExtremeWeather

At the time of writing, the experiments were still running, so no results can be presented.

3.7.1 Task presentation

The ExtremeWeather dataset [39] is composed of 4 measurements per day, 365 days per year (no extra measurement for leap years). All measurements have 16 different channels corresponding to a different feature, and the world is represented by an equiangular grid of 768 by 1152 pixels, which results in a total of approximately 885k pixels.

For their climate pattern segmentation problem, Jiang and Cohen used a tessellation of an icosahedron of order 5, having $10 \cdot 4^5 + 2 = 10'242$ pixels using bilinear interpolation. Inspired by them, a HEALPix map is used using bilinear interpolation on the latitude, longitude grid, with a density N_{side} of 32.

Concerning the labels, only the bounding boxes are given. Before creating the groundtruth accurately, all pixels inside bounding box is considered belonging to the corresponding class.

3.7.2 Implementation

As an HEALPix map is used, average pooling can be used, and an architecture similar to U-NET[44] is built.

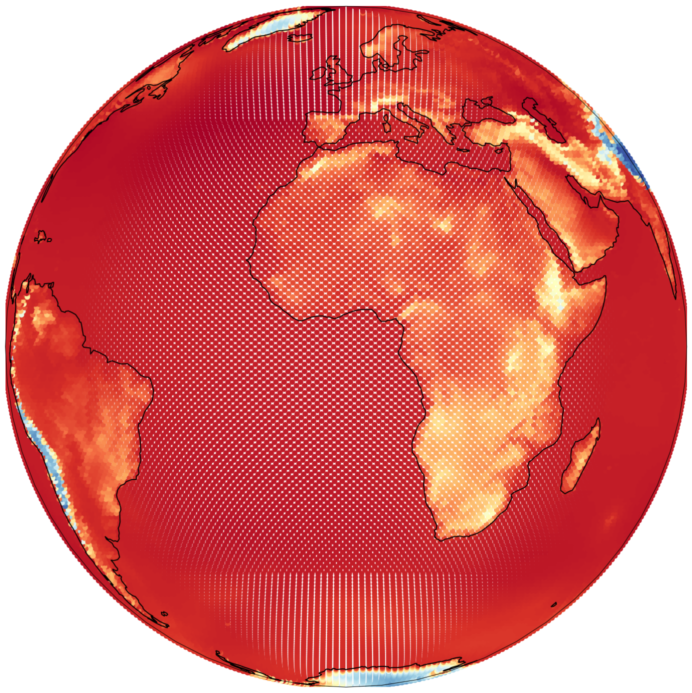

Figure 3.28: Bilinear interpolation of Atmospheric Pressure at sea level at $N_{side} = 32$

The deconvolution, the transpose convolution, is exactly the same graph convolution used by DeepSphere. Additionally, it is easier to use average pooling than max pooling in order to do the unpooling.

3.8 Climate pattern segmentation

This section is still incomplete as the dataset is still not public at the time of the writing of the report. This will hopefully be completed at a later time.

3.8.1 Task presentation

Other spherical CNNs have already run this experiment but with another sampling, based on the icosahedron. The sCNNs are made by Jiang [21] and Cohen [20].

4. Conclusion

4.1 Synthesis

In this section, the conclusions from all the tasks will be summed up in order to conclude on the differences and potential advantages of DeepSphere. In other words, we analyze how well our model competes against other state-of-the-art spherical CNNs.

Concerning the speed of the computation, and thus of the learning time, it was principally measured in the section 3.3, during the shape retrieval task. Thus, DeepSphere runs 4 to 40 times faster than other spherical CNNs using Fourier transform to reach peak performance, and an inference time 10 times faster with a small local filter.
Furthermore, SHFT-SCNNs are not able to run the learning of the cosmological maps classification in an acceptable time (section 3.2).

Furthermore, the flexibility of DeepSphere is proven with the regression task in section 3.6. Using an irregular sampling of the sphere is not doable by spherical CNNs using Fourier transform or planar projection, unless one interpolates it into another sampling. Additionally, it is shown that having an idea of the locality increases the performance, rather than working only on the time series of each nodes. Thus using a CNN makes indeed more sense.

Regarding the shape retrieval and shape classification task, in sections 3.3 and 3.5, the obtained results were similar to the results of the other models, even though lower for ModelNet40. Thus it can be safely stated that equivariance to the third rotation of $SO(3)$ is an unnecessary price to pay.

Concerning the approximation of the spherical harmonics with the eigenvectors of the Laplacian of the HEALPix graph, leading to an almost equivariant to rotation convolution, it is shown during the SHREC17 task that using a theoretically better graph does not lead to better results. This is mostly due to the fact that the important information lies in the low frequency. And this explains why the model still performs well with a poor sampling resolution ($N_{side} = 8$). Again, using a better graph did not lead to a better result for the ModelNet40 task. Thus, it can be safely stated that the model is sufficiently equivariant to rotation for the considered tasks.
For the regression task, and for the not experimented segmentation task, there is no indication that equivariance to rotation helps.

The other tasks not presented (segmentation and object detection on weather data) were chosen to show different uses of DeepSphere and illustrate the multiple applications it can have.

Finally, DeepSphere is not able to outperform the state of the art on established tasks, but can reach similar results in a shorter time.

4.2 Future Work

As there are some tasks still in suspense, the first thing to do is to complete them. Concerned tasks are (1) the climate pattern segmentation in section 3.8 and (2) the climate event detection in section 3.7.

Another task that would have been nice to have is a scene classification or segmentation, to have another point of comparison with [20] and [21]. That would also allow to work on a different kind of data, omnidirectional images, where equivariance to rotation is not necessary a good thing to have, like an isotropic filter. Indeed, some objects in these images are often linked with another one in a fixed configuration, and an anisotropic filter may capture this information.

Again, an intensive analysis of speed versus performance of different configurations of DeepSphere would have been great, and should be done in a future work.

Finally, going out of the scope of this study, other symmetries could be explored and easily reached with another graph. For example, in the case of omnidirectional imaging, and more specifically, panoramas, the images are always oriented with the gravity.
On another note, in this example, a cylinder could be a better graph, as the axis of the cylinder corresponds to the gravity's orientation, and the angle to the rotation of the panoramic point of view.
It would be possible to create a dataset from panoramas of cameras, having only a 360°(or less) on view with the same pitch, and could be use as a proof of concept.

Acknowledgements

I would like to thank Carlos Esteves for helping understanding his code and providing assistance for the SHREC17 task, and Chiyu "Max" Jiang and Steve Farrell for providing access to climate data for the climate pattern segmentation task. Finally, I would like to thank Michaël Defferard and Nathanaël Parraudin for their help and guidance throughout this project, as well as Martino Milani, Charles Gallay and Luca Zampieri with whom I enjoyed worked with and whose valuable advice and insight I appreciated.

Bibliography

[1] Taco S. Cohen et al. “Spherical CNNs”. In: *arXiv:1801.10130 [cs, stat]* (Jan. 30, 2018). arXiv: `1801.10130`. URL: `http://arxiv.org/abs/1801.10130` (visited on 02/13/2019).

[2] Carlos Esteves et al. “Learning SO(3) Equivariant Representations with Spherical CNNs”. In: *arXiv:1711.06721 [cs]* (Nov. 17, 2017). arXiv: `1711.06721`. URL: `http://arxiv.org/abs/1711.06721` (visited on 02/19/2019).

[3] Y-Lan Boureau, Jean Ponce, and Yann LeCun. “A Theoretical Analysis of Feature Pooling in Visual Recognition”. In: *Proceedings of the 27th International Conference on International Conference on Machine Learning* (2010), pp. 111–118.

[4] Nathanaël Perraudin et al. “DeepSphere: Efficient spherical Convolutional Neural Network with HEALPix sampling for cosmological applications”. In: *arXiv:1810.12186 [astro-ph]* (Oct. 29, 2018). arXiv: `1810.12186`. URL: `http://arxiv.org/abs/1810.12186` (visited on 02/13/2019).

[5] K. M. Gorski et al. “HEALPix – a Framework for High Resolution Discretization, and Fast Analysis of Data Distributed on the Sphere”. In: *The Astrophysical Journal* 622.2 (Apr. 2005), pp. 759–771. ISSN: 0004-637X, 1538-4357. DOI: `10.1086/427976`. arXiv: `astro-ph/0409513`. URL: `http://arxiv.org/abs/astro-ph/0409513` (visited on 06/13/2019).

[6] Martino Milani et al. *About the spectrum of the Discrete Laplace-Beltrami operator on the Sphere for Rotation Equivariant Neural Networks*, p. 50.

[7] Joan Bruna et al. “Spectral Networks and Locally Connected Networks on Graphs”. In: *arXiv:1312.6203 [cs]* (Dec. 20, 2013). arXiv: `1312.6203`. URL: `http://arxiv.org/abs/1312.6203` (visited on 04/16/2019).

[8] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. “Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering”. In: *arXiv:1606.09375 [cs, stat]* (June 30, 2016). arXiv: `1606.09375`. URL: `http://arxiv.org/abs/1606.09375` (visited on 02/19/2019).

[9] Taco Cohen et al. “Convolutional Networks for Spherical Signals”. In: *arXiv:1709.04893 [cs]* (Sept. 14, 2017). arXiv: `1709.04893`. URL: `http://arxiv.org/abs/1709.04893` (visited on 04/12/2019).

[10] J. R. Driscoll and D. M. Healy. “Computing Fourier Transforms and Convolutions on the 2-Sphere”. In: *Advances in Applied Mathematics* 15.2 (June 1, 1994), pp. 202–250. ISSN: 0196-8858. DOI: `10.1006/aama.1994.1008`. URL: `http://www.sciencedirect.com/science/article/pii/S0196885884710086` (visited on 05/27/2019).

[11] D.M. Healy et al. “FFTs for the 2-Sphere-Improvements and Variations”. In: *Journal of Fourier Analysis and Applications* 9.4 (July 1, 2003), pp. 341–385. ISSN: 1069-5869, 1531-5851. DOI: `10.1007/s00041-003-0018-9`. URL: `http://link.springer.com/10.1007/s00041-003-0018-9` (visited on 03/27/2019).

[12] Jens Keiner and Daniel Potts. "Fast evaluation of quadrature formulae on the sphere". In: *Mathematics of Computation* 77.261 (Jan. 1, 2008), pp. 397–419. ISSN: 0025-5718, 1088-6842. DOI: `10.1090/S0025-5718-07-02029-7`. URL: `http://www.ams.org/journal-getitem?pii=S0025-5718-07-02029-7` (visited on 05/27/2019).

[13] John R. Baumgardner and Paul O. Frederickson. "Icosahedral Discretization of the Two-Sphere". In: *SIAM Journal on Numerical Analysis* 22.6 (Dec. 1985), pp. 1107–1115. ISSN: 0036-1429, 1095-7170. DOI: `10.1137/0722066`. URL: `http://epubs.siam.org/doi/10.1137/0722066` (visited on 06/13/2019).

[14] C. Ronchi, R. Iacono, and P.S. Paolucci. "The "Cubed Sphere": A New Method for the Solution of Partial Differential Equations in Spherical Geometry". In: *Journal of Computational Physics* 124.1 (Mar. 1996), pp. 93–114. ISSN: 00219991. DOI: `10.1006/jcph.1996.0047`. URL: `http://linkinghub.elsevier.com/retrieve/pii/S0021999196900479` (visited on 04/29/2019).

[15] U. Elahi, Z. Khalid, and R. A. Kennedy. "Comparative analysis of geometrical properties of sampling schemes on the sphere". In: *2016 10th International Conference on Signal Processing and Communication Systems (ICSPCS)*. 2016 10th International Conference on Signal Processing and Communication Systems (ICSPCS). Dec. 2016, pp. 1–7. DOI: `10.1109/ICSPCS.2016.7843316`.

[16] Yu-Chuan Su and Kristen Grauman. "Learning Spherical Convolution for Fast Features from 360 Imagery". In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., 2017, pp. 529–539. URL: `http://papers.nips.cc/paper/6656-learning-spherical-convolution-for-fast-features-from-360-imagery.pdf` (visited on 04/12/2019).

[17] Benjamin Coors, Alexandru Paul Condurache, and Andreas Geiger. "SphereNet: Learning Spherical Representations for Detection and Classification in Omnidirectional Images". In: *Computer Vision – ECCV 2018*. Ed. by Vittorio Ferrari et al. Vol. 11213. Cham: Springer International Publishing, 2018, pp. 525–541. ISBN: 978-3-030-01239-7 978-3-030-01240-3. DOI: `10.1007/978-3-030-01240-3_32`. URL: `http://link.springer.com/10.1007/978-3-030-01240-3_32` (visited on 05/27/2019).

[18] Wouter Boomsma and Jes Frellsen. "Spherical convolutions and their application in molecular modelling". In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., 2017, pp. 3433–3443. URL: `http://papers.nips.cc/paper/6935-spherical-convolutions-and-their-application-in-molecular-modelling.pdf` (visited on 04/12/2019).

[19] Yeonkun Lee et al. "SpherePHD: Applying CNNs on a Spherical PolyHeDron Representation of 360 degree Images". In: *arXiv:1811.08196 [cs]* (Nov. 20, 2018). arXiv: `1811.08196`. URL: `http://arxiv.org/abs/1811.08196` (visited on 05/27/2019).

[20] Taco S. Cohen et al. "Gauge Equivariant Convolutional Networks and the Icosahedral CNN". In: *arXiv:1902.04615 [cs, stat]* (Feb. 11, 2019). arXiv: `1902.04615`. URL: `http://arxiv.org/abs/1902.04615` (visited on 04/03/2019).

[21] Chiyu "Max" Jiang et al. "Spherical CNNs on Unstructured Grids". In: *arXiv:1901.02039 [cs]* (Jan. 7, 2019). arXiv: `1901.02039`. URL: `http://arxiv.org/abs/1901.02039` (visited on 05/25/2019).

[22] Nicoletta Krachmalnicoff and Maurizio Tomasi. "Convolutional Neural Networks on the HEALPix sphere: a pixel-based algorithm and its application to CMB data analysis". In: *arXiv:1902.04083 [astro-ph]* (Feb. 11, 2019). arXiv: `1902.04083`. URL: `http://arxiv.org/abs/1902.04083` (visited on 06/15/2019).

[23] Taco S. Cohen and Max Welling. "Group Equivariant Convolutional Networks". In: *arXiv:1602.07576 [cs, stat]* (Feb. 24, 2016). arXiv: `1602.07576`. URL: `http://arxiv.org/abs/1602.07576` (visited on 06/04/2019).

[24] Taco S. Cohen, Mario Geiger, and Maurice Weiler. "Intertwiners between Induced Representations (with Applications to the Theory of Equivariant Neural Networks)". In: *arXiv:1803.10743 [cs, stat]* (Mar. 28, 2018). arXiv: `1803.10743`. URL: `http://arxiv.org/abs/1803.10743` (visited on 02/19/2019).

[25] Risi Kondor, Zhen Lin, and Shubhendu Trivedi. "Clebsch– Gordan Nets: a Fully Fourier Space Spherical Convolutional Neural Network". In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio et al. Curran Associates, Inc., 2018, pp. 10117–10126. URL: `http://papers.nips.cc/paper/8215-clebschgordan-nets-a-fully-fourier-space-spherical-convolutional-neural-network.pdf` (visited on 06/12/2019).

[26] Xinyang Feng et al. "Discriminative analysis of the human cortex using spherical CNNs - a study on Alzheimer's disease diagnosis". In: *arXiv:1812.07749 [cs]* (Dec. 18, 2018). arXiv: `1812.07749`. URL: `http://arxiv.org/abs/1812.07749` (visited on 06/15/2019).

[27] Maurice Weiler et al. "3D Steerable CNNs: Learning Rotationally Equivariant Features in Volumetric Data". In: *arXiv:1807.02547 [cs, stat]* (July 6, 2018). arXiv: `1807.02547`. URL: `http://arxiv.org/abs/1807.02547` (visited on 02/25/2019).

[28] Daniel Worrall and Gabriel Brostow. "CubeNet: Equivariance to 3D Rotation and Translation". In: *arXiv:1804.04458 [cs, stat]* (Apr. 12, 2018). arXiv: `1804.04458`. URL: `http://arxiv.org/abs/1804.04458` (visited on 06/15/2019).

[29] Jonathan Masci et al. "Geodesic convolutional neural networks on Riemannian manifolds". In: *arXiv:1501.06297 [cs]* (Jan. 26, 2015). arXiv: `1501.06297`. URL: `http://arxiv.org/abs/1501.06297` (visited on 06/16/2019).

[30] Federico Monti et al. "Geometric deep learning on graphs and manifolds using mixture model CNNs". In: *arXiv:1611.08402 [cs]* (Nov. 25, 2016). arXiv: `1611.08402`. URL: `http://arxiv.org/abs/1611.08402` (visited on 06/03/2019).

[31] Michael M. Bronstein et al. "Geometric deep learning: going beyond Euclidean data". In: *IEEE Signal Processing Magazine* 34.4 (July 2017), pp. 18–42. ISSN: 1053-5888. DOI: `10.1109/MSP.2017.2693418`. arXiv: `1611.08097`. URL: `http://arxiv.org/abs/1611.08097` (visited on 06/16/2019).

[32] Renata Khasanova and Pascal Frossard. "Geometry Aware Convolutional Filters for Omnidirectional Images Representation". In: (), p. 9.

[33] Pascal Frossard and Renata Khasanova. "Graph-Based Classification of Omnidirectional Images". In: *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*. 2017 IEEE International Conference on Computer Vision Workshop (ICCVW). Venice, Italy: IEEE, Oct. 2017, pp. 860–869. ISBN: 978-1-5386-1034-3. DOI: `10.1109/ICCVW.2017.106`. URL: `http://ieeexplore.ieee.org/document/8265315/` (visited on 04/12/2019).

[34] Jan Eric Lenssen, Matthias Fey, and Pascal Libuschewski. "Group Equivariant Capsule Networks". In: *arXiv:1806.05086 [cs]* (June 13, 2018). arXiv: `1806.05086`. URL: `http://arxiv.org/abs/1806.05086` (visited on 06/15/2019).

[35] Carlos Esteves et al. "Equivariant Multi-View Networks". In: *arXiv:1904.00993 [cs]* (Apr. 1, 2019). arXiv: `1904.00993`. URL: `http://arxiv.org/abs/1904.00993` (visited on 04/03/2019).

[36] *Matterport3D: Learning from RGB-D Data in Indoor Environments*. URL: `https://niessner.github.io/Matterport/` (visited on 04/16/2019).

[37] Iro Armeni et al. "Joint 2D-3D-Semantic Data for Indoor Scene Understanding". In: (), p. 9.

[38] *Global Historical Climate Network Daily - Description | National Centers for Environmental Information (NCEI) formerly known as National Climatic Data Center (NCDC)*. URL: `https://www.ncdc.noaa.gov/ghcn-daily-description` (visited on 06/04/2019).

[39] Evan Racah et al. "ExtremeWeather: A large-scale climate dataset for semi-supervised detection, localization, and understanding of extreme weather events". In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., 2017, pp. 3402–3413. URL: `http://papers.nips.cc/paper/6932-extremeweather-a-large-scale-climate-dataset-for-semi-supervised-detection-localization-and-understanding-of-extreme-weather-events.pdf` (visited on 06/06/2019).

[40] Mayur Mudigonda et al. "Segmenting and Tracking Extreme Climate Events using Neural Networks". In: (), p. 5.

[41] *SHREC 2017: Large-scale 3D Shape Retrieval from ShapeNet Core55*. URL: `https://shapenet.cs.stanford.edu/shrec17/` (visited on 04/25/2019).

[42] Zhirong Wu et al. "3D ShapeNets: A deep representation for volumetric shapes". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Boston, MA, USA: IEEE, June 2015, pp. 1912–1920. ISBN: 978-1-4673-6964-0. DOI: `10.1109/CVPR.2015.7298801`. URL: `http://ieeexplore.ieee.org/document/7298801/` (visited on 06/06/2019).

[43] Richard B Neale et al. "Description of the NCAR Community Atmosphere Model (CAM 5.0)". In: (), p. 289.

[44] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *arXiv:1505.04597 [cs]* (May 18, 2015). arXiv: `1505.04597`. URL: `http://arxiv.org/abs/1505.04597` (visited on 06/04/2019).

A. Miscellaneous

Improving DeepSphere implementation

The original version of DeepSphere was implemented with python using TensorFlow[1] (TF). As it was using an old way to transfer the data to the GPU, some modifications were necessary for the experiments of this project. Indeed, the shape retrieval task deals with a great amount of data, and the whole dataset cannot be loaded in the RAM at the same time, especially when augmentation is used and different perturbed versions of the same instance exist. The old way is refered as "Dataset load on RAM" as it consisted in loading the whole data directly on the RAM, and transferring it to the GPU batch by batch.

The code for the experiment is based on Cohen's work[2], and was using npy files[3] to store the preprocessed data. The intermediate solution consisting in reading the files when needed is called "Dataset loaded on the fly".
To reduce the unwanted idle time between each batch, the TF dataset pipeline is used.
It was only later discovered that the dataset pipeline handling is optimized with the use of TFrecords files to store the preprocessed data. The amount of work necessary to rebuild the dataset was deemed too much and the improvements unnecessary.

The performance of each setting is tested on the architecture *Cohen-like*, with 20 epochs and an evaluation of a validation set using the method "Dataset loaded on RAM" every 0.5 epoch.

	Time per batch	CPU time	Wall time	Load average
Dataset loaded on RAM	0.12 s	1598 s	2693 s	80%
Dataset loaded on the fly	0.11 + 0.06 s (loading time)	2541 s	3667 s	50%
TF Dataset pipeline	0.09 s	2400 s	2046 s	90%

Inference time analysis

A quick analysis of the inference time of the architecture *Cohen-like* is performed on the shape retrieval task in order to see the influence of the HEALPix graph (in function of the N_{side} parameter) and the size of the filter K. The batch size is 32.

[1] `https://www.tensorflow.org/`
[2] `https://github.com/jonas-koehler/s2cnn/tree/master/examples/shrec17`
[3] `https://www.numpy.org/devdocs/reference/generated/numpy.lib.format.html`

N_{side}	32	64	128
Batch time [s]	0.07	0.26	1.15

K	2	4	5
Batch time [s]	0.11	0.19	0.26

Table A.1: Evolution of inference time

The evolution of the time is then linear in function of K and the number of pixels $(12 \cdot N_{side}^2$

B. Pooling non-uniform graph

One of the problem with a graph built from a non-uniform sampling is that the pooling operation is non-trivial since the sampling is not hierarchical.

Thus, different pooling methods are explored for this graph.

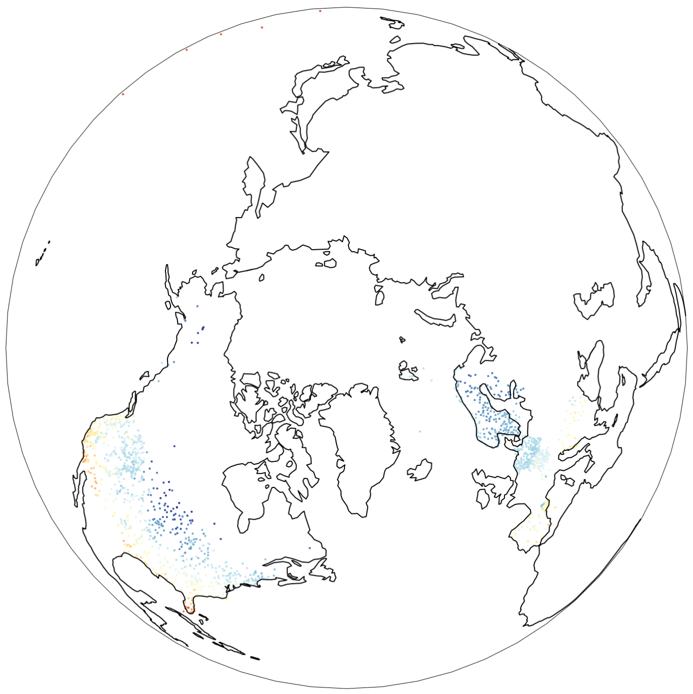

Figure B.1: Original signal on non-uniform sampling
low values are represented in blue and high values in red

A first one consists to using a hierarchical sampling with less pixels and interpolating the signals. An example is made from the popular HEALPix. A N_{side} is chosen such that the number of pixels is inferior than the number of nodes from the graph. The four HEALPix pixels closest to each node are chosen, and their value is computed from the original pixel.

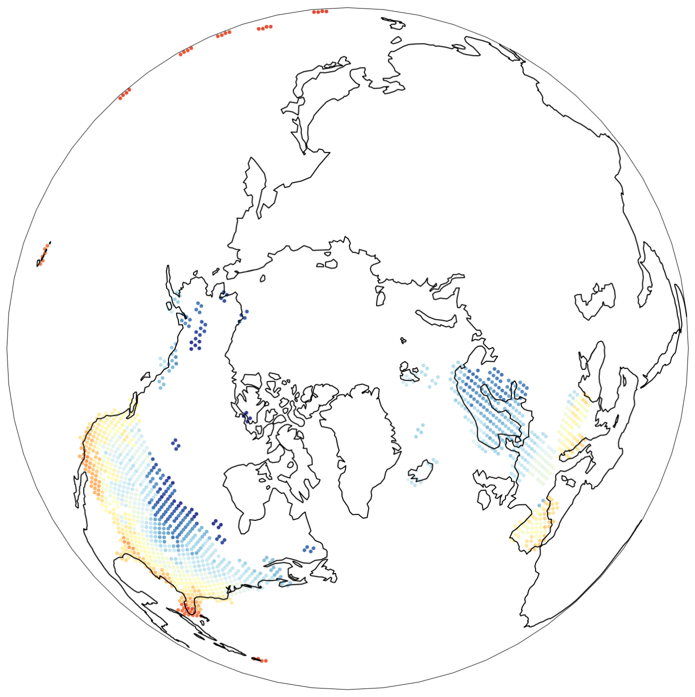

Figure B.2: Pooled signal to HEALPix sampling

A second one consists to carry out a spectral clustering of the graph using a number of cluster corresponding to the desired number of pixels after the pooling operation. The goal of this method is to keep the pseudo-structure of the previous graph.

Practically, KMeans is applied on the eigenvectors of the combinatorial Laplacian to identify the clusters, and new pixels are the center of these clusters.

The main problem with this method is the treatment of far isolated pixels. If these kind of pixels is clustered with other pixels far from it, the resulting cluster will hold no sense.

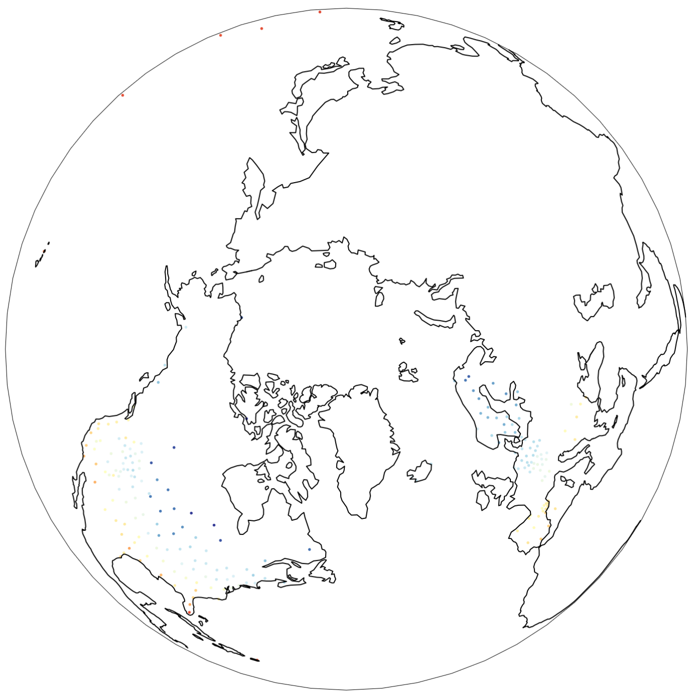

Figure B.3: Pooled signal to clusters

Finally, a more general method will be to sparsify the graph, thus removing nodes that are not important. The methods implemented on PyGSP[1] are tried, but do not give satisfying results as the resulting graph loses its coordinates.

As it was just an exploration of possible methods, no implementation in TensorFlow was done to exploit them.

[1]`https://pygsp.readthedocs.io/en/stable/reference/reduction.html`