Imperial College London

MEng Individual Project

Imperial College London

Department of Computing

Geometric Deep Learning for Subcortical Brain Shape Analysis

Author:
Jedrzej Blaszyk

Supervisor:
Dr Ben Glocker

Second Marker:
Dr Wenjia Bai

March 6, 2023

Abstract

Magnetic Resonance Imaging (MRI) is one of the leading methods for brain analysis, as it helps clinicians to image the structure, function, and pharmacology of the nervous system. Geometric Deep Learning (GDL), a novel field in machine learning, has emerged to generalize deep learning models to non-Euclidean domains, such as 3D shapes. In our work, we survey recent developments in GDL and benchmark existing convolutional operators on classical brain modelling tasks such as biological sex prediction and age regression. Moreover, we propose a novel convolutional operator, LocalEdgeConv, which guarantees invariance to rotations and translations of the 3D shape, thus avoiding the need for image registration. We train and evaluate our models on the UK Biobank dataset (14,503 brain scans), then we test the robustness of our approach on the Cam-CAN dataset (652 brain scans). We achieve competitive results, which substantially outperform the base-line brain analysis methods, and show that GDL models are robust and can generalize well. Furthermore, we benchmark the LocalEdgeConv operator on the ModelNet40 dataset and show that, in certain scenarios, it significantly outperforms state-of-the-art methods. Our results are not constrained to medical settings and can be easily applied to any shape classification tasks.

Acknowledgements

I would like to thank my supervisors, Dr Ben Glocker and Dr Loic Le Folgoc, for their guidance and support.

Further, I would like to thank my brother Piotr and my friend Eduardo for proof-reading the report.

Lastly, I would like to thank my parents and all my friends for giving me advice throughout the project and always supporting me.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

From the ancient Egyptian mummifications to 18th-century scientific research on "globules" and neurons, there is evidence of neuroscience practice throughout the early periods of history. It took several thousand years for the human race to learn that the seat of intelligence in humans is the brain and not the heart. Only in the advent of the Renaissance, did doctors realise the importance of the human brain. Since then, the scientific world has been able to progressively deepen its research. It's now common knowledge that the subcortical brain regions have a pivotal role in the cognitive, affective, and social functions in humans.

Neuroimaging techniques, such as MRI scans, are leading methods for the analysis of the brain. These methods help modern doctors to image the structure, function, and pharmacology of the nervous system. MRI came to dominate brain mapping due to its low invasiveness, lack of radiation exposure, and relatively wide availability. Clinicians can use brain scans to perform a neurological examination or to locate cancerous tissue growths causing lesions.

The advent of machine learning (ML) and computer vision allowed healthcare to benefit substantially. Nowadays, ML models often help clinicians with brain scan analysis, as they match or even surpass the human-level performance [29].

Although the existing, conventional models already excel at various benchmarks, we believe that we should never cease experimenting with novel approaches to the brain analysis problem. One of such new fields, Geometric Deep Learning, is the main focus of our work.

1.1 Objectives

Geometric deep learning (GDL), a term first proposed by Bronstein et al. [7], has emerged aiming to generalize deep learning models to non-Euclidean domains. These novel techniques have been successfully used for building recommender systems [28], fake news detection [40], protein function prediction [17], and detection of cancer-beating molecules in food [36]. GDL owes its success to the fact that it operates directly on the relational structure of a given problem. An example of such a structure is a graph. It can describe various concepts ranging from a social network to a chemical compound.

Recent progress has demonstrated the applicability of GDL in tasks involving learning on 3D objects directly. It laid the foundations for 3D shape classification and 3D shape correspondence tasks. We take recent work on GDL and apply it to the medical setting for brain shape analysis. To the best of our knowledge, this is a pioneer work in surveying and implementing various GDL techniques for the analysis of brain geometry.

In our work, we explore different ways in which we can represent a 3D shape and evaluate existing architectures on a few benchmark brain modelling tasks (e.g. biological sex prediction and age regression). Moreover, we investigate the correlation of brain geometry with certain conditions: hypertension and long-term smoking. We make contributions by introducing a novel convolutional operator, *LocalEdgeConv*, which avoids the need for shape pre-alignment, because it operates on the shapes' intrinsic features. We benchmark our method on ModelNet40 and show that, in certain scenarios, our work substantially outperforms state-of-the-art methods.

For our work, we use data coming from the UK Biobank, which contains roughly 15,000 brain scans. Moreover, we use scans coming from the Cam-CAN dataset to evaluate the robustness of the proposed methods. For benchmarking the LocalEdgeConv operator, we use data coming from the ModelNet40 dataset [41].

The results of our work need not be constrained to the medical setting, and can be easily applied to any shape recognition task – such as the geospatial mapping, using LiDAR scanners.

To complete our project, we have split it into several sub-tasks:

1. Investigate Geometric Deep Learning techniques and identify which are suitable for our work.

2. Preprocess the UK Biobank and Cam-CAN datasets into graph and point cloud representations.

3. Compare several different Geometric Deep Learning models trained and evaluated on UK Biobank dataset.

4. Propose a translation- and rotation invariant-operator, LocalEdgeConv, which achieves good performance on the UK Biobank dataset. This model enables us to avoid doing 3D shape registration.

5. Verify that our models are robust and generalize well to scans coming from a different dataset (Cam-CAN).

6. Benchmark LocalEdgeConv on the ModelNet40 dataset and evaluate its performance.

1.2 Challenges

During the project, we found the following to be the biggest challenges:

1. **Access to powerful hardware.** This project involves training deep neural networks, which require powerful GPUs for training. The departmental machines are shared among many students also working on their projects and therefore access to resources was limited.

2. **Training times.** Training times vary from model to model and are between 45min and 24hrs. It was a limiting factor when it came to hyperparameter search and exploring different architectures for shape classification.

3. **Lack of domain knowledge.** Medical imaging and brain analysis require a domain knowledge. Although not strictly required when training models, lack of domain knowledge may be a drawback compared to other research and might hinder the performance of our methods.

1.3 Contributions

1. **Survey and benchmark multiple methods suitable for brain analysis.** We carry out an extensive survey of different GDL architectures and decide which ones are the most suitable for learning on 3D objects.

2. **Define LocalEdgeConv: rotation and translation invariant convolutional operator.** We propose a novel convolutional operator, which avoids the need to pre-register shapes coming from different sources. This is because our operator is designed to be translation- and rotation-invariant.

3. **Demonstrate how our architecture can be applied to different datasets.** We build a pipeline which allows us to apply our models on brain scans coming from arbitrary datasets.

1.4 Report layout

Chapter 2 focuses on the advances in Geometric Deep Learning for object classification and outlines the models specific to medical imaging. A brief comparison of different approaches, and motivation behind them, can also be found in this chapter.

The main body is split into three chapters. Chapter 3 focuses on how to apply GDL to brain shape analysis. It introduces and explains the architectures used in our evaluation. It talks about the issues arising from using different datasets; that is, the added complexity of image registration. The novel translation- and rotation-invariant architecture, dubbed LocalEdgeConv, is also presented in this chapter. Chapter 4 first evaluates the performance of different architectures on the UK Biobank dataset, showing how some architectures significantly outperform the baseline method. It is shown that our novel convolutional operator avoids the need to perform any shape registration and guarantees robustness to data coming from different sources. In this chapter we benchmark and evaluate LocalEdgeConv operator on the ModelNet40 dataset.

Chapter 5 presents potential real-world applications of our findings.

Chapter 6 concludes this report by summarising the work which was done and suggests several areas of future research.

Chapter 2

Background

Data represented in non-Euclidean domains is used extensively within computer science and related fields. Examples of such data include social networks, 3D computer shapes, recommender systems, citation graphs, or molecular graph structures. All of those examples and many more can be formulated as graphs or manifolds which capture interactions (edges) between individual units (vertices). For those purposes, Geometric Deep Learning – a term coined by Bronstein et al. [7] – has emerged aiming to generalize deep learning models to non-Euclidean domains.

In this chapter we provide preliminaries to Geometric Deep Learning and survey current state-of-the-art approaches for graph classification.

2.1 Learning on Euclidean domains

The classical notion of deep learning has been developed in Euclidean domains. In this section, we provide a brief overview of Convolutional Neural Networks (CNN). Knowledge of these will serve as a stepping stone in understanding models on non-Euclidean domains.

2.1.1 Convolutional Neural Networks

Convolutional Neural Networks are a class of deep neural networks. They are characterized by their shared-weights architecture and shift-equivariance. They are commonly used in image and video recognition, natural language processing and image classification [45].

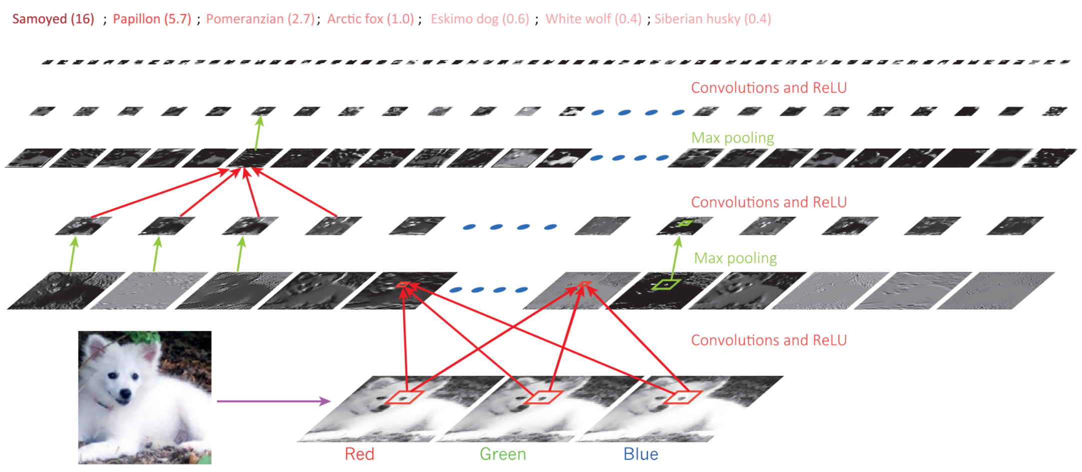

Samoyed (16) ; Papillon (5.7) ; Pomeranzian (2.7); Arctic fox (1.0) ; Eskimo dog (0.6) ; White wolf (0.4) ; Siberian husky (0.4)
Convolutions and ReLU
Max pooling
Convolutions and ReLU
Max pooling
Convolutions and ReLU
Red
Green
Blue

Figure 2.1: Typical Convolutional Neural Network architecture, used in computer vision applications. Figure from Lecun et al. [23].

2.1.2 Convolutional layer

A CNN typically consists of several convolutional layers. Each convolutional layer is made up of a set of learnable filters (also known as kernels) and is of the form $g = C_r(f)$. It acts on a p-dimensional input $f(x) = (f_1(x), ..., f_p(x))$ by applying a set of filters $\Gamma = (\gamma_{l,l'})$ where $l = 1, ..., q$, $l' = 1, ..., p$ and a point-wise linearity ξ,

$$g_l(x) = \xi \left(\sum_{l'=1}^{p} (f_{l'} * \gamma_{l,l'})(x) \right) \tag{2.1}$$

which produces a q-dimensional output $g(x) = (g_1(x), ..., g_q(x))$ – referred to as feature maps (See Fig. 2.1).

Feature maps are stacked along the depth dimension to form the output volume of the convolution layer (See Fig. 2.2). CNNs exploit spatially-local correlation by enforcing a local connectivity pattern between neurons of adjacent layers, so that each neuron is connected to a small region of the input. Such an architecture ensures that the learnt filters produce the response to a spatially-local input pattern. Weight sharing of convolution operation, another characteristic feature of CNNs, controls the number of free parameters and makes the CNN parameters more efficient compared to Multi-Layer Perceptron (MLP) networks. Parameter sharing makes CNNs translation equivariant and allows the operator at each layer to have a constant number of parameters, independent of the input size n. The convolution operation may be further categorized into different classes – based on the type and size of filters, type of padding, and the direction of convolution [23].

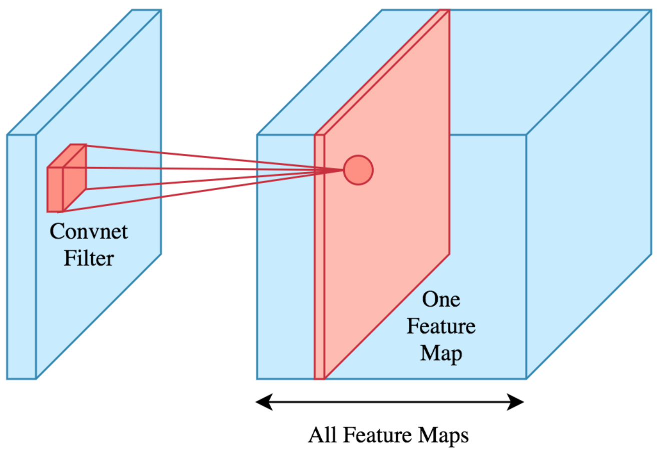


Figure 2.2: Neuron of the single activation map and its local receptive field. Figure from [5].

2.1.3 Pooling layer

One limitation of feature maps is that they respond to the precise position of features on the input (See Fig. 2.2), meaning that a small perturbation in the position of a feature on the input image can result in a different feature map. This can happen with cropping, shifting, or with other small changes on the input image. An effective approach to fixing this problem is called pooling. It works by partitioning the input into a set of non-overlapping rectangles and, for each of these rectangles, producing an output of a single scalar value (e.g. the maximum). The intuition is that the exact location of a feature is less important than its rough location, relative to other features. The pooling layer serves to progressively reduce the spatial size of the representation and decrease the number of parameters, thereby lowering the computational cost of training the network and controlling over-fitting.

The pooling operation $g = P(f)$ is defined as:

$$g_l(x) = P(\{f_l(x') : x' \in N(x)\}), \quad l = 1, ..., q \tag{2.2}$$

where $N(x) \subseteq \Omega$ is a neighborhood around x and P is a permutation-invariant function, such as the L_p-norm (the choice of $p = 1, 2$ or ∞ results in average-, energy-, or max-pooling respectively) [7].

2.1.4 Activation function

The activation is a nonlinear transformation that is done over the input signal, using an activation function. The transformed output is then sent to the next layer of neurons as input. With the non-linearities introduced by the activation function, the neural network is capable of expressing complex non-linear models.

One of the most commonly used activation functions is Rectified Linear Unit (ReLU). It was first used for deep learning in Restricted Boltzman Machines by Nair et al. [30] and is defined as follows:

$$ReLU(x) = max(0, x) \tag{2.3}$$

Exponential Linear Unit (ELU) is another activation function. According to Clevert et al., it speeds up learning in deep neural networks and leads to higher classification accuracies [10] and is therefore widely used in our work. It is defined as:

$$ELU(x) = \begin{cases} x & x > 0 \\ \alpha(e^x - 1) & x \leq 0 \end{cases} \tag{2.4}$$

where α is a hyperparameter. In PyTorch, it defaults to $\alpha = 1$ and that is what we use in our work.

2.1.5 Fully connected layer

The main objective of a fully connected layer in CNNs is to take the result of the convolution and pooling layers and use it to classify the data (e.g. image) into a class (See Fig. 2.1).

2.2 Intrinsic and extrinsic shape descriptors

The substantial success of deep learning, especially CNNs, in computer vision has led to a big interest in applying these methodologies to geometric problems – like shape classification, graph signal processing or graph clustering. There are two ways, intrinsic and extrinsic, to reason about the geometrical properties of 3D shapes. To the best of our knowledge, the best performing shape classification architectures use a combination of both [38], [32], [31].

2.2.1 Extrinsic features

Extrinsic approaches treat the geometric data as structures embedded in a Euclidean space. The main problem with this paradigm is that it's not invariant to shape perturbations, like rotations or deformations (see Fig. 2.3). It is challenging to develop successful models which use only extrinsic descriptors as features, since they would require a huge dataset given the number of possible rotations of the shape. The problem can be alleviated by assuming the shapes are aligned (e.g w.r.t. an axis).

2.2.2 Intrinsic features

Intrinsic approaches try to apply learning techniques to geometric data by generalizing the main ingredients, such as convolutions, to non-Euclidean domains. For example, reasoning about a 3D shape as a topological space of itself (rather than embedding it in a Euclidean space) is an intrinsic approach. In an intrinsic representation, the filter is applied on the very surface of some data – thus being independent of the embedding of the construction (see Figure 2.3).

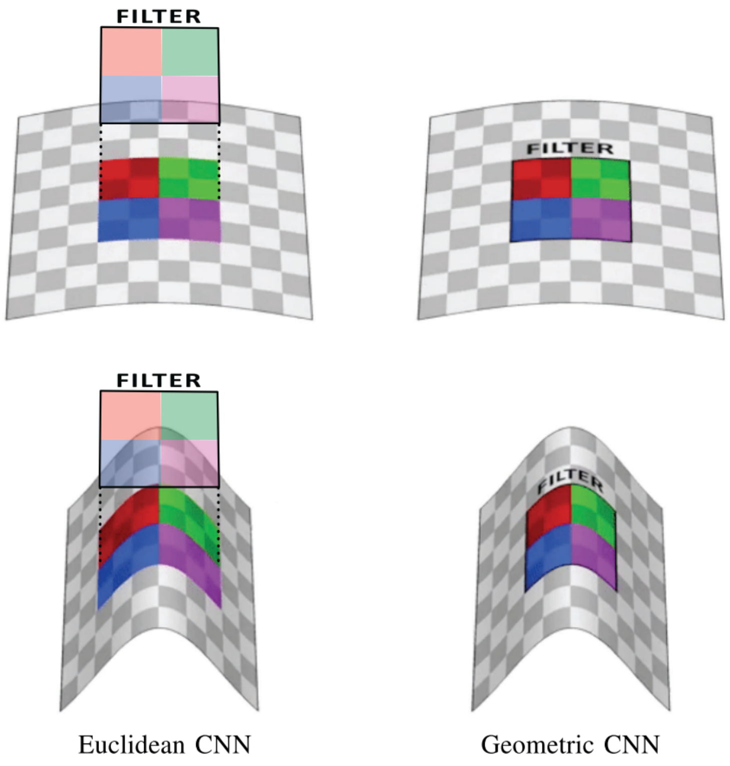


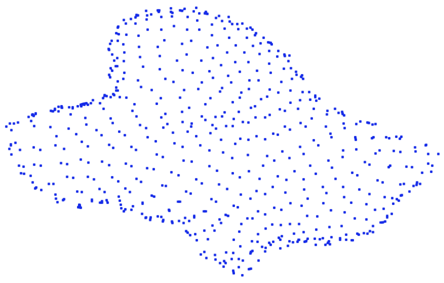
Figure 2.3: Illustration of the difference between classical CNN (left), applied extrinsically to a 3D shape considered as a Euclidean object, and a geometric CNN (right), applied intrinsically on the surface. The figure is from Bronstein et al. [7].

2.3 Preliminaries of Geometric Deep Learning

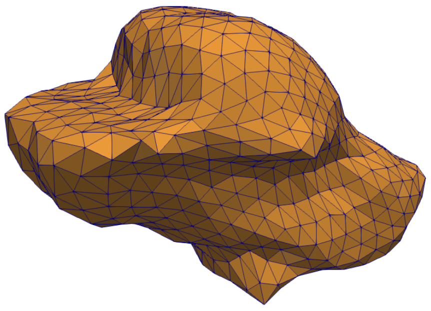
Here we present and explain the main concepts of Geometric Deep Learning. Our work will build on the notions introduced in this section.

2.3.1 Graph

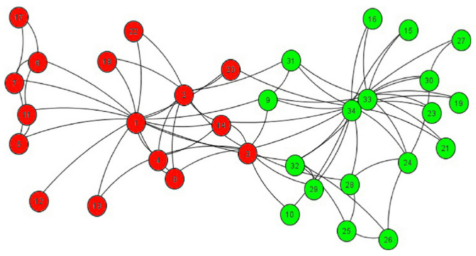
A graph G is a pair (V, E), with a finite set of vertices $V = \{v_1, ..., v_n\}$, $|V| = n$ and a finite set of edges $E \subseteq V \times V$, $|E| = m$. A graph can be represented by an adjacency matrix W of size $n \times n$, where:



(a) Point cloud of a brain stem



(b) Triangular mesh of a brain stem



(c) Graph representing the social network of a university karate club, taken from [43].

Figure 2.4: Examples of non-Euclidean constructs.

$$W[i,j] = \begin{cases} 1 & \text{if there is an edge connecting } v_i \text{ and } v_j, \\ 0 & \text{otherwise} \end{cases} \tag{2.5}$$

2.3.2 Manifold

A manifold is a space that is locally Euclidean. Around point $x \in X$, the manifold is homeomorphic to a d-dimensional Euclidean space referred to as the tangent space and denoted by T_xX. An inner product $\langle ., . \rangle_{T_xX} : T_xX \times T_xX \to \mathbb{R}$, depending smoothly on x, is called the Riemannian metric [27]. Attributes which are expressible entirely in terms of Riemannian metric, and are therefore independent of the way the surface is embedded, are called intrinsic. Such quantities are invariant to isometric deformations [4]. A Riemannian metric makes it possible to define several geometric notions on a Riemannian manifold – such as area of a surface, higher-dimensional analogues like volume and intrinsic curvature of the manifold itself.

In computer graphics, shapes are represented as discrete 2-dimensional manifolds embedded in $\mathbb{R}^3$. A discrete manifold has vertices uniformly sampled from the surface it is trying to represent, with edges expressing the local structure of the shape. In order to preserve the geometry of the underlying continuous manifold, a discrete manifold is a polyhedral surface of small adjacent faces (triangles). Each *face* $F \in V \times V \times V$ and each edge is shared by exactly two triangular faces [7]. The tuple (V, E, F) is referred to as a triangular mesh. See Fig. 2.4b.

2.3.3 Point cloud

From a data structure point of view, a point cloud is an unordered set of vectors. From a computer vision perspective, it is represented as a set of 3D points $\{P_i | i = 1, ..., n\}$ – where each point P_i is a vector of (x, y, z) coordinates plus extra feature channels, such as color, normal, curvature etc. For object classification, the input point cloud is either directly sampled from a shape or pre-segmented from a scene point cloud. Point clouds provide a flexible geometric representation, suitable for countless applications in computer graphics and comprise the raw output of most 3D data acquisition devices (like LiDAR sensors mounted on autonomous vehicles). Hand-designed features on point clouds have long been used, however, the recent overwhelming success of CNNs in image analysis suggests the value of adapting insights from CNN to point clouds. See Fig. 2.4a for a point cloud sampled from the surface of a 3D shape.

2.3.4 Curvature

Shape curvature is an important concept for our work. At any point on the surface of a manifold, we can define a normal vector orthogonal to the surface. The planes containing the normal vector are called normal planes. The intersection of a normal plane with the surface forms a curve called a normal section and it is the normal curvature. The maximum and minimum curvatures of the sections, at a given point, are called the principal curvatures: κ_1, κ_2. The Gaussian curvature is the product of the two principal curvatures $K = \kappa_1 \kappa_2$. The mean curvature is given by the arithmetic mean of the two principal curvatures: $H = \frac{1}{2}(\kappa_1 + \kappa_2)$.

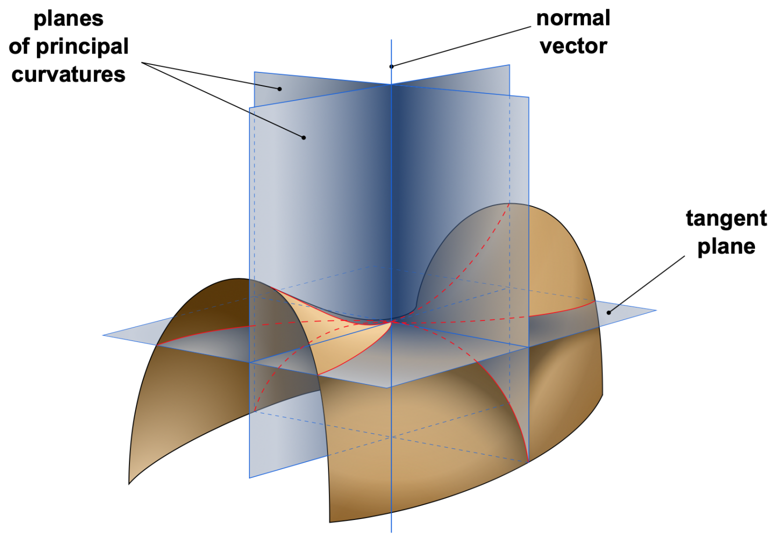


Figure 2.5: Manifold with normal planes in the direction of principal curvatures. The figure is from Bhate el al. [2].

Curvature is an intrinsic property of the surface, meaning it does not depend on the particular embedding of the surface or the shape. If the shape is translated or rotated in the embedding space, the curvature at any given point will not change. This property is an important concept when designing architectures that are invariant to translations and rotations.

2.3.5 Calculus on manifolds

The Laplacian operator $\Delta : L^2(X) \to L^2(X)$, is defined as:

$$\Delta f = -div(\nabla f) \tag{2.6}$$

where *div* is divergence and ∇ is the gradient operator. The *gradient operator* $\nabla f : L^2(X) \to L^2(TX)$ is similar to the classical notion of the gradient defining the direction of the steepest change of the function at a point, with the only difference being that the direction is now a tangent vector. The *divergence operator* $div : L^2(TX) \to L^2(X)$ is acting on tangent vector fields and (formal) adjoint to the gradient operator [33]. Note that the Laplacian operator is intrinsic, as it is expressed solely in terms of the Riemannian metric. Intuitively, the Laplacian of f on a manifold is the difference between $f(x)$ and the average value of f around x. (See Fig. 2.6b)

Discrete Laplacian operator In computer graphics, 3D shapes are represented in the form of a triangular mesh (V, E, F) – that is why it is useful to define the discrete Laplacian operator. Let's associate a weight $a_i > 0$ with each vertex $i \in V$, and a weight $w_{ij} \geq 0$ with each edge $(i, j) \in E$.

Real functions $f : V \to \mathbb{R}$ and $F : E \to \mathbb{R}$ on the vertices and edges of the graph are the discrete analogy to scalar and tangent vector fields.

The *graph gradient* is an operator $\nabla : L^2(V) \to L^2(E)$, mapping functions defined on vertices to functions defined on edges:

$$(\nabla f)_{ij} = f_i - f_j \tag{2.7}$$

The *graph divergence* is an operator $div : L^2(E) \to L^2(V)$ defined as:

$$(divF)_i = \frac{1}{a_i} \sum_{j:(i,j)\in E} w_{ij} F_{ij} \tag{2.8}$$

The *graph Laplacian* is an operator $\Delta : L^2(V) \to L^2(V)$, defined as $\Delta = -div(\nabla)$. Combining Eq. 2.7 and 2.8, we have

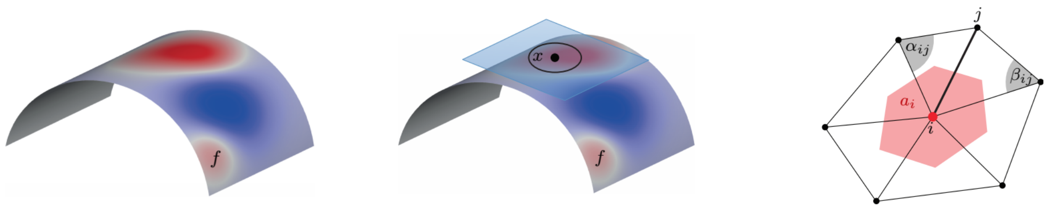
$$(\Delta f)_i = \frac{1}{a_i} \sum_{j:(i,j)\in E} w_{ij}(f_i - f_j) \tag{2.9}$$

Eq. 2.9 captures the intuitive geometric interpretation of the Laplacian: the difference between the local average of a function around a point and the value of the function at the point itself.

We can write Eq. 2.9 in matrix-vector notation. By denoting $W = (w_{ij})$ as an $n \times n$ matrix of edge weights, $A = diag(a_1, ..., a_n)$ the diagonal matrix of vertex weights, and $D = diag(\sum_{j:j\neq i} w_{ij})$ the *degree matrix*, the graph Laplacian application to a function $f \in L^2(V)$ represented as a column vector $f = (f_1, ..., f_n)^T$ is expressed as:

$$\Delta f = A^{-1}(D - W)f \tag{2.10}$$

Setting A as $A = I$ in Eq. 2.10 is called *unnormalized graph Laplacian.* In this case the unnormalized graph Laplacian Δ is a real symmetric positive semidefinite matrix that has complete set of orthogonal eigenvecotrs, which we denote by $\Phi = (\Phi_1, ..., \Phi_n)$. These eigenvectors have associated real, non-negative eigenvalues $\lambda_1, ..., \lambda_n$, which satisfy $\Delta\Phi_i = \lambda_i\Phi_i$, for $i = 1, ..., n$.



(a) Scalar field (b) Laplacian operator (c) Triangular mesh

Figure 2.6: Visualisation of the scalar field (left), Laplacian operator (middle) on the manifold and the triangular mesh (right). Figure from [6].

2.4 Spectral methods

Given a weighted graph, one way to generalize a convolutional architecture is to look at linear operators that commute with the graph Laplacian. This property implies operating on the spectrum of the graph weights, given by the eigenvectors of the graph Laplacian [7].

2.4.1 Spectral CNN

Spectral convolutional layer was defined by Bruna et al. [8] as

$$g_l = \xi \left(\sum_{l'=1}^{p} \Phi_k g_{\theta_{l,l'}}(\Lambda) {\Phi_k}^T f_{l'} \right) \tag{2.11}$$

where $(f_1, ..., f_p)$ and $(g_1, ..., g_q)$ represent the p- and q-dimensional input and output signals on the vertices of the graph, $g_{\theta_{l,l'}}(\Lambda)$ is a $k \times k$ diagonal matrix of spectral multipliers representing a filter in the frequency domain, Φ_k represents the first k Laplacian eigenvectors sorted by eigenvalues from lowest to highest $(\lambda_1 \leq ... \leq \lambda_k \leq ... \leq \lambda_n)$, and ξ is a nonlinearity applied on the vertex function values.

For spectral methods, the choice and design of spectral filter influences the computational complexity and the number of parameters in the convolutional layer, which affects the network performance.

Because the eigendecomposition of the Laplacian matrix is needed to obtain Φ_k, Spectral CNN faces some limitations. First, the eigendecomposition is of $\mathcal{O}(n^3)$ computational complexity. Second, any perturbation in the graph results in a change to its eigenbasis. Third, the learned filters are dependent on the domain, meaning they cannot be applied to a graph with a different structure [42]. It means that if we have a filter w.r.t. a basis Φ_k, and then apply it on another domain with different basis Ψ_k, the filter might not work as expected (see Fig. 2.7).

2.4.2 ChebNet

The *Chebyshev polynomials* are defined by the recurrence relation:

$$\begin{aligned} T_j(\lambda) &= 2\lambda T_{j-1}(\lambda) - T_{j-2}(\lambda) \\ T_0(\lambda) &= 1 \\ T_1(\lambda) &= \lambda \end{aligned} \tag{2.12}$$

Chebyshev Spectral CNN (ChebNet) approximates the spectral filter by Chebyshev polynomials of the diagonal matrix of eigenvalues [11]. We know that a polynomial of the

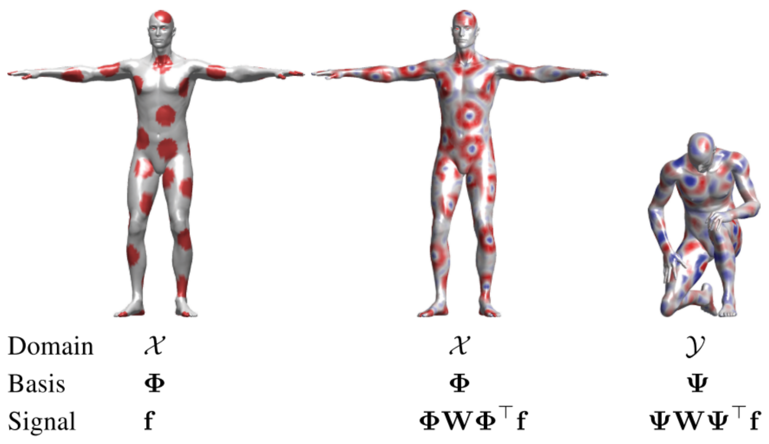

Domain $\mathcal{X}$ $\mathcal{X}$ $\mathcal{Y}$
Basis $\mathbf{\Phi}$ $\mathbf{\Phi}$ $\mathbf{\Psi}$
Signal $\mathbf{f}$ $\mathbf{\Phi W \Phi^\top f}$ $\mathbf{\Psi W \Psi^\top f}$

Figure 2.7: A toy example illustrating the difficulty of generalizing spectral filtering across non-Euclidean domains. Left: a function defined on a manifold (function values are represented by color); middle: result of the application of an edge-detection filter in the frequency domain; right: the same filter applied on the same function but on a different (nearly-isometric) domain produces a completely different result. The figure is from Bronstein et al. [7]

Laplacian acts as a polynomial of the eigenvalues. Therefore, it is possible to represent the filters via a polynomial expansion, instead of spectral multipliers (Spectral CNN):

$$g_\theta(\Delta) = \Phi g_\theta(\Lambda)\Phi^T \quad \text{corresponding to:} \quad g_\theta(\lambda) = \sum_{j=0}^{r-1} \theta_j \lambda^j \tag{2.13}$$

We have that θ is the r-dimensional vector of polynomial coefficients and $g_\theta(\Lambda) = diag(g_\theta(\lambda_1), ..., g_\theta(\lambda_n))$ is a diagonal matrix of spectral multipliers, resulting in filter matrices $g_{\theta_{l,l'}}(\Lambda)$ whose entries have an explicit form in terms of the eigenvalues. A filter can therefore be parameterized uniquely as an expansion with order $r-1$:

$$g_\theta(\tilde{\Delta}) = \sum_{j=0}^{r-1} \theta_j \Phi T_j(\tilde{\Lambda})\Phi^T = \sum_{j=0}^{r-1} \theta_j T_j(\tilde{\Delta}) \tag{2.14}$$

where $\tilde{\Delta} = 2{\lambda_n}^{-1}\Delta - I$ and $\tilde{\Lambda} = 2{\lambda_n}^{-1}\Lambda - I$ is a rescaling of the Laplacian eigenvalues from the interval $[0, \lambda_n]$ to $[-1, 1]$, since Chebyshev polynomials form an orthogonal basis in $[-1, 1]$.

The convolutional layer is then defined as:

$$g_l = \xi\left(\sum_{l'=1}^{p}\sum_{j=0}^{r-1} \theta_j T_j(\tilde{\Delta}) f_{l'}\right) \tag{2.15}$$

The computational complexity of this procedure is $\mathcal{O}(rn) = \mathcal{O}(n)$, as there is no need to compute the forward and backward Fourier transforms. We also do not need to compute the eigendecomposition of the Laplacian Δ, which is an improvement over Spectral CNNs.

2.5 Spatial methods

The issues with spectral approaches (see Fig. 2.7) can be tackled by using methods that extract representations from local Euclidean neighborhoods on discrete manifolds. The spatial convolution is considered a more versatile method for learning on non-Euclidean structures.

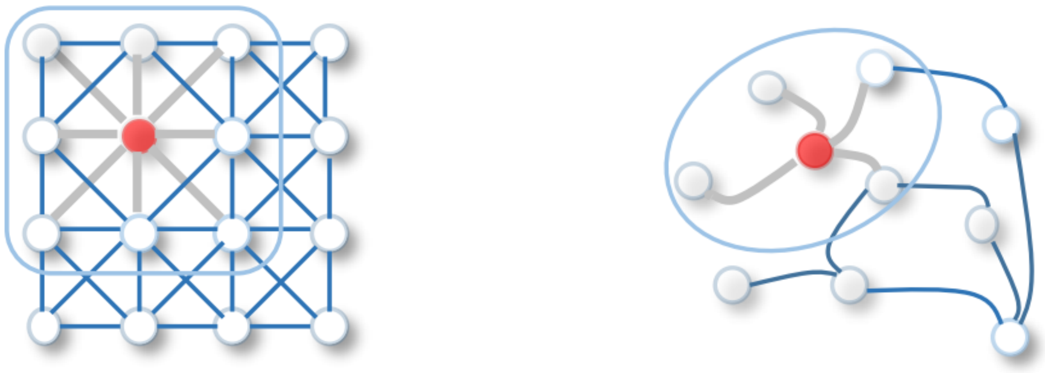

(a) 2D Convolution on an image (b) Graph Convolution

Figure 2.8: 2D Convolution vs. Graph Convolution. Figure taken from Wu et al. [42].

Spatial methods define graph convolutions based on a node's spatial relations, which is analogous to the convolution operation on a classical CNN. Images can be considered a special form of a graph with each pixel representing a node, connected to each neighbouring pixels. A filter would be applied on the patch of the image including the pixel and its neighbouring nodes. Similarly, spatial methods convolve a given node's features, using a patch operator, with its neighbors' features (See Fig. 2.8b). The intuition about the spatial graph convolutions is that this operation propagates and updates node features along edges.

2.5.1 GCN

Kipf et al. [22] simplified the Chebyshev polynomial of order $r-1$ further, by assuming $r=2$ and $\lambda_n \approx 2$:

$$g_l = \xi\left(\sum_{l'=1}^{p} \alpha_0 f_{l'} + \alpha_1(\Delta - I) f_{l'}\right) = \xi\left(\sum_{l'=1}^{p} \alpha_0 f_{l'} + \alpha_1 D^{-1/2} W D^{-1/2} f_{l'}\right) \tag{2.16}$$

where $\tilde{W} = W + I$ (adding self loops) and $\tilde{D} = diag(\sum_{j \neq i} \tilde{w}_{ij})$.

The intuition of this method is that it can alleviate the problem of overfitting on local neighborhood structures for graphs with very wide node degree distributions, such as social networks, citation networks and many other real-world graph datasets. The computational complexity of this approach is also $\mathcal{O}(n)$, similar to ChebNets. Both apply simple filters acting on the r- or *1*-hop neighborhood of the graph in the spatial domain.

2.5.2 GeodesicCNN

Masci et al. [26] proposed the first intrinsic version of convolutional neural networks, Geodesic CNN (GCNN), on manifolds applying filters to local patches represented in geodesic polar coordinates. Since manifolds come with a low-dimensional tangent space at each point, it follows naturally to work in a local system of coordinates in the tangent space. In particular, on two-dimensional manifolds (like 3D shapes) one can create a polar system of coordinates around x where the radial coordinate is given by some intrinsic distance $\rho(x') = d(x, x')$, and the angular coordinate $\theta(x)$ is obtained by ray shooting from a point at equispaced angles. The *patch operator* in GCNNs is defined on discrete manifold as follows:

$$(D(x)f)(\rho, \theta) = \sum_{y \in N(x)} v_{\rho,\theta}(x, y) f(y) \tag{2.17}$$

The patch operator $D(x)f$ maps values of function f to polar coordinates ρ, θ. Then the intuition behind the *convolution operator* is that it is a matching a template $g(\rho, \theta)$ with the extracted patch at each vertex of the discrete manifold, where the maximum is taken over all possible rotations of the template in order to resolve the origin ambiguity in the angular coordinate. The *geodesic convolution* is defined for manifolds as follows

$$(f * g) = \max_{\Delta\theta \in [0, 2\pi)} \int_0^{2\pi} \int_o^{\rho_{max}} g(\rho, \theta + \Delta\theta)(D(x)f)(\rho, \theta) d\rho d\theta \tag{2.18}$$

The weighting functions v localized around ρ, θ, in this case, can be obtained as a product of Gaussians:

$$v_{ij}(x, x') = e^{-(\rho(x') - \rho_i)^2 / 2\sigma_\rho{}^2} e^{-(\theta(x') - \theta_j)^2 / 2\sigma_\theta{}^2} \tag{2.19}$$

where $i = 1, ..., J$ and $j = 1, ..., J'$ denote the indices of the radial and angular bins, respectively. The resulting JJ' weights are bins of width $\sigma_p \times \sigma_\theta$ in polar coordinates.

2.5.3 Mixture Model Network (MoNet)

Monti et al. [27] proposed a general construction of patches by defining a local system of d-dimensional pseudo-coordinates $u(x, y)$ around each point x.

Each pseudo-coordinate is put through a weighting function, which provides the effect of a traditional image convolution kernel. Learning the filters and the patch operators affords additional degrees of freedom for the architecture, which makes it a state-of-the-art approach in several applications. In the case of a Gaussian Mixture Model (GMM) convolution, the weighting function is a set of parametric kernels $v_1(u), ..., v_J(u)$ with learnable parameters that operates on the pseudo-coordinates. The patch operator $D(x)f$ can be written as follows:

$$D_j(x)f = \sum_{y \in N(x)} v_j\big(u(x, y)\big) f(y), \quad j = 1, ..., J \tag{2.20}$$

where J represents the dimensionality of the patch on a discrete manifold. Then, a general convolution operator is defined by a template-matching procedure:

$$(f * g) = \sum_{j=1}^{J} g_j D_j(x) f \tag{2.21}$$

where one of the possible kernels can be a Gaussian kernel [27]:

$$v_j(u) = e^{-\frac{1}{2}(u-\mu_j)^T \Sigma_j{}^{-1}(u-\mu_j)} \tag{2.22}$$

where $d \times d$ covariance matrices $\Sigma_1, ..., \Sigma_J$ and $d \times 1$ mean vectors $\mu_1, ..., \mu_J$ are learnable parameters of the kernels. The choice of a Gaussian kernel makes Eq. 2.20, 2.21 a Gaussian Mixture Model.

Method	Pseudo-coordinates	$u(x, y)$	Weight function $w_j(u)$
CNN	Local Euclidean	$y - x$	$\delta(u - \bar{u}_j)$
GCN	Vertex degree	$\deg(x)$, $\deg(y)$	$(1 - \lvert 1 - \frac{1}{\sqrt{u_1}} \rvert)(1 - \lvert 1 - \frac{1}{\sqrt{u_2}} \rvert)$
GCNN	Local polar geodesic	$\rho(x, y)$, $\theta(x, y)$	$\exp(-\frac{1}{2}(u - \bar{u}_j)^T \begin{pmatrix} \bar{\sigma}_\rho^2 & \\ & \bar{\sigma}_\theta^2 \end{pmatrix} (u - \bar{u}_j))$

Table 2.1: CNN, GCN and GCNN as a particular setting of this framework. Table adapted from Monti et al. [27]. $\bar{\sigma}_\rho, \bar{\sigma}_\theta, \bar{u}$ denote fixed parameters of the weight functions.

Several CNN-type Geometric Deep Learning methods on graphs and manifolds can be obtained as a particular setting of the proposed framework with an appropriate definition of u and $w(u)$, see Table 2.1. For example, these methods can be applied on general graphs using the pseudo-coordinates x as the features, such as vertex degree, making it effectively a GCN. Fig. 2.9 shows how patch operator kernel functions $v_j(u)$ of GCNN, ACNN (Anisotropic CNN [3]) and MoNet are used in different generalizations of convolutions on manifolds. Note that for GeodesicCNN the kernels were fixed, whereas for MoNets the kernels are trained.

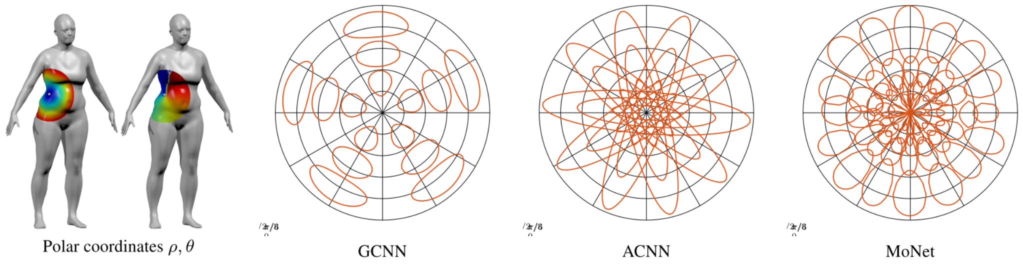


Figure 2.9: Representation of the weighting functions in the local polar (ρ, θ) system of coordinates (hand-crafted in GCNN and ACNN and learned in MoNet). Figure is from Monti et al. [27].

2.5.4 PointNet

PointNet is an architecture proposed by Qi et al. [31]. It is a continuous set function approximator, designed to work on point clouds. Formally, given an unordered point set $\{x_1, x_2, ..., x_n\}$ with $x_i \in \mathbb{R}^d$, one can define a set function $f : X \to \mathbb{R}$ that maps a set of points to a vector:

$$f(x_1, x_2, ..., x_n) = \gamma \left(\underset{i=1,...,n}{\text{MAX}} \{h(x_i)\} \right) \tag{2.23}$$

where γ and h are Multi-Layer Perceptron (MLP) networks. The function f in Eq. 2.23 is invariant to permutations of points in the set. The response of h can be interpreted as the spatial encoding of a point.

In contrast with operators like GCN or MoNet, the patch operator in PointNet is not constrained by the local connectivity of vertices, because point clouds don't have a notion of edges. The patch operator in PointNet assumes that the neighboring points form a meaningful subset, since all the points are from a space with a distance metric. Therefore, the model captures local structures from nearby points, and the combinatorial interactions among local structures.

2.5.5 PointNet++

PointNet++ architecture by Qi et al. [32] builds atop the original PointNet framework. It introduces hierarchical feature learning, to address the PointNet's inability to capture local context at different scales.

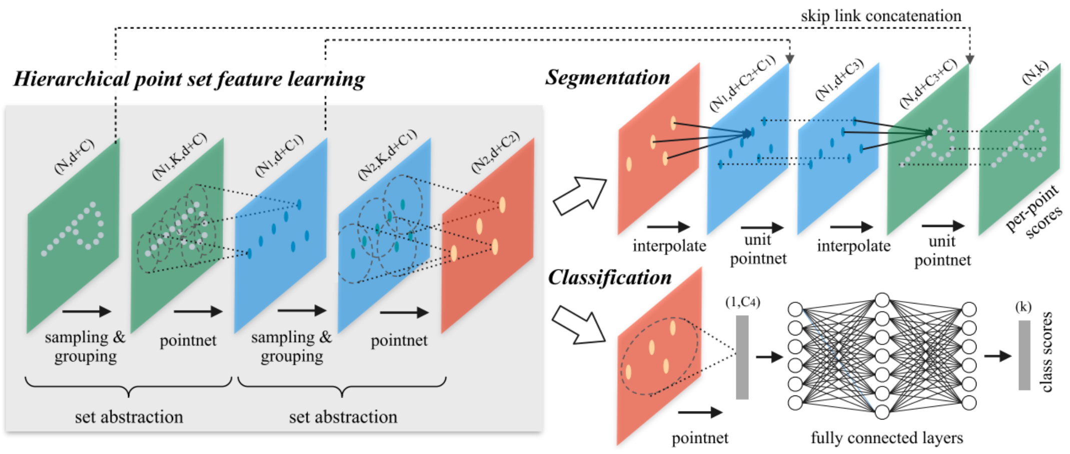


Figure 2.10: Illustration of a hierarchical feature learning architecture and its application for set segmentation and classification using points in 2D Euclidean space as an example. Figure is from Qi et al. [32]

The hierarchical structure is composed of a number of set abstraction levels, see Fig. 2.10. At each level, a set of points is processed and abstracted to produce a new set with fewer elements. The set abstraction level is made of three key layers: Sampling layer, Grouping layer and PointNet layer.

The Sampling layer selects a set of points from input points which defines the centroids of local regions. Given input points $\{x_1, x_2, ..., x_n\}$, the iterative *farthest point sampling* (FPS) is used to choose a subset of points $\{x_{i_1}, x_{i_2}, ..., x_{i_m}\}$, such that x_{i_j} is the most distant point, in terms of metric, from the set $\{x_{i_1}, x_{i_2}, ..., x_{i_{j-1}}\}$ with regard to the rest of the points.

Grouping layer then constructs local region sets by finding "neighboring" points around the centroids.

PointNet layer uses a mini-PointNet networks to encode local region patterns into feature vectors.

2.5.6 Dynamic Graph CNN

Dynamic Graph CNN, by Wang et al. [38], draws inspiration from PointNet and convolutional operations, but instead of working on individual points, it exploits the geometric structure by constructing a local neighboring graph and applying convolution-like operation on the edge connecting the neighborhood pair of points. It therefore has the property of translation-invariance and non-locality. This convolutional operator is referred to in the paper as *Edge Convolution* (EdgeConv).

In this approach the graph is not fixed, and rather it is dynamically updated after each layer of the network. It means that the set of the k-nearest neighbors of a point changes after each layer. This property leads to nonlocal diffusion of information throughout the point cloud.

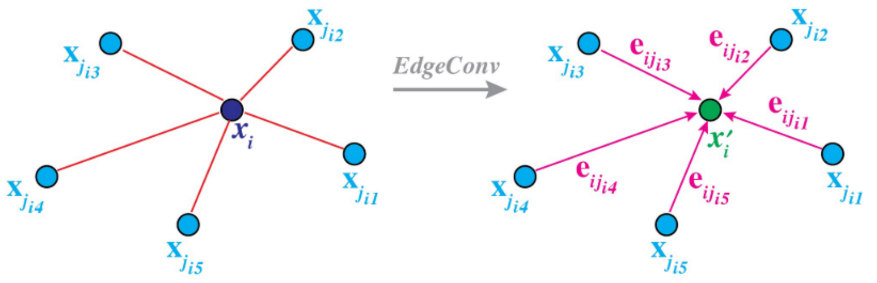


Figure 2.11: EdgeConv operator: The output of EdgeConv is calculated by aggregating the edge features associated with the edges from each connecting vertex. Figure is from Wang et al. [38].

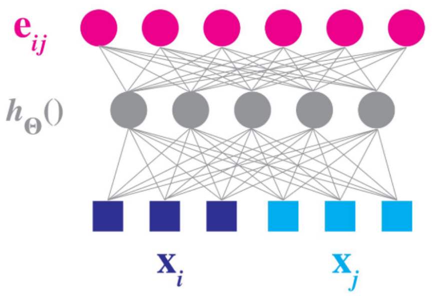


Figure 2.12: Computing edge feature e_{ij} from a point pair (X_i, X_j). In this example h_Θ is an MLP with weights being the learnable parameters. Figure is from Wang et al. [38].

Formally for a point cloud $X = \{x_1, x_2, ..., x_n\}$, with $x_i \in \mathbb{R}^F$, a directed graph $G = (V, E)$ representing a local point cloud structure, where V are vertices and E are edges. G is constructed as a k-nearest neighbour graph of X in $\mathbb{R}^F$. Edge feature is defined as $e_{ij} = h_\Theta(x_i, x_j)$, where $h_\Theta : \mathbb{R}^F \times \mathbb{R}^F \to \mathbb{R}^{F'}$ is a non linear function with learnable parameters Θ. EdgeConv operation applies channel-wise symmetric aggregation operation $\Box$, like mean, max or sum, on the edge features associated with all the edges coming out from each vertex. The result of the application of EdgeConv operator on the i-th vertex is therefore defined as:

$$x'_i = \mathop{\Box}_{j:(i,j)\in E} h_\Theta(x_i, x_j) \tag{2.24}$$

Choice of the edge function h and the aggregation operation $\Box$ influences the properties of EdgeConv operatror, see Table 2.2.

The classification model takes as input n points, calculates an edge feature set of size k for each point at the EdgeConv layer, and aggregates features within each set to compute EdgeConv responses for corresponding points. The output features of the last EdgeConv

h function	Properties
$h_\Theta(x_i, x_j) = \theta_j x_j$ $\theta = (\theta_i, ..., \theta_k)$	Classical Euclidean convolution.
$h_\Theta(x_i, x_j) = h_\Theta(x_j)$	Global information of the local neighboring structure. (PointNet)
$h_\Theta(x_i, x_j) = h_\Theta(x_j - x_i)$	Local information, considering the shape as a collection of small patches and losing the global information.
$h_\Theta(x_i, x_j) = h_\Theta(x_i, x_j - x_i)$	Asymmetric edge function combining the global shape structure (x_i and local shape features ($x_j - x_i$).

Table 2.2: Possible choices for function h with their properties.

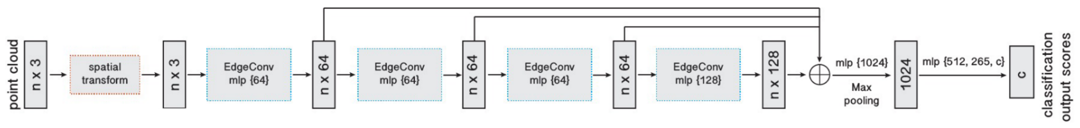


Figure 2.13: Example architecture of Dynamic Graph CNN classification model. Figure is from Wang et al. [38].

layer are aggregated globally (mean or max pool) to form a 1D global descriptor, which is used to generate classification scores for the classes.

2.6 Comparison between spectral and spatial methods

The theoretical foundation for graph spectral methods lies in graph signal processing and therefore new ConvGNNs can be built by designing new graph signal filters. However, spatial models are preferred over spectral models due to efficiency, generality, and flexibility issues. Spectral models are less efficient than spatial models as they need to perform eigendecomposition or handle the whole graph at the same time (e.g. mesh completion scenario) [42]. Spatial models are more scalable to large graphs as they directly perform convolutions in the graph domain via information propagation (i.e. message passing). The computation can be performed in a batch of nodes instead of the whole graph. Moreover, spectral models assume a fixed graph and because they rely on a graph Fourier basis they generalize poorly to new graphs. This is because any perturbation to a graph results in a change of eigenbasis. Spatial models perform graph convolutions locally on each node, which allows for weight sharing across different structures and locations. Finally, spectral methods are limited to undirected graphs whereas spatial methods can handle a bigger variety of graphs such as edge inputs, directed graphs, signed graphs and heterogenous graphs because of the flexibility of the aggregation function [42].

2.7 Generic architecture for graph classification

The generic convolutional architecture for graph classification is similar to the classical CNN architecture e.g. LeNet by Lecun et al. [24]. It consists of:

- *Convolutional layer.* Spectral or spatial convolutional layers, e.g GCN, EdgeConv.
- *Pooling layer.* The layer where a vertex-wise pooling operation is applied, e.g. max-pool, mean-pool or sort-pool [44].

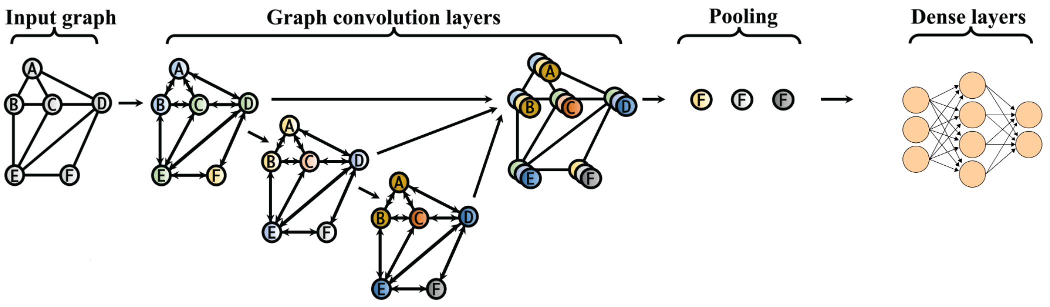


Figure 2.14: Generic architecture for graph classification. Original figure from Zhang et al. [44].

- *Dense layers*. This part of the architecture takes the result of the convolutional and pooling layers and classifies the data.

All the architectures included in our work will follow the pattern presented in Fig. 2.14, but they will differ in the kind of convolutional operator and the pooling operator used.

2.8 Generic message passing scheme

Message passing scheme generalizes the convolutional operator to irregular domains, it is also known as a *neighbourhood aggregation*. Let's have that $\mathbf{x}_i^{(k-1)} \in \mathbb{R}^F$ denotes node features of node i in layer $(k-1)$ and $\mathbf{e}_{j,i} \in \mathbb{R}^D$ denotes edge features from node j to node i. Message passing graph neural networks can be described as

$$\mathbf{x}_i^{(k)} = \gamma^{(k)} \left(\mathbf{x}_i^{(k-1)}, \square_{j \in \mathcal{N}(i)} \, \phi^{(k)} \left(\mathbf{x}_i^{(k-1)}, \mathbf{x}_j^{(k-1)}, \mathbf{e}_{j,i} \right) \right) \tag{2.25}$$

where $\square$ denotes a differentiable, permutation invariant function, e.g. sum, mean or max, and γ and ϕ denote differentiable functions such as Multi Layer Perceptrons. All spatial methods can be expressed in terms of a generic message passing scheme.

2.9 PyTorch Geometric

PyTorch Geometric [15] is a Geometric Deep Learning extension library for PyTorch. It is a library for deep learning on irregularly structured input data such as graphs, point clouds and manifolds from a variety of published papers. It consists of an easy-to-use mini-batch loader for many small and single big graphs, multi gpu-support and a large number of common benchmark datasets.

2.10 UK Biobank

UK Biobank (UKBB) is a major national and international health resource, and a registered charity in its own right, with the aim of improving the prevention, diagnosis and treatment of a wide range of serious and life-threatening illnesses – including cancer, heart diseases, stroke, diabetes, arthritis, osteoporosis, eye disorders, depression and forms of dementia [19].

For our work we use data collected from 14,503 subjects. For each subject we have 3D brain scan (see Fig. 2.4b) along with a medical record.

2.11 Cam-CAN dataset

The Cambridge Centre for Ageing and Neuroscience (Cam-CAN) is a large-scale collaborative research project at the University of Cambridge, launched in October 2010. The Cam-CAN project uses epidemiological, cognitive, and neuroimaging data to understand how individuals can best retain cognitive abilities into old age [34].

We use 652 brain scans from that dataset for the evaluation of our methods.

Chapter 3

Analysis of Subcortical Brain

Healthcare institutions generate and capture enormous amounts of data containing extremely valuable signals and information, at a pace far surpassing what traditional methods of analysis can process. With the advent of better computer hardware and more ubiquitous development frameworks, came the rise of Deep Learning (DL). DL was therefore able to enter the picture, as it is one of the best ways to integrate, analyze and make predictions based on large, heterogeneous data sets.

Brain imaging has drawn great interest among researchers in recent years, as it is used to diagnose metabolic diseases and lesions on a finer scale. Using deep learning methodologies – like convolutional neural networks (CNN) – has proven to be successful for tasks such as brain tissue segmentation [12], lesion detection [46] and regression and classification models (e.g. biological sex prediction, age regression). Nowadays, state-of-the-art models for brain analysis use brain MRI scans as input and 3D convolutions as main operator [39]. Models are able to accurately segment brain tissues and make predictions thereon. While those methods perform very well, they suffer from several constraints. Firstly, different scanners produce scans with slightly different intensity ranges and therefore a model trained on data coming from one scanner might have issues generalising to data coming from a different scanner. Secondly, 3D brain images need to be uniformly aligned, as the prediction coming out of rotated brain scan might not be meaningful. Thirdly, modern CNN models developed on isotropic scans cannot be easily applied to anisotropic images from different scanners.

Geometric deep learning (GDL) is a novel field in the world of machine learning. It offers powerful techniques to develop models which can learn directly on 3D shapes (see Section 2.7). For brain imaging purposes, we can represent the subcortical brain as a 3D shape. Our work is to try and evaluate the usability of geometric deep learning in the area of brain imaging. GDL operators offer benefits such as being rotation invariant, under certain assumtpions, and are not constrained by the dimensionality of the scan.

3.1 Problem definition

3.1.1 Role of subcortical brain

Subcortical brain structures are a group of diverse neural formations deep within the brain, which include regions such as: Hippocampus, Thalamus and Putamen. They are located just below the cerebral cortex in the human brain and are involved in complex activities such as memory, emotion, pleasure and hormone production. They act as information hubs of the nervous system, as they relay and modulate information passing to different areas of the brain [35]. Table 3.1 summarizes the roles of subcortical brain.

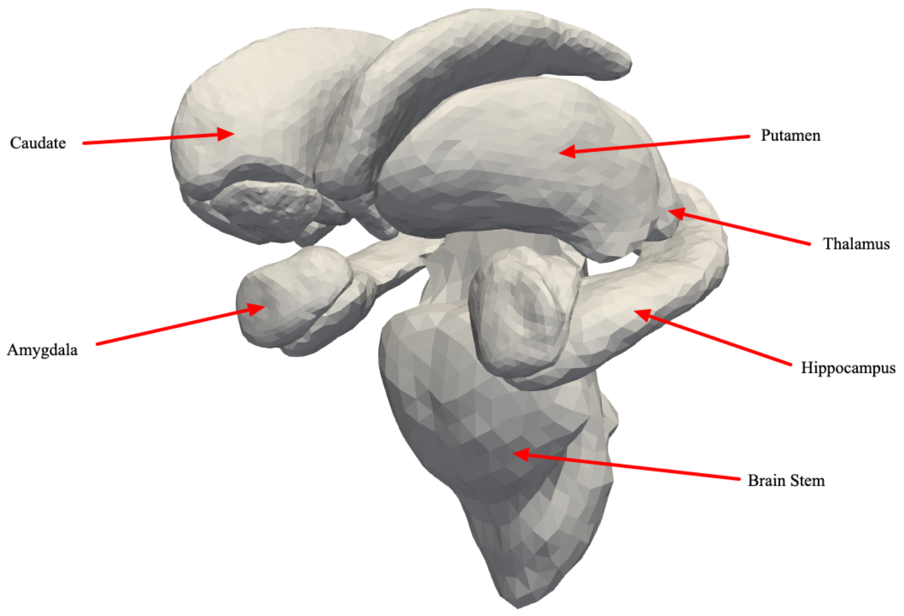


Figure 3.1: Subcortical brain with annotations. This is an example taken directly from the UK Biobank dataset, which contains 14,503 segmented 3D shapes of subcortical brain structures.

Brain part	**Role**
Brainstem	Conduction of information
Hippocampus	Spatial information processing, temporary memory
Thalamus	Relay function, controls sleep/wakefulness
Caudate	Motor system control, reward function control
Amygdala	Fear, anxiety, aggression control
Accumbens	Cognitive processing, motivation/aversion control
Putamen	Movement control
Pallidus	Movement control

Table 3.1: Subcortical brain parts and their roles.

3.1.2 Related work on subcortical brain

There are numerous works investigating the correlation between the geometric properties of the subcortical brain and human features. Wang et al. [37] suggest that the subcortical brain volume depends on age and biological sex. Foster-Dingley el al. [16] argue that there might be a relation between blood pressure and the shape of the subcortical brain, for older people. Moreover, Durazzo et al. [14] show that long-term cigarette smoking might be related with amplified age-related volume loss of the subcortical brain.

3.2 Analysis goals

The main goal of our work is to investigate the power of Geometric Deep Learning methods for brain shape analysis. Our evaluation builds upon the existing work described in section 3.1.2. We focus on the following tasks:

- *biological sex classification* - predict biological sex from brain shape
- *age regression* - predict age from brain shape
- *hypertension classification* - predict if a subject has hypertension (blood pressure higher than 140/90) or not (blood pressure lower than that)
- *long-term smoker classification* - predict if a subject is a long-term smoker, meaning one who has smoked most or all days in the past

We will analyse the above relations in the following ways:

- Create a model for each separate brain part
- Create an ensemble model, which utilises information from all of the brain parts

3.3 Data pre-processing

Data is provided as *vtk* files. We first remeshed the data to a common resolution, so that each shape class had the same amount of vertices (512). Then data was transformed into the format supported by PyTorch Geometric [15], that is a list of vertices with their features and the adjacency matrix. PyTorch Geometric library supports features such as, mini-batch loader for graphs and helpful data transforms which sped up the work on this project.

3.4 Multi-layer graph convolutional network

We compare several Geometric Deep Learning methods for the classification and regression tasks outlined in the section 3.2. Here we provide the description of the architectures used.

We consider two approaches to reason about a 3D shape. In the first approach we treat the triangular mesh of a shape as a graph. A graph encompasses information about the local neighbourhood of each node (adjacency matrix), node features (e.g XYZ coordinates) and edge attributes (e.g Euclidean or polar distances between connected nodes). The other approach we consider is treating the 3D shape as a point cloud, where an arbitrary number of points are sampled from the surface of the shape. In this scenario the only piece of information we have are the features of the point. The features are usually XYZ coordinates but they are not limited to that, as they can also include e.g. color, curvature or normal at that point. The local neighbourhood for the patch operator is dynamically determined e.g. k-nearest neighbours algorithm.

We benchmark 4 different methods. GCN and MoNet treat the 3D shape as a graph. EdgeConv and PointCloud++ treat the shape as a point cloud.

3.4.1 Network architecture

We propose a multi-layer graph convolution architecture for classification and regression tasks. Then we compare and evaluate the performance of models which are based on different geometric deep learning operators.

We keep the network architectures similar. Refer to Fig. 3.2 for the high-level overview of our approach. For GCN, MoNet and EdgeConv we used 2 convolutional layers followed by a dense layer, a pooling operator and a final dense layer. For PoinNet++, every convolution operator is preceded by farthest point sampling and grouping layers as described in the paper.

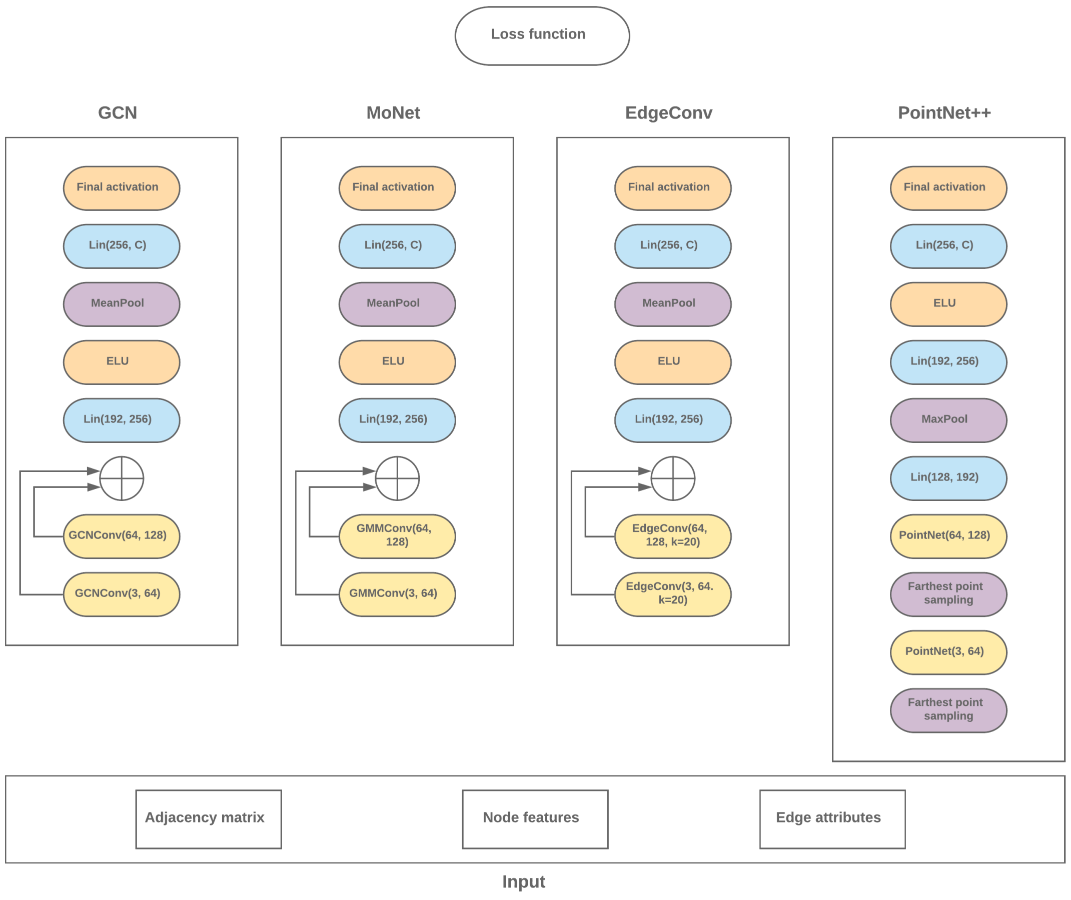


Figure 3.2: High-level overview of architectures employed for our experiment, comparison between GCN, MoNet, EdgeConv and PointNet++. $\bigoplus$ means tensor concatenation. For PointNet++ every sampling layer is followed by a grouping layer as described in the paper [31].

GCN convolution

We take a generic graph classification architecture decribed in Section 2.7. The convolutional operator is a graph convolutional layer by Kipf et al. [22] described in Section 2.5.1. This convolutional operator, can be expressed in terms of generic message passing scheme (see Section 2.8), where:

$$\mathbf{x}_i{}' = \sum_{j \in \mathcal{N}(i) \cup \{i\}} \frac{1}{\sqrt{\deg(x_i)} \cdot \sqrt{deg(x_j)}} \cdot (\mathbf{x}_j \mathbf{\Theta}) \tag{3.1}$$

where $\mathcal{N}(i)$ is the local 1-hop neighborhood of node x_i and $\mathbf{\Theta}$ is a learnable weight matrix.

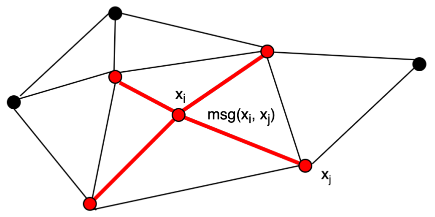
Mixture Model Network (MoNet)

Here the convolution operator is GMMConv (described in detail in Section 2.5.3). We use Euclidean distances between nodes as pseudo-coordinates required by this method. This convolutional operator is expressed in the generic message-passing scheme as:

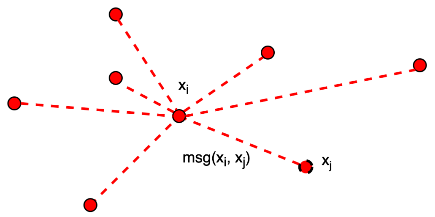
$$\mathbf{x}'_i = \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \frac{1}{K} \sum_{k=1}^{K} \mathbf{w}_k(\mathbf{e}_{i,j}) \odot \mathbf{\Theta}_k \mathbf{x}_j \tag{3.2}$$

$$\tag{3.3}$$

where $\mathbf{w}_k(\mathbf{e}) = \exp\left(-\frac{1}{2}(\mathbf{e} - \mu_k)^\top \Sigma_k^{-1} (\mathbf{e} - \mu_k)\right)$ is a k-the parametric kernel with learnable parameters μ_k and Σ_k.



(a) Patch operator at x_i in a graph representing a triangular mesh.



(b) Patch operator at x_i in a point cloud representing a 3D shape.

Figure 3.3: Comparison between patch operators of a 3D shape represented as a graph (triangular mesh) and a point cloud. Note that the receptive field of a point cloud patch operator is dynamic and therefore can be bigger, e.g. k-nearest neighbors grouping, where k determines the size of the patch.

EdgeConv

EdgeConv operator is described in detail in Section 2.5.6. The main idea behind it is constructing a local neighborhood graph and applying convolution-like operation on the edge connecting the neighboring pair of points. It can be expressed in the generic message passing scheme as:

$$\mathbf{x}'_i = \sum_{j \in \mathcal{N}(i)} h_{\mathbf{\Theta}}(\mathbf{x}_i \| \mathbf{x}_j - \mathbf{x}_i) \tag{3.4}$$

where $\mathcal{N}(i)$ is a dynamically determined k-nearest neighbourhood of node $\mathbf{x}_i$, $\|$ means feature concatenation and $h_{\mathbf{\Theta}}$ is a learnable function.

PointNet++ convolution

The convolution operator of *PointNet++* from Section 2.5.5 consists of three layers: Sampling layer, Grouping layer and PointNet layer.

The Sampling layer selects a set of points from the input points which defines the centroids of local regions using the iterative *farthest point sampling* (FPS). It selects the most distant points, in terms of metric, with regard to the rest of points. Grouping layer then constructs local region sets by finding "neighboring" points around the centroids (see Fig. 3.4). PointNet layer uses mini-PointNet networks to encode local region patterns into feature vectors.

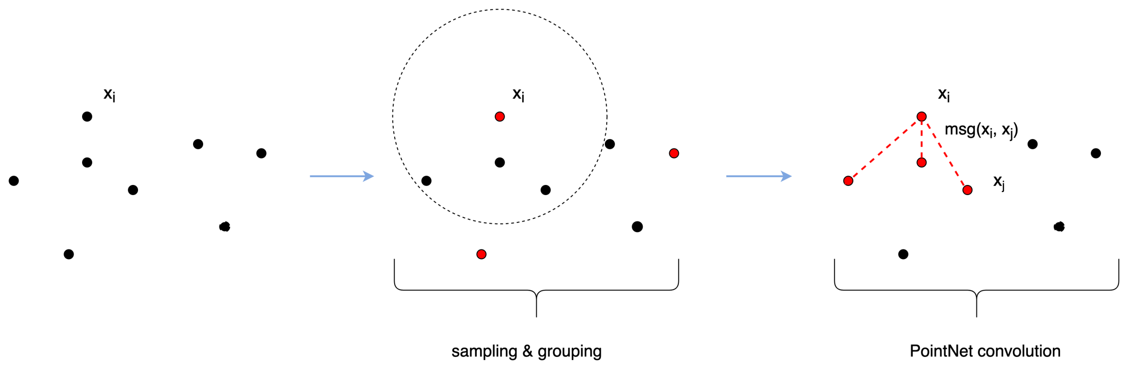


Figure 3.4: *PointNet++* convolution steps. First, *farthest point sampling* algorithm samples points from the point cloud (red nodes in the 2nd step), then *PointNet convolution* on node x_i is performed with points within a certain threshold (dotted circle). Notice how local neighbourhood of node x_i is determined by the dotted circle.

A PointNet layer is defined as follows:

$$\mathbf{x}'_i = \gamma_{\mathbf{\Theta}} \left(\max_{j \in \mathcal{N}(i) \cup \{i\}} h_{\mathbf{\Theta}}(\mathbf{x}_j, \mathbf{p}_j - \mathbf{p}_i) \right) \tag{3.5}$$

where $\gamma_{\mathbf{\Theta}}$ and $h_{\mathbf{\Theta}}$ denote neural networks, *i.e.* MLPs, and $\mathbf{P} \in \mathbb{R}^{N \times F}$ defines the position of each point.

Pooling

For GCN, MoNet and EdgeConv channel-wise mean pooling operator was used. For PointNet++ a channel-wise max pooling was used, to keep it consistent with the implementation in the paper. In general, a *channel-wise pooling operator* for graphs can be defined as:

$$p_i = f(x_i | x \in X) \tag{3.6}$$

where p_i is the pooling output for the i-th channel, X is the set of node features and f is a permutation-invariant function such as: *max*, *mean*, *min*.

Activation Functions

All the activation functions used in the experiment, apart from the final activation, are *Exponential Linear Unit* (ELU) functions. As mentioned in the Section 2.1.4, ELU has better properties than the classic ReLU - according to Clevert at al. it speeds up learning in deep neural networks and leads to higher classification accuracies [10].

The final activation functions are dependent on the task. For regression it is the *identity* function. For classification we apply the *log_softmax* function, defined as:

$$\sigma(z)_i = \log \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \text{ for } i = 1, .., K \tag{3.7}$$

where K is the number of output classes.

Loss Functions

For regression we use the *Mean Squared Error* (MSE) loss, defined as:

$$MSE = \frac{1}{n}\sum_{i=0}^{n}(y_i - \hat{y}_i) \tag{3.8}$$

where y_i is a true value, the $\bar{y}_i$ is a predicted value and n is the number of data points. For classification we use *Negative Log Likelihood* (NLL) loss, defined as

$$NLL = -\sum_{i=1}^{n} y_i \log(\hat{y}_i) \tag{3.9}$$

Note: combination of *log_ softmax* activation function together with NLL loss is equivalent to applying *Cross-Entropy* loss.

3.5 Ensemble method

We also consider an ensemble model, which can utilise information from all of the subcortical brain parts. The high-level overview of our method is presented in Fig. 3.5. Here, the *conv architecture* (a building block) for a given brain part can be replaced by any of the models mentioned above that is: GCN, MoNet, EdgeConv and PointNet++. Each building block returns an embedding of a given brain part, then the embeddings are concatenated into a vector, which is passed to MLP to produce the output.

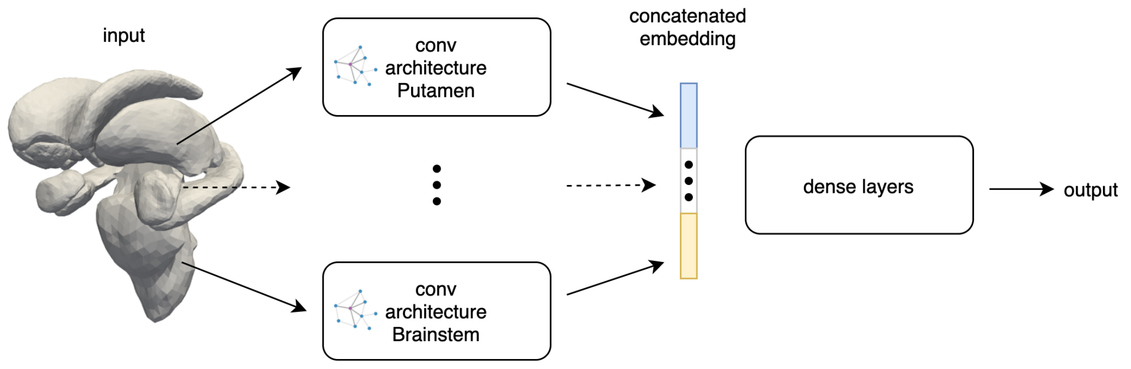


Figure 3.5: Overview of the ensemble method.

3.6 LocalEdgeConv - convolutional operator with translation and rotation invariance property

In the medical setting it is often the case that different datasets are not aligned in the Euclidean space. A common solution is the process of image registration, which transforms data into a common position, scale and orientation. The drawback of image registration is that it is a complicated process with many local minima and therefore it adds an additional layer of complexity to the task.

We can avoid the image registration altogether by using the shape features which are invariant to the location and translation. The only assumption we make is that the shapes have the same scale, which is usually the milimeter space. If data has a different scale, the scan metadata should easily allow to scale any two datasets accordingly.

LocalEdgeConv

In Table 2.2 we described the different possible choices of node features included in the message, in the generic message passing scheme, of the EdgeConv layer. The features suggested in the original paper, presented in the last row of this table, are $[x_i, x_j - x_i]$, where x describes node features and indices i, j specify the nodes exchanging the message. In this setting x_i can be considered a global shape descriptor and the feature difference $x_i - x_j$ can be considered a local shape descriptor. We think that for a setting, where shapes are not pre-registered, using features derived from the local descriptor $x_i - x_j$ can benefit the usability and applicability of the model.

We suggest the following translation and rotation invariant convolutional layer dubbed by us *LocalEdgeConv*, which can be expressed in a generic message passing scheme as:

$$\mathbf{x}'_i = \sum_{j \in \mathcal{N}(i)} h_{\mathbf{\Theta}} \Big(\|\mathbf{d_{j,i}}\|, \angle(\mathbf{r}_i, \mathbf{d_{j,i}}), \angle(\mathbf{r}_j, \mathbf{d_{j,i}}), \angle(\mathbf{r}_i, \mathbf{r}_j) \Big) \tag{3.10}$$

where $\mathcal{N}(i)$ is a dynamically determined k-nearest neighbourhood of the node $\mathbf{x}_i$, $h_{\mathbf{\Theta}}$ is a learnable function, the norm is the L_2 norm, $\angle(.,.)$ is the angle between vectors, $\mathbf{d_{j,i}}$ is the distance between points $\mathbf{x_j}$ and $\mathbf{x_i}$, and $\mathbf{r_i}$, $\mathbf{r_j}$ are reference vectors for the points.

The reference vector $\mathbf{r}$ is any vector which can be uniquely computed from the local neighbourhood of the node $\mathbf{x}$. An example of a reference vector is the normal vector to the surface at the point. In our work we don't use the surface normal as a reference vector because it is not straightforward to compute with the local neighbourhood message-passing scheme. We define a reference vector as follows.

$$\mathbf{r}_i = \frac{1}{k} \sum_{j \in \mathcal{N}(i)} (\mathbf{p}_j - \mathbf{p}_i) \tag{3.11}$$

where $\mathcal{N}(i)$ is a dynamically determined k-nearest neighbourhood of the node $\mathbf{x}_i$, and the $\mathbf{p}_j$ and $\mathbf{p}_i$ are the XYZ coordinates of the points $\mathbf{x}_j$ and $\mathbf{x}_i$ respectively.

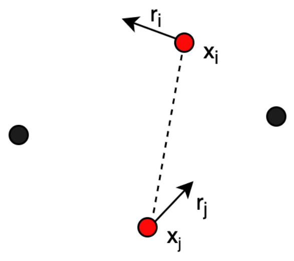


Figure 3.6: Visualisation of translation and rotation invariant features. Let $\mathbf{d_{j,i}}$ be the dotted line. $\|\mathbf{d_{j,i}}\|, \angle(\mathbf{r}_i, \mathbf{d_{j,i}}), \angle(\mathbf{r}_j, \mathbf{d_{j,i}}), \angle(\mathbf{r}_i, \mathbf{r}_j)$ are invariant to any translations and rotations of the point cloud in the embedding space.

Our implementation of LocalEdgeConv operator consists of two message passing steps. In the first step we calculate the reference vectors $\mathbf{r}$ for all the points in the point cloud (see Eq. 3.11). The second step is to perform the actual convolution from the Eq. 3.10. Whenever we refer to the LocalEdgeConv operator we implicitly mean those two steps.

Below is the proposed rotation and translation invariant architecture. The first layer is LocalEdgeConv layer, which is invariant to translation and rotation and works as shows in Fig. 3.7. Then it is followed by the conventional (dynamic) EdgeConv layer with pooling, activation and dense layers producing the final output. This model is used in our evaluation, and we refer to it as *LocalEdgeConv* architecture.

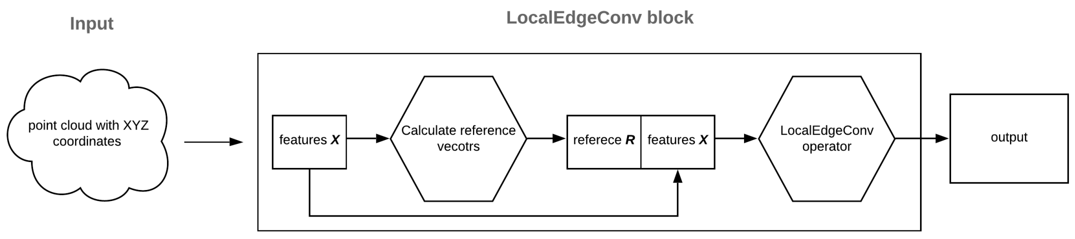


Figure 3.7: *EdgeConvBlock* operator consists of two steps. In the first step the referece vectors are calculated from the raw XYZ coordinates. In the second step the convolution is performed as described in Eq. 3.10. ⬡ stands for a message passing operation and the rectangle stands for the feature matrix.

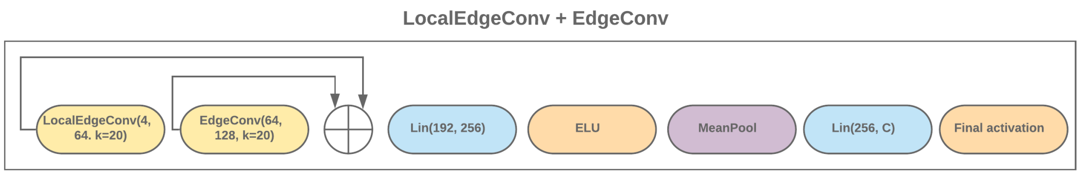


Figure 3.8: Overview of the translation and rotation invariant architecture proposed by us. $\bigoplus$ means tensor concatenation.

Since we are giving up the global shape descriptors there might be a decrease in the expressive power of the original EdgeConv. This potential performance decrease is traded for robustness of this method to translated and rotated data. In the evaluation we will investigate this trade-off further.

3.7 Shape curvature - intrinsic feature with translation and rotation invariance property

Another way to avoid the image registration is to use intrinsic shape features, such as the shape curvature, which do not change with translation or rotation of the object.

We propose a pre-processing pipeline which takes the raw *.vtk* shape and calculates the *mean curvature* at every node (see more in Section 2.3.4). The curvature calculation method can be extended to the raw noisy point cloud data, e.g. data coming from a LiDAR scanner, as described in [9], [18].

We evaluate this approach using this architecture, which is similar to the one used with LocalEdgeConv operator. The input for our model is a point cloud with the mean curvature as the single feature. Note that we do not use the XYZ coordinates in this approach. We expect a drop in the performance because now the local neighbourhood of a point is not based on its position, but solely on the mean curvatures. Remember that for EdgeConv and LocalEdgeConv the local neighbourhood is determined by the k-nearest neighbours algorithm in the feature metric space. Therefore, the patch operator will select the points with similar mean curvatures. The comparison between the patch operators is summarised in the Fig. 4.11.

The main drawback of this method is that the mean curvature must be pre-computed for a given shape in the current setup. Therefore, it makes it less suitable for a real-time systems, such as data coming from the LiDAR sensor.

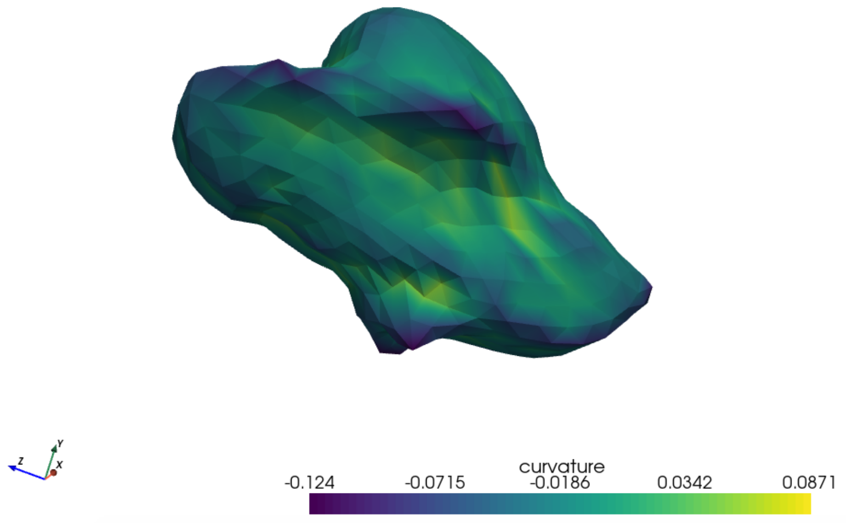
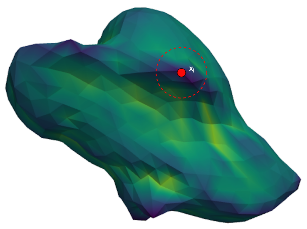
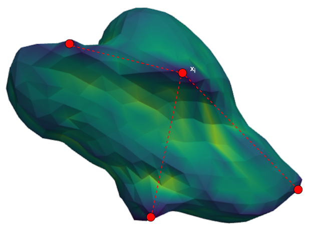


Figure 3.9: Visualisation of the brainstem with the mean shape curvature as a feature.



(a) XYZ coordinates



(b) mean curvature

Figure 3.10: An overview of the differences in k-nn patch operators when using different features. On the left: the patch includes neighbours local in the Euclidean space. On the right: the patch includes points with similar curvatures. The locality is determined by the metric space of the features.

3.8 Image registration

Image registration can transform the data into a common position, scale and orientation. In the Evaluation section we investigate the robustness of the GDL models on the data coming from the Cam-CAN dataset.

The UK Biobank and Cam-CAN scans are expressed in the same milimeter scale, but they are non-aligned in the embedding space. There exists an affine transformation, including translation and rotation, which allows to match those data points. The image registration pipeline used in our work consists of finding a template shape for each part of subcortical brain and then registering Cam-CAN to match this template.

UK Biobank template. There exists a one-to-one vertex correspondence between shapes in the UK Biobank. The template is constructed as follow: vertex positions are the mean of the vertex positions of given brain shape in the dataset, the vertex connectivity is unchanged.

Shape registration. The solution for the registration of multiple 3D shapes uses unit quaternions. It was suggested by Benjemaa et al. [1]. It aligns a given shape to the template shape by optimising rotation and translation parameters. We use Adam optimizer with a learning rate of 0.001. The loss is defined in terms of the point-to-surface distance, suitable for rigid shape registration, which tries to minimize the distance between the output reconstructed points and the input template mesh.

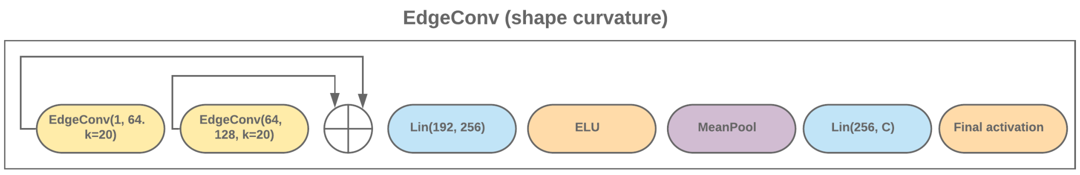


Figure 3.11: Overview of the translation and rotation invariant architecture using the mean curvature as the input. $\bigoplus$ means tensor concatenation.

Since 3D shape registration is a very complex task on its own, our aim is not to find the best registration pipeline. Instead, we treat this simple registration pipeline as a baseline against which we compare our other approaches.

Chapter 4

Evaluation

4.1 Evaluation setup

Each model is trained for 30 epochs with a learning rate of 0.0002, mini-batch gradient descent with a batch size of 16, and Adam [21] optimizer.

Each shape is remeshed to contain a fixed number of vertices: 512. GCN and MoNet use the graph representation of the shape as the input, whereas EdgeConv and PointNet++ treat the shape as a point cloud.

We train separate models for each brain part, as well as ensemble models, which take all parts of the subcortical brain as the input. Our benchmarking tasks are: biological sex classification, age regression, hypertension classification and smoking status classification (See Section 3.2).

We split BioBank dataset into 60/20/20 train, validation, and test subsets to perform parameter tuning and the final evaluation.

As our naive baseline method, we use logistic regression on brain part volumes. We did not include methods based on classical Euclidean convolutions, operating on 3D voxel-based representation of the brain, in our evaluation due to the abundance of different architectures. UK Biobank is a well-researched dataset and there exists work evaluating classical approaches to the brain analysis.

As the last step, we evaluate the robustness of our methods on data coming from a different source: the Cam-CAN dataset (see Section 2.11).

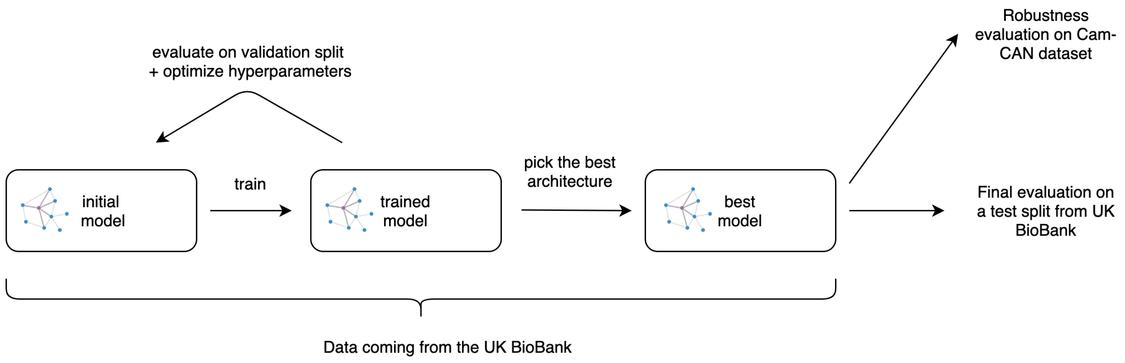

evaluate on validation split
+ optimize hyperparameters
initial model
train
trained model
pick the best architecture
best model
Robustness evaluation on Cam-CAN dataset
Final evaluation on a test split from UK BioBank
Data coming from the UK BioBank

Figure 4.1: A high-level overview of the training and evaluation pipeline.

4.2 Models trained on a single brain part

In this section, we overview the performance of the models trained on a single subcortical brain part.

4.2.1 Biological sex classification

In the task of biological sex classification, the Geometric Deep Learning methods substantially outperformed logistic regression on brain volumes. Also, this evaluation showed that GDL methods operating on point clouds perform better on average than methods treating the object as a graph with XYZ features. The Thalamus proved to be the brain part most useful for this task, with *PointNet++* architecture achieving 83.7% and *EdgeConv* achieving 82.6% accuracy. Other parts, like Brainstem and Caudate, also had similar results. In general, *PointNet++* architecture appeared to be the best, which is probably due to the fact that it uses sampling and grouping layers before each convolutional layer. These extra steps allow PointNet++ to extract the most useful global shape descriptors. Brainstem, Thalamus, and Caudate are responsible for reward function, sleep/wakefulness control, and the conduction of information, therefore it would be interesting to research whether the underlying geometrical differences have any tangible impact on the psychological differences between sexes.

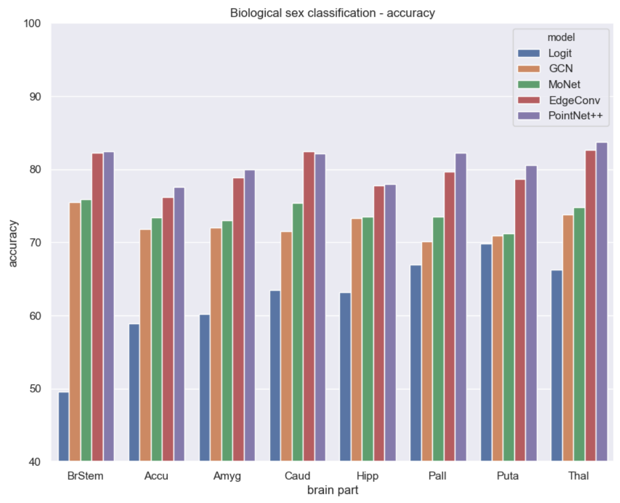


Figure 4.2: Biological sex classification accuracy.

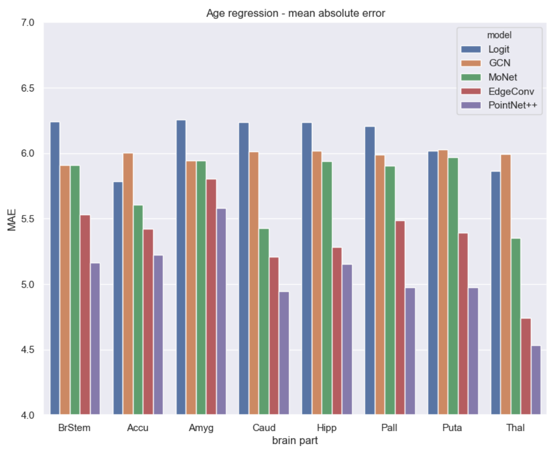
4.2.2 Age regression

For brain age regression, the point cloud based architectures performed significantly better than the rest, with (again) *PointNet++* performing the best in terms of all metrics considered – MSE, MAE and, R^2 correlation coefficient. The best MAE, of 4.53 years, was achieved on the model trained on the Thalamus. See Fig. 4.3 for a full summary of results.

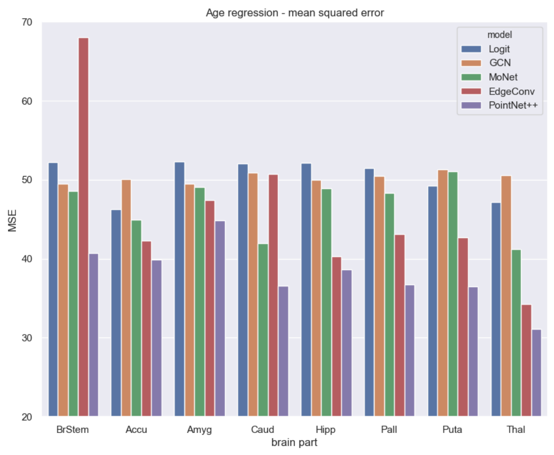
4.2.3 Hypertension classification

For the classification of hypertension, we split the dataset into 2 stratified subsets containing male and female subjects, each with similar age distribution. We consider male and female subjects separately, because of the noticeable differences in brain geometry between sexes (see Section 4.2.1).

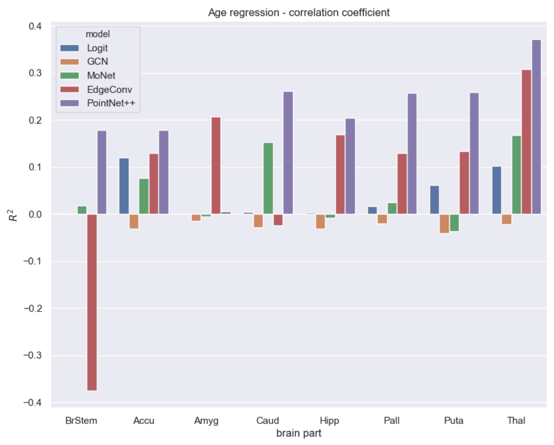
For the male subset of subjects, all models had problems with finding the geometrical traits of an advanced hypertension stage. The best result was obtained on Thalamus with



(a) Mean Absolute Error



(b) Mean Squared Error



(c) R^2 Correlation Coefficient

Figure 4.3: Evaluation metrics of age regression.

EdgeConv, yielding a classification accuracy of 57.3%. For the female subset models, we were able to better distinguish the features correlated with hypertension, especially in brain parts like the Accumbens, Caudate, Pallidus, and Thalamus. Models based on point cloud representations performed better than other models, the best result was on the Pallidus with the *PointNet++* architecture – yielding an accuracy of 61.2%.

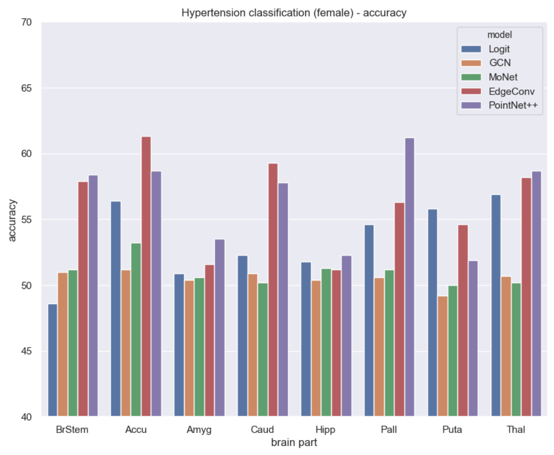
4.2.4 Smoking status classification

For the smoking status classification, we split the dataset into 2 stratified subsets containing male and female subjects, each with a similar age distribution.

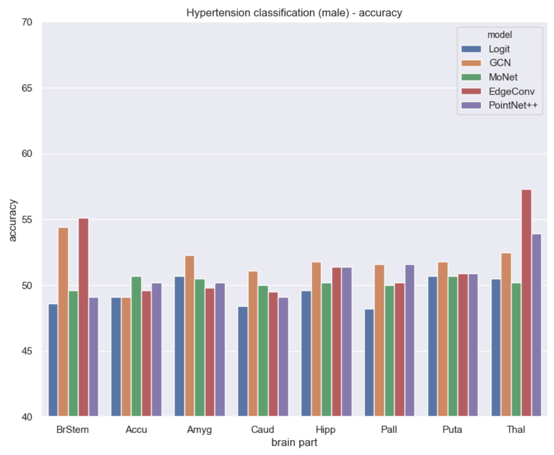
In this setting, male subjects exhibit more traits correlated to being a long-term smoker. We observed a greater variance in the performance of GDL methods – sometimes, they performed marginally worse than the baseline method. One possible explanation is that long-term smoking does not have a substantial effect on the geometrical features of the subcortical brain.

4.2.5 Single brain part models summary

In our evaluation, the GDL models outperformed the baseline volume-based logistic regression. Among the GDL models, the ones operating on the point cloud representation of a shape were able to achieve significantly better results than the graph-based models,

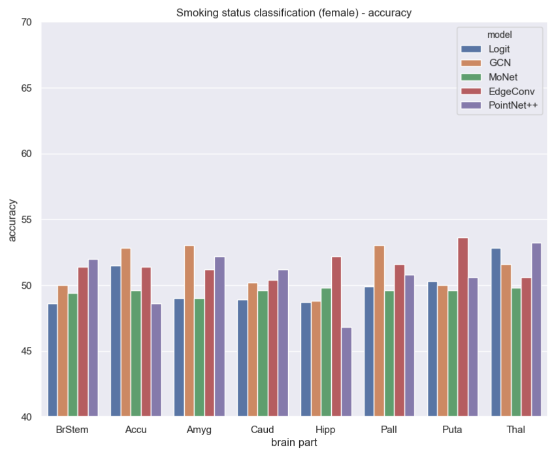


(a) Female subjects.

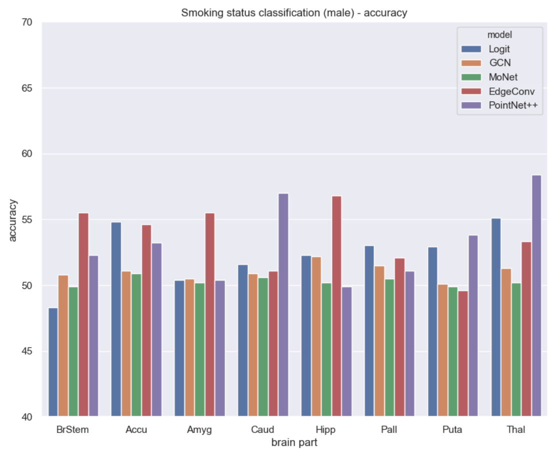


(b) Male subjects.

Figure 4.4: Hypertension status classification.



(a) Female subjects.



(b) Male subjects.

Figure 4.5: Smoking status classification.

which used triangular-mesh representation. We think that the main reason for the performance difference is that point cloud convolutions use patch operator with arbitrarily larger receptive field (e.g. k nearest neighbours, where k is a parameter). On the other hand, the graph representation models use the adjacency matrix to determine the local neighbourhood, which is a computationally faster approach yet it limits the performance of such methods. Table 4.1 summarizes the comparison of the average receptive field, expressed in terms of neighbourhood size, of each method.

We see strong dependence between the method performance and the size of the receptive field of the shapem, and therefore argue that point cloud based methods are more suitable techniques for tasks involving learning on geometrical structures.

4.3 Ensemble models

In this question we evaluate if there is any performance gain when using the ensemble model. That is, a model which learns on *all* the parts of the subcortical brain, and not just a single part.

For this evaluation we only consider the best performing methods (EdgeConv and PointNet++) as well as the baseline logistic regression model. We evaluate the ensemble

Method	Average neighbourhood size
GCN	5.98 (fixed)
MoNet	5.98 (fixed)
EdgeConv	constant 20 (parameter)
PointNet++	upper bounded by 64 (parameter)

Table 4.1: Comparison of the average neighbourhood size across the models. In the graph representation (triangular mesh), every node has on average 5.98 neighbours. For point cloud methods the neighbourhood size can vary. For EdgeConv, we used used k-nn algorithm, with k equal to 20, to determine the neighbourhood as suggested in the original paper. For PointNet++ we took up to 64 neighbours within certain distance from the node, this was also a default parameter in the original paper.

methods on the biological sex classification and age regression tasks.

Model	Accuracy	Change
Logit	73.3%	+3.5%
EdgeConv	83.2%	+0.6%
PointNet++	**84.3%**	+0.4%

Table 4.2: Performance summary of ensemble models for sex classification. The "Change" column represents the change over the best performing single brain part model.

Model	MAE	Change in MAE
Logit	5.02	-0.75
EdgeConv	4.86	+0.12
PointNet++	**4.74**	+0.21

Table 4.3: Performance summary of ensemble models for age regression. The "Change in MAE" column represents the change over the best performing single brain part model.

The data in Table 4.2 and 4.3 suggests that using an ensemble of models is not definitively better than using single-brain-part models. For biological sex classification, the performance gain is noticeable for all models: Logit, EdgeConv, and PointNet++. The gain varies between 0.4% and 3.5%. For age regression, Logit's performance increased the most with the MAE lowered by 0.75 years. Interestingly, EdgeConv and PointNet++ achieved worse results than the ones achieved in the single-brain-part setting. We think that 30 epochs might be not enough for those models to converge given the number of parameters.

4.4 Cam-CAN evaluation

In order to evaluate the robustness of the models trained on the UK Biobank dataset, we perform an additional test on the Cambridge Centre for Ageing and Neuroscience (Cam-CAN) dataset. Each subject in the Cam-CAN dataset has a feature set containing age and biological sex, therefore we will evaluate our models on sex classification and age regression tasks.

Cam-CAN dataset contains subjects with age ranging from 18 years to 88 years, whereas the UK Biobank (UKBB) has subjects ranging from 44 years to 73 years. Also, the age distribution is different between those two datasets (see Fig. 4.6). We will evaluate how

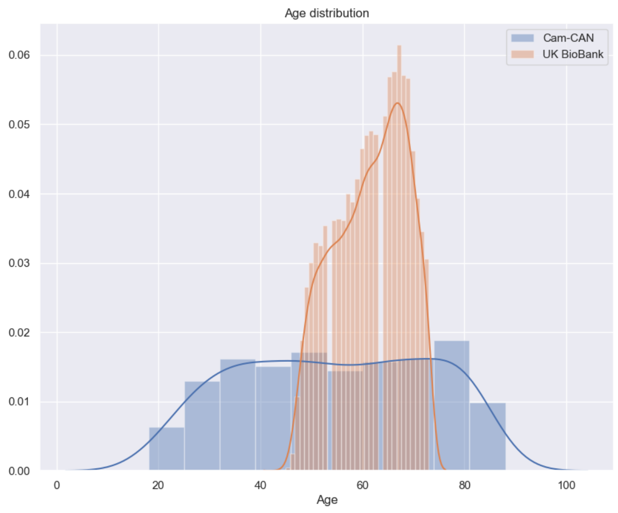


Figure 4.6: Age distribution in Cam-CAN and UK Biobank datasets.

well our models can generalize to unseen data. That is, subjects with age outside of UK Biobank's age range.

One of the important tasks when dealing with data coming from a different dataset is image registration, which is the process of transforming different sets of data into a common position, scale, and orientation. It adds a layer of complexity when building models, as it can be the case that shape registration can create some unwanted noise within the dataset. This is due to the fact that the loss function for shape registration has multiple local minima; so there is no guarantee that the dataset variance will be preserved, as sometimes the optimizer can get stuck in one of the local minima.

Without any shape pre-registration, we evaluated the sex classification models (trained on UKBB) on Cam-CAN and the accuracy was no better than a coin flip. The reason is that Cam-CAN and UK Biobank are not aligned, see Fig. 4.7, and therefore some form of shape registration is required.

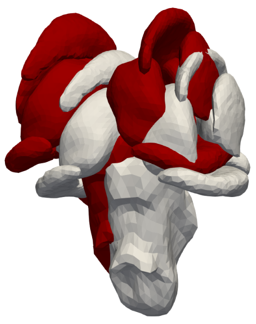


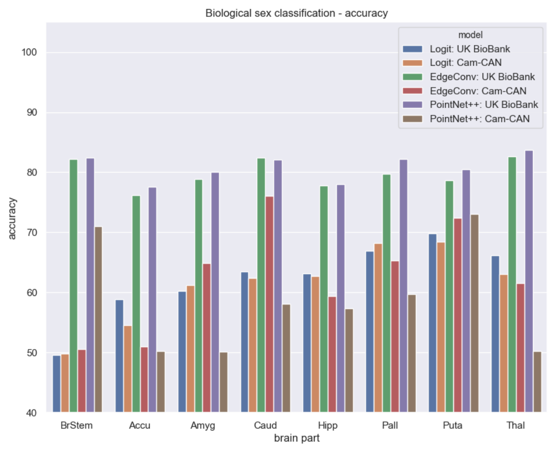
Figure 4.7: Visualisation of shape misalignment between scans coming from the UK Biobank (white) and Cam-CAN (red) datasets.

In this evaluation, we will take two approaches. The first one would use a UK Biobank template to align the Cam-CAN dataset accordingly (more details on image registration are present in Section 3.8). The second approach would make use of local, intrinsic shape descriptors, which are independent of shape location and rotation in the embedding space, see Section 3.6.

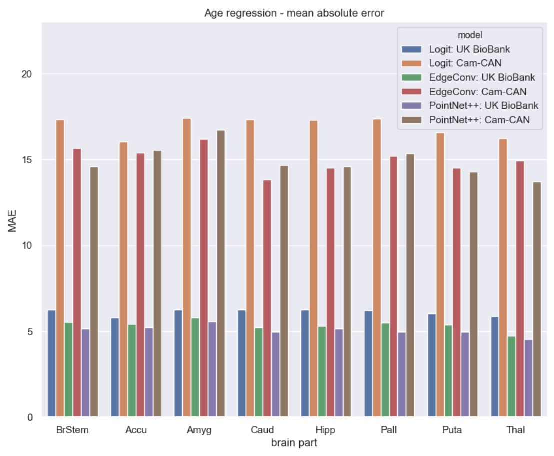
Evaluation With Image Registration Pipeline

For this evaluation, we take two point cloud architectures, EdgeConv and PointNet++, which were the best-performing models in the previous tests.

The results in Fig. 4.8 suggest that the models trained on the UK Biobank and evaluated on the preregistered Cam-CAN dataset suffer from a significant performance drop, e.g. ≈20%, on the task of sex classification. This tendency is consistent across all the models and brain parts that we performed the evaluation on. There are two possible explanations for the performance drop. On the one hand, our models generalize poorly to data coming from a dataset with a vastly different age distribution. On the other hand, the complicated shape registration pipeline might be adding some unaccounted noise to the data and therefore hinder the models' performance.



(a) Biological sex classification accuracy.



(b) Age regression MAE.

Figure 4.8: Biological sex classification and age regression evaluation on pre-registered Cam-CAN data.

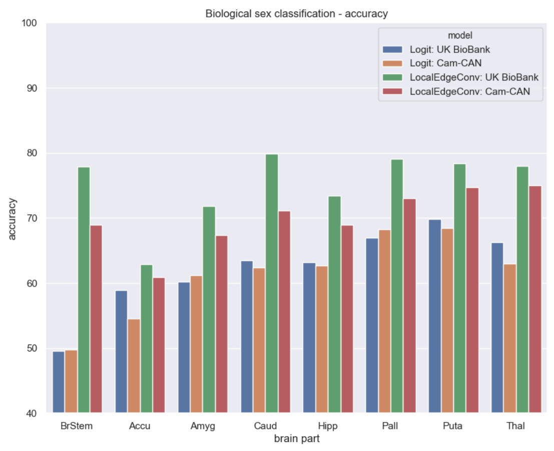
Evaluation With LocalEdgeConv Operator

Using local shape descriptors as features can benefit the robustness of the model. In our scenario, we use *LocalEdgeConv* for local feature learning, which helps us avoid doing any registration. For this evaluation, we train the LocalEdgeConv models on the UK Biobank dataset, and perform the evaluation on both the UKBB and the Cam-CAN. We use volume-based logistic regression as the baseline as it also uses rotation and translation invariant feature: shape volume.

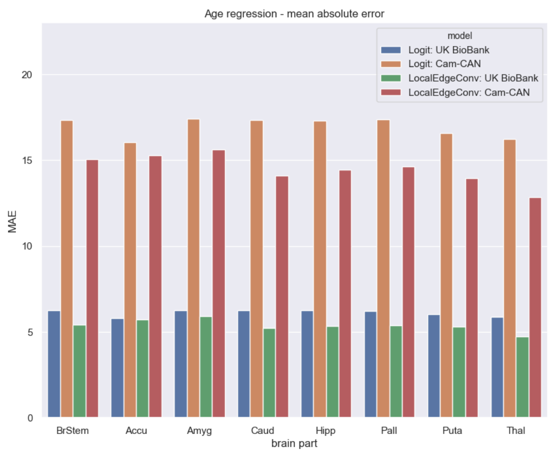
In our evaluation, summarized in Fig. 4.9, we see a much smaller performance drop in sex classification, which is ≈5% vs ≈20% when using simple image registration. We think that the small drop in performance can be attributed to the difference in age distribution across datasets and the fact that Cam-CAN dataset's age range is from 18 years to 88 years, whereas the UK Biobank is from 44 years to 73 years.

The task of age regression shows an expected performance drop. The MAE achieved on the UKBB test split was 4.7-5.4 years, whereas the MAE on Cam-CAN was ≈14.8 years. In our setting, we ask our model to extrapolate its predictions to the data far outside of the training data range. Since ML models are data-driven, it is to no surprise that our model does not perform well on this task (see Fig. 4.6 to see the age distribution difference).

For this evaluation, we conclude that the smaller performance drop in the biological sex classification task shows that our model was able to learn sex-related geometrical features which are not strictly correlated with the age of the subject. We also see that the models



(a) Biological sex classification accuracy.

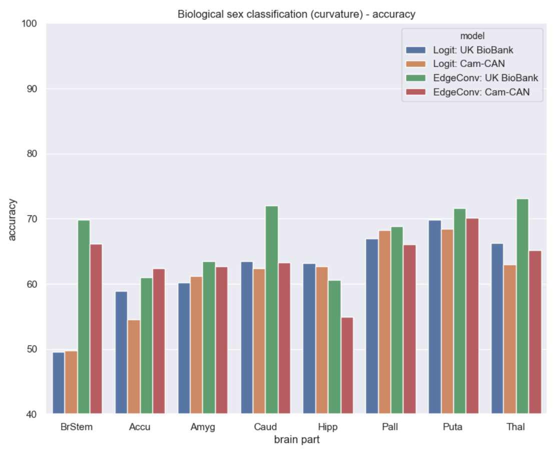


(b) Age regression MAE.

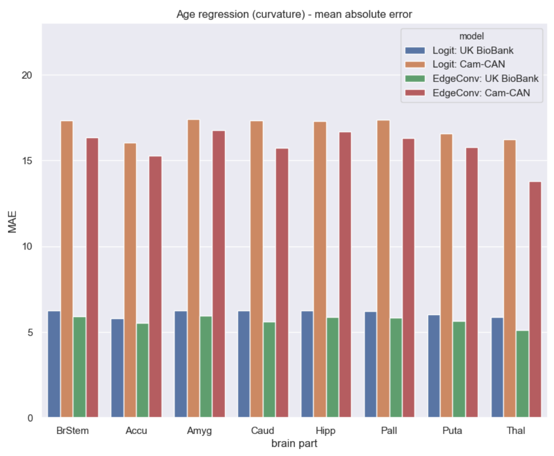
Figure 4.9: Biological sex classification and age regression evaluation on Cam-CAN data using logistic regression and LocalEdgeConv operator.

trained with LocalEdgeConv operator performed significantly better than the ones which used global shape descriptors and the shape registration pipeline.

Evaluation with shape curvature as the input feature



(a) Biological sex classification accuracy.



(b) Age regression MAE.

Figure 4.10: Biological sex classification and age regression evaluation on Cam-CAN data using logistic regression and EdgeConv operator which uses the shape curvature as the input feature.

Another way to avoid the need for shape registration is to use the intrinsic shape features. Intrinsic features are robust to perturbations in the orientation and the position of the shape. For this evaluation, we use the EdgeConv model, which uses the mean curvature as the single feature of a point. We train the model on the UK Biobank dataset and perform the evaluation on both the UKBB and the Cam-CAN. We use volume-based logistic regression as the baseline as it also uses rotation and translation invariant feature: shape volume.

The results show that this method, for the sex classification task, performs on average worse than LocalEdgeConv, for both UKBB and Cam-CAN, by about 6%. Interestingly the performance difference, of this method, between the UKBB and Cam-CAN is about

5%, the same as for LocalEdgeConv. For the age regression task this method achieved for UKBB ≈5.7 years MAE and for Cam-CAN ≈15.8 years MAE, both worse than the results achieved by LocalEdgeConv. It shows that LocalEdgeConv makes use of intrinsic features, which are more useful than the mean curvature of the shape. It also means that the on-the-fly computed intrinsic features used by LocalEdgeConv are more meaningful than the pre-computed shape curvature. The architectures and the number of parameters are largely the same for both: LocalEdgeConv and EdgeConv with the curvature as a feature.

4.5 LocalEdgeConv benchmark - ModelNet40

We evaluate the *LocalEdgeConv* model on the ModelNet40, from Wu et al. [41], classification task, consisting of predicting the category of a previously unseen shape. The dataset contains 12,311 meshed CAD models from 40 categories. 9,843 models are used for training and 2,468 models for testing. We follow the experimental settings of Qi et al. [32]. For each model, 1,024 points are uniformly sampled from the mesh faces. The point cloud is rescaled to fit into the unit sphere. Only the (x, y, z) coordinates of the sampled points are used, and the original meshes are discarded.

We use mini-batch gradient descent with Adam optimizer with the learning rate of 0.0005. The batch size is set to 20. LocalEdgeConv has the local neighbourhood limited to $k = 20$ neighbours. We train the models for 50 epochs.

In our evaluation, we use graph classification architectures with two convectional layers as described in the previous section (see Fig. 3.2 and Fig. 3.8). Convolutional layers from EdgeConv and PointNet++ are used in the state-of-the-art architectures [38], [32]. Note that the results achieved by us differ slightly from the reported results: EdgeConv 92.9% and PointNet++ 90.7%.

Model	Overall accuracy
EdgeConv	88.2%
PointNet++	**88.9%**
LocalEdgeConv	71.4%

Table 4.4: Classification results on ModelNet40.

EdgeConv and PointNet++ perform substantially better than LocalEdgeConv in this scenario. It is because EdgeConv and PointNet++ use global shape descriptors which are more powerful than the local, intrinsic shape descriptors used by LocalEdgeConv.

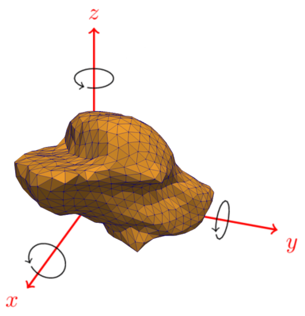
In the next step, we evaluate how the models, trained on the standard ModelNet40 dataset, perform when shapes are randomly rotated and shifted in the embedding space. The results are summarized in Table 4.5.

Model	Overall accuracy
EdgeConv	9.3%
PointNet++	11.3%
LocalEdgeConv	**70.2%**

Table 4.5: Classification results on ModelNet40. Each shape is randomly rotated and shifted.

The results support our claim that LocalEdgeConv is robust to perturbations in the position and the orientation of the shape. LocalEdgeConv significantly outperforms other methods in this setting, achieving 70% accuracy vs 9-11% accuracy of other methods.

(a) image



(b) 3D shape

Figure 4.11: Rotation axis of a 2D image and a 3D shape. The added degrees of freedom in higher dimensions cause the number of possible rotations to increase.

Data augmentation, like the rotation, can benefit the performance of other methods considered. Rotating 2D images is a commonly used technique which improves the computer vision models' robustness. Shapes are embedded in the 3D Euclidean space and it means that rotations have an added degree of freedom. It implies that the 3D data augmentation becomes increasingly more expensive. To visualise the issue let us consider the task in which we augment an image with rotation increments of 10°. Since we rotate the image along one axis, for every image we will have 35 additional rotated images (with rotation increments of: 10°, 20°, 30°, ...), so in total 36 images. 3D shape can be rotated along 3 axes, therefore the number of total possible orientations, with 10° rotation increment, is equal to $36^3 = 46,656$. This makes 3D data augmentation an infeasible task, the curse of dimensionality. We think that our model, LocalEdgeConv, addresses this issue well.

LocalEdgeConv avoids the need to pre-align the point cloud representation of the shape. We showed that it can not only model human brains but also classify regular objects such as a person, a car, and a bench (categories of ModelNet40). Data coming from the LiDAR scanners, equipped in autonomous vehicles and some drones, is used for geospatial mapping of the surroundings, e.g. the surroundings of the car on a street. We think that our technique has a potential be applied in such scenarios to classify and detect objects, as the incoming data is not pre-aligned.

Chapter 5

Applications

In this chapter, we investigate potential applications for our proposed convolutional operator. Our evaluation on the ModelNet40 dataset has shown that LocalEdgeConv can make models robust to perturbations in the orientation and the position of the 3D shape (see Section 4.5).

5.1 Shape representation learning

One application of LocalEdgeConv we have in mind is building a *shape2vec* model for the extraction and representation of shape features in the latent space. Nowadays, physicians use properties such as volume and diameter to describe, e.g. a tumor. Those features are useful for human medical experts when deciding, e.g. on the kind of treatment for tumors. But for an ML model, using a learned representation of a shape instead of human-readable features might increase the models' performance. Therefore, we suggest building a *shape2vec* model which would be able to learn the representation of 3D shapes such as tumors.

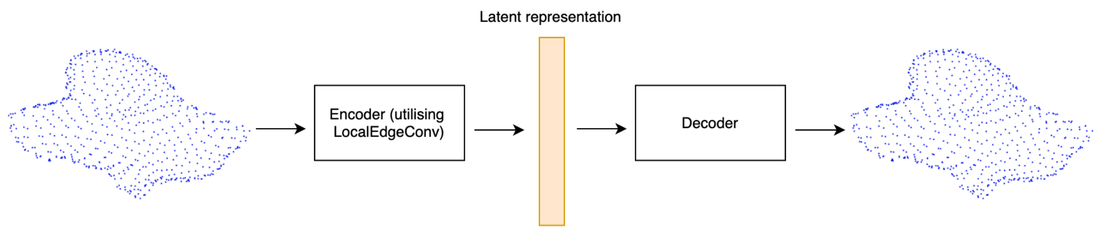


Figure 5.1: Proposed architecture of the shape2vec model.

The *shape2vec* model can be implemented as an autoencoder, trained on point cloud representation of a shape, which utilises LocalEdgeConv in the encoding step.

5.2 Photogrammetry

Photogrammetry is the science of making measurements from photographs. The input to photogrammetry are photographs, and the output is typically a map or a 3D model of some real-world object or scene. Many of the maps we use today are created with photographs taken from an aircraft.

A model can be built, using LocalEdgeConv, to classify objects in the 3D representation of a scene, e.g. a street. In this scenario, a number of points are sampled from the 3D scene. We can use LocalEdgeConv for object detection, implemented as a point classification task. For object detection, we want to predict whether a given point is the part of an object

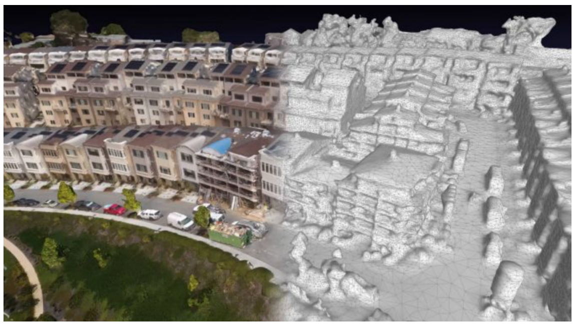


Figure 5.2: Street map obtained using the photogrammetry technique. Image taken from the DroneDeploy website [13].

or not. It is similar to finding a bounding box in the classical object detection in an image. Once we have a point cloud representation of the detected 3D shape, we can use an architecture similar to the one in Section 3.6, to perform object classification and detect, e.g. people and cars on the street.

5.3 Geospatial mapping

Geospatial mapping is a spatial analysis technique that typically employs systems capable of capturing and processing spatial data. A common sensor that can capture the spatial data is the LiDAR scanner. Since LiDAR scanners are commonly used in autonomous vehicles, the detection of objects such as pedestrians, cars, or trees is an important task.

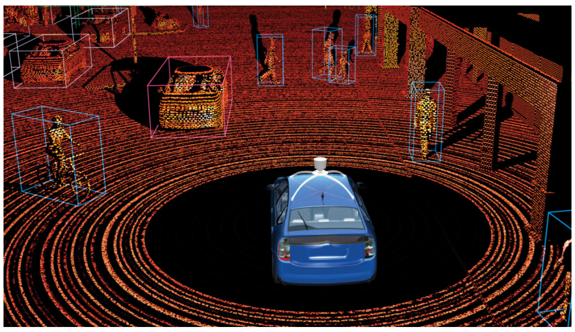


Figure 5.3: LiDAR scanner output representation the surroundings [20].

In this setting, we can use similar architecture to the one used for photogrammetry. A set of points needs to be sampled from the data captured by the LiDAR scanner. Then we need to perform two steps: object detection and object classification.

Chapter 6

Conclusion and Future Work

We have successfully implemented geometric deep learning models for brain shape analysis and achieved better than baseline results on two datasets: UK Biobank and Cam-CAN.

This was done by surveying and benchmarking a few different architectures and selecting the best ones. We applied a generic graph classification architecture with different convolutional operators and were able to compare the expressiveness of each setup. It can be observed that models treating shapes as point clouds achieve markedly better performance than methods treating shapes as meshes (e.g. in biological sex classification, $\sim$10% difference in the accuracy). We also noticed that shape registration creates an additional layer of complexity when building models aimed to be robust for data coming from different sources. Therefore; we proposed a new convolutional operator, dubbed LocalEdgeConv. LocalEdgeConv is robust to perturbations in the embedding space of the shape, making it rotation and translation invariant. Our experimental evaluation on the data, performed on two different datasets (UK Biobank and Cam-CAN), showed that the model can generalize well for sex classification. For the age regression, we attribute the performance loss to the very different age distributions between the datasets, with some age groups not represented at all in the UK Biobank. We also benchmarked LocalEdgeConv on the ModelNet40 dataset and achieved better than the state-of-the-art results in certain settings of the problem.

Our work can be considered as a pioneer in applying Geometric Deep Learning for brain shape analysis. We used the properties of convolutions on non-Euclidean domains to our advantage, by designing an operator which avoids the need of shape registration.

We believe there are more applications in the medical world, other than brain shape analysis, that would benefit from our findings. One of the applications we have in mind is building a shape2vec model for the extraction and representation of shape features. Nowadays, physicians use properties such as volume and diameter to describe (say) a tumor. We think that shape2vec model would be able to extract more meaningful shape features, which can be later used for other modelling tasks.

Our results are not constrained to medical settings only. We see a lot of potential in applying our findings to a wide range of object classification tasks. One those tasks is classifying data coming from LiDAR scanners for the geospatial mapping. In such scenarios, the point cloud data does not come pre-aligned. LocalEdgeConv, an operator robust to perturbations in location and orientation, can help existing systems detecting cars or people on the streets.

Although we managed to achieve good results, there is still a lot of room for improvements before our methods are deployed in systems dealing with real-world medical problems.

6.1 Future work

Even though we were able to achieve very good results with our models, there is a potential for future work. The performance of machine learning models usually reflects the quality of the training data. We think that the expansion of the UK Biobank dataset would be beneficial, so that it includes subjects from underrepresented age groups. This would be crucial for developing better models. One interesting area to research in greater detail is data augmentation, as (for conventional computer vision applications) this can usually yield substantial increases in the performance and generalizability of the model. Another key area of improvement is refining the model design, by potentially employing some ideas from deep convolutional neural networks (e.g. residual connections).

6.1.1 Data augmentation

Although there exist many standard data augmentation techniques for classical 2D computer vision problems (like cropping, rotation, horizontal and vertical flipping), there are no universal methods to augment 3D data such as shapes. Liu et al. [25] proposed a new 3D shape dataset augmentation method by learning the deformation between shapes in a highly reduced latent space, while affording interactive control of shape generation. Applying their findings to augment the UK Biobank dataset could potentially further improve the performance of the models.

6.1.2 Model design

In our work, we defined a generic graph classification architecture with two convolutional layers, pooling, and dense layers. All the models considered in our work are defined in terms of generic graph classification architecture. Alas, experimentation with deeper networks with skip connections, like in ResNet, can potentially improve the performance as well.

Bibliography

[1] Raouf Benjemaa and Francis Schmitt. A solution for the registration of multiple 3d point sets using unit quaternions. In *European Conference on Computer Vision*, pages 34–50. Springer, 1998.

[2] Dhruv Bhate, Clint Penick, Lara Ferry, and Christine Lee. Classification and Selection of Cellular Materials in Mechanical Design: Engineering and Biomimetic Approaches. *Designs*, 3(1):19, 2019.

[3] D. Boscaini, J. Masci, E. Rodolà, M. M. Bronstein, and D. Cremers. Anisotropic diffusion descriptors. *Computer Graphics Forum*, 35(2):431–441, 2016.

[4] Davide Boscaini, Jonathan Masci, Emanuele Rodolà, and Michael Bronstein. Learning shape correspondence with anisotropic convolutional neural networks. *Advances in Neural Information Processing Systems*, (Nips):3197–3205, 2016.

[5] Brilliant.org. Convolutional neural network. `https://brilliant.org/wiki/convolutional-neural-network/`, 2020. [Retrieved 12:13, January 15, 2020].

[6] Michael Bronstein. Intrinsic deep learning on manifolds (spectral cnns, geodesic cnns, anisotropic cnns, mixture model networks, embedding-based techniques). University Lecture, 2017.

[7] Michael M. Bronstein, Joan Bruna, Yann Lecun, Arthur Szlam, and Pierre Vandergheynst. Geometric Deep Learning: Going beyond Euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

[8] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral Networks and Locally Connected Networks on Graphs. pages 1–14, 2013.

[9] Yueqi Cao, Didong Li, Huafei Sun, Amir H Assadi, and Shiqiang Zhang. Efficient curvature estimation for oriented point clouds. *arXiv preprint arXiv:1905.10725*, 2019.

[10] Djork Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (ELUs). *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, pages 1–14, 2016.

[11] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in Neural Information Processing Systems*, (Nips):3844–3852, 2016.

[12] L. Dora, S. Agrawal, R. Panda, and A. Abraham. State-of-the-art methods for brain tissue segmentation: A review. *IEEE Reviews in Biomedical Engineering*, 10:235–249, 2017.

[13] DroneDeploy. Dronedeploy.com, 2020. [Online; accessed 08-June-2020].

[14] Timothy C et al. Durazzo. Cigarette smoking is associated with amplified age-related volume loss in subcortical brain regions.

[15] Matthias Fey and Jan Eric Lenssen. Fast Graph Representation Learning with PyTorch Geometric. (1):1–9, 2019.

[16] Moonen JE et al. Foster-Dingley JC, van der Grond J. Lower blood pressure is associated with smaller subcortical brain volumes in older persons.

[17] P Gainza, F Sverrisson, F Monti, E Rodolà, D Boscaini, M M Bronstein, and B E Correia. Deciphering interaction fingerprints from protein molecular surfaces using geometric deep learning. *Nature Methods*, 2019.

[18] Paul Guerrero, Yanir Kleiman, Maks Ovsjanikov, and Niloy J Mitra. Pcpnet learning local shape properties from raw point clouds. In *Computer Graphics Forum*, volume 37, pages 75–85. Wiley Online Library, 2018.

[19] http://www.ukbiobank.ac.uk/. Uk biobank. `http://www.ukbiobank.ac.uk/about-biobank-uk/`, 2020. [Retrieved 18:13, January 19, 2020].

[20] Tony Kerr. How autonomous cars map the environment, 2020. [Online; accessed 08-June-2020].

[21] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pages 1–15, 2015.

[22] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. pages 1–14, 2016.

[23] Yann Lecun, Yoshua Bengio, and Geoffrey Hinton. Deep learning, 2015.

[24] Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.

[25] Jiarui Liu, Qing Xia, Shuai Li, Aimin Hao, and Hong Qin. Quantitative and flexible 3D shape dataset augmentation via latent space embedding and deformation learning. *Computer Aided Geometric Design*, 71:63–76, 2019.

[26] Jonathan Masci, Davide Boscaini, Michael M Bronstein, and Pierre Vandergheynst. Geodesic Convolutional Neural Networks on Riemannian Manifolds. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2015-Febru, pages 832–840, 2015.

[27] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodolà, Jan Svoboda, and Michael M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model CNNs. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017-Janua:5425–5434, 2017.

[28] Federico Monti, Michael Bronstein, and Xavier Bresson. Geometric matrix completion with recurrent multi-graph neural networks. In *Advances in Neural Information Processing Systems*, pages 3697–3707, 2017.

[29] Lia Morra, Silvia Delsanto, and Loredana Correale. *Artificial Intelligence in Medical Imaging: From Theory to Clinical Practice.* 11 2019.

[30] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[31] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017-Janua:77–85, 2017.

[32] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in Neural Information Processing Systems*, 2017-Decem:5100–5109, 2017.

[33] Steven Rosenberg. pages vii–x. London Mathematical Society Student Texts. Cambridge University Press, 1997.

[34] Jason R. Taylor, Nitin Williams, Rhodri Cusack, Tibor Auer, Meredith A. Shafto, Marie Dixon, Lorraine K. Tyler, Cam-CAN, and Richard N. Henson. The Cambridge Centre for Ageing and Neuroscience (Cam-CAN) data repository: Structural and functional MRI, MEG, and cognitive data from a cross-sectional adult lifespan sample. *NeuroImage*, 144:262–269, 2017.

[35] Jana Vasković. Subcortical structures, 2020.

[36] Kirill Veselkov, Guadalupe Gonzalez, Shahad Aljifri, Dieter Galea, Reza Mirnezami, Jozef Youssef, Michael Bronstein, and Ivan Laponogov. Hyperfoods: Machine intelligent mapping of cancer-beating molecules in foods. *Scientific Reports*, 9:1–12, 2019.

[37] Yanpei Wang, Qinfang Xu, Jie Luo, Mingming Hu, and Chenyi Zuo. Effects of age and sex on subcortical volumes. *Frontiers in Aging Neuroscience*, 11(SEP):1–12, 2019.

[38] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph Cnn for learning on point clouds. *ACM Transactions on Graphics*, 38(5), 2019.

[39] Z. Wang, Y. Sun, Q. Shen, and L. Cao. Dilated 3d convolutional neural networks for brain mri data classification. *IEEE Access*, 7:134388–134398, 2019.

[40] Wikipedia. Fabula AI — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Fabula%20AI&oldid=947615277`, 2020. [Online; accessed 06-June-2020].

[41] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[42] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A Comprehensive Survey on Graph Neural Networks. XX(Xx):1–22, 2019.

[43] Wayne Zachary. An information flow model for conflict and fission in small groups1. *Journal of anthropological research*, 33, 11 1976.

[44] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, pages 4438–4445, 2018.

[45] Qianru Zhang, Meng Zhang, Tinghuan Chen, Zhifei Sun, Yuzhe Ma, and Bei Yu. Recent advances in convolutional neural network acceleration. *Neurocomputing*, 323:37–51, 2019.

[46] Martin Zlocha, Qi Dou, and Ben Glocker. Improving RetinaNet for CT Lesion Detection with Dense Masks from Weak RECIST Labels. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11769 LNCS:402–410, 2019.