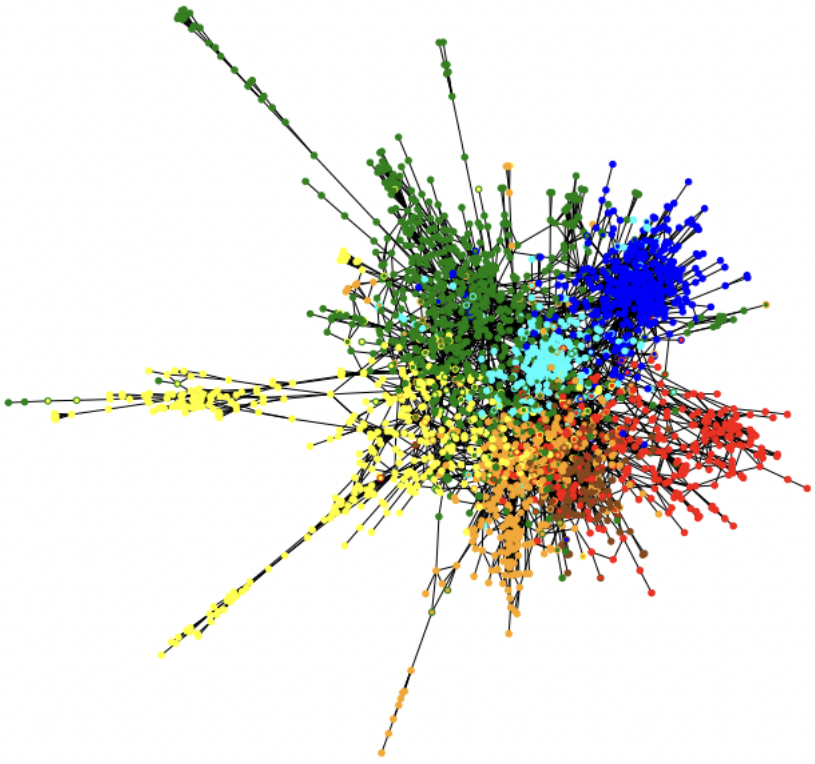
Geometric Deep Learning Course : Lecture Notes

taught by Michael M. Bronstein, Joan Bruna, Taco Cohen, Petar Veličković

Giancola Simone Maria[1†]

[†]Bocconi University, Milan

July 14, 2022



[1]simonegiancola09@gmail.com

Abstract

Geometric Deep Learning (GDL) is a fast emerging field of research. Where Deep Learning fails to present a unifying framework, GDL is a valid proposal for a theoretical framework under high dimensional learning. This document is a collection (hopefully in continuous expansion) of the Lecture Notes of the course held at the 2021 African's Master in Machine Intelligence (AMMI). Recordings are available online [Vel21], and a book is currently being written [Bro+21]. To craft such a framework notions from Statistics, Group Theory, Geometry, Fourier Analysis, Continuous and Discrete Spaces are beautifully aligned together. The result is a principled and contained collection of basics from which many architectures can be derived. Up to my knowledge, this is the first set of lecture notes from the course. In the future, I plan to complete it in its entirety, and enlarge it with deeper theoretical insights on the methods referenced, or with the help of additional material available (there are recorded seminars [Vel21], and an upcoming Summer School in Pescara, Italy).

Disclaimer 1 The document is subject to major updates. There are 50 **TODO** sections with potential expansions or missing proofs!

Disclaimer 2 Chapters are in the order of teaching. Images are entirely taken from the presentations, and come from different sources, please refer to the slides to find the exact origins. Hopefully, it will be fixed in future versions.

Contents

List of Figures

List of Tables

List of Algorithms

List of Theorems

List of Symbols

Ω	data domain
ν	data distribution
$\mathcal{X}$	distribution space
f^*	data generating process
$\mathcal{F}$	model hypothesis class
$\gamma(\cdot)$	complexity measure
ℓ	loss
$\mathcal{R}(f)$	population loss
$\widehat{\mathcal{R}}(f)$	empirical loss
$\mathcal{H}^s$	s-Sobolev Space
$\mathcal{X}(\Omega, \mathcal{C})$	space of signals
$\mathfrak{g}$	symmetry **TODO check**
$\mathfrak{G}$	algebraic group
S	group generators
ρ	group representation map
O_x	orbit of a signal
$S_{\mathfrak{G}}$	group smoothing operator
A	group average
Σ_n	permutation group
$\bigoplus$	permutation invariant operator
$\mathcal{G}$	graph
$\mathbf{A}$	adjacency matrix
$\mathcal{N}(\cdot)$	neighborhood function
S	shift operator
$\widehat{x}$	Fourier transformed vector
$\star$	convolution operation
$\mathfrak{H}$	stabilizer subgroup
$\mathfrak{g}\mathfrak{H}$	left coset
$\mathfrak{G}/\mathfrak{H}$	quotient of subgroup
$T_u\Omega$	manifold tangent space
$g_u(\cdot, \cdot)$	Riemaniann metric
γ	geodesic
$\Gamma_{u \to v}$	parallel transport
$exp_u(\cdot)$	exponential map
$\eta(\cdot)$	domain deformation
$d\eta_u(x)$	linearization differential

Δ	laplacian
div, ∇^*	divergence operator
$\mathcal{T}$	triangular mesh
$\mathcal{C}_u, F$	fiber
ω_u	gauge
π	bundle
B, Ω	Base space
E	total space
σ	section
P	principal bundle
$\mathbf{z}^{(t)}$	sequential result
$\tau(\cdot)$	time warping function

Chapter 1

Introduction

Symmetry, as wide or as narrow as you may define its meaning, is one idea by which man through the ages has tried to comprehend and create order, beauty, and perfection.

Hermann Weyl

The aim of this course is to propose a framework for Deep Learning that follows that of the *erlangen programme* for Geometry. The concepts of symmetry will arise frequently. Hopefully, with the help of such lectures, fundamental principles underlying deep representation learning architectures will be uncovered.

These lecture notes will cover the theoretical objects to understand concepts explained.

Rather than reporting the whole introduction, we reroute the reader to the next Chapters for formalisms. This Chapter will be mostly a presentation of the topic and the **general approach**.

If there is one takeaway from this approach it would be that symmetry is fundamental to charachterize structures and methods. The term itself has a long history. In ancient greek $\sigma\nu\mu\mu\epsilon\tau\rho\acute{\iota}\alpha$ roughly means *same measure*, and was used to describe the beauty of proportion in arts and music. Platonic solids were considered to be the basic buidling blocks by Plato and his Philosophy. Later in the 17^{th} century, Kepler explored snowflake shapes and attempted to understand its reasons.

Also modern geometry is based on the Greek Euclid. 4 basic postulates made the basis of his reasoning, and for hundreds of years, the 5^{th} defied any attempt to prove its redundancy[1]. What ended the *Euclidean Monopoly* is the 17^{th} century, when Poncelet and Desargues developed projective geometry, where points and lines are interchangeable. This was the first attempt to undermine Euclidean Geometry, by still being inherently not so far apart from it. The first non-Euclidean geometry is credited to Lobachevsky, a Russian mathematician that considered the 5^{th} action to be too restrictive, and relaxed it, giving birth to Hyperbolic Geometry. Though

[1] In a plane, given a line and a point not on it, at most one line parallel to the given line can be drawn through the point

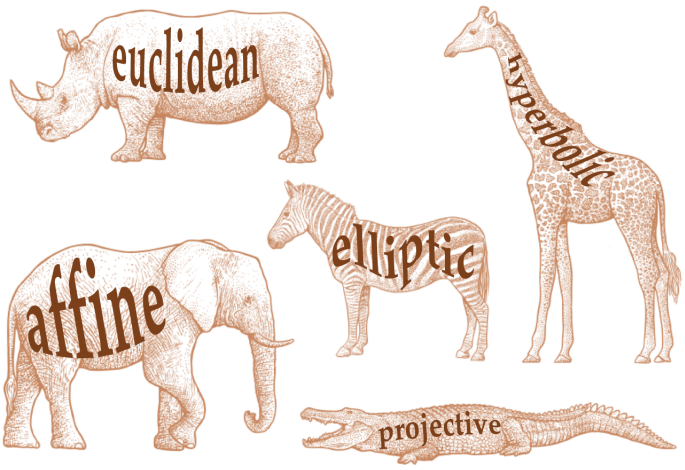


Figure 1.1: 19^{th} century Zoo of Geometries

highly criticized, it gave rise to the contributions of Bolyai and Gauss. Lastly, Riemann, and his work on differential geometry gave an end to *Euclid's Monopoly*.

As a result of all these developments an entire Zoo of geometries was created, with research moving towards understanding what defined a geometry and its properties.

Klein, a young German mathematician, when asked to deliver his research program [2], proposed his approach of treating geometry by the study of symmetries. A geometry coudl be defined by specifying specific transformations formalized by Group Theory, a field born in the same century by the contributions of Galois and Lie, and Klein himself.

Even though the Erlangen Programme originated in Erlangen, Klein later moved to Göttingen, where Gauss worked as well. There, his impact had profound spillovers in Physics, thanks to Noether, his colleague there, who proved that the translational symmetry of time naturally motivated conservation of energy. Also Hermann Weyl used the same principles to study Gauge Invariance, leading to the unification of Forces by Yang and Mills, with the Standard Model of Physics.

This historical context is very much similar to the current state of affairs of Deep Learning. There is no doubt that Deep Learning brought a great revolution in Data Science, thanks to the contributions of many researchers.

On the other hand, there is a zoo of networks with **few unifying principles**. As a consequence it is difficult ot define relations between different methods, leading to publishing the same methods over different domains and names, and lack of directions to follow.

Geometric Deep Learning (GDL) proposes a minset and a teaching framework to understand and study these architectures. This field was popularized by a recent publication and book preview [Bro+17], [Bro+21].

Large and High Quality datasets, combined with sufficient computational resources spurred the success of Machine Learning methods that interpolate information by estimating functions

[2] The Erlangen Programme

Figure 1.2: (some of) the Founding Fathers of Deep Learning: Geoffrey Hinton, Yoshua Bengio, Yann LeCun, Jürgen Schmidhuber

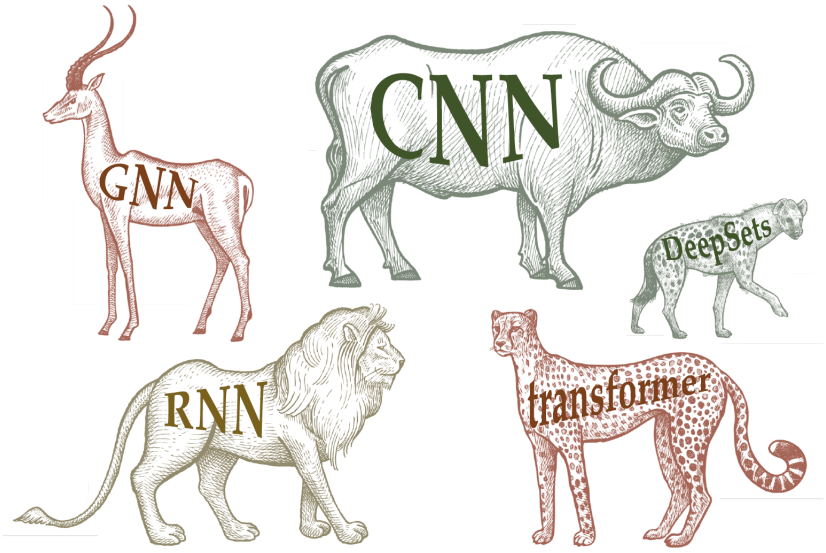


Figure 1.3: Deep Learning Zoo

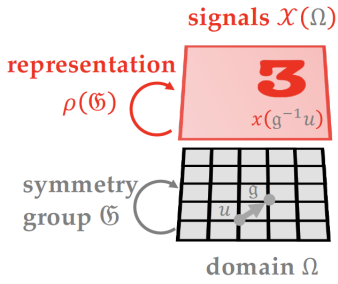


Figure 1.4: Symmetry Prior Diagram

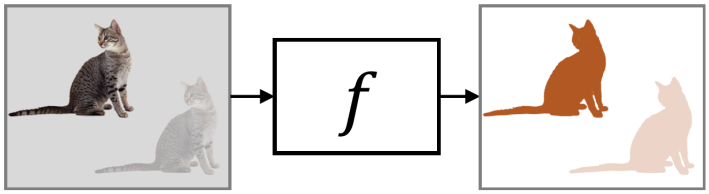


Figure 1.5: Shift Equivariant Function

from large classes. A simple choice of architecture is already much expressive. A Multi Layer Perceptron is granted to be able to Universally Approximate any continuous function (more later in Chapter 2). What appears is that in high dimensions though, the number of samples required for a given estimation error is exponentially increasing in the dimensionality. This is commonly known as the **curse of dimensionality**, one of the most difficult obstacles in learning. Indeed, event the easiest images are highly dimensional. Yet, any object of this kind presents a well defined structure (e.g. a grid), over which the signal is mapped (e.g. the colored pixels).

The culmination of local weight sharing ideas was the ultimately famous method of Convolutional Neural Networks for Computer Vision tasks [LeC+89]. This concept extends quickly to other topics where graphs are used to study their characteristics.

The additional geometric structure of some domains can be used to break the curse of dimensionality. Indeed, symmetry priors, introduced in Chapter 3 (Figure 1.4) are able to unify under structural properties different samples with common features.

This symmetry operation acts on the function specification. In some cases we wisht the output to be *equivariant* to transformations (Figure 1.5, in others, we wish an *invariant* output. Both thse concepts will be explored throughout the lectures, to eventually get to the most general function form arising from their combination, drawn in Figure 1.6.

A collection of these principles naturally defines a Blueprint for Geometric Deep Learning (Definition 3.43), which layes the foundations of many already known architectures. The power of this line of reasoning will be noticed when those architectures will arise from the first principles outlined in the Blueprint.

What we ultimately will show is that a wide range of structures (Figure 1.7) can be treated equivalently if paired with a well specified group symmetry (Figure 1.8), named the five Gs

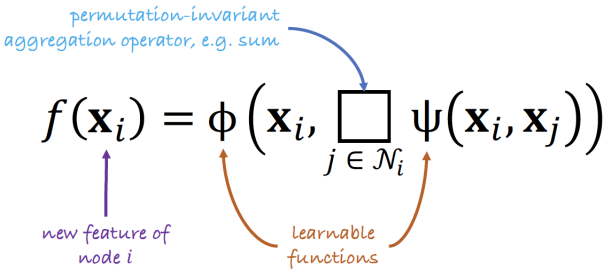


Figure 1.6: Invariance and Equivariance combined

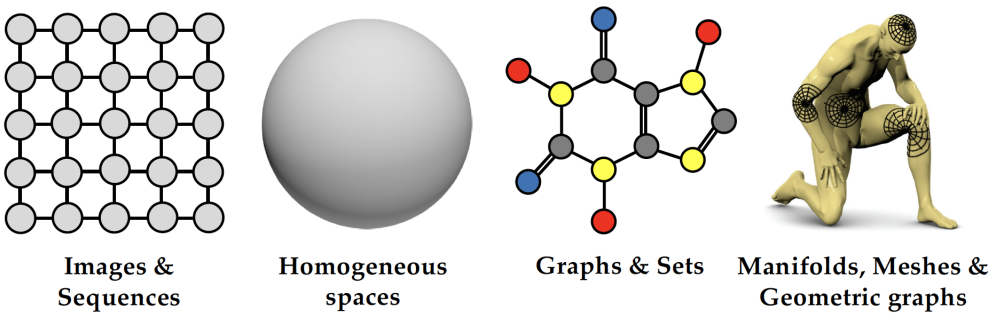


Figure 1.7: Structures (Domains) in GDL

of Geometric Deep Learning. An outline of the architectures that can be derived by these fundamental principles is found in Figure 1.9.

The outline of the lectures can be found in Table 1.1, with references to the book chapters, and lecture notes chapters.

For the foundational bricks, we instead suggest reading Chapters 2 and 3, toegether with Chapters $2, 3$ of the book [Bro+21]. For Manifolds and gauges, another interesting reference for symmetry topics is reported [Wei+21].

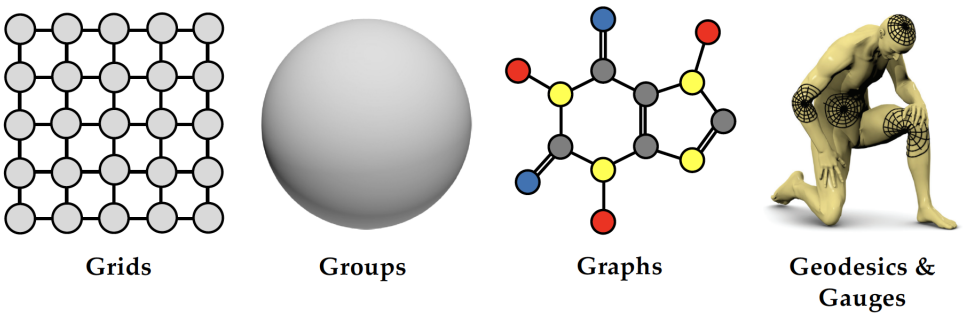


Figure 1.8: The five Gs of GDL

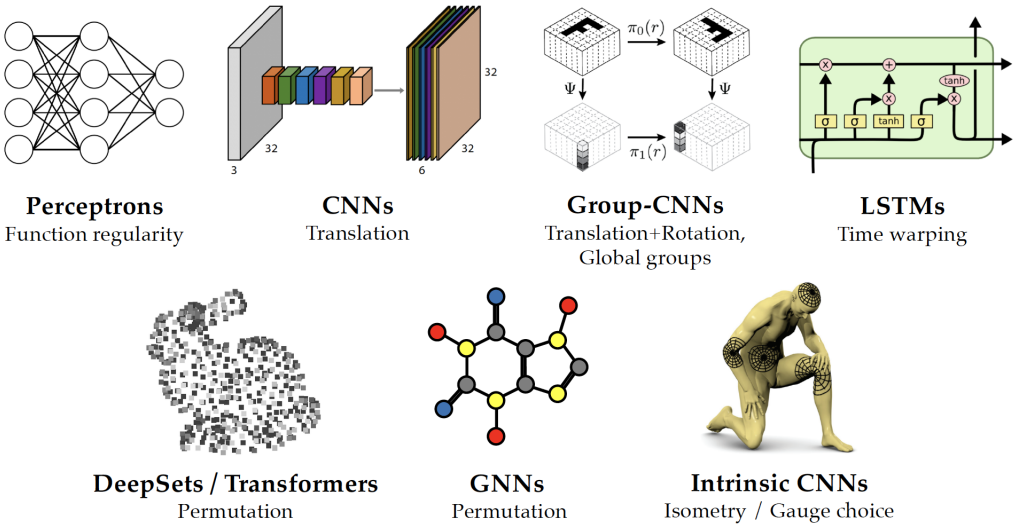


Figure 1.9: Architectures from Fundamental Principles of GDL

Architecture	Domain Ω	Symmetry Group $\mathfrak{G}$	Lecture Notes Chapter	Book [Bro+21] Chapters
CNN	Grid	Translation	5	4.2, 5.1
Spherical CNN	Sphere $SO(3)$	Rotation $SO(3)$	6, 6.1.1	4.3, 5.2
Intrinsic/Mesh CNN	Manifold	Isometry $Iso(\Omega)$/ Gauge symmetry $SO(2)$	7	4.4, , 4.5, 4.6, 5.6
GNN	Graph	Permutation Σ_n	4	4.1, 5.3, 5.4
DeepSets	Set	Permutation Σ_n	4	4.1, 5.3, 5.4
Transformer	Complete Graph	Permutation Σ_n	4	4.1, 5.3, 5.4
LSTM	1D Grid	Time Warping	9	5.7, 5.8

Table 1.1: Teaching Content Structure

Chapter 2

High-Dimensional Learning

In this lecture, the topics addressed will be:

- Statistical Learning error decomposition
- the Curse of Dimensionality
- an approach to tackle the curse

2.1 Data and Error Decomposition

The ingredients for Statistical Learning are

- the data distribution
- an approximation model
- an error metric
- an estimation algorithm

Below, the notation and the basic concepts will be introduced.

Definition 2.1 (Data Domain)**.** Data is assumed to be a set of pairs:

$$\{(x_i, y_i)\}_{i=1}^{n} \; x_i \in \Omega, \; y_i \in \mathbb{R} \quad \forall i \tag{2.1}$$

Definition 2.2 (Data Distribution and space $\nu, \mathcal{X}$)**.** Of these samples, it is assumed that:

$$x \sim \nu \quad x \, defined \, over \, \mathcal{X} \tag{2.2}$$

Definition 2.3 (Data Generating Process f^*)**.** The relationship between x_i and signals y_i is estabilished through f^* as:

$$f^* : \mathcal{X} \to \mathbb{R} : f^*(x_i) = y_i \forall i \tag{2.3}$$

The aim of supervised learning is that of finding a good estimate of f^*, which is defined over the distribution of observations over an observed space $\mathcal{X}$, coming from a domain space Ω.

Observation 2.4 (No free lunch)**.** *For the purpose of obtaining guarantees, assumptions on both ν and f^* must be established.*

Most of the process is backed on the definition of a space of action, introduced below.

Definition 2.5 (Model Hypothesis class $\mathcal{F}$)**.** For estimation, a set of feasible functions to mimic f^* is assumed.

$$\mathcal{F} = \left\{ f : \mathcal{X} \to \mathbb{R} \right\} \tag{2.4}$$

Definition 2.6 (Complexity measure $\gamma(\cdot)$)**.** Associated to a family of functions, an indicator of their flexibility/complexity is considered to "rank" members.

$$\gamma : \mathcal{F} \to \mathbb{R}_+ \tag{2.5}$$

Definition 2.7 (Sobolev Norm)**.** The Sobolev Norm is a measure of wiggliness.

$$\int (1 + \omega^2)^S |f(\omega)|^2 d\omega \tag{2.6}$$

Example 2.8 (Examples of $\mathcal{F}, \gamma$ pairs)**.** Some of the most intuitive function spaces and complexity measures are:

- $\mathcal{F} = \{neuralnets\}$, $\gamma(f) = \#neurons$
- **TODO add F** the Sobolev Norm $\gamma(f) = \int (1 + \omega^2)^S |f(\omega)|^2 d\omega$

Definition 2.9 (Path norm for Neural Network)**.** For a neural network:

$$f(x) = \sum_{j \leq m} a_j \rho(w_j^T x + b) \tag{2.7}$$

The path norm can be defined as:

$$\|f\| = \sum_{j \leq m} |a_j| (\|w_j\| + |b_j|) \tag{2.8}$$

Which is a weighted sum of the neurons' contributions in the final prediction.

Error metrics are used to estimate the performance of the selected model f.

Definition 2.10 (Error measure ℓ)**.** The foundational brick is a pointwise convex error measure[1]

$$\ell : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_+ \quad as\ \ell(f(x), f^*(x)) \tag{2.9}$$

.

Definition 2.11 (Population Loss $\mathcal{R}(f)$)**.**

$$\mathcal{R}(f) = \mathbb{E}_\nu[\ell(f(x), f^*(x))]$$

Definition 2.12 (Empirical Loss $\widehat{\mathcal{R}}(f)$)**.** For a dataset $\{(x_i, y_i)\}_{i=1}^n$

$$\widehat{\mathcal{R}}(f) = \frac{1}{n} \sum_i \ell(f(x_i), f^*(x_i)) \tag{2.10}$$

In principle, the population loss is not achievable and an ML practitioner works with the population loss.

[1] aka loss

Lemma 2.13 (Facts about Error metrics)**.** *These pointwise identities are not useful as $f \in \mathcal{F}$ depends on the training set. Later, more informative measures will be introduced.*

1. $\mathbb{E}[\widehat{\mathcal{R}(f)}] = \mathcal{R}(f)$
2. $V[\widehat{\mathcal{R}}(f)] = \frac{1}{n}\mathbb{E}[\ell(f(x), f^*(x)) - \mathcal{R}(f)]$

In terms of estimation algorithm the implemented strategy is *empirical risk minimization*, to achieve *population loss minimization*. To achieve this goal, a restriction on the hypothesis space and a linked framework is introduced.

Definition 2.14 (Empirical Risk Minimization (ERM) Framework)**.** Consider a **convex** ball shaped constraint on the complexity:

$$\mathcal{F}_\delta = \{f \in \mathcal{F} : \gamma(f) \leq \delta\}$$

Aim to find the best function, using as strategies:

- plain ERM
$$\hat{f}_\delta = \underset{\mathcal{F}_\delta}{argmin}\{\widehat{\mathcal{R}}(f)\}$$
- penalized
$$\hat{f}_\delta = \underset{\mathcal{F}_\delta}{argmin}\{\widehat{\mathcal{R}}(f) + \lambda\gamma(f)\}$$
- interpolation
$$\hat{f}_\delta = \underset{\mathcal{F}_\delta}{argmin}\{\gamma(f)\} \; s.t. \; \widehat{\mathcal{R}}(f) = 0 \; \lambda > 0$$

Observation 2.15 (On the interpolation form)**.** *The model chosen perfectly predicts on available data and is the least complex among those belonging to this subset. In contexts where noise is predominant, it is not advised as a choice, as the observed date is not a trustworthy sample of the Ω space.* ***TODO check****.*

$$\widehat{\mathcal{R}}(f) = 0 \iff \hat{f}_\delta(x_i) = f^*(x_i) \forall i \tag{2.11}$$

Using a lagrangian formulation, it is the "opposite" of the penalized form:

$$\hat{f}_\delta = \underset{\mathcal{F}_\delta}{argmin}\{\lambda\widehat{\mathcal{R}}(f) + \gamma(f)\}$$

The advantage of such approach is that the error can be analyzed through meaningful decompositions. As a starting point, the best performance for the hypothesis class is introduced to rank estimators [BB07].

Definition 2.16 (Baseline)**.** The baseline[2] for a family $\mathcal{F}$ is:

$$\underset{f\in\mathcal{F}}{inf}\{\mathcal{R}(f)\} \tag{2.12}$$

Theorem 2.17 (Error Decomposition)**.** *Denote $\hat{f}_\delta$ as $\hat{f}$. Then:*

$$\mathcal{R}(\hat{f}) \leq \underset{f\in\mathcal{F}}{inf}\{\mathcal{R}(f)\} + \varepsilon_{opt} + \varepsilon_{stat} + \varepsilon_{appr} \tag{2.13}$$

[2]In other references it is referred to as *prophet*

Where:

$$\begin{cases} \varepsilon_{appr} = + \inf_{f \in \mathcal{F}_\delta} \{\mathcal{R}(f)\} - \inf_{f \in \mathcal{F}} \{\mathcal{R}(f)\} \\ \varepsilon_{opt} = + \widehat{\mathcal{R}}(\hat{f}) - \inf_{f \in \mathcal{F}_\delta} \{\widehat{\mathcal{R}}(f)\} \\ \varepsilon_{stat} = 2 \sup_{f \in \mathcal{F}_\delta} \left\{ |\mathcal{R}(\hat{f}) - \widehat{\mathcal{R}}(\hat{f}) \right\} \end{cases} \tag{2.14}$$

Proof. Consider the population loss minus the baseline. The aim is to build known quantities by adding and subtracting. Denote by $\mathcal{E} = \mathcal{R}(\hat{f}) - \inf_{f \in \mathcal{F}} \{\mathcal{R}(f)\}$ and work out the new identity

$$\mathcal{E} = \mathcal{R}(\hat{f}) - \inf_{f \in \mathcal{F}} \{\mathcal{R}(f)\} + \inf_{f \in \mathcal{F}_\delta} \{\mathcal{R}(f)\} - \inf_{f \in \mathcal{F}_\delta} \{\mathcal{R}(f)\} \qquad \text{add \& subtract} \tag{2.15}$$

$$= \mathcal{R}(\hat{f}) - \inf_{f \in \mathcal{F}_\delta} \{\mathcal{R}(f)\} \underbrace{+ \inf_{f \in \mathcal{F}_\delta} \{\mathcal{R}(f)\} - \inf_{f \in \mathcal{F}} \{\mathcal{R}(f)\}}_{\varepsilon_{appr}} \qquad \text{reordering} \tag{2.16}$$

$$= \mathcal{R}(\hat{f}) - \inf_{f \in \mathcal{F}_\delta} \{\mathcal{R}(f)\} + \varepsilon_{appr} \qquad \text{first error} \tag{2.17}$$

$$= \mathcal{R}(\hat{f}) - \inf_{f \in \mathcal{F}_\delta} \{\mathcal{R}(f)\} + \varepsilon_{appr} + \widehat{\mathcal{R}}(\hat{f}) - \widehat{\mathcal{R}}(\hat{f}) + \inf_{f \in \mathcal{F}_\delta} \{\widehat{\mathcal{R}}(f)\} - \inf_{f \in \mathcal{F}_\delta} \{\widehat{\mathcal{R}}(f)\} \qquad \text{add \& subtract} \tag{2.18}$$

$$= \underbrace{+ \widehat{\mathcal{R}}(\hat{f}) - \inf_{f \in \mathcal{F}_\delta} \{\widehat{\mathcal{R}}(f)\}}_{\varepsilon_{opt}} + \mathcal{R}(\hat{f}) - \widehat{\mathcal{R}}(\hat{f}) + \inf_{f \in \mathcal{F}_\delta} \{\widehat{\mathcal{R}}(f)\} + - \inf_{f \in \mathcal{F}_\delta} \{\mathcal{R}(f)\} + \varepsilon_{appr} \qquad \text{reordering} \tag{2.19}$$

$$= \varepsilon_{opt} + \mathcal{R}(\hat{f}) - \inf_{f \in \mathcal{F}_\delta} \{\mathcal{R}(f)\} - \widehat{\mathcal{R}}(\hat{f}) + \inf_{f \in \mathcal{F}_\delta} \{\mathcal{R}(f)\} + \varepsilon_{appr} \qquad \text{second error} \tag{2.20}$$

$$\leq \varepsilon_{opt} + \underbrace{2 \sup_{f \in \mathcal{F}_\delta} \left\{ |\mathcal{R}(\hat{f}) - \widehat{\mathcal{R}}(\hat{f})| \right\}}_{\varepsilon_{stat}} + \varepsilon_{appr} \qquad \textbf{TODO explain} \tag{2.21}$$

$$\leq \varepsilon_{opt} + \varepsilon_{stat} + \varepsilon_{appr} \qquad \text{third error} \tag{2.22}$$

Thus:

$$\mathcal{R}(\hat{f}) \leq \inf_{f \in \mathcal{F}} \{\mathcal{R}(f)\} + \varepsilon_{opt} + \varepsilon_{stat} + \varepsilon_{appr} \tag{2.23}$$

□

Observation 2.18 (On Error Decomposition)**.** *The three ε can be interpreted as follows:*

- ε_{appr}*: the error dependent on the ball chosen with respect to the family specified. It is big if the hypothesis space is too far from* f^*

- ε_{opt}*: the optimizable empirical loss error difference with respect to the best possible in the ball. It is small if the algorithm can be efficiently solved.* ***TODO check***
- ε_{stat}*: the maximum population vs empirical distance within the ball chosen. Given a finite dataset size n it grows as* $\mathcal{F}$ *grows in size.*

2.2 The Curse of Dimensionality

This concept was introduced by Bellman in 1962. It is a formal analysis of interpolation in a high dimensional regime, which shows its limitations. An example is used to break guarantees.

Definition 2.19 ($\beta-Lipschitz$ function)**.** A (real) function $f : \mathcal{X} \subseteq \mathbb{R}^d \to \mathbb{R}$ is $\beta-Lipschitz$ if:

$$\forall x, x' \in \mathcal{X} \quad |f(x) - f(x')| \leq \beta \|x - x'\| \tag{2.24}$$

It is a regularity condition which ensures at most β label distance with respect to feature space distance. The bound induces a proximity in both spaces, with a β proportionality limit.

Example 2.20 (Cursing a Lipschitz Function)**.** Let f^* be $1-Lipschitz$, so that:

$$\forall x, x' \in \mathcal{X} \quad |f(x) - f(x')| \leq \beta \|x - x'\|$$

Assume also that:

$$x_i \sim \mathcal{N}(0, I_d) \forall i \tag{2.25}$$

The aim is to estimate how many samples are needed to bound the error by a pre-specified constant ε.
First of all it can be observed that:

$$\mathcal{F} = \{f : R^d \to \mathbb{R} \; f \; 1-Lip \; f \; bounded\} \tag{2.26}$$

Is a Banach Space if endowed with the usual norm. Considering a set of Lipschitz functions as estimators, the complexity measure will be the Lipschitz constant itself, and the aim is to find the lowest complex interpolant function $\hat{f}$.

$$\hat{f} = \underset{f \in \mathcal{F}_\delta}{argmin}\{\gamma(f)\} \quad s.t. \; f(x_i) = f^*(x_i) \forall i \quad \gamma(f) = sup\left\{ \frac{|f(x) - f(x')|}{\|x - x'\|} \right\} \tag{2.27}$$

Picking $x \sim \nu$ the approach is similar to the proof of Theorem 2.17. Let x_{i0} be the closest point to x in the training set. By construction, $\hat{f}$ interpolating f^* is **at most** $1-Lipschitz$. Then:

$$\begin{aligned}
|est - real| &= |\hat{f}(x) - f^*(x)| && \text{estimated vs real} \\
&= |\hat{f}(x) - f^*(x) + \hat{f}(x_{i0}) - \hat{f}(x_{i0}) + f^*(x_{i0}) - f^*(x_{i0})| && \text{add \& subtract} \\
&\leq |\hat{f}(x) - \hat{f}(x_{i0})| + |\hat{f}(x_{i0}) - f^*(x_{i0})| + |f^*(x_{i0}) - f^*(x)| && \text{triang ineq} \\
&\leq |\hat{f}(x) - \hat{f}(x_{i0})| + |f^*(x_{i0}) - f^*(x)| && \text{by interpolation} \\
&\leq 2\beta \|x_{i0} - x\| && \text{by construction} \\
&\leq 2\|x_{i0} - x\| && \text{as } \beta = 1
\end{aligned}$$

The average variation from the prediction (squared) is then:

$$\mathcal{E} = \mathbb{E}_{\mathcal{X}}\left[|\hat{f}(x) - f^*(x)|^2\right] \tag{2.28}$$

$$\leq 4\mathbb{E}_{\mathcal{X}}\left[\|x_{i0} - x\|^2\right] \tag{2.29}$$

$$\leq 4\mathcal{W}_2^2(\nu, \hat{\nu_n}) \tag{2.30}$$

$$\sim n^{\frac{-1}{d}} \tag{2.31}$$

Where the penultimate equality is obtained by recognizing that it is the well known Wasserstrein distance between the true and the n dependent estimated distribution, which is of order $n^{\frac{-1}{d}}$. Then:

$$\mathcal{E} \sim n^{\frac{-1}{d}} \implies n \sim \mathcal{E}^{-d} \to \infty \tag{2.32}$$

Which tends to infinity with an exponential rate.
Having established an upper bound, a lower bound is obtained similarly as:

$$\textbf{TODO} \tag{2.33}$$

The curse of dimensionality also pops up in approximation. Considering a family of shallow Neural Networks:

$$\mathcal{F} = \left\{ f(x) = \sum_{j \leq m} a_j \rho(w_j^T x + b) \right\} \quad : \quad m = \#neurons \tag{2.34}$$

The universal approximation Theorem by [HSW89],[Pin99],[Cyb89],[Bar93] ensures that with one layer any function can be approximated.

Theorem 2.21 (Universal Approximation Theorem)**.** ***TODO***

However, in practice, introducing complexity measures such as the number of neurons or the path norm leads to the same problems.

A result by Maiorov finds [Mai99] **TODO**

Theorem 2.22 (Sobolev Curse)**.** *If $f \in \mathcal{H}^s$ where s is the number of finite derivatives of f then:*

$$\implies \mathcal{E} = \inf_{g \in \mathcal{F}} \left\{ \sup_{x \in \mathcal{X}} |f(x) - g(x)| \right\} \in \Theta(m^{-\frac{s}{d}}) \tag{2.35}$$

Which is again a tight exponential bound.

Similarly Barron proved that for a very limited class the curse can be avoided [Bar93; Bar94]. The restriction is nevertheless too strong for practical purposes.

Theorem 2.23 (Barron Class avoids the Curse)**.** *If $f \in \mathcal{H}_1$ is such that it belongs to the Barrow class where:*

$$\int |\hat{f}(\omega)| \|\omega\|^2 d\omega \leq \infty \tag{2.36}$$

Then:

$$\implies \mathcal{E} = \inf_{g \in \mathcal{F}} \left\{ \sup_{x \in \mathcal{X}} |f(x) - g(x)| \right\} \in O(m^{-1}) \perp\!\!\!\perp d \tag{2.37}$$

Which is not exponential in d.

A publication by Jin et. al proposes one important result, formalized in the next Theorem [JNJ17].

Definition 2.24 ($\beta-$*smooth* functions)**.** A function is $\beta-$*smooth* if its gradient is $\beta-$*Lipschitz*. Namely:

$$\forall x, x' \in \mathcal{X} \quad \|\nabla f(x') - \nabla f(x)\| \leq \beta \|x - x'\| \tag{2.38}$$

Assumption 2.25 ($\widetilde{O}$ notation)**.** The notation of a function $f(n) \in \widetilde{O}(n)$ means that:

$$f(n) \in O(nlog(n)) \tag{2.39}$$

Theorem 2.26 (Perturbed GD quasi-dimension independence)**.** *noisy GD*
Perturbed Gradient Descent finds an $\mathcal{E}$ approximate second order stationary points of a $\beta-$smooth function in $\widetilde{O}\left(\frac{\beta}{\mathcal{E}^2}\right) \perp\!\!\!\perp d$ iterations.

To summarize:

- Lipschitz Functions family is too large, impacting the statistical error
- Sobolev and Barron classes are too small, impacting approximation error

In the next lectures, it will be shown that the geometric low dimensional substructure Ω can be exploited to solve the high dimensionality of signals $\mathcal{X}(\Omega)$.

Chapter 3

Geometric Priors

The contents of this chapter are mostly taken from two lectures.

In the first lecture the topics introduced will be:

- domains & signals
- symmetries
  - of the domain
  - groups
  - group actions and representations
  - action of the group on the space of signals
- invariant & equivariant functions.

In the second lecture, we will encouter:

- more on invariance and equivariance
- scale separation
- the blueprint of GDL

3.1 Domains & Signals

In Geometric Deep Learning, data are signals on spaces. Exploiting this feature, learning becomes tractable by symmetry and scale separation, two concepts that will be outlined.

In practice, the size and complexity of $\mathcal{F}$ is decreased without compromising the hypothesis. The result is the same ε_{appr} and a reduction in ε_{stat}.

In Definition 2.1 it is declared that $x_i \in \Omega$. In the setting of GDL, Ω possibly has an additional structure, part of the 5Gs:

- Grids
- Groups
- Graphs
- Geodesics & Gauges

As a key message, neural networks which process geometric data should make use of the structure of Ω, and respect it.

Definition 3.1 (Space of signals $\mathcal{X}(\Omega, \mathcal{C})$)**.** To be considered as the space $\mathcal{X}$, but generated from Ω. In principle, the observations x_i become functions from the domain to a vector space, with dimensions called channels. Then, $\mathcal{X}(\Omega, \mathcal{C})$ is the set of those signals.

$$\mathcal{X}(\Omega, \mathcal{C}) = \{x : \Omega \to \mathcal{C}\} \quad : \quad \Omega \textit{ domain}, \;\; \mathcal{C} \textit{ v.s.} \tag{3.1}$$

Example 3.2 (Image as Signal)**.** Let $\Omega = \mathbb{Z}_n \times \mathbb{Z}_n$ be an image grid, and $\mathcal{C} = \mathbb{R}^3 = \mathbb{R}_R \times \mathbb{R}_G \times \mathbb{R}_B$ the color assigned to a pixel in RGB quantization. Then:

$$\mathcal{X}(\Omega, \mathcal{C}) = \{x : \mathbb{Z}_n \times \mathbb{Z}_n \to \mathbb{R}^3\} \tag{3.2}$$

Example 3.3 (Molecular Graph)**.** Let $\Omega = \{1, \dots, n\}$ be the vertices of a molecular graph, and $\mathcal{C} = \mathbb{R}^m$ be the possible atoms. Then:

$$\mathcal{X}(\Omega, \mathcal{C}) = \{x : \{1, \dots, n\} \to \mathbb{R}^m\} \tag{3.3}$$

TODO check

While Ω might not be a vector space, $\mathcal{X}(\Omega, \mathcal{C})$ is a vector space, as functions mapping to $\mathcal{C}$ satisfy the vector space axioms. The possible outcomes of additional structures are included below:

Definition 3.4 (Operation on $\mathcal{X}(\Omega, \mathcal{C})$)**.** Addition of $x \in \mathcal{X}(\Omega, \mathcal{C})$ is defined as:

$$(\alpha x + \beta x')(u) = \alpha x(u) + \beta x'(u) \quad \forall u \in \Omega, \forall \alpha, \beta \in \mathbb{R} \tag{3.4}$$

Definition 3.5 (Additional Hilbertian structure of $\mathcal{X}(\Omega, \mathcal{C})$)**.** Given an inner product $\langle \cdot, \cdot \rangle_{\mathcal{C}}$ on $\mathcal{C}$ and a measure μ on Ω an inner product on $\mathcal{X}(\Omega, \mathcal{C})$ is obtained as[1]:

$$\langle x, x' \rangle_{\mathcal{X}(\Omega, \mathcal{C})} = \int_{\Omega} \langle x(u), x'(u) \rangle d\mu(u) \tag{3.5}$$

And the pair $\{\mathcal{X}(\Omega, \mathcal{C}), \langle \cdot, \cdot \rangle_{\mathcal{X}(\Omega, \mathcal{C})}\}$ forms a Hilbert space.

While a function maps element of the domain to elements of the channel space, it is possible to consider more general structures known as fields. There, each element $u \in \Omega$ is mapped to a specific feature space $\mathcal{C}_u$ named *fiber*, which can be thought of as tangent planes to the point u. The main requirement is that all $\{\mathcal{C}_u\}_{u \in \Omega}$ are isomorphic. Furthermore, if a way to identify fibers is given, they can be formulated as functions. For the purpose of this class, the analysis is restricted to function $\mathcal{X}(\Omega, \mathcal{C})$ spaces.

Observation 3.6 (Domain as Data)**.** *In some occasions, it could be the case that there is no signal on Ω. In simple unweighted graphs nothing can be treated as such. Nevertheless, it is possible to consider the adjacency matrix and define signals over the space:*

$$\mathcal{X}(\Omega \times \Omega, \mathcal{C} = \mathbb{R}^n \times \mathbb{R}^n) \tag{3.6}$$

[1] A weighted (by the measure) integral of the inner products.

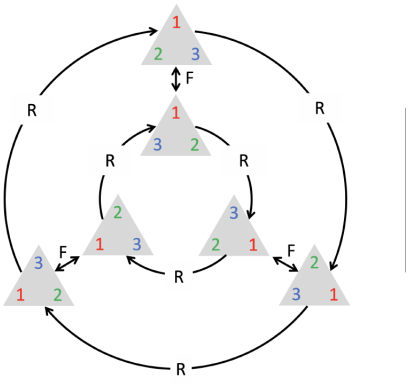


Figure 3.1: Symmetries of a triangle in a Cayley diagram

Figure 3.2: Symmetry of the Parametrization

3.2 Symmetries

Informally it is possible to assert that:

A symmetry of an object is a transformation that leaves some property unchanged (invariant)

An example for a triangle is shown in Figure 3.1. It is also possible to form multiplication tables which show all the possible combinations of *moves*.

Among the many types of symmetries, some are proposed below.

Definition 3.7 (Symmetry of the Parametrization $\mathfrak{g}$)**.** Let $\mathcal{X}$ and $\mathcal{Y}$ be the input and label space. Consider $\mathcal{W}$ as the weight space. A symmetry of the parametrization of $f : \mathcal{X} \times \mathcal{W} \to \mathcal{Y}$ is a transformation $\mathfrak{g} : \mathcal{W} \to \mathcal{W}$ such that:

$$f(x, \mathfrak{g}w) = f(x, w) \quad \forall x \in \mathcal{X}\ w \in \mathcal{W} \tag{3.7}$$

Example 3.8 (Fully connected one layer Neural Network)**.** For a fully connected one layer NN, exchanging the weights of two nodes in the hidden layer does not change the final output prediction. A graphical example is shown in Figure 3.2

Definition 3.9 (Symmetry of the Label Function)**.** Let $\mathcal{X}$ and $\mathcal{Y}$ be the input and label space, interacting through the ground truth label function $L : \mathcal{X} \to \mathcal{Y}$. A transformation $\mathfrak{g} : \mathcal{X} \to \mathcal{X}$ is a symmetry of the label when:

$$L(\mathfrak{g}x) = L(x) \forall x \in \mathcal{X} \implies L \circ \mathfrak{g} = L \tag{3.8}$$

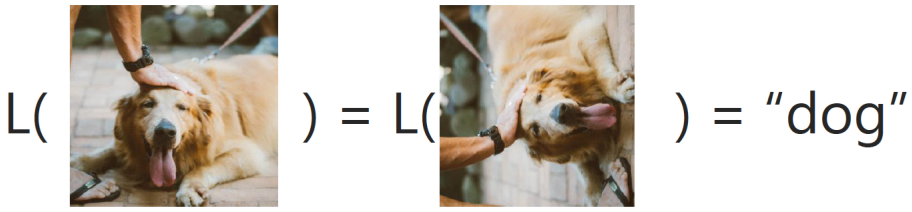


Figure 3.3: Symmetry of the Label Function

Example 3.10 (Image Rotation)**.** Considering a labeling function that recognizes dog images, a transformation of the input $\mathfrak{g}$ that rotates the image is not supposed to change the labeling of a dog as a dog. In Figure 3.3, $\mathfrak{g}$ is a 90 degrees rotation.

In some sense, learning classes for a labeling means learning its symmetries. Indeed, any invertible map $\mathfrak{g}$ which respects class bounds is a label symmetry. As an thought exercise, any classification task can be devised as a distinction class, i.e. find the symmetries that do not escape subsets corresponding to classes that partition $\mathcal{Y}$. However, knowing all the symmetries of a a set of objects is a priori infeasible.

Definition 3.11 (Symmetries on Ω domains $\mathfrak{g}$, Symmetry group)**.** Let Ω be a domain space with a structure. A transformation $\mathfrak{g} : \Omega \to \Omega$ is a symmetry if the *structure* is preserved[2]. The set of those elements is the **Symmetry Group**

Example 3.12 (Domain Symmetries)**.** Among the easiest examples we find:

- For an (unordered) set $\Omega = \{1, \dots, n\}$ a permutation
- For $\Omega = \mathbb{R}^d$, which is a Euclidean space, any Euclidean Isometry such as rotations, translations or reflections. A metric is needed.**TODO check define**
- A diffeomorphism for a smooth manifold Ω

The concept is highly dependent on what is Ω

A group of symmetries for an object must necessarily include:

- the identity transformation
- the inverse for each transformation
- the composition any series of transformations

These innate requirements are included in the much broader definition of algebraic group.

Definition 3.13 (Group $\mathfrak{G}$)**.** A group is a pair made of a set and an operation defined on it. The set is denoted as $\mathfrak{G}$, while the operation is shadowed $\mathfrak{gh} : \mathfrak{g}, \mathfrak{h} \in \mathfrak{G}$. The conditions to fulfill for it to be valid are:

- Associativity:

$$(\mathfrak{gh})\mathfrak{l} = \mathfrak{g}(\mathfrak{hl}) \forall \mathfrak{g}, \mathfrak{h}, \mathfrak{l} \in \mathfrak{G} \tag{3.9}$$

[2]This concept will be expanded throughout the book

- Identity:
$$\exists! \mathfrak{e} \in \mathfrak{G} \mid \mathfrak{g} = \mathfrak{e}\mathfrak{g} = \mathfrak{g}\mathfrak{e} \forall \mathfrak{g} \in \mathfrak{G} \tag{3.10}$$
- Inverse:
$$\forall \mathfrak{g} \in \mathfrak{G} \exists! \mathfrak{g}^{-1} \in \mathfrak{G} \mid \mathfrak{g}\mathfrak{g}^{-1} = \mathfrak{g}^{-1}\mathfrak{g} = \mathfrak{e} \tag{3.11}$$
- Closure:
$$\forall \mathfrak{g}, \mathfrak{h} \in \mathfrak{G} \ \mathfrak{g}\mathfrak{h} \in \mathfrak{G} \tag{3.12}$$

Definition 3.14 (Abelian Groups)**.** A commutative group $\mathfrak{G}$ namely satisfying:

$$\mathfrak{g}\mathfrak{h} = \mathfrak{h}\mathfrak{g} \forall \mathfrak{g}, \mathfrak{h} \in \mathfrak{G} \tag{3.13}$$

Is Abelian.

Definition 3.15 (Group generators S)**.** Let $S \subseteq \mathfrak{G}$ where $\mathfrak{G}$ is a group with some operation. S is a generator of $\mathfrak{G}$ if every element of the group can be expressed as a **finite** composition of elements and inverses of the elements of S.

$$S : \langle S \rangle = \left\{ \mathfrak{g} = s_1 s_2 \dots s_n \mid n \leq \infty, \ s_i \in S \vee s_i^{-1} \in S, \ i \in \{1, \dots, n \right\} = \mathfrak{G} \tag{3.14}$$

In some cases, the infinite size of a group will be reduced to its set of generators, thus easing the formalization process.

Example 3.16 (Back to triangle symmetries)**.** The symmetries of a triangle of Figure 3.1 are generated by $S : |S| = 2$ where the two elements are:

- 60 degrees rotation
- Simple mirror reflection

Observation 3.17 (Groups are more general than symmetries)**.** *Definition 3.15 does not imply that the elements are necessarily symmetries. For example, considering again the symmetries of a triangle of Figure 3.1, these can be interpreted as permutations over the set* $\Omega = \{1, 2, 3\}$ *where each vertex is a number and a reference system is chosen.*

From Definition 3.11 the algebraic structure of the group is stated as the set of transformations:

$$\mathfrak{g} : \Omega \to \Omega \tag{3.15}$$

The interest for these objects arises when they are applied to Ω and transferred in the signal space $\mathcal{X}(\Omega, \mathcal{C})$.

Definition 3.18 (Group Action)**.** A group action is a map defined on $\mathfrak{G} \times \Omega \to \Omega$ denoted as $(\mathfrak{g}, u) \to \mathfrak{g}.u$. It must be compatible with the group axioms of Definition 3.15$\forall \mathfrak{g}, \mathfrak{h} \in \mathfrak{G} \forall u \in \Omega$.

$$\begin{aligned}
&(\mathfrak{g}\mathfrak{h})\mathfrak{l}.u = \mathfrak{g}(\mathfrak{h}\mathfrak{l}.u) \\
&\exists! \mathfrak{e} \in \mathfrak{G} \mid \mathfrak{g}.u = \mathfrak{e}\mathfrak{g}.u = \mathfrak{g}\mathfrak{e}.u \\
&\exists! \mathfrak{g}^{-1} \in \mathfrak{G} \mid \mathfrak{g}\mathfrak{g}^{-1}.u = \mathfrak{g}^{-1}\mathfrak{g}.u = \mathfrak{e}.u \\
&\mathfrak{g}\mathfrak{h}.u \in \mathfrak{G}
\end{aligned}$$

Theorem 3.19 (Sufficient conditions for Group Action)**.** *For a $\mathfrak{G}$ to act on Ω it is sufficient to check that the following holds:*

$$\mathfrak{g}.(\mathfrak{h}.u) = (\mathfrak{g}\mathfrak{h}).u \tag{3.16}$$

Proof. If $\mathfrak{g}.(\mathfrak{h}.u) = (\mathfrak{g}\mathfrak{h}).u$ then any composition of elements counts as a group element applied to that map. Closure is enough as it guarantees the existance of identities, inverses and associative properties trivially by using the condition. □

Example 3.20 (Euclidean group and images)**.** Let $\mathfrak{G} = E(2)$ the Euclidean group on $\Omega = \mathbb{R}^2$ consisting of translations on both axes and rotations. It is expressed as a matrix:

$$\mathfrak{g} = (\theta, t_x, t_y) = \begin{bmatrix} \cos\theta & \sin\theta & t_x \\ -\sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.17}$$

The map from element of the euclidean space $\mathbb{R}^2$ and this matrix to the new element is a group action:

$$(\mathfrak{g}, u) = ((\theta, t_x, t_y), (x, y)) \to \mathfrak{g}.u = \begin{bmatrix} \cos\theta & \sin\theta & t_x \\ -\sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{3.18}$$

The same group action on a set can act on its signals $\mathcal{X}(\Omega, \mathcal{C})$. Indeed, if the euclidean space element is rotate, so is its image. The requirement for validity is stated in Proposition 3.21.

Proposition 3.21 (Group Action on signal space)**.** *Let $\mathfrak{G}$ act on Ω, then the action of $\mathfrak{G}$ on $\mathcal{X}(\Omega, \mathcal{C})$ is:*

$$(\mathfrak{g}.x)(u) = x(\mathfrak{g}^{-1}u) \tag{3.19}$$

Proof. By Theorem 3.19 it is sufficient to check the composition distribution $\forall \mathfrak{g}, \mathfrak{h} \in \mathfrak{G}, \forall x \in \mathcal{X}(\Omega, \mathcal{C})$.

$$\begin{aligned}
(\mathfrak{g}.(\mathfrak{h}.x))(u) &= && \text{outer composition} && (3.20)\\
&= (\mathfrak{h}.x)(\mathfrak{g}^{-1}u) && \text{assumption applied on } \mathfrak{g} && (3.21)\\
&= x(\mathfrak{h}^{-1}\mathfrak{g}^{-1}u) && \text{assumption applied on } \mathfrak{h} && (3.22)\\
&= ((\mathfrak{g}\mathfrak{h}).x)(u) && \text{equivalent to the assumption} && (3.23)
\end{aligned}$$

Where the last passage comes from the composition of inverse group elements which is of inversed order, i.e. $(\mathfrak{g}\mathfrak{h})^{-1} = \mathfrak{h}^{-1}\mathfrak{g}^{-1}$. Thus, for the defined group action, composition is verified, and disjoint group operations join together. Thus, by Theroem 3.19 the defined operation is sufficient to form a group action satisfying the axioms. □

A graphical example from the slides is shown in Figure 3.4

Of this broad collection of group actions, an interesting kind that will be reminded often is that of Group Representations, which add a linearity property to the map.

Definition 3.22 (Group Representation, with map ρ)**.** A group representation is a linear group action such that:

$$\mathfrak{g}.(\alpha x + \beta x') = \alpha \mathfrak{g}.x + \beta \mathfrak{g}.x' \quad \forall \alpha, \beta \in \mathbb{R},\ \forall x, x' \in \mathcal{X}(\Omega, \mathcal{C}) \tag{3.24}$$

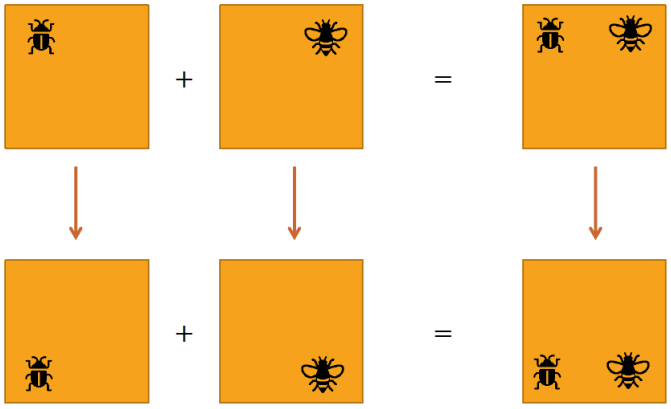


Figure 3.4: Group action on signal space

It has an n dimensional representation[3] denoted by a matrix $\rho : \mathfrak{G} \to \mathbb{R}^n \times \mathbb{R}^n$ which has an inverse for each group element:

$$\exists \rho^{-1} \quad \forall \mathfrak{g} \in \mathfrak{G} \tag{3.25}$$

And works with composition:

$$\rho(\mathfrak{g}\mathfrak{h}) = \rho(\mathfrak{g})\rho(\mathfrak{h}) \quad \forall \mathfrak{g}, \mathfrak{h} \in \mathfrak{G} \tag{3.26}$$

Proposition 3.23 (Group Representation map on signal space)**.** *As in Proposition 3.21 it can be verified that:*

$$\rho(\mathfrak{g})x(u) = x(\mathfrak{g}^{-1}u) \implies (\rho(\mathfrak{g}))(\rho(\mathfrak{h}x)(u) = (\rho(\mathfrak{g}\mathfrak{h})(x))(u) \quad \forall \mathfrak{g}, \mathfrak{h} \in \mathfrak{G}, \forall x \in \mathcal{X}(\Omega, \mathcal{C}) \tag{3.27}$$

Proof. Use the same concepts of Proposition 3.21 □

Some examples below will clarify important concepts linked to representations.

Example 3.24 (Translating Pixels)**.** Let $\mathfrak{G} = (\mathbb{Z}, +)$ the set of integer translations with addition. Consider as domain $\Omega = \mathbb{Z}_5$. An action is:

$$\mathfrak{g} = n \; on \; u \in \Omega : (\mathfrak{g}, u) \to (n + u)(mod 5) \tag{3.28}$$

Which is a 5 cyclic shift, represented in the signals space $\mathcal{X}(\Omega, \mathcal{C})$ as a matrix such that the single shift is repeated n times:

$$\rho(n) = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}^n \quad \forall \mathfrak{g} = n \in \mathfrak{G}$$

[3] Where n is in general arbitrary and not linked to the size of $\mathfrak{G}$ or Ω, but will often coincide in the use case of Neural Networks.

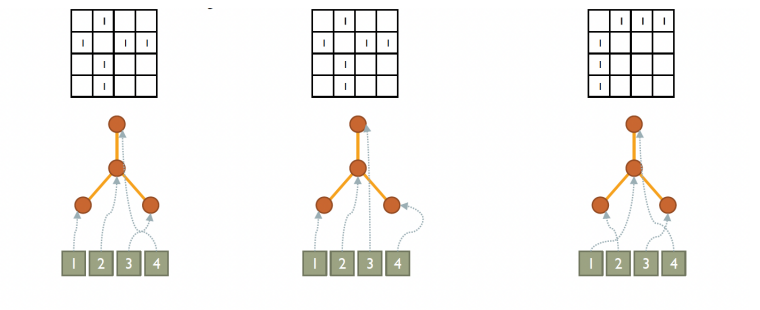


Figure 3.5: Symmetry of the description

Example 3.25 (Permuting Vertices)**.** Let $Gr = \mathbb{S}_n$ the group of permutations on the set $\{1, \ldots, n\}$ and $\Omega = V$ the set of vertices of a graph. Then, the elements of the group are equivalent to permutations represented as $P_{indices}$. There are three kinds of representations for $\mathbb{S}_n$.
For a classification task, the wish is that up to permutation the given label is the same, namely a transformation ρ_0 such that:

$$\rho_0(P)s = 1 \cdot s \forall P \in \mathfrak{G} \tag{3.29}$$

If a feature is associated to the nodes, the representation ρ_2 should mimic a permutation, as to preserve the assignment to each node:

$$\rho_1(P)s = Ps \forall P \in \mathfrak{G} \tag{3.30}$$

If a property is stored in a binary relation (such as edge connections with an adjacency matrix, possibly weighted), the representation ρ_2 should maintain the feature up to transformations on both ends. Namely for an *adjacency like* matrix M it will be the case that:

$$\rho_2(P)M = PMP^T \quad \forall P \in \mathfrak{G} \tag{3.31}$$

It is possible to check that all of the above satisfy the axioms of a representation.

While this approach is well defined in a mathematical framework, it is worth stressing that sometimes the implementation at a computer level adds layers of potential representations that are not proper to the object. For example, a graph is an unordered collection of vertices and edges, but when coded, it is intrinsically assigned an order, which however should not be considered when defining representations. It is important to consider symmetries of the **representation** and **not** symmetries of the description.

Example 3.26 (Graphs and order)**.** Given a graph, a number is assigned for each vertex to denoted it. The usual example is $\{1, \ldots, n\}$. For this graph, many adjacency matrices cna be defined by swapping the order of the rows and columns, where the symmetry is of the representation. On the contrary, assigning different numbers to different vertices woudl give rise to the same graph, but from a symmetry of the description. Figure 3.5 shows the symmetry we are **not** interested in.

In later lectures, grids groups and manifolds will be explored further in terms of the just introduced objects.

Figure 3.6: Partial rotation damages learning

In the context of a symmetry of the label, it is possible to give a more formal treatment of the subject through the concept of orbits and representations.

Definition 3.27 (Orbit O_x)**.** For a given signal $x \in \mathcal{X}(\Omega, \mathcal{C})$ an orbit O_x with respect to a group $\mathfrak{G}$ is the set of all possible representations of such signal.

$$O_x = \{\mathfrak{g}.x \in \mathcal{X}(\Omega, \mathcal{C}) \mid x \in \mathcal{X}(\Omega, \mathcal{C}), \mathfrak{g} \in \mathfrak{G}\} \tag{3.32}$$

3.3 Invariance and Equivariance

The symmetry of signals trivially returns a structured function to learn, as symmetric elements must map to the same output space. This characterization is that of invariant representations.

Definition 3.28 (Invariant Representation)**.** A function $f : \mathcal{X}(\Omega, \mathcal{C}) \to \mathcal{Y}$ is $\mathfrak{G}$-invariant if:

$$f(\rho(\mathfrak{g})x) = f(x) \quad \forall \mathfrak{g} \in \mathfrak{G}, x \in \mathcal{X}(\Omega, \mathcal{C}) \tag{3.33}$$

Namely, for each orbit of symmetry the function does not change its output.

On the other hand, in the context of Convolutional Neural Networks, the microscopic to macroscpic judgment process of the classifier would be damaged by invariant representations at intermediate levels. Consider as an example rotating only part of the image, as shown in Figure 3.6

For this purpose, while invariant subparts are not good, equivariant subparts are what is needed.

Definition 3.29 (Shift Equivariant functions)**.** A function $f : \mathcal{X}(\Omega, \mathcal{C}) \to \mathcal{X}(\Omega, \mathcal{C})$ is $\mathfrak{G}$-equivariant if:

$$f(\rho(\mathfrak{g})x) = \rho(\mathfrak{g})f(x) \quad \forall \mathfrak{g} \in \mathfrak{G} \tag{3.34}$$

Namely, input and output change accordingly in the same way for intermediate layers of representation.

Example 3.30 (Shift Invariance and equivariance in Computer Vision)**.** Moving an object across the space should not contaminate its classification as for example a cat or a dog.
For a Convolutional Neural Network, intermediate layers are equivariant.

Definition 3.31 (Equivariant Networks)**.** Build a general architecture where:

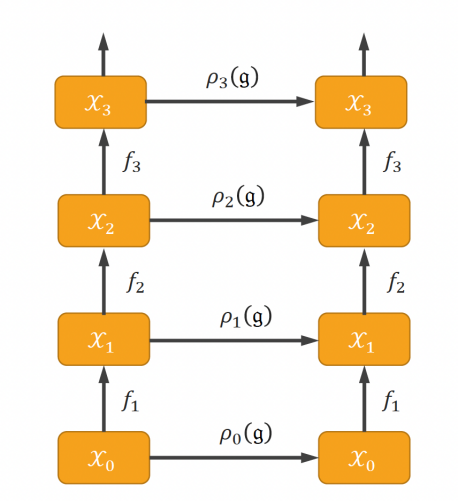


Figure 3.7: Equivariant Network

- $i = 1, \ldots, L$ are layers fo the network
- $\mathcal{X}_i$ are feature vector spaces
- f_i are maps between layers
- $\mathfrak{G}$ is a symmetry group with representations $\rho_i \forall \mathcal{X}_i$ such that:

$$f_i \circ \rho_{i-1}(\mathfrak{g})(x_i) = \rho_i(\mathfrak{g}) \circ f_i(x_i) \quad \forall x_i \in \mathcal{X}_\rangle \tag{3.35}$$

A graphical example is shown in Figure 3.7 It is also possible to show that if all layers are equivariant then their composition is.

Equivariance provides consistency across orbits. Indeed it enforces that objects in the input space that map to the same output also do so if the same representation is mapping is applied to them. Formally, and equivariant network is such that:

$$x, y \in \mathcal{X}(\Omega, \mathcal{C}) : f(x) = f(y) \implies f \circ \rho_1(\mathfrak{g})x = f \circ \rho_1(\mathfrak{g})y = \rho_2(\mathfrak{g}) \circ f(x) \tag{3.36}$$

TODO Isomorphisms and Automorphisms

It was found that CNNs learn equivariant representations up to some degree [LV15].
Summarizing it is worth recalling that:

- equivariance is layerwise, not on data as a whole
- a symmetry on the domain acts linearly, with a group representation
- often, this leads to label symmetry
- **TODO check**

Recalling previous steps we have that:

- The aim is to learn $f^* : \mathcal{X} \to \mathbb{R}$
- The input space is a signal on a geometric domain Ω denoted as $\mathcal{X}(\Omega, \mathcal{C})$
- Ω presents transformations through a group $\mathfrak{G}$
- The latter are seen as group representations on the signal space $\mathfrak{g} : \mathcal{X}(\Omega, \mathcal{C}) \to \mathcal{X}(\Omega, \mathcal{C})$ as in Proposition 3.21

There, the aim is translated to learning an optimal candidate in the hypothesis class[4] $\mathcal{F}(\mathcal{X}(\Omega, \mathcal{C}), \Omega)$. An additional assumption ensures that the curse of dimensionality can be avoided.

Assumption 3.32 (Invariance of the target function)**.** The target function f^* is $\mathfrak{G}$-invariant.

$$f^*(\mathfrak{g}.x) = f^*(x) \quad \forall x \in \mathcal{X}(\Omega, \mathcal{C}), \mathfrak{g} \in \mathfrak{G} \tag{3.37}$$

TODO book 3.2 3.3 A very natural and intuititive object is the group smoothing operator.

Definition 3.33 (Group Smoothing Operator $S_{\mathfrak{G}}$)**.** For a group $\mathfrak{G}$ which is discrete and finite define:

$$S_{\mathfrak{G}} f = \frac{1}{|\mathfrak{G}|} \sum_{\mathfrak{g} \in \mathfrak{G}} f \circ \mathfrak{g} \implies S_{\mathfrak{G}} f(x) = \frac{1}{|\mathfrak{G}|} \sum_{\mathfrak{g} \in \mathfrak{G}} f(\mathfrak{g}.x) \quad \forall x \in \mathcal{X}(\Omega, \mathcal{C}) \tag{3.38}$$

Which is the average over an orbit.
For the general case, the definition becomes:

$$f(x) = \frac{1}{\mu(\mathfrak{G})} \int_{\mathfrak{G}} f(\mathfrak{g}.x) d\mu(\mathfrak{g}) = f\left(\frac{1}{\mu(\mathfrak{G})} \int_{\mathfrak{G}} \mathfrak{g}.x d\mu(\mathfrak{g}) \right) \tag{3.39}$$

Where $\mu(\cdot)$ is the *Haar measure* of a group **TODO maybe expand**

Theorem 3.34 (Smoothing operator properties)**.** *Considering a smoothing operator the following hold.*

1. *The smoothing does not affect the original function.*

$$S_{\mathfrak{G}} f^* = f^* \tag{3.40}$$

2. *it is possible to create an invariant function class as:*

$$S_{\mathfrak{G}} \mathcal{F} = \{ S_{\mathfrak{G}} f : f \in \mathcal{F} \} \tag{3.41}$$

Proof. Consider the easy case of Equation 3.38 for simplicity. For point 1 observe that:

$$S_{\mathfrak{G}} f^*(x) = \frac{1}{|\mathfrak{G}|} \sum_{\mathfrak{g} \in \mathfrak{G}} f^*(\mathfrak{g}.x) = \frac{1}{|\mathfrak{G}|} \sum_{\mathfrak{g} \in \mathfrak{G}} f^*(x) = \frac{1}{|\mathfrak{G}|} |\mathfrak{G}| f^*(x) = f^*(x) \quad \forall x \in \mathcal{X}(\Omega, \mathcal{C}) \tag{3.42}$$

Where the first inequality is by definition and the second by assumption.
Considering point 2, it is also sufficient to apply the definition. Indeed a smoothed operator $S_{\mathfrak{G}} f$ is such that:

$$S_{\mathfrak{G}} f(\mathfrak{h}.x) = \frac{1}{|\mathfrak{G}|} \sum_{\mathfrak{g} \in \mathfrak{G}} f(\mathfrak{g}.(\mathfrak{h}.x)) = S_{\mathfrak{G}} f(x) \quad \forall x \in \mathcal{X}(\Omega, \mathcal{C}) \tag{3.43}$$

TODO check
Which is the definition of invariance. □

Definition 3.35 (Invariant Function Class $S_{\mathfrak{G}}\mathcal{F}$)**.** Using the objects introduced before and the result of Theorem 3.34, the invariant function class is defined as:

$$\{ S_{\mathfrak{G}} f : f \in \mathcal{F} \} \tag{3.44}$$

For a family of functions $\mathcal{F}$

[4] For simplicity often denoted as $\mathcal{F}(\mathcal{X}(\Omega, \mathcal{C}))$

An example is proposed below.

Example 3.36 (Grids and invariance groups)**.** Let $\Omega = \{1, \dots, d\}$ and $\mathfrak{G} = C_d$ the cyclic group of order d. Consider as Function class $\mathcal{F} = \{f(x) = p_k(x_1, \dots, x_d)\}$ the set of polynomials of degree k. Then, the invariant class is:

$$S_{\mathfrak{G}}\mathcal{F} = \{S_{\mathfrak{G}} f : f \in \mathcal{F}\} \tag{3.45}$$

$$\implies S_{\mathfrak{G}} f = \frac{1}{d} \sum_{\mathfrak{g} \in \mathfrak{G}} f \circ \mathfrak{g} \tag{3.46}$$

$$\implies S_{\mathfrak{G}} f(x) = \frac{1}{d} \sum_{\mathfrak{g} \in \mathfrak{G}} p_k(g.(x_1, \dots, x_d)) \tag{3.47}$$

$$= \frac{1}{d} \sum_{\mathfrak{g} \in \mathfrak{G}} p_k(\mathfrak{g}.x) \tag{3.48}$$

The coefficients of the polynomial are fixed, while the x_is are cyclically exchanged. This means that the mean of all the d transformations is:

$$S_{\mathfrak{G}} f(x) = p_k(\overline{x} \dots, \overline{x}) : \overline{x} = \frac{1}{d} \sum_{i=1}^{d} x_i \quad \forall p_k \in \mathcal{F}, x \in \mathcal{X}(\Omega, \mathcal{C}) \tag{3.49}$$

If instead the group is the symmetric group $\mathfrak{G} = \mathbb{S}_d$ then there are $|\mathfrak{G}| = d!$ elements and:

$$S_{\mathfrak{G}} f(x) = \frac{1}{d!} \sum_{\mathfrak{g} \in \mathfrak{G}} p_k(\mathfrak{g}.x)$$

$$= \frac{1}{d!} \textbf{TODO}$$

Proposition 3.37 (Approximation error and Smoothing)**.** *Implementing a smoothing operator, the approximation error is not affected.*

Proof. Applying the definition:

$$\varepsilon_{appr} = \inf_{f \in \mathcal{F}} \left\{ \|f - f^*\|^2 \right\}$$

Where the smoothing operator is an orthogonal projection such that:

$$\begin{aligned} \|f - f^*\|^2 &= \|S_{\mathfrak{G}} f - S_{\mathfrak{G}} f^*\|^2 + \|(I - S_{\mathfrak{G}}) f - (I - S_{\mathfrak{G}}) f^*\|^2 && \text{By orthogonalization} \\ &= \|S_{\mathfrak{G}} f - f^*\|^2 + \|(I - S_{\mathfrak{G}}) f\|^2 && \text{By Theorem 3.34} \end{aligned}$$

Where taking the infimum the result is the same in the restricted class:

$$\inf_{f \in \mathcal{F}} \left\{ \|f - f^*\|^2 \right\} = \inf_{f \in S_{\mathfrak{G}}\mathcal{F}} \left\{ \|f - f^*\|^2 \right\} \tag{3.50}$$

TODO check definitions do not coincide □

While the approximation error is not reduced, it can be shown that the statistical error is. Making the hypothesis class smaller, this is guaranteed, a quantification of the error for Lipschitz functions is proposed in the next exercise [MMM21],[BVB21][LB03].

Example 3.38 (Lipschitz smoothing)**.** Let $\mathcal{F}$ be the class of β-Lipschitz functions. Its smoothed version takes into account the group structure in the *Lipschitzness* as:

$$f \in S_{\mathfrak{G}}\mathcal{F} : |f(x) - f(x')| \leq \beta \inf_{\mathfrak{g}\in\mathfrak{G}} \left\{ \|x - \mathfrak{g}.x'\| \right\} \tag{3.51}$$

TODO check or prove

Theorem 3.39 (Kernel Ridge Regression)**.** *Using a $\mathfrak{G}$-invariant Kernel Ridge Regression the generalization error of a Lipschitz $\mathfrak{G}$-invariant function f^* is such that:*

$$\mathbb{E}[\mathcal{R}(\tilde{f})] \lesssim (|\mathfrak{G}|n)^{-\frac{1}{d}} \tag{3.52}$$

The above result is still exponential in the size of the Group, and proves that invariance is not sufficient to avoid the curse of dimensionality, due to the exponential in d relation.
Some open problems in this direction give a degree of confidence for the tightness of this degree, but are still to be explored. Summarizing, the two key conclusions are:

1. Group invariance through symmetries is not sufficient to break the curse of dimensionality
2. It is still unclear how to derive the classes practically

3.4 Scale Separation

TODO book 3.4

The methodology introduced by many researchers makes use of the compositionality principle to break the curse of dimensionality. A hierarchical structure could be useful, but a formalization in this framework is missing. Inferencing from the Physics field, multiscale structures are considered. The applications include but are not limited to:

- turbulence
- percolation

In this document, the basics of Multi Resolution Analysis (MRA) will be outlined in the context of a 2D grid as Ω. The treatment is informal.
A signal $x \in L^2(\Omega)$ is decomposed into:

- a new signal $\widetilde{x} \in L^2(\widetilde{\Omega})$ where $\widetilde{\Omega}$ is a coarser grid
- information to obtain x from $\widetilde{x}$

This tool goes far beyond ML and is based on concepts from Fourier Analysis. The link with high dimensional learning is the use of a scale separating prior.
If the target function is such that $f^* \approx \widetilde{f^*}(\widetilde{x})$ so f^* can be approximated in a coarser domain, then it can be stated that:

$$\textit{since } dim(\mathcal{X}) \propto |\Omega|, |\widetilde{\Omega}| \ll |\Omega| \implies \mathcal{X}(\widetilde{\Omega}) \textit{ avoids curse} \tag{3.53}$$

As the coarsed signal is enough. However, as shown in Example 3.40, this is not always enough, and additional considerations must be made.

Figure 3.8: Mountain vs Beach

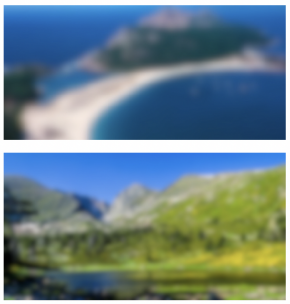


Figure 3.9: Mountain vs Beach coarsed

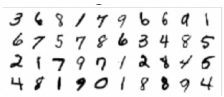


Figure 3.10: MNIST

Figure 3.11: MNIST coarsed

Figure 3.12: Patches are the image

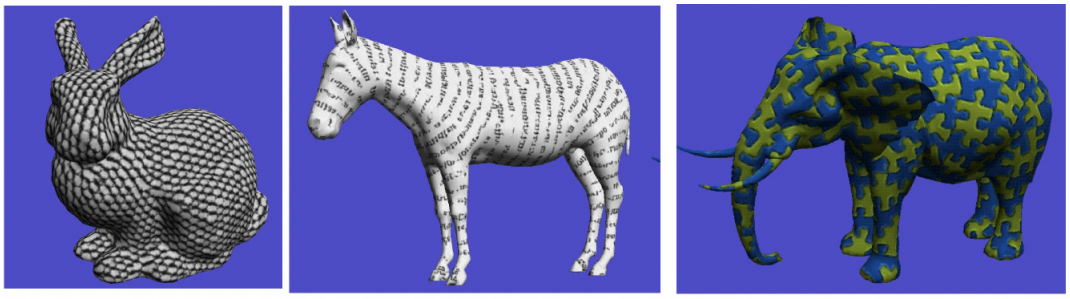


Figure 3.13: Patches are not the image

Example 3.40 (Mountains, Beaches, MNIST)**.** The task of distinguishing mountains from beaches is easy if the image is downgraded, while MNIST is not trivial if the quality is reduced. See Figures 3.8, 3.9, 3.10, 3.11 for reference.

In another situation, assume that the target function is such that: $f^*(x) \approx \sum_u g(x_u)$ where f^* is approximated by a sum of patches. This case happens when the structure has a spatial homogeneity underlining. The relevant dimension is that of the patch, and the curse is avoided. More details can be found in [FCW21]. In Figures 3.12 and 3.13 it is possible to see a good and a bad approximation. In the latter, approximating with local classification makes classifies an animal as the texture on the surface.

While these two cases present weaknesses, a combination of both presents high adaptability and potential. Moreover, adding a coarse level function $\widetilde{f}$ that captures information about the transformed input it is possible to build a compositional chain where the signal is coarsed to a different space through a non linear transformation and a non linear function is learnt. The hierarchical structure imposed will hopefully come with an improved performance. Figure 3.14 provides a graphical example of the process.

So far, compositional models have not been completely understood. There are instances in which the approach is guaranteed to come with a benefit such as Dynamic Programming. Yet, the theoretical landscape is still nebulous. Nevertheless, as shown in the next section, group invariance and scale separation can characterize Deep Learning models completely.

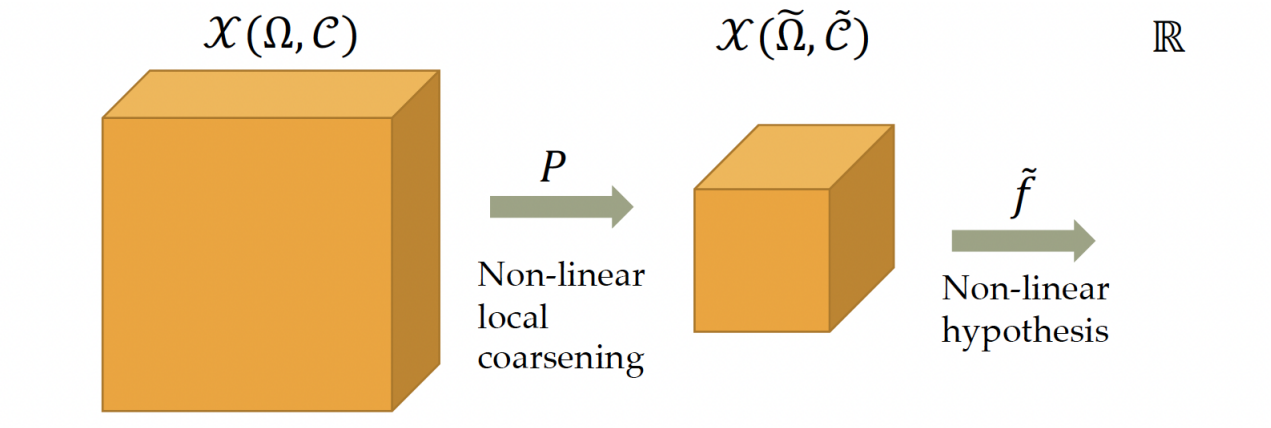


Figure 3.14: Compositional Model

3.5 The Blueprint of Geometric Deep Learning

Considering the just reported notions of simmetry and scale separation it is possible to learn representations of high dimensional functions f on the space of signals $\mathcal{X}(\Omega, \mathcal{C})$, with an underlying structure Ω and a paired symmetry group $\mathfrak{G}$.

For this purpose, the group average and a composition property of maps are introduced.

Definition 3.41 (Group Average Operator A)**.** For a signal and a group symmetry, simply define the group average as the average over the orbit of such signal.

$$Ax = \frac{1}{\mu(\mathfrak{g})} \int_{\mathfrak{G}} \mathfrak{g}.x d\mu(\mathfrak{g}) \tag{3.54}$$

Theorem 3.42 (Composition of Linear Equivariants and Local non Linear Maps)**.** *Let $B : \mathcal{X}(\Omega, \mathcal{C}) \to \mathcal{X}(\Omega, \mathcal{C}')$ be equivariant and $\sigma : \mathcal{C}' \to \mathcal{C}''$ be a non linear (possibly local) map. Assume further that σ is applied element wise, such that: $(\sigma(x))(u) := \sigma(x(u))$. Then, the composition:*

$$U := (\sigma \circ B) : \mathcal{X}(\Omega, \mathcal{C}) \to \mathcal{X}(\Omega, \mathcal{C}'') \tag{3.55}$$

Is $\mathfrak{G}$-equivariant.

Proof. Observe that U maps signals to different channel spaces, keeping the structure fixed. The aim is to show that:

$$U(\mathfrak{g}.x) = \mathfrak{g}.U(x) \quad \forall x \in \mathcal{X}(\Omega, \mathcal{C}), \mathfrak{g} \in \mathfrak{G} \tag{3.56}$$

Working out the steps:

$$\begin{aligned}
U(\mathfrak{g}.x) &= \sigma \circ B(\mathfrak{g}.x) && \text{By definition} && (3.57)\\
&= \sigma \circ \mathfrak{g}.B(x) && \text{By } B \text{ equivariant} && (3.58)\\
&= (\sigma(x'))(u) && \text{Where } x' = \mathfrak{g}.B(x) && (3.59)\\
&= \sigma(x'(u)) && \text{By definition} && (3.60)\\
&= \mathfrak{g}.\sigma(B(x)(u)) && \text{replacing } x' \text{ and by element wise } \sigma && (3.61)\\
&= \mathfrak{g}.U(x) && \text{By definition} && (3.62)
\end{aligned}$$

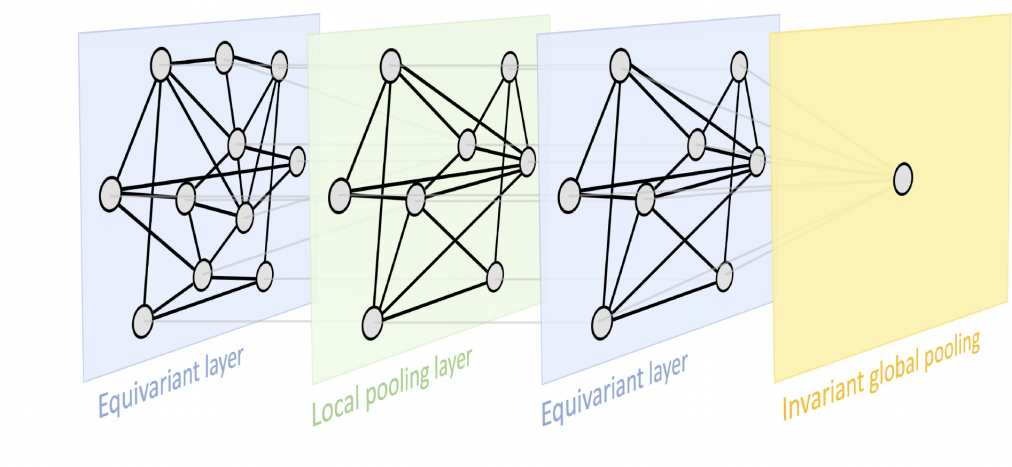


Figure 3.15: Graph neural network GDL Blueprint

Thus, the composed map is $\mathfrak{G}$-equivariant. □

In the following Chapters, it will be shown that the combination of a linear $\mathfrak{G}$ invariant map through the average smoothing operator $S_{\mathfrak{G}}f$ and a series of equivariant maps U is sufficient to build most of the currently known network architectures, and to adapt to universal approximation theorems. This set of objects is referred to as the GDL blueprint, and is summarized below to put all the ideas together.

Definition 3.43 (The GDL Blueprint)**.** Collect the following building blocks:

- A linear $\mathfrak{G}$-equivariant layer $B : \mathcal{X}(\Omega, \mathcal{C}) \to \mathcal{X}(\Omega', \mathcal{C}')$
- A non linearity $\sigma : \mathcal{C} \to \mathcal{C}'$ applied element wise as $(\sigma(x))(u) = \sigma(x(u))$
- A coarsening operator (local pooling) $P : \mathcal{X}(\Omega, \mathcal{C}) \to \mathcal{X}(\omega', \mathcal{C})$ where $\Omega' \subseteq \Omega$
- A $\mathfrak{G}$-invariant (global pooling) layer $A : \mathcal{X}(\Omega, \mathcal{C}) \to \mathcal{Y}$

By Theorems 3.34 and 3.42 a composition[5] such as:

$$f = A \circ \sigma_J \circ B_J \circ P_{J-1} \circ \dots P_1 \circ \sigma_1 \circ B_1 \tag{3.63}$$

Makes $f : \mathcal{X}(\Omega, \mathcal{C}) \to \mathcal{Y}$ $\mathfrak{G}$-invariant. An example for a Neural Network on a Graph is shown in Figure 3.15.

Observation 3.44 (On the Blueprint)**.** *It is worth clearing out that:*

- *B is non linear and $\mathfrak{G}$-Equivariant since the non linearity is provided by σ*
- *Global pooling aggregates information of the entire domain*

In conclusion, the concepts of Geometrics Priors, invariance and equivariance provide a principled construction of Neural Network architectures through the introduction of symmetries and scale separation. While the precise assumptions are still to be cleared form a theoretical perspective, it is possible that the hypothesis spaces designed with such method may break the curse of dimensionality. In the next Chapters, the building blocks of such method will be cleared further.

[5]Where input and output dimensions of adjacent maps coincide

Popular architectures, with the right choice of the domain Ω, perfectly align with the GDL Blueprint.

Chapter 4

Graphs & Sets

The image of the world around us, which we carry in our head, is just a model. Nobody in his head imagines all the world, government or country. He has only selected concepts, and relationships between them, and uses those to represent the real system.

Jay Wright Forrester

In the previous discussions, it was shown that there are guiding principles to synthesize and steer the design of Deep Learning architectures. This framework allows to reason about old approaches and propose new methods. The guiding principles can be summarized as:

- Symmetries
- $\mathfrak{G}$-Equivariance
- $\mathfrak{G}$-Invariance
- Locality
- Scale Separation

This Chapter is focused on Graphs and sets, and more precisely on implementations such as:

- Graph Neural Networks (GNN)
- *Deepsets*
- Transformers

The first lecture, from Section 4.1.1 to 4.1.3 is devolved to understanding:

- the blueprint on graphs & sets
- permutation invariance and equivariance notions
- *Deepsets* [Zah+18] as the blueprint for set neural networks
- permutation equivariant GNN layers

4.1 Foundational Bricks

Observation 4.1 (Why start with graphs?)**.** *Graphs are a convenient discrete domain with minimal geometric assumptions, and great generalization power. Most of the structures expanded in later Chapters are up to some point graphs.*

For general graphs and sets, symmetry is instantiated whenever two objects present the same underlying structure but a different appearance.

Definition 4.2 (Permutation Group Σ_n)**.** A permutation group Σ_n for a set of unordered elements $\{1, \dots, n\}$ is the set of bijections paired with the composition of permutations.
It is the symmetry group of Graphs and Sets, i.e. $\mathfrak{G} = \Sigma_n$.

4.1.1 Sets

Sets are unconnected graphs (i.e. vertices with not edges), with the peculiarity of a simpler domain, and many conclusions carrying over well to the general setting.

Definition 4.3 (Setup for learning on a set)**.** The domain is an unordered collection of points $\Omega = \mathcal{V}$, where $|\mathcal{V}| = n$. Each point has a feature $x_i \in \mathbb{R}^k \forall i$, which is the channel of the signal[1].
The features can be stacked in a node feature matrix:

$$X = (x_1, \dots, x_n)^T \in \mathbb{R}^{|\mathcal{V}|} \times \mathbb{R}^k \tag{4.1}$$

While this is needed for processing purposes, it inherently specifies an **order** (row wise), which is to be inhibited.
For n elements, there are $n!$ ways to order them, unambiguously specified by permutations, which have as group representation of the group action (from Definition 3.22) the set of permutation matrices:

$$\forall \mathfrak{g} \in \mathfrak{G} \exists! \rho(\mathfrak{g}) = P \in \mathbb{R}^{|\mathcal{V}|} \times \mathbb{R}^{|\mathcal{V}|} \quad where \quad \begin{cases} P_{vu} \in \{0,1\} & \forall v, u \in \mathcal{V} \\ \sum_u P_{vu} = 1 & \sum_v P_{uv} = 1 \end{cases} \tag{4.2}$$

Following the Blueprint from Definition 3.43, we impose that:

$$f(PX) = f(X) \forall P \tag{4.3}$$

Where in this case $f \equiv A$ the group average and the role of $\mathfrak{g}$ is expressed by P.

The final invariant aggregation that f performs can be of many types. For the *Deepsets* model, which will be analized further in other subsections, we have that[Zah+18]:

$$f(X) = \phi\left(\sum_{i \in \mathcal{V}} \psi(x_i) \right) \tag{4.4}$$

Where the invariance is guaranteed by the operator $\sum$. However, there are other aggregators that are not affected by the order such as:

[1] Namely $\mathcal{C} = \mathbb{R}^k$

- avg
- max
- min

For this reason, a general notation is implemented.

Definition 4.4 (Permutation Invariant Operator $\bigoplus$)**.** Denote a general permutation invariant operator over a set $\mathcal{V}$ as $\bigoplus_{v \in \mathcal{V}}$

At the intermediate layers, where the computations are performed at a node level, invariance destroys the diversity of the input. Permutation equivariance, again from the Blueprint of Definition 3.43, is adapted as:

$$F(PX) = PF(X) \forall P \tag{4.5}$$

As discussed before **TODO not discussed, expand**, an important concept is preserving locality of operations to guarantee stability under deformations. Indeed, locality ensures that errors do not propagate at a global level. For a set, locality is implemented by imposing that the transformation for each layer is done node wise as:

$$h_i = \psi(x_i) \forall i \quad stacked\ as \quad H = F(X) \tag{4.6}$$

Without additional structure or assumptions, this approach is usually the most descriptive possible.

The formalization, as before, is an aggregation of equivariant $\psi(\cdot)$ functions with an invariant tail to aggregate if needed $\psi(\cdot)$, summarized as:

$$f(X) = \phi\left(\bigoplus_{i \in \mathcal{V}} \psi(x_i) \right) \tag{4.7}$$

4.1.2 Graphs

Definition 4.5 (Graph $\mathcal{G}$)**.** A graph is a set with connections, denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. THe easiest representation of a graph is its adjacency matrix $\mathbf{A}$:

$$\mathbf{A} = \begin{cases} a_{ij} = 1 & (i,j) \in \mathcal{E} \\ a_{ij} = 0 & otherwise \end{cases} \tag{4.8}$$

Definition 4.6 (Setup for learning on a graph)**.** The domain now includes the edges $\Omega = (\mathcal{V}, \mathcal{E})$. Each vertex has a feature $x_i \in \mathbb{R}^k$. While edges can have features, for the moment they will be saved for the more general case discussed just after.
In this setting, an *unwanted* order is set for the rows of X and rows and columns of $\mathbf{A}$.

As before, permutation invariance and equivariance must hold in the respective cases. A function on the space of signals takes two inputs: features X and connections $\mathbf{A}$. The two are permuted differently. X is permuted on the row order only, $\mathbf{A}$ is permuted on both rows and columns.

$$f \mid f(X, A) \to \mathbb{R} \quad F \mid F(X, A) \to \mathcal{X}(\Omega', \mathcal{C}') \tag{4.9}$$

For this reason, the properties are expressed as:

$$Invariance \qquad f(PX, P\mathbf{A}P^T) = f(X, \mathbf{A}) \tag{4.10}$$

$$Equivariance \qquad F(PX, P\mathbf{A}P^T) = PF(X, \mathbf{A}) \tag{4.11}$$

For the second property, locality concepts are exploited. In the context of graphs, which differently from sets present relationships, it is possible to make use of neighborhoods.

Definition 4.7 (Neighborhood function $\mathcal{N}(\cdot)$)**.** A neighborhood function $\mathcal{N} : \mathcal{V} \to Pow^{\mathcal{V}}$ returns all vertices connected to a given input. The image is the power set of the vertex set. By convention, a node u has itself as a neighbor.

$$\mathcal{N}_u = \{u, v_1, \ldots\} : \forall v_i \exists (u, v_i) \in \mathcal{E} \tag{4.12}$$

For a gievn graph $\mathcal{G}$ with features X, the multiset of features of neighbors of a given node $i \in \mathcal{V}$ is denoted as $X_{\mathcal{N}_i}$

A local function operates over the node itself and its neighbors as $\phi(x_i, X_{\mathcal{N}_i})$. Aggregating, a function $F(\cdot, \cdot)$ is of the form:

$$F(X, \mathbf{A}) = \begin{bmatrix} — & \phi(x_1, X_{\mathcal{N}_1}) & — \\ — & \ldots & — \\ — & \phi(x_n, X_{\mathcal{N}_n}) & — \end{bmatrix} \tag{4.13}$$

Again, nodes' features have a custom order in the matrix, and so does the adjacency matrix $\mathbf{A}$, which itself induces an order in $X_{\mathcal{N}_i} \forall i$. Nevertheless, to build an equivariant function, only one condition is needed, as stated below.

Proposition 4.8 (Graph permutation equivariance sufficient conditions)**.** *For $F(\cdot, \cdot)$ to be Σ_n-Equivariant for a graph $\mathcal{G}$ it is sufficient that ϕ is independent of the order of the nodes in $X_{\mathcal{N}_i} \forall i \in \mathcal{V}$.*

Proof. Assume ϕ is independent of order of the nodes in the neighbors multiset. Namely, ϕ is invariant:

$$\phi(x_i, PX_{\mathcal{N}_i}P^T) = \phi(x_i, X_{\mathcal{N}_i}) \forall i \in \mathcal{V}, \forall P \tag{4.14}$$

Then, for a general permutation P on F, the order of the rows will be swapped. At the same time, the order of the multiset for each row will be swapped without impacting ϕ due to its equivariance. Denoting as $p(i)$ the vertex returned from the bijection permutation for a given $i \in \mathcal{V}$:

$$F(PX, P\mathbf{A}P^T) = \begin{bmatrix} — & \phi(x_{p(1)}, PX_{\mathcal{N}_{p(1)}}P^T) & — \\ — & \ldots & — \\ — & \phi(x_{p(n)}, PX_{\mathcal{N}_{p(n)}}P^T) & — \end{bmatrix} \tag{4.15}$$

$$= \begin{bmatrix} — & \phi(x_{p(1)}, X_{\mathcal{N}_{p(1)}}) & — \\ — & \ldots & — \\ — & \phi(x_{p(n)}, X_{\mathcal{N}_{p(n)}}) & — \end{bmatrix} \tag{4.16}$$

$$= P \begin{bmatrix} — & \phi(x_1, X_{\mathcal{N}_1}) & — \\ — & \ldots & — \\ — & \phi(x_n, X_{\mathcal{N}_n}) & — \end{bmatrix} \tag{4.17}$$

$$= PF(X, \mathbf{A}) \tag{4.18}$$

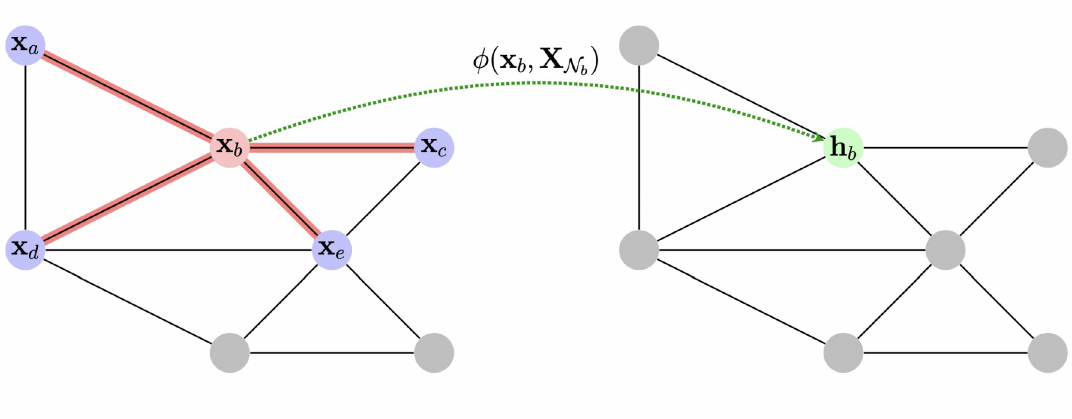


Figure 4.1: Latent node features generation

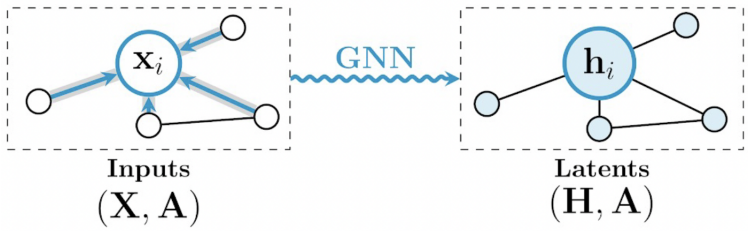


Figure 4.2: GNN equivariant transformation of features for a node i

Where the first equality follows by the notation implemented, the second by assumption, and the third by reordering the vertices at the original setup. □

For a single vertex $i \in \mathcal{V}$, a function $\phi(\cdot, \cdot)$ of its features and those of its neighbors is used to generate its latent form h_i. A visualization of the update for a single returning the latent $h_i = \psi(x_i, X_{\mathcal{N}_i}$ is proposed in Figure 4.1. The resulting transformation is obtained through stacking permutation invariant functions and is permutation equivariant.

4.1.3 The blueprint on graphs

For a general GNN it is common to refer to F as a GNN layer and ϕ as a *local permutation invariant* diffusion / propagation / message passing function. These claims are justified by the previous subsection. Analyzing a series of equivariant layers, a GNN can be summarized as a transformation of features of a graph $(X, \mathbf{A})$ into latent versions $(H, \mathbf{A})$, with the option of a final permutation invariant aggregation. Figure 4.2 again depicts such a scenario. The most explored inference problems include:

- **individual nodes:** output a decision for each node $i \in \mathcal{V}$, Figure 4.3:

$$z_i = f(h_i) \forall i \in \mathcal{V} \tag{4.19}$$

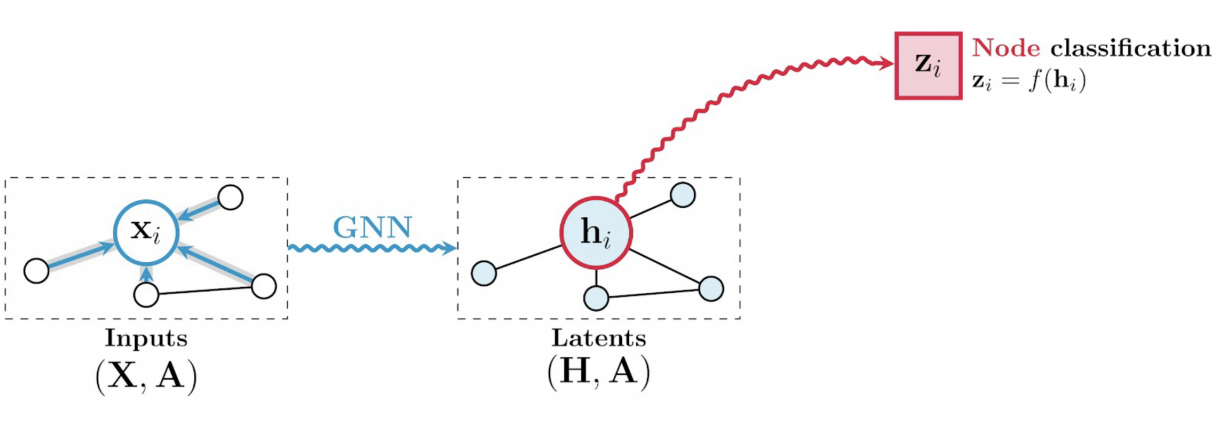


Figure 4.3: GNN node classification

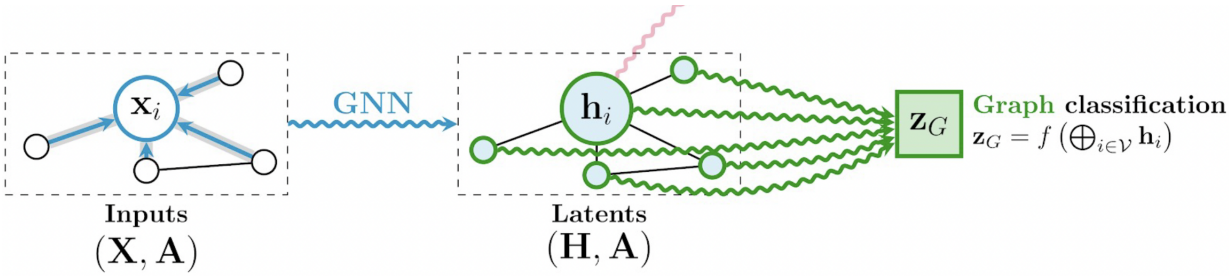


Figure 4.4: GNN graph classification

- **aggregated nodes:** return a permutation invariant result at a graph level, Figure 4.4:

$$z_{\mathcal{G}} = f(\bigoplus_{i \in \mathcal{V}} h_i) \tag{4.20}$$

- **link prediction:** estimate the probability that a connection of vertices i, j exists at all, Figure 4.5

$$z_{ij} = f(h_i, h_j, e_{ij}) \tag{4.21}$$

A very intense area of research is the choice of the $\phi(\cdot, \cdot)$ function. Most of the implementations fall under the umbrella of either of the three methods proposed below.

Definition 4.9 (Convolutional CNN ϕ)**.** For each neighbor j of the vertex $i \in \mathcal{V}$, fix the weight to c_{ij}. Often the degree of j is used: $c_{ij} = \frac{1}{deg(j)}$, or some measure depending on the adjancency matrix $\mathbf{A}$. Determine the latent features of the node as:

$$h_i = \phi\left(x_i, \bigoplus_{j \in \mathcal{N}_i} c_{ij} \psi(x_j)\right) \tag{4.22}$$

Convolutional GNNs are used for *homophilous graphs*[2], where the edge weight is intepreted as the likelihood of the two vertices having the same label. See Figure 4.6 for a graphical instance

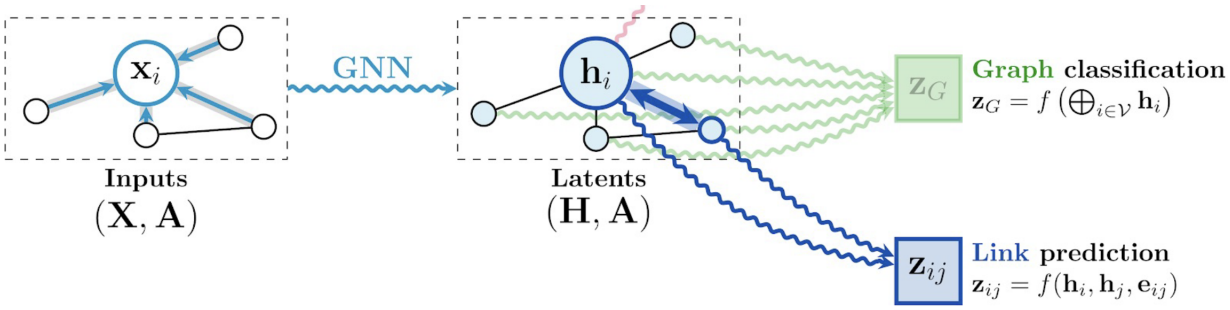


Figure 4.5: GNN link prediction

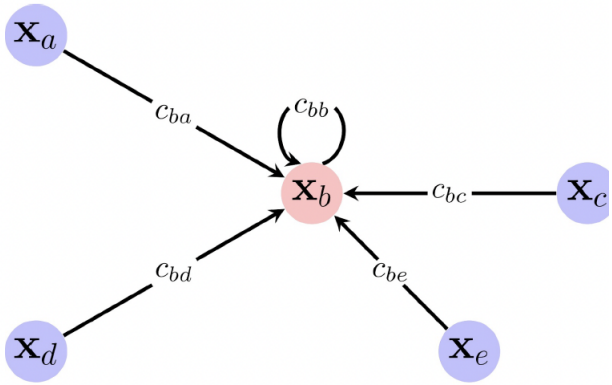


Figure 4.6: Convolutional GNN

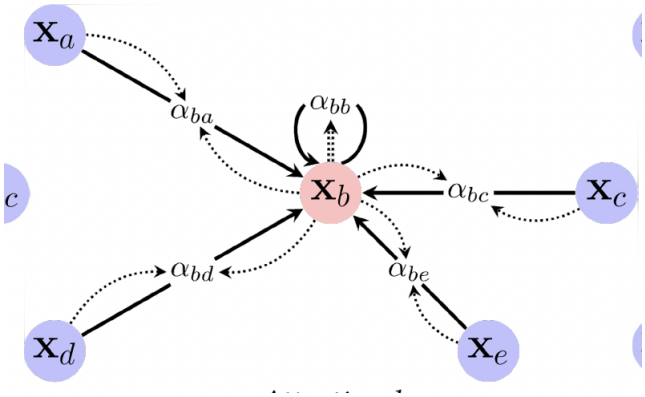


Figure 4.7: Attentional GNN

of the computations. It is rather fast to implement and thus highly scalable.
Computes one scalar for each edge to weight the ψ contribution.
There are many architectures using this technique [DBV17], [KW17], [Wu+19].

While this approach already comes with a wide applicability spectrum, consider the case in which a user retweets 10 tweet of someone else, strongly disagreeing. In this scenario, though the edge connection might be strong, the label of the opinion should be opposite. Nevertheless, a fixed weight does not allow for granularity in the orientation choice. This motivates a slight modification, which moves away from considering the graph topology only.

Definition 4.10 (Attentional GNNs ϕ)**.** Define an implicit weight attention function as $\alpha_{ij} = a(x_i, x_j)$ that changes across different pairs (i, j). Determine latent features of node $i \in \mathcal{V}$ as:

$$h_i = \phi\left(x_i, \bigoplus_{j \in \mathcal{N}_i} \alpha_{ij}\psi(x_j)\right) \tag{4.23}$$

By construction, it does not encode strictly homophilous type connections, as it is computed based on features x_i, x_j. Due to its formulation, it can be interpreted as a fair tradeoff between interpretability, scale and capacity. In Figure 4.7 the reader can find a pictorial representation of the operation. Attention networks are widely implemented in literature [Mon+17], [Vel+18], [BAY22].

While providing more flexibility at a higher computational cost, it is possible to extend further the design to incorporate a broader method.

Definition 4.11 (Message passing ϕ)**.** Compute and aggregate arbitrary vectors (messages) $m_{ij} = \psi(x_i, x_j)$ to be sent across neighboring edges to the candidate node i. The new latent vectors will be quantified as:

$$h_i = \phi\left(x_i, \bigoplus_{j \in \mathcal{N}_i} \psi(x_i, x_j)\right) \tag{4.24}$$

Here, the edges give hints on how to communicate information, where the message function results' m_{ij} decides what data to pass. Figure 4.8 shows a small sized example. Though being

[2] i.e. graphs that present connections when the node features are similar

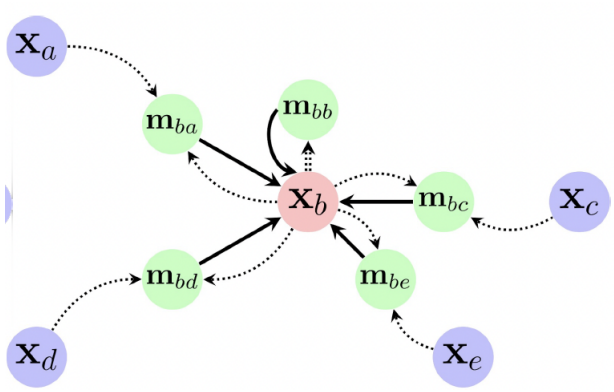


Figure 4.8: Message Passing GNN

the most general form known for a GNN layer level computation, it comes at the cost of high implementation and intepretability costs.
Similarly to the previous methods, many architectures implement message passing [Bat+16], [Gil+17], [Bat+18].

Theorem 4.12 (Generalization power of ϕ specifications)**.** *From Definitions 4.9, 4.10 and 4.11 it is possible to infer that:*

$$Convolutional \subset Attentional \subset Message\ Passing \tag{4.25}$$

Where $a \subset b$ means "a is a special case of b"

Proof. To prove the claims, we go on by showing that the more general representation can be adapted to the less general one.
(Attentional Message Passing) For each vertex $i \in \mathcal{V}$ set $\psi(x_i, x_j)\forall j$ to be such that:

$$\psi(x_i, x_j) = \alpha_{ij}\psi'(x_j) = a(x_i, x_j)\psi'(x_j) \tag{4.26}$$

Where clearly the first inequality is a restriction on the possible function that $\psi(x_i, x_j)$ can represent as from having x_i in the argument only a scalar α_{ij} is the result of its contribution. Notably, x_j makes both a scalar contribution in α_{ij} and a vector contribution in ψ'.
(Convolutional is Attentional) For each vertex $i \in \mathcal{V}$ set $\alpha_{ij} = c_{ij}\forall j$ to be constant and only determined by j, ignoring the features of x_i. This makes the coefficient in front of $\psi'(x_j)\forall j$ constant across different nodes i for the same fixed j.

$$a(x_i, x_j)\psi'(x_j) = c_{ij}\psi'(x_j) \tag{4.27}$$

Thus:

$$Attentional \subset Message\ Passing \land Convolutional \subset Attentional$$
$$\overset{transitivity}{\Longrightarrow} Convolutional \subset Attentional \subset Message\ Passing$$

□

4.2 Graph Neural Networks (GNNs)

The second lecture, concluding the Chapter, will be an overview of:

- Implementing graph GNNs through *GraphNets*
- Latent graph inference in absence of a graph instance
- Power of GNNs analysis through graph isomorphism testing

4.2.1 Maximally Potent GNNs Specifications

Previously, as underlined in Definition 4.5, some potential features of a graph instance $\mathcal{G}$ were omitted. Having observed the *easy* case, we turn to a more general formulation.

Definition 4.13 (General Attributed Graphs)**.** A very wide family of graphs $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ has the following information sources:

- node features $x_u \in \mathbb{R}^k$
- edge features $x_{uv} \in \mathbb{R}^l$
- graph features $x_{\mathcal{G}} \in \mathbb{R}^m$

Analogously, for each layer latents are indexed as $h_u, h_{uv}, h_{\mathcal{G}}$.
While not being as general as possible, the framework is easily extendable to hypergraphs with multiple edges between vertices.

Example 4.14 (General Attributed Graphs in Chemistry)**.** Consider where atoms are vertices and bonds are edges. For an atom u x_u could be the atom type, for a bond uv x_{uv} could be the type of the bond, for graph level properties $x_{\mathcal{G}}$ could store the molecular weight.

Due to its highly adaptive power, *Graph Network*[3] researchers will be used [Bat+18].

Definition 4.15 (Spatial GNN Blueprint)**.** As a dataflow for each layer do the following equivariant operations in order, where as in definition 4.4 the aggregation is invariant:

- Update edge features with graph and relevant nodes information, Figure 4.9

$$h_{uv} = \psi(x_u, x_v, x_{uv}, x_{\mathcal{G}}) \tag{4.28}$$

- Update node features with new edge and graph information, Figure 4.10

$$h_u = \phi\left(x_u, \bigoplus_{v \in \mathcal{N}_u} h_{uv}, x_{\mathcal{G}}\right) \tag{4.29}$$

- Update graph features with new nodes and edges information, Figure 4.11

$$h_{\mathcal{G}} = \rho\left(\bigoplus_{v \in \mathcal{V}} h_v, \bigoplus_{(u,v) \in \mathcal{E}} h_{uv}, x_{\mathcal{G}}\right) \tag{4.30}$$

The just introduced layer formalization is the foundational brick of many architectures.

Example 4.16 (GNN Architecture and blueprint matching)**.** Specify functions ψ, ϕ, ρ to obtain standard GNNs (e.g. GCN, GAT, MPNN), assuming edge and graph features are given. **TODO**

[3] by Deepmind

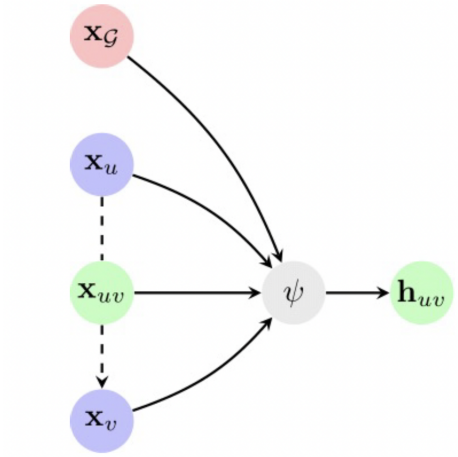


Figure 4.9: Edge update ψ

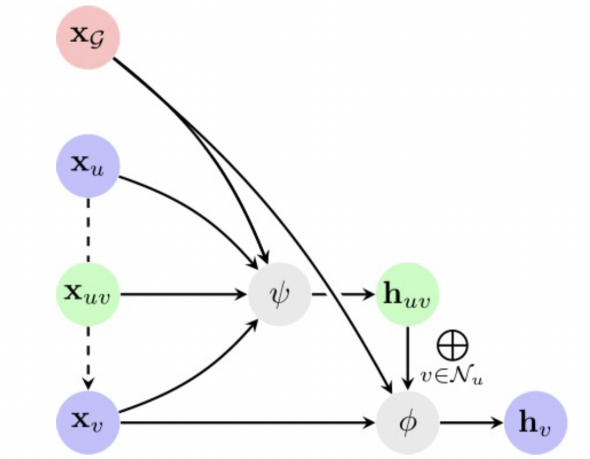


Figure 4.10: Node update ϕ

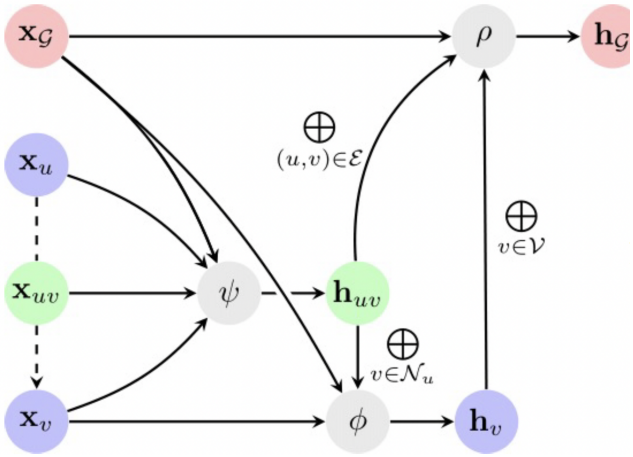


Figure 4.11: Graph update ρ

4.2.2 Latent Graph Inference

Until now, it was assumed that the structure of the problem, represented as a graph $\mathcal{G}$ was given. In science, this is not always the case. The problem of latent graph inference is currently a hot topic in graph representation learning.

To justify its importance, consider a connectivity query task, which is a decision problem returning an answer to the question *is node i connected to node j*. The data structure implemented and graph shape can highly impact performance. For this reason, being able to correctly infer whether edges and nodes exist is a game changer.

The discussion that follows draws from two extreme cases to give an idea of the achievable spectrum of potential. For simplicity, edge and graph features are ignored, only node features are present, and $X \in \mathbb{R}^{|\mathcal{V}|} \times \mathbb{R}^{|\mathcal{V}|}$.

A pessimistic approach could be assuming $\mathcal{G}$ is completely disconnected, making $\mathcal{V}$ a set and $\mathcal{E} = \emptyset$. Then:

$$\mathcal{E} = \emptyset \implies \mathbf{A} = \mathbf{1} \quad \mathcal{N}_u = \{u\} \forall u \in \mathcal{V} \implies \bigoplus_{v \in \mathcal{N}_u} x_v = x_u \tag{4.31}$$

And the node update of Equation 4.29 becomes that of *Deepsets*:

$$h_u = \psi(x_u) \tag{4.32}$$

On the opposite side, assume the graph $\mathcal{G}$ is fully connected. Then:

$$\mathcal{E} = \mathcal{V} \times \mathcal{V} \implies \mathbf{A} = \mathbf{1}\mathbf{1}^T \quad \mathcal{N}_u = \mathcal{V} \forall u \in \mathcal{V} \tag{4.33}$$

Claim 4.17. *Deepsets* $\equiv$ Conv GNNs with no edge information A node update with no edge and no graph information is of the form:

$$h_u = \psi(x_u, \bigoplus_{v \in \mathcal{V}}) \psi(x_v) \; \forall u \in \mathcal{V} \tag{4.34}$$

Where the second argument is equal for all node $u \in \mathcal{V}$, and thus non influent in the establishing differences across vertices. Clearly, *Deepsets* and Convolutional Neural Networks cannot be distinguished. **TODO**

Claim 4.17 suggests enriching latent updates as to include features from connected nodes $\mathcal{N}_u = \mathcal{V}$.

$$h_u = \phi\left(x_u, \bigoplus_{v \in \mathcal{V}} a(x_u, x_v)\psi(x_v)\right) \tag{4.35}$$

Other than being the fully connected case of the Attentional flavour proposed in Definition 4.10, Equation 4.35 is a common Transformer.

Very popular in Natural Language Processing, a Transformer infers sequential structural information by adding positional embdeddings.

In this intepretation, Attentional GNNs could be seen as a method to infer **soft** adjacency matrices. There are good references to expand the topic [Jos20].

Definition 4.18 (Latent Graph Inference Problem Setup)**.** For a latent graph, X the feature matrix is given, while the adjacency matrix $\mathbf{A}$ is unknown.

Observation 4.19 (The truth is in between: Latent Graph Inference)**.** *The disconnected case loses lots of potential information. On the other hand, the fully connected case may be redundant and computationally expensive for large graphs. The problem of inferring* $\boldsymbol{A}$ *is that inclusion/exclusion choices for edges are discrete*[4]*, non differentiable, and thus hard to backpropagate.*

For the purpose of having a glimpse of available options, 4 common methods will be quickly outlined below.

Variational [Kip+18]. Assume a prior distribution over edges is given $p(a_{uv} = 1)$. Of course, the choice determines how potentially connected or sparse the hypothesis is.

- Given a fully connected graph with node features x_u run a GNN on it, obtaining latent edge features estimates h_{uv}.
- Obtain a posterior q for the adjacency matrix as:

$$q(a_{uv}|X) = \sigma(\psi(h_{uv})) \tag{4.36}$$

  Where σ is an activation function such as *sigmoid* or *softmax*.
- Sample new adjacency estimates from q as $a_{uv} \sim q(a_u v|X)$.
- Use the new edges in a second Graph Network for the desired final output.

These steps are one forward run. To backpropagate and train the network the Gumbel trick from Variational Auto Encoders is used (quick tutorial [Ada13]).

Despite the nice bayesian probabilistic framework, the complexity of running over a fully connected graph is still present. Other options allow to infer a sparse graph without ever running on a dense GNN.

k-NN graphs Assume the graph is a k-NN graph, where each node has features h_u, and is connected to the k nearest in h nodes, according to some notion of distance/similarity.

- *Dynamic Graph CNN* [Wan+19] is an easy non parametric method. It can be summarized as follows:
  - Use as similarity the inner product $h_u^T h_v$
  - for every layer and every node evaluate the k nearest neighbors in h $\mathcal{N}_u = \underset{v \in \mathcal{V}}{topk}\{h_u^T h_v\}$
  - in the following layer use $\mathcal{N}_u$ as edges.
- *Differentiable Graph Module* [Kaz+22] is a technique from reinforcement learning. A snapshot of the pipeline is shown in Figure 4.12. The key aspects are:
  - Use as similarity the inner product $h_u^T h_v$
  - at every layer, compute the probabilities of the transformed node features

$$p(a_{uv} = 1) \propto \sigma(\psi(x_u)^T \phi(x_v))$$

[4] either in or out

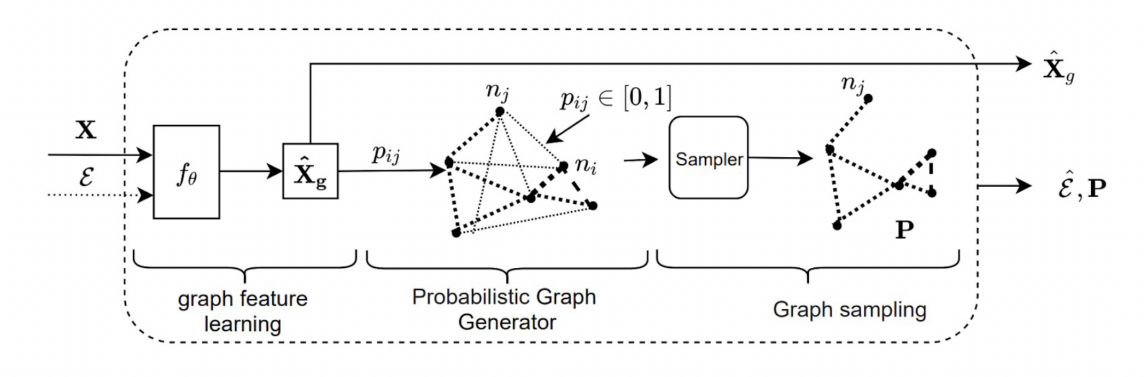


Figure 4.12: *Differentiable Graph Module layer*

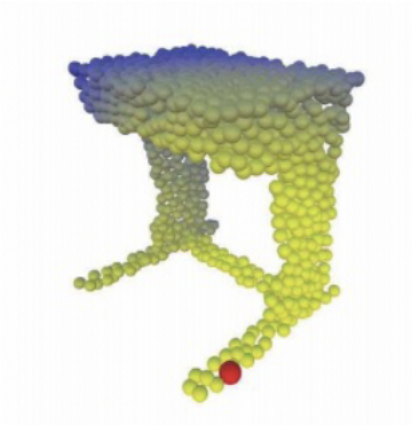


Figure 4.13: Bad leg layout semantics

- as policy implement an RL agent that chooses k neighbors for every node.
- use as reward downstream performance[5]
- optimize via policy gradient methods[6]

Example 4.20 (*Dynamic Graph CNN in action*)**.** Consider the task of recognizing parts of an table from a dataset of point cloud objects. Using a purely euclidean distance alone, the result is a smooth diffusion of latents. A point in a leg is not well linked with any point in the other leg, as in Figure 4.13. On the contrary, a local GNN with the above implementation allows to have a better result in terms of feature similarity. The outcome is shown in Figure 4.14. Observe also that the legs are **symmetric**!

A final option is *Pointer Graph Networks*, a supervised method which makes use of assumed ground truth edges [Vel+20].

4.2.3 GNN Power Assessment through Graph Isomorphism Tests

There is extensive knowledge about cases in which GNNs tend to fail. One such example is deciding whether two graphs are isomorphic.

[5]for example, an accuracy measure
[6]for example: REINFORCE

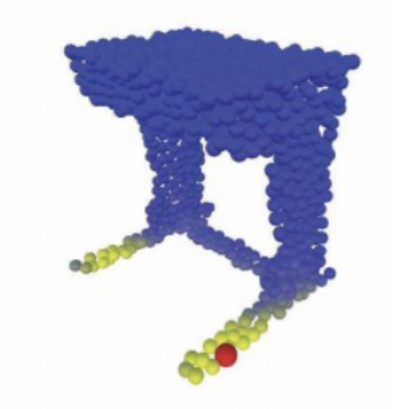


Figure 4.14: Good leg layout semantics

Definition 4.21 (Graph Isomorphism)**.** Consider two graphs $\mathcal{G} = (\mathcal{V}, \mathcal{E}), \mathcal{G}' = (\mathcal{V}', \mathcal{E}')$. Two graphs are isomorphic when there is a bijection between the two vertex sets that preserves connections across the two.

$$\exists f : \mathcal{V} \to \mathcal{V}' \; bijective \mid (u, v) \in \mathcal{E} \iff (f(u), f(v)) \in \mathcal{E}' \tag{4.37}$$

A graph isomorphism is denoted as $\mathcal{G} \cong \mathcal{G}'$.
Consequently, the graph isomorphism problem is a decision problem asking whether two graphs are isomorphic to each other.

In particular, e key aspect of a well working GNN is distinguishing two non isomorphic graphs in the features, namely:

$$\mathcal{G}_1 \ncong \mathcal{G}_2 \implies h_{\mathcal{G}_1} \neq h_{\mathcal{G}_2} \tag{4.38}$$

An easy and effective method for isomorphisms tests will be used as a benchmark for assessing a GNN in this task. Formulated by Weisfeiler and Lehman in 1968, it consists in passing hashes **TODO define what is a hash in this case**of sums along the edges until convergence. At the end, two graphs are *possibly* isomorphic if the color histograms do not change. Algorithm 4.2.3 shows the steps in order.

Algorithm 1 Weisfeiler & Lehman Test (1-WL)

Input: initial node coloring $(h_1^{(0)}, \dots h_n^{(0)})$
Output: final node coloring $(h_1^{(T)}, \dots h_n^{(T)})$

$t \leftarrow 0$
repeat
  for $v_i \in \mathcal{V}$ **do**
    $h_i^{(t+1)} \leftarrow hash\left(\sum_{j \in \mathcal{N}_i} h_j^{(t)} \right)$
    $t \leftarrow t + 1$
  end for
until $h^{(t)}$ stable
return $h^{(T)}$ final coloring

Figure 4.15: Example graph

Figure 4.16: First iteration

Example 4.22 (1-WL in action)**.** Consider the graph in Figure 4.15, and use Algorithm 4.2.3. Iterating, the steps can be explained as:

1. Each node has two or three connections, two colors are assigned (Figure 4.16)
2. Green nodes have either one or two neighbors, assign them two different colors. Yellow nodes have the same number of neighbors, all of them are colored as red (Figure 4.17)
3. the coloring is stable

Applying the same procedure to another graph, isomorphic to the former, ti can be noticed that the frequence of colors (labels) is the same (Figure 4.18).

A couple worthy remarks are:

1. Untrained GNNs are basically random hashes
2. the 1-WL test can fail at times
3. over discrete features, GNNs can at best be as powerful as the 1-WL test.
4. GNNs can be strengthened by analyzing failure cases of Algorithm 4.2.3 by modifying
   - features
   - message passing technique
   - graph structure

Point 4 is an active research area. Any modification with improved performance leads to *higher-order* GNNs. A very easy example of feature fix is inspired by the failure of distinguishing a 6 cycle like Figure 4.19 from a double 3 cycle like Figure 4.20. The issue lies in the hop vision of both 1-WL and GNNs, that view each node as always being connected to two nodes (Figure 4.21.

A solution cleverly assigns a color (a randomized feature) to each node [SYK21], to count how many hops are needed to encounter it again (Figure 4.22)

Continuous features present more difficulties. Some instances can be distinguished through one (e.g. avg) and some through another (e.g. min). The publication that presents *Principal*

Figure 4.17: Stable configuration

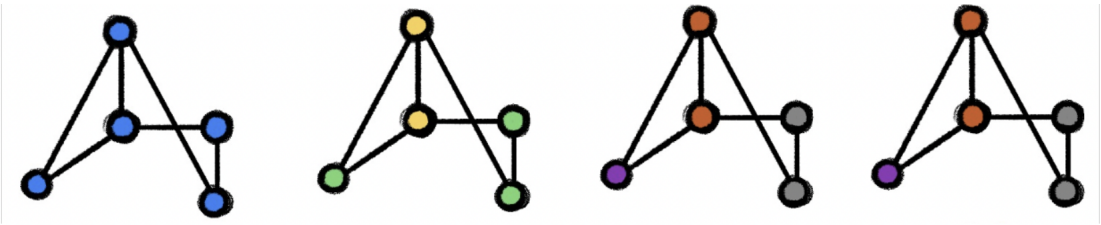


Figure 4.18: Isomorphic Graph (1-WL)

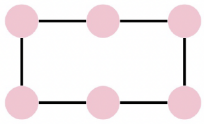


Figure 4.19: 6 cycle

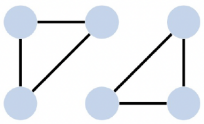


Figure 4.20: 3 cycle

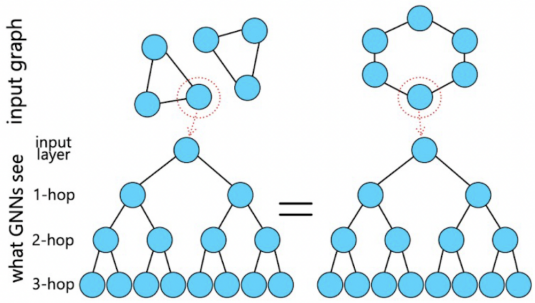


Figure 4.21: Rollout of exploration steps

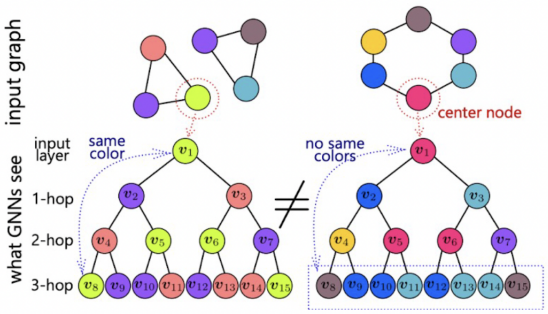


Figure 4.22: Coloured vertices exploration steps

Neighborhood Aggregation contains a formal analysis of this problem for continuous uncountable spaces [Cor+20].

In the next Chapters, stronger assumptions on the structure Ω will be enforced, leading to grids and groups. Additionally, remarks about spectral GNNs and the graph Fourier transform will be touched.

Chapter 5

Grids

TODO 4.2, 5.1

In the previous Chapter, it was shown that:

- the GDL blueprint can be formalized for sets and graphs
- as the metric and topologic structure change, a rich variety emerges

In this Chapter, we will see:

- the mathematical structure of a grid is very powerful
- the spectral and multiscale formulations have some implementation details and innate limitations
- the GDL blueprint arises even in this case

Contents will be carried out as follows:

1. the translation group and its link with grids will be presented, along with the Fourier transform as the natural tool to implement it
2. limitations that suggest using wavelet transforms instead will be covered
3. to conclude, a gentle ride towards CNNs will be outlined

Due to the very heavy mathematical background of Fourier and Wavelet transforms, some arguments may sound too informal, but hopefully they will allow to give an idea of what happens.

5.1 Translation Group and Fourier Transform

Consider a Grid as structure Ω. There can be many such types, with different forms. Some examples are shown below.

Example 5.1 (Simple Grid structures)**.**

- A one dimensional grid $\Omega = \{1, \dots, d\}$
- A two dimensional grid $\Omega = \{(i, j) : i, j = 1, \dots, d\}$

Remark (A disclaimer on dimension)**.** When referring to a $s-d$ dimensional Grid, it is intended that the size of the underlying set is d and not the dimension of the grid, which is denoted as s. In the above example, d is the size of the side of the hypercube, while $s = 2$ as the hypercube is a square. and we have a $2 - d$ grid.

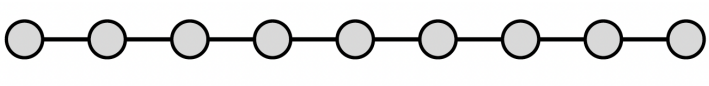


Figure 5.1: $1-d$ grid

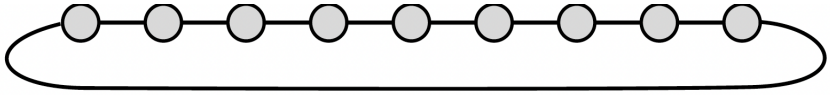


Figure 5.2: $1-d$ ring graph

In the general case, what could be the group transformation $\mathfrak{g} : \Omega \to \Omega$? Intuitively, it would be a **translation**, which consists in moving one point in the space. Assuming one direction of translation, a problem arises at the boundary, as there is no explicit move to make. To deal with this structural flaw, a further assumption needs to be stated.

Assumption 5.2 (Periodic boundary condition holds)**.** Given a grid Ω, treat it as a ring graph, where the endpoints are linked by an additional edge. A graphical representation of a Grid is in Figure 5.1, while its ring graph custom form is found in Figure 5.2

With this setting, we further consider the easy case of a $1-d$ grid with $1-d$ translations. Such instances suggest the following translation description. Define a (bijective) map:

$$\mathfrak{g} : \Omega \to \Omega \tag{5.1}$$

$$u \to (u+1)(mod\, d) \tag{5.2}$$

Which is the generator of the cyclic group $\mathfrak{G} = \mathbb{Z}_d = \Big(\{1, \ldots, d\}, +(mod\ d)\Big)$ paired with addition modulo d, coming with nice properties such as:

- inverse existance

$$\mathfrak{g}^{-1} : \Omega \to \Omega$$
$$u \to (u-1)(mod\, d)$$

- composition as sum

$$\mathfrak{g}^{k} : \Omega \to \Omega$$
$$u \to (u+k)(mod\, d)$$

The easiest case can be quickly generalized to any structure Ω.

Definition 5.3 (Grid flavours and Associated translation groups)**.** We recognize two approaches:

- The s-dimensional grid:
  For an $s-d$ grid of the form $\Omega = \{1, \ldots, d\}^{\otimes s}$ define the translation group as: $\mathfrak{G} = \bigtimes_{i=1}^{s} \mathbb{Z}_d^{(i)}$
- The continuous domain[1]:
  wrap the grid around a Torus such that $\Omega = \mathbf{T} = S^1 \times S^1$ and $\mathfrak{G} = S^1 \times S^1$. For a graphical idea, see Figure 5.3

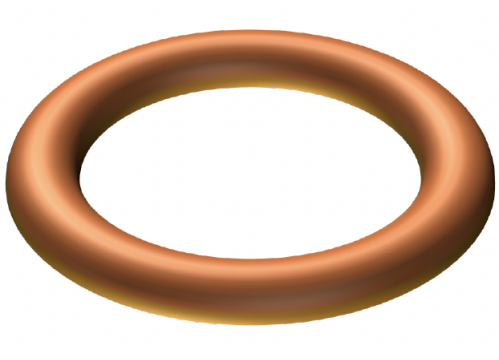


Figure 5.3: Torus

For the s-dimensional grid, one must be careful. A very important result is that the cartesian product of cyclic groups is cyclic if and only if $\{s_i\}$ are *coprime*[2]. For s dimensions of $\mathbb{Z}_d$ this is **not** the case. What must be done in terms of maps is defining them separately for each of the cyclic groups and using the composition. More specifically, there will be s generators.

An example for $s = 2$ and arbitrary d with two generators can be found below.

Example 5.4 ($s-d$ Grid bijective maps)**.** Consider Ω to be a $2-d$ grid. Then $\mathfrak{G} = \mathbb{Z}_d \times \mathbb{Z}_d$, and the map $\mathfrak{g}$ will be:

$$\mathfrak{g} = \mathfrak{h} \circ \mathfrak{l} \quad where \quad \begin{cases} \mathfrak{h}(u) = (u+1)(mod\, d) \\ \mathfrak{l}(v) = (v+1)(mod\, d) \end{cases} \tag{5.3}$$

Observation 5.5 (Structure and Group)**.** *Observe that for a grid and its symmetry group* $\Omega \cong \mathfrak{G}$ *but this is not always the case. In the future, more discussion will be added.*

Having in hand a group symmetry, recall that on the signal space $\mathfrak{g}$ is *lifted* to linear transformations in the form of a group representation matrix $\rho(\mathfrak{g}) \forall \mathfrak{g} \in \mathfrak{G}$. For the sake of simplicity, we will again consider the $1-d$ case and focus on the main ideas.

When the map is lifted to the space of signals we have that $\mathcal{X}(\Omega, \mathcal{C}) = \{x : \Omega \to \mathbb{R}\} \cong \mathbb{R}^d$ and:

$$\mathfrak{g} : \mathcal{X}(\Omega, \mathbb{R}) \to \mathcal{X}(\Omega, \mathbb{R}) \tag{5.4}$$

$$x \to \mathfrak{g}.x : \Omega \to \mathbb{R} \tag{5.5}$$

Whereas in terms of structure:

$$x \to \mathfrak{g}.x : \Omega \to \mathbb{R} \tag{5.6}$$

$$u \to x(\mathfrak{g}^{-1}u) \tag{5.7}$$

With this characterization, the aim is to find a set of matrices that could represent inequivocably each element of $\mathfrak{G}$. The following Proposition is the desired result.

[1] informal **TODO**

[2] Chinese remainder theorem, namely if the greatest common divisor is 1.

Definition 5.6 (Shift Operator S)**.** For $d \in \mathbb{Z}_d$ let S be a square matrix of dimension d representing the 1 position shift for each of the functions operated by it.

$$S = \begin{bmatrix} 0 & 1 & 0 & \dots & \dots & \dots & 0 \\ 0 & 0 & 1 & 0 & \dots & \dots & 0 \\ 0 & 0 & 0 & 1 & 0 & \dots, & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & 0 & \dots & \dots & \dots & \dots & 0 \end{bmatrix} \tag{5.8}$$

Which is a matrix with zero entries apart from the those to the right of the diagonal entries and the low left box S_{d0}.

Observation 5.7 (What does S represent)**.** *For a grid Ω, S can be seen as the adjacency matrix of the directed ring graph, where the direction is defined by the transformation convention (e.g. to the right).*

Proposition 5.8 (Shift Operator represents Groups on $\mathcal{X}(\Omega, \mathcal{C})$)**.** *We claim that the shift operator is such that there exists a power $S^k : k \in \mathbb{N}$ such that it represents any translation.*

$$\exists k \in \mathbb{N} \mid \mathfrak{h}.x = S^k x \forall \mathfrak{h} \in \mathfrak{G} \tag{5.9}$$

Proof. Consider the Shift Operator S of Definition 5.6, clearly it is equivalent to translating to the right the points in a $1-d$ grid. As argued previously, the symmetry group is $\mathfrak{G} = \mathbb{Z}_d$ and the group element (the map) corresponding to it is:

$$\mathfrak{g} : \Omega \to \Omega$$
$$u \to (u+1)(mod\, d)$$

By the definition of cyclic group, there is a generator[3] $\langle \mathfrak{g} \rangle$. Clearly, powers of $\mathfrak{g}$ are powers of S. Thus, $S^k x, k \in \mathbb{N}$ cyclically recover all the elements in Ω, and the claim is justified. The pairing between S and $\mathfrak{g}$ is unique, and also the compositions, by simple induction. □

Proposition 5.9 (Cyclic Group is Abelian)**.** *The group of the one dimensional grid is Abelian. Namely:*

$$\mathfrak{g} \circ \mathfrak{h} = \mathfrak{h} \circ \mathfrak{g} \forall \mathfrak{g}, \mathfrak{h} \in \mathfrak{G} \tag{5.10}$$

Proof. Even though there are other ways to prove it, we follow the exercise assigned by the lecturer, and exploit Proposition 5.8.
By the matrix representation of group actions $\mathfrak{g}, \mathfrak{h} \in \mathfrak{G}$, assume that they are such that:

$$\mathfrak{g}.x = S^k x \quad \wedge \quad \mathfrak{h}.x = S^q x \quad k, q \in \mathbb{N} \tag{5.11}$$

[3] An element such that its powers are *all* the elements in the group, often denoted as $\langle \mathfrak{g} \rangle$.

Composing the operation one gets:

$$\begin{aligned}(\mathfrak{h} \circ \mathfrak{g}).x &= \mathfrak{h}.(\mathfrak{g}.x) \\ &= \mathfrak{h}.(S^k x) \\ &= S^q S^k x \\ &= S^{q+k} x \\ &= S^k S^q x = \mathfrak{g}.(S^q x) = \mathfrak{g}.(\mathfrak{h}.x) = (\mathfrak{g} \circ \mathfrak{h}).x\end{aligned}$$

And the group is commutative (Abelian). □

To remark even further the power of the shift operator, we report here a very important result.

Definition 5.10 (Normal matrices)**.** Denoting as † the complex conjugate of a matrix $A \in \mathbb{R}^{d \times d}$ is normal if it commutes with its conjugate transpose[4]:

$$AA^\dagger = A^\dagger A \tag{5.12}$$

Theorem 5.11 (Shift operator is diagonalizable in complex space)**.** *Let S be the shift operator, then it is:*

1. *normal*
2. *diagonalizable in the complex space*

Proof. **(Claim 1)** Consider S as in Definition 5.6. S represents a shift to the (conventional) right with boundary conditions of a ring graph. The shift to the left happens to be exactly $S^\dagger$, where the indices are inverted, shifting all points to the (conventional left). For any configuration x, a shift to the left and to the right is equivalent to a null move. The same is true for the opposite order of shifts. Then:

$$SS^\dagger = I = S^\dagger S \tag{5.13}$$

This fact emerges also from a combination of Propositions 5.8 and 5.9.
(Claim 2) We prove *S normal $\iff$ S unitary diagonalizable*
($\implies$ direction) Assume S is normal, use the Schur decomposition $S = UTU^\dagger$ where U is unitary and T is upper triangular. By the normality of S it holds that:

$$SS^\dagger = UTU^\dagger UT^\dagger U = UTT^\dagger U = S^\dagger S = UT^\dagger TU^\dagger \implies TT^\dagger = T^\dagger T = UD^\dagger DU^\dagger = S^\dagger S$$

And T is normal as well. By T being upper triangular and $T^\dagger$ being lower triangular, commutativity holds $\iff$ T diagonal, and S is unitary diagonalizable.
($\impliedby$ direction) trivial, as:

$$SS^\dagger = UDU^\dagger UDU^\dagger = UDD^\dagger U^\dagger$$

Where we only use the fact that $DD^\dagger = D^\dagger D$ for a diagonal matrix.
Thus by S normal then it is unitarily diagonalizable. □

Observation 5.12 (On Theorem 5.11)**.** *It is clear that with such a diagonalization there will be an eigenvalue eigenvector decomposition of the shift operator S. With some work, it is possible to show that the eigenvalues of the shift operator are complex. This result will be shown later when needed.*

[4]Note taht this does not necessarily mean it is Hermitian i.e. $A = A^\dagger$.

Remark (Extendability to higher dimensions)**.** For $s-d$ grids the same reasonings can be made. The key point is noticing that each of the s *directions* of the grid define independent maps that compose together. From a physical perspective, the generators could be seen as unitary vectors. In the higher dimensional case, the matrix will be a tensor **TODO check**.

Sticking to the $1-d$ case, the GDL blueprint of Definition 3.43 suggests that it is sufficient to manage *local linear invariants and equivariants.* For this purpose, an interpretation in the grids context will be given.

Theorem 5.13 (Linear Invariants for Grids, Properties)**.** *If $\Omega = \{1, \dots, d\}$ then a linear invariant map $\mathcal{J} : \mathcal{X}(\Omega, \mathcal{C}) \to \mathbb{R}$ satisfies:*

1. *$\mathcal{J}(x) = \langle x, v \rangle$ for some $v \in \mathbb{R}^d$*
2. *$Sv = v$ where S is the shift operator*
3. *$v =$ **1 TODO check** does not have norm 1*

Proof. **(Claim 1)** The signals space is $\mathcal{X}(\Omega, \mathcal{C}) = \mathcal{X}(\Omega, \mathbb{R}) = \{x : \Omega \to \mathbb{R}\} \cong \mathbb{R}^d$ when we are in the $1-d$ case. Thus, $\mathcal{J} : \mathbb{R}^d \to \mathbb{R}$ and for the map to be linear it must be an dot product of the form:

$$\mathcal{J}(x) = \langle x, v \rangle \quad for\ some\ v \in \mathbb{R}^d$$

Where the vector v is applied to any $x \in \mathbb{R}^d$. To impose invariance, as per Definition 3.28 it must be the case that:

$$\mathcal{J}(\mathfrak{g}.x) = \mathcal{J}(x) \forall x \in \mathbb{R}^d, \forall \mathfrak{g} \in \mathfrak{G}$$

(Claim 2) By the invariance and basic operations on the dot product:

$$\begin{aligned}
& \mathcal{J}(x) = \mathcal{J}(\mathfrak{g}.x) && \text{invariance condition} \\
\implies & \langle x, v \rangle = \langle \mathfrak{g}.x, v \rangle && \text{invariance dot product form} \\
& = \langle S^k x, v \rangle && \text{by Proposition 5.8} \\
& = \langle x, (S^k)^\dagger v \rangle && \text{Where } \dagger \text{ is the complex conjugate}
\end{aligned}$$

Observe that the operator S^k has an inverse operator $(S^k)^\dagger \forall k \in \mathbb{N}$. This holds by Theorem 5.11.
Thus, we have that taking the second arguments in the dot product at the last two equalities:

$$\begin{aligned}
& v = (S^k)^\dagger v && \\
\implies & S^k v = S^k (S^k)^\dagger v && \text{left multiplication} \\
& = S \cdot \ldots \cdot S S^\dagger \cdot \ldots \cdot S^\dagger v && \text{expand product } k \text{ times for both} \\
& = v \forall k \in \mathbb{N} && \text{as } S S^\dagger = I
\end{aligned}$$

From this fact, since it holds $\forall k$, it also holds for $k = 1$. Notice that given the nature of the additive translation group we could have started from any translation to get all the others.
(Claim 3) TODO □

Having established the requirements for a grid, it is worth noticing that the vector v for the linear invariant map is an eigenvector of the shift operator. Below, the same is done for equivariant maps.

Theorem 5.14 (Linear Equivariants for Grids, Properties)**.** *If $\Omega = \{1, \dots, d\}$ then a linear equivariant map $\mathcal{L} : \mathcal{X}(\Omega, \mathcal{C}) \to \mathcal{X}(\Omega, \mathcal{C}')$ satisfies:*

1. *$\mathcal{L}(x) = Cx$ for some $C \in \mathbb{R}^{d \times d}$*
2. *where C commutes with the shift operator S, namely $C : CS = SC$*

Proof. **(Claim 1)** The signals space is $\mathcal{X}(\Omega, \mathcal{C}) = \mathcal{X}(\Omega, \mathbb{R}) = \{x : \Omega \to \mathbb{R}\} \cong \mathbb{R}^d$ when we are in the $1-d$ case. Thus, $\mathcal{L} : \mathbb{R}^d \to \mathbb{R}^d$ and for the map to be linear it must be a matrix product of the form:

$$\mathcal{L}(x) = Cx \quad for\ some\ C \in \mathbb{R}^{d \times d} \tag{5.14}$$

Where the matrix C is applied to any $x \in \mathbb{R}^d$. To impose equivariance, as per Definition 3.29 it must be the case that:

$$\mathcal{L}(\mathfrak{g}.x) = \mathfrak{g}.\mathcal{L}(x) \forall x \in \mathbb{R}^d, \forall \mathfrak{g} \in \mathfrak{G}$$

(Claim 2) By the equivariance requirement and basic operations on the matrix product:

$$\begin{aligned}
&\mathcal{L}(\mathfrak{g}.x) = \mathfrak{g}.\mathcal{L}(x) && \text{equivariance condition} \\
\implies &C(\mathfrak{g}.x) = \mathfrak{g}.(Cx) && \text{equivariance matrix product form} \\
\iff &CS^k x = S^k C x && \text{by Proposition 5.8} \\
\iff &CS^k = S^k C && \text{which holds } \forall k \in \mathbb{N}
\end{aligned}$$

By similar arguments, we take $k = 1$ and conclude that:

$$CS = SC$$

Which is an equivalent condition as $\forall k$. □

Coming back to the observation on the shift operator done in Theorem 5.11 it is possible to characterize even further the equivariant map represented by C. In particular, the eigenvalues of S will be complex. We have that:

Theorem 5.15 (Eigendecomposition of Shift Operator)**.** *For $S \in \mathbb{R}^{d \times d}$ as in Definition 5.6 the eigenvectors v_k and eigenvalues λ_k where $k \in \{1, \dots, d\}$ are:*

$$v_k = \frac{1}{\sqrt{d}} \begin{bmatrix} 1 \\ e^{i2\pi \frac{k}{d}} \\ e^{i2\pi \frac{2k}{d}} \\ \dots \\ e^{i2\pi \frac{k(d-1)}{d}} \end{bmatrix} \in \mathbb{C}^d \tag{5.15}$$

$$\lambda_k = e^{i2\pi \frac{k}{d}} \tag{5.16}$$

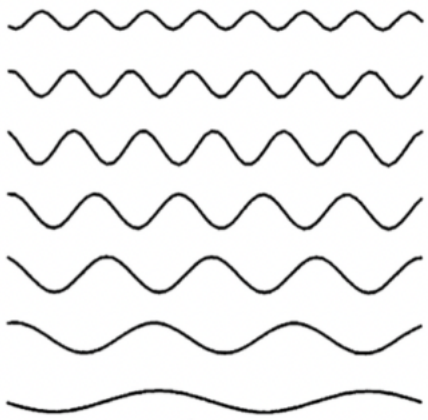


Figure 5.4: Some sinusoidal functions

Proof. We only check that the values claimed are correct. Consider $\{v_k, \lambda_k\}_{k=1}^{d}$. Let s_j. be the j^{th} row of S. In practice, s_j maps an element of a vector to its neighbor to the right. Then:

$$\begin{aligned}
\langle v_k, s_j \rangle &= v_k[j+1] && \text{as } s_j \text{ shifts } v_k \text{ in its } j+1 \text{ index} \\
&= \frac{1}{\sqrt{d}} e^{i2\pi \frac{k(j+1)}{d}} && \text{by definition of } v_k[j+1] \\
&= e^{i2\pi \frac{k}{d}} \frac{1}{\sqrt{d}} e^{i2\pi \frac{kj}{d}} && \\
&= e^{i2\pi \frac{k}{d}} v_k[j] && \text{by definition of } v_k[j]
\end{aligned}$$

Stacking the rows $j = 1, \ldots, d$ with a vector v_k we get that:

$$Sv_k = \begin{bmatrix} - & s_1 v_k & - \\ - & s_2 v_k & - \\ - & \cdots & - \\ - & s_d v_k & - \end{bmatrix} = \begin{bmatrix} - & e^{i2\pi \frac{k}{d}} v_k[1] & - \\ - & e^{i2\pi \frac{k}{d}} v_k[2] & - \\ - & \cdots & - \\ - & e^{i2\pi \frac{k}{d}} v_k[d] & - \end{bmatrix} = e^{i2\pi \frac{k}{d}} v_k = \lambda_k v_k \tag{5.17}$$

And the eigendecomposition is of the claimed form. □

Going further into this topic, it is possible to argue that these are deeply related to the Fourier transform, which will be *informally* discussed.

Definition 5.16 (Discrete Fourier Transformed (DFT) vector $\widehat{x}$)**.** For a vector x and a stacked matrix of eigenvectors of S denoted as $V = \begin{bmatrix} v_1, \ldots, v_d \end{bmatrix}^T$, we define its Fourier transformed version $\widehat{x}$ as:

$$\widehat{x} = Vx \quad where \quad \widehat{x}[k] = \langle x, v_k \rangle = \frac{1}{\sqrt{d}} \sum_{j=0}^{d-1} x_j e^{-i2\pi \frac{kj}{d}} \tag{5.18}$$

Which can be seen as a decomposition of x into a sum of sinusoidal functions (see Figure 5.4 for an idea of such functions).

Parceval's identity establishes a similarity of original and spectral domain, while the second result is useful to revert the transformation.

Theorem 5.17 (Parceval's Identity)**.** *For vectors $x, y \in \mathbb{R}^d$ the Fourier transform is an isometry*[5]*.*

$$\langle \widehat{x}, \widehat{y} \rangle = \langle x, y \rangle \forall x, y \in \mathbb{R}^d \tag{5.19}$$

Proof. Step by step:

$$\begin{aligned}
\langle \widehat{x}, \widehat{y} \rangle &= \langle Vx, Vy \rangle && \text{by Definition of DFT} \\
&= (Vy)^\dagger (Vx) && \text{by definition of dot product} \\
&= y^\dagger V^\dagger V x && \text{by Theorem 5.11 (see comment below)} \\
&= y^\dagger x = \langle x, y \rangle && \text{again by Definition of dot product}
\end{aligned}$$

Where the implication from Theorem 5.11 ensures that V is unitary, so that: $V^\dagger V = I$. □

Proposition 5.18 (Inverse Discrete Fourier Transform (IDFT))**.** *For a DFT where $\widehat{x} = Vx$ and $x \in \mathbb{R}^d$ we have that the inverse transformation is:*

$$x_j = \frac{1}{\sqrt{d}} \sum_{k=1}^{d} \widehat{x}[k] e^{i2\pi \frac{kj}{d}} \tag{5.20}$$

Proof. Again by Theorem 5.11 if $\widehat{x} = Vx$ then V is orthonormal with norm 1 and:

$$\widehat{x} = Vx \implies V^{-1}\widehat{x} = V^\dagger \widehat{x} = x \implies x_j = V^\dagger_{j\bullet} \widehat{x} \tag{5.21}$$

Where $V^\dagger_{j\bullet}$ denotes the j^{th} row of the conjugate transpose of V. By simply conjugate transposing the matrix V one gets that: **TODO check coefficient in front and general reasoning**

$$V^\dagger_{j\bullet} = V_{\bullet j} = \frac{1}{\sqrt{d}} \begin{bmatrix} 1 & e^{i2\pi \frac{kj}{d}} & e^{i2\pi \frac{(k+1)j}{d}} & \dots & e^{i2\pi \frac{(k+d)j}{d}} \end{bmatrix}$$

And the claim is satisfied □

Such a framework extends naturally to the continuous domain T of the torus we saw earlier through the concept of windowed Fourier transform, and even in unbounded/multidimensional regions through the general Fourier transform.

The application areas flourished thanks to the Fast Fourier Transform FFT Algorithm and include:

- Signal Processing
- Number Theory
- Partial Differential Equations
- Mathematical Physics
- Geometry
- Graph Theory

[5] i.e. dot products are preserved in the transformed version of the vectors.

TODO maybe Fast DFT algorithm outline

Presenting the DFT and highlighting its properties is useful for our task of understanding how the equivariant C is. Only the final steps are missing:

Lemma 5.19 (Commutative diagonalizable matrices and eigenvectors)**.** *Two diagonalizable matrices commute if and only if they share the same eigenvectors.*

$$AB = BA \iff \forall (x, \lambda) \in Eig(A) \;\; Bx = \lambda' x \;\&\; opposite \tag{5.22}$$

Proof. **($\implies$ direction)** Consider A and one of its eigenpairs:

$$\begin{aligned} A(Bx) &= B(Ax) && \text{commutativity} && (5.23)\\ &= B(\lambda x) && \text{eigenpair of } A && (5.24)\\ &= \lambda(Bx) && \text{reordering} && (5.25)\\ \iff & (Bx, \lambda) \in Eig(A) && && (5.26) \end{aligned}$$

If we let $Bx \neq 0 \iff B \neq \mathbf{0}$ which is the trivial case, one can assert that the vectors Bx and x share the same eigenvalue λ. If for simplicity we assume that the eigenvalues are distinct, then Bx is a multiple of x for some $\lambda' \in \mathbb{R}$ (namely $Bx = \lambda' x$) and x is an eigenvector of both A and B.
($\impliedby$ direction) If two matrices share the same eigenvectors, and are diagonalizable, then they are simultaneously diagonalizable *i.e.* there exists an invertible matrix P such that:

$$P^{-1}AP = D_A \quad \wedge \quad P^{-1}BP = D_B \tag{5.27}$$

Where D_A and D_B are diagonal. Checking commutativity:

$$\begin{aligned} AB &= PD_AP^{-1}PD_BP^{-1} && \text{inverting the Equations above}\\ &= PD_AD_BP^{-1} && \text{as } P^{-1}P = I\\ &= PD_BD_AP^{-1} && \text{diagonal matrices commute}\\ &= PD_BP^{-1}PD_AP^{-1} && \text{expanding } I\\ &= BA && \text{by assumption} \end{aligned}$$

And $AB = BA \iff$ *simultaneously diagonalizable*, where simultaneous diagonalization is equivalent to diagonalizability and same eigenvectors as P is the matrix of eigenvectors. □

The just proved result is exploited in many fields. In Lie algebra, a set of simultaneously diagonalizable matrices generates a *Toral Lie Algebra*, which should ring a bell.

Lastly, we are able to characterize operators C with an *almost* formal justification.

Definition 5.20 (Convolutional Operation $\star$)**.** For two functions f, g a convolution is defined as:

$$(f \star g)(t) = \int_{-\infty}^{\infty} f(\tau) g(t - \tau) d\tau \tag{5.28}$$

Which can be adapted to the discrete countable domain easily.

Theorem 5.21 (Characterization of Equivariant linear map for Grids)**.** *If C is the matrix representing an equivariant linear map on a $1-d$ grid $\Omega = \{1, \dots, d\}$ then it holds that:*

1. *C is diagonal in the Fourier domain.*

$$C = V^\dagger diag(\widehat{\alpha})V \quad \alpha \in \mathbb{C}^d \tag{5.29}$$

2. *C is a convolutional operation with the resulting rows being convolutions of x and the complex diagonal.*

$$(Cx)_j = (x \star \alpha)_j \tag{5.30}$$

Proof. **(Claim** 1**)** From Theorem 5.14 we have $SC = CS$. Using Lemma 5.19 if we assume for simplicity both are diagonalizable (S is diagonalizable) it is possible to state that they share the same eigenvectors and that they are simultaneously diagonalizable. Then, since V is the eigenmatrix of S:

$$VSV^\dagger = D_S \underset{sim\ diag}{\implies} VCV^\dagger = D_C \underset{isolating\ C}{\implies} C = V^\dagger D_C V \tag{5.31}$$

Where we set $D_C = diag(\widehat{\alpha})$ as notation for the diagonal matrix with $\alpha \in \mathbb{C}^d$.
(Claim 2**)** Following the lecture, we verify that:

$$\begin{aligned}
Cx &= C\left(\sum_k \widehat{x}[k] v_k^\dagger\right) && \text{IDFT, Theorem 5.18} \\
&= \sum_k \widehat{x}[k] C v_k^\dagger && \text{as } \widehat{x}[k] \in \mathbb{C} \text{ and } C \text{ is a linear operator} \\
&= \sum_k \widehat{x}[k] \widehat{\alpha}_k v_k^\dagger && \text{by Claim 1, } C \text{ diagonalizes in the Fourier domain}
\end{aligned}$$

Considering a row then **TODO check dimension match and shift of signal and if x is daggered at the end or not**:

$$
\begin{aligned}
(Cx)_j &= \sum_k \left[\widehat{x}^\dagger[k] e^{-i2\pi\frac{jk}{d}}\right]^\dagger \widehat{\alpha}_k && \text{reordering for a row the last result} \\
&= \sum_k \left[(\langle x, v_k\rangle)^\dagger e^{-i2\pi\frac{jk}{d}}\right]^\dagger \widehat{\alpha}_k && \text{by DFT definition 5.16} \\
&= \sum_k \left[(v_k^\dagger x)^\dagger e^{-i2\pi\frac{jk}{d}}\right]^\dagger \widehat{\alpha}_k && \text{by Definition of dot product} \\
&= \sum_k \left[x^\dagger v_k e^{-i2\pi\frac{jk}{d}}\right]^\dagger \widehat{\alpha}_k && \text{reordering} \\
&= \sum_k \left[S^{-j} v_k x^\dagger\right]^\dagger \widehat{\alpha}_k && \text{Theorem 5.15 applied } -j \text{ times eigendecomposition} \\
&= \sum_k \left[\widehat{S^{-j} x^\dagger}\right]^\dagger \widehat{\alpha}_k && \text{recognize Fourier transf of } -j \text{ shifted transposed signal} \\
&= \langle \widehat{S^{-j} x^\dagger}, \widehat{\alpha}_k \rangle && \text{by Definition of dot} \\
&= \langle S^{-j} x^\dagger, \alpha_k \rangle && \text{Parceval's Identity 5.17} \\
&= \sum_l x_{j-l} \alpha_l && \text{expanding the dot product} \\
&= (x \star \alpha)_j && \text{by Definition of convolution}
\end{aligned}
$$

□

Having shown that:

- any convolution is diagonal in the Fourier space
- any linear operator commuting with S is a convolution

We now sketch the opposite implication, namely that any convolution commutes with S

$$(Cx)_j = \sum_l x_{j-l}\alpha_l = \sum_l \alpha_l S^{-l} x_j \tag{5.32}$$

Where we have just set $x_{j-l} = S^{-l}x_j$. We can check that this implies that any discrete convolution commutes with S as per the last equation it is equivalent to applying first S^{-j} and then multiplying by $\widehat{\alpha}_l$. Given that $SS^{-l} = S^{-l}S$ and $\widehat{\alpha}_l$ is a coefficient, convolving and translating commute as operations.

The connection between convolutions and Fourier transforms is visualized in Figure 5.5

While Fourier transforms are great tools, there are drawbacks, especially when constraining the transformations to be local. To justify this claim, we will rather provide an example.

Example 5.22 (Fourier invariant Modulus)**.** Assume a very local Fourier transformation is carried out, where:

$$\widehat{\alpha}[\tilde{k}] = 1 \wedge \widehat{\alpha}[k] = 0 \forall k \neq \tilde{k} \tag{5.33}$$

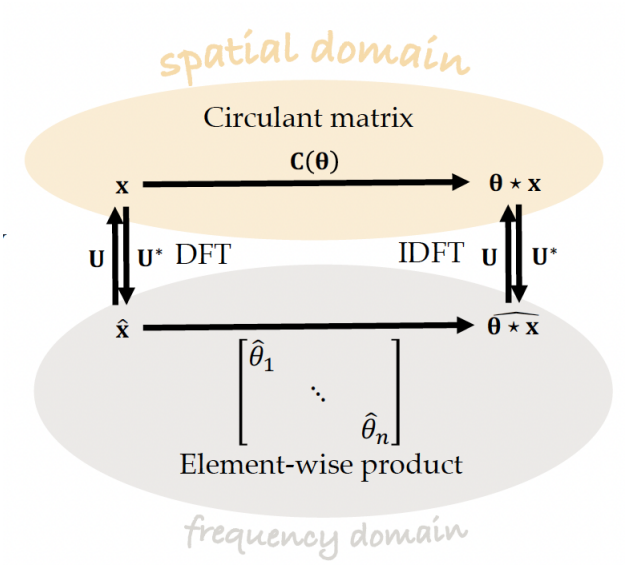


Figure 5.5: Convolutions and Shifts

Considering the operator $\Phi(x) = \rho(\widehat{x}) = |\widehat{x}|$ we check that it is translation invariant:

$$\begin{aligned}\Phi(S^n x) &= |e^{-i2\pi\frac{nk}{d}}\widehat{x}| = |(e^{i(-2\pi)})^{\frac{nk}{d}}\widehat{x}| \\ &= |(\cos(-2\pi) + i\sin(-2\pi))^{\frac{nk}{d}}\widehat{x}| = |(1)^{\frac{nk}{d}}\widehat{x}| = |\widehat{x}| = \Phi(x)\end{aligned}$$

While this might look sufficient, as we exploit the translation properties, and the modulus kills the imaginary phase, most of semantic information is conserved in the phase and the transformed value will lose information. A graphical example of the outcome of a Fourier invariant operation is proposed in Figure 5.6

Using as a reference the continuous domain where $\Omega = T$, translations are a subgroup of Automorphisms $Aut(\Omega)$ of the form $\tau : \Omega \to \Omega$. In some cases, to relax the daunting invariance restriction, it is possible to consider slightly relaxed functions and impose *deformation stability* requirements such as:

$$|f(x) - f(\tau.x)| \leq c(\tau) \tag{5.34}$$

Where $c(\cdot)$ is a smoothness measure that controls the distance from the symmetry group:

$$c(\tau) = \sup\{\|\nabla\tau(u))\|\} \tag{5.35}$$

To check that these conditions include invariant transformations it is enough to see that:

$$c(\tau) = 0 \forall \mathfrak{g} \in \mathfrak{G} \implies |f(x) - f(\tau.x)| \leq 0 \implies f(x) = f(\tau.x) \forall x, \forall \tau \tag{5.36}$$

Doing so, the allowed transformations would be part of a set of functions that is controlled in the deformation effect but stores more information.

However, in the enlarged space, Fourier invariants prove to be unstable to deformations, as the following argument will (*informally*) show. Assume we are in the continuous domain. There,

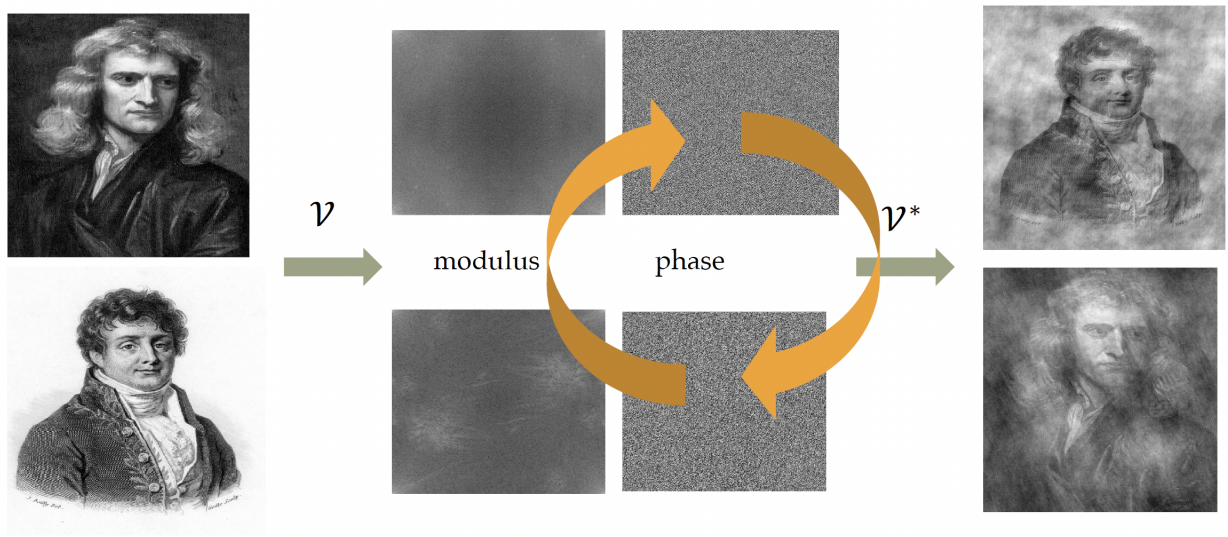


Figure 5.6: Loss of information of Newton vs Fourier portraits

a signal $x(u) = h(u)e^{i\xi u}$ is controlled by a window $h(u)$ of bandwidth σ_u, and a deformation $(\tau.x)(u) = x((1+s)u) : s \ll 1$. Then the translated vector x_τ is:

$$x_\tau(u) = h((1+s)u)e^{i\xi(1+s)u}$$

We claim that if the frequence ξ and a small translation s are much bigger than the bandwidth, then the Fourier space distance of the modulus is equivalent to the norm of the original vector. Namely:

$$(1+s)\xi - \xi = s\xi \gg \sigma_h(2+s) \implies \| |\widehat{x}| - |\widehat{\tau.x}| \| \sim \|x\| \tag{5.37}$$

Making such invariant operations highly dependent on the size of the input, and not reliable in terms of stability.

Thus, even though there is an idea of how to avoid information loss, it does not generalize well. A representation by spectral properties of the domain is unstable as soon as we deviate from strict translations. We must resort to different techniques, and will in particular focus on scale separation with wavelets.

5.2 Wavelet Scattering Representations

Due to the difficulty of the topic, this section will be presented in a *very informal* manner.

As an alternative to frequency extraction we consider information at different *scales*. For an idea of how this is practically carried out, see a comparison with Fourier transforms in Figure 5.7.

To achieve such decomposition, we will implement *filter banks* of the form:

$$Wx = (x \star \psi_\alpha) \tag{5.38}$$

Where:

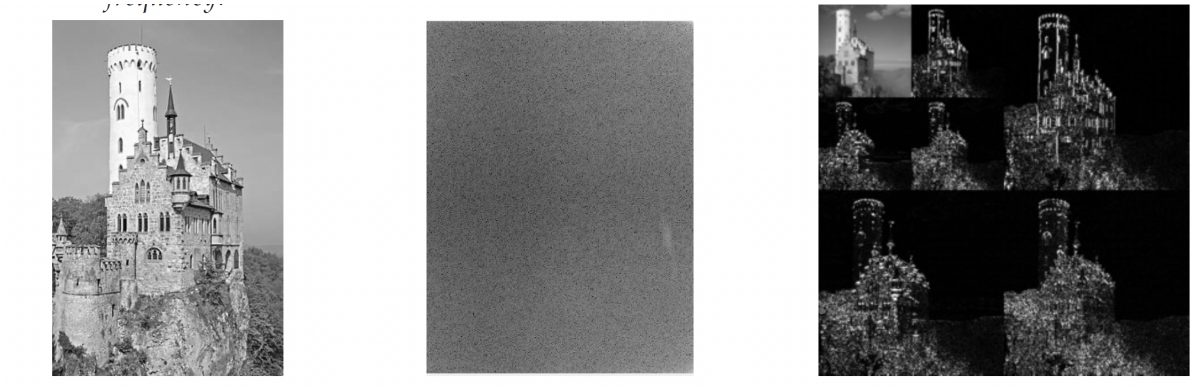


Figure 5.7: In order: original, Fourier, Scale

- ψ is the *mother* wavelet, an wave localized in space
- $\psi_\alpha(u) = 2^{-j}\psi(2^{-j}R_\theta u)$ where $\alpha = (j, \theta)$ are polar indicators for dilation and rotation

A visualization of the forms wavelets can take is found on Figure 5.8.

In this context, if they are designed through the Littlewood-Paley transform:

$$(1-\epsilon)\|x\| \leq \|Wx\| \leq \|x\| \tag{5.39}$$

Then some theoretical guarantees about deformation stability and linear equivariance can be made. The first is by Mallat [Mal09], the second by Mallat and Waldspurger [Mal12; Wal17], the third by Mallat [Mal12].

Theorem 5.23 (Wavelet Linear Equivariance)**.** *For such a setting, we have that:*

$$\|W(\tau.x) - \tau.(Wx)\| \leq \|\nabla\tau\|\|x\| \tag{5.40}$$

Where the norm of the transformation controls closeness to equivariant transformations.

Theorem 5.24 (Wavelet Deformation Stability)**.** *For appropriate choice of mother wavelets:*

$$\|\Phi(x)\| = \|x\| \tag{5.41}$$

Moreover, energy convergence is exponential in depth, making the overall representation stable to deformations.

Theorem 5.25 (Wavelet Linear Invariance)**.** *For $x \in L^2(\Omega)$ compactly supported and $c(\tau) = g(\|\nabla\tau\|, \|\nabla^2\tau\|)$ as a smoothness measure* ***TODO check what is*** *g* ***in paper****:*

$$\|\Phi(\tau.x) - \Phi(x)\| \leq c(\tau)\|x\| \tag{5.42}$$

Where the norm and the smoothness measure control closeness to invariant transformations

Theorems 5.23, 5.24 and 5.25 together justify the design of a specific Wavelet Geometric Deep Learning blueprint, through what are commonly referred to as scattering representations.

Definition 5.26 (Wavelet GDL Blueprint)**.** Adapt the building blocks of Definition 3.43 to Wavelets as:

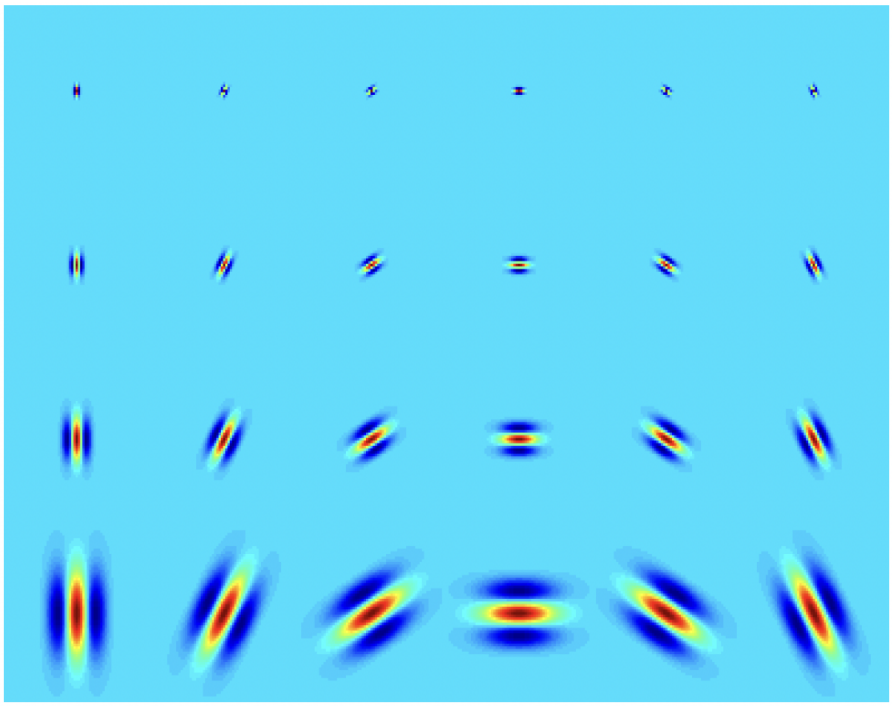


Figure 5.8: Wavelet filters example

- Linear local invariant filters A,
- linear local equivariant wavelet filter bank W
- a pointwise non linear function ρ applied after each wavelet

To better visualize the effects of such maps, the correpondence with graphical effects for an image is established below.

Example 5.27 (Wavelet GDL Blueprint on a hermit crab)**.** The final invariant filter alone acts as in Figure 5.9. The combination of a single layer $\{A, A\rho W\}$ produces the scale separation of Figure 5.10. As soon as more layers are introduced, deeper separations are obtained. In Figure 5.11, we have $\{A, A\rho W, A\rho W\rho W\}$.

While the representation power of the Wavelet blueprint can be implemented in Computer Vision tasks[6], experimental evidence shows that it is **not** enough to obtain good performances on hard to solve benchmarks. Figure 5.12 is a comparison between DeepNets and Wavelet scattering combined with a linear classifier.
The problem lies in the absence of interactions between feature maps. In the next section a brief exaplanation of why CNNs overcome this issue and maintain the requirements will be outlined.

[6]N.B.: Being just a representation, it requires no training and needs to be combined with a classifier.

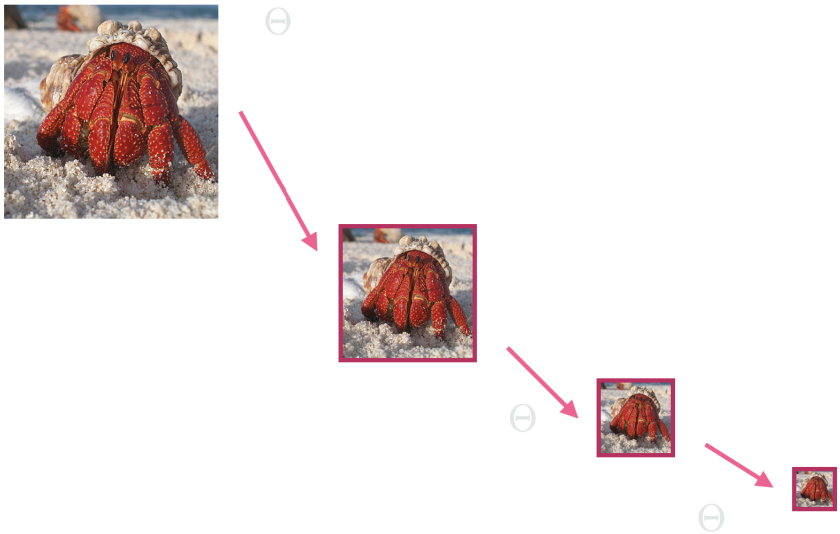


Figure 5.9: Invariant filter alone

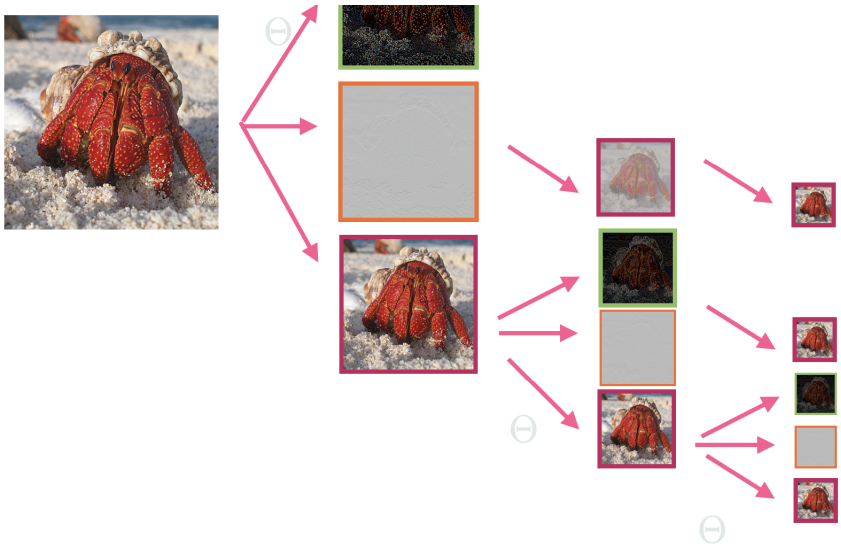


Figure 5.10: One layer scale separation $\{A, A\rho W\}$

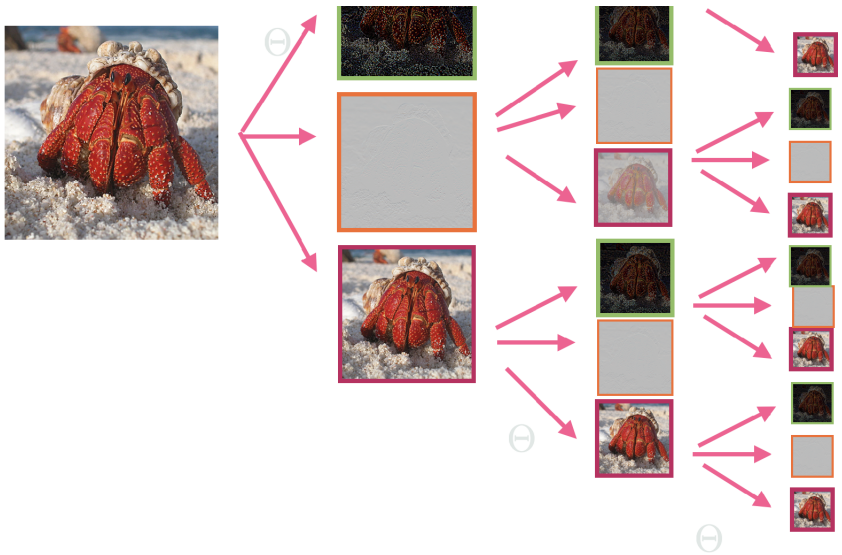


Figure 5.11: Two layers scale separation $\{A, A\rho W, A\rho W \rho W\}$

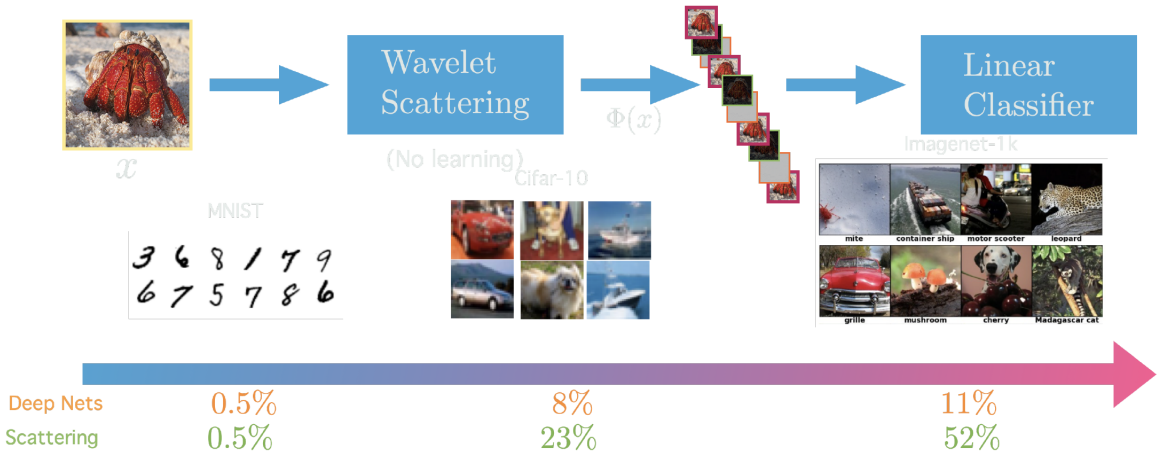


Figure 5.12: Performance comparison Wavelet Scattering vs DeepNets

5.3 Convolutional Neural Networks

Convolutional Neural Networks combine wavelet spatial filters and cross channel trainable coefficients θ. These θ coefficients capture connections across wavelet coefficients, providing more flexibility [Oya+19], [Zar+20]. From one layer to the other, the transformation is of the form:

$$x_{m+1} = \rho\left(\sum_k (\Psi_k x_m)\theta_k \right) \quad \theta_k \in \mathbb{R}^{C_m \times C_{m+1}} \tag{5.43}$$

Where:

- m indexes layers
- Ψ_k are the wavelet transformations of x_m, in the form $\Psi_k x = x \star \psi_k$
- θ_k captures interactions across layers and is indeed dimension compatible with both

The above architecture almost closes the gap with state of the art models, that contain lots of heuristics to improve the performance such as:

- residual layers
- normalization layers

On the theory side, some still open questions are:

- identifying the functional space of CNNs
- quantitative assessment of the role of depth and width of filters
- understanding the interaction between the architecture and the optimization process

To conclude the analysis, we briefly draw the most important conclusions:

- Grids have a rich theoretical structure, with geometric and analytic properties. Thanks to this, we are able to characterize clearly the GDL blueprint.
- Wavelet Scattering is the simplest instance that respects all the requirements, exploiting the concept of multiscale prior. It requires no training.
- CNNs are the result of a mix of theory and heuristics. Currently, they seem to achieve the best known tradeoff between geometric priors and representational power.
- there are a lot of research questions yet to be solved

For Scattering Representation applications, we reroute the reader to some interesting sources [BM19], [Che+20], [Eic+18].

In the next chapters, we will explore non Euclidean domains, tackling them with groups, meshes and gauges.

Chapter 6

Groups

In this chapter we will dive deeper into the Group theoretic foundations of GDL. The concepts covered are:

- Group convolution
  - The intuitive roto-translation discrete case
  - how to generalize to continuous 3D rotations
- General theory of groups and Homogeneous Spaces
  - how to define a convolution
  - a theory that works for any Homogeneous space
- The naive convolution on Homogeneous Spaces
  - why is it limited
  - the problem of isotropic filters
- Induced representations and Steerable CNNs
  - final version of the *Convolution is all you need* Theorem

6.1 Group Convolution

Consider the simplest case of group convolutions, with discrete integer translations and rotations of $\frac{\pi}{2}$ radiants. Many image recognition problems wish to be equivariant to those two. Our disucssion can be seen applied to the case of Figure 6.1. There ρ_θ is a θ rotation on the signal, while $\mathcal{R}_\theta$ is a rotation on the channel space. The filter is rotated through $\rho_\theta \Psi$ instead.

In a convolution, a filter is slided over the image, and gets a strong response up to matching with the portion of the signal $x \in \mathcal{X}(\Omega, \mathcal{C})$. For the rotated filter, there is no match with the eye or mouth filter unless the channel itself is oriented as the image. For each element in a roto translation group we apply a 2D filter and compute the inner product with the filter to evaluate the response. The filter is indeed equivariant to translations but **not** to rotations. The feature maps (the channels) rotate, and the match is found with the second filter orientation.

The transformation between the feature maps makes the identification appear through a rotation and a cyclic shift of the filters involved. In an RGB image as an input, there is only one way of

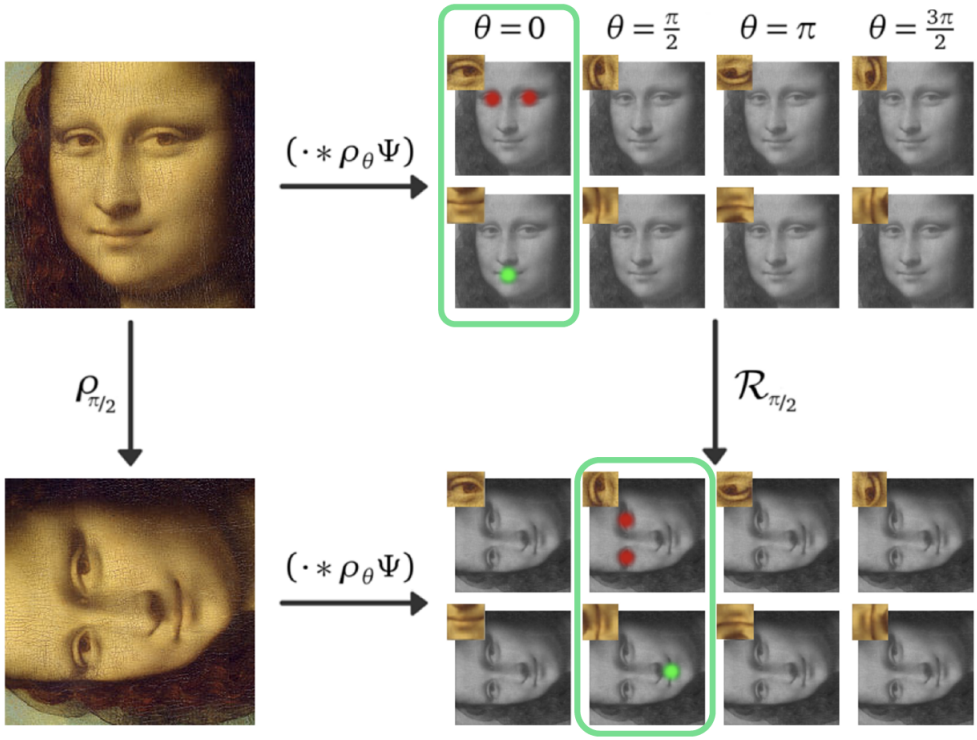


Figure 6.1: A discrete roto-translation map

transforming, but in the feature space, since there is an additional action on the channel, even though the group is the same, the way that it acts on data is different.

Example 6.1 (Discrete roto-translations on a letter)**.** Consider another example. In Figure 6.2 we have an input image, and doing the convolution we obtain 4 feature maps. There, the filter is rotated 4 times, and the feature map is the same above and below but rotated (blue arrow). In the rotated layer, the input has now 4 different input channels (feature maps). Convolving it, we get one output channel and get to rotate the filter 4 times. There is a different transformation law. One can show that the output of the operation below has the same transformation law up to rotation.

As a last operation, what one could do is a pooling of channels through some invariant operation. With the resulting image having the same transformation behavior of the input, as in Figure 6.4 In the input $\Omega = \mathbb{Z}_2$, transformed as a scalar field (we will see this later). In the first layer, the structure is $\Omega = p4$, where each pixel has spatial coordinates (translations) and channel coordinates (rotations). Another way to think of the 4 channels is as a single function on the group $p4$, this is a regular representation (we will see later what it means). The other hidden layer has the same properties. The final layer is the same as the input space.

To understand what a group convolution is, we proceed by analogy with the usual convolution in Table 6.1.

We recognize that a group convolution acts on the orbit of a signal's transformations. The inner product is weigthed by the Haar measure $d\mu(u)$ adn the convolution resembles that of an inner

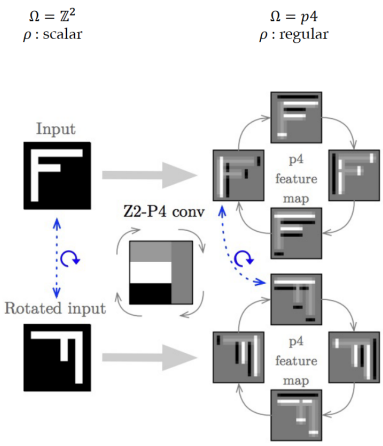


Figure 6.2: Discrete input roto-translation

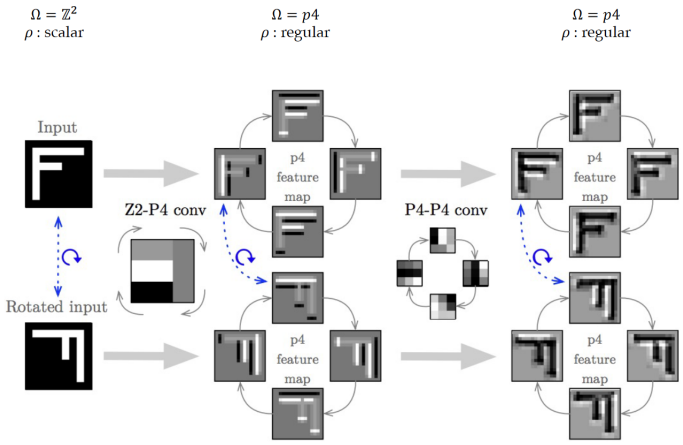


Figure 6.3: Discrete rotated layer roto-translation

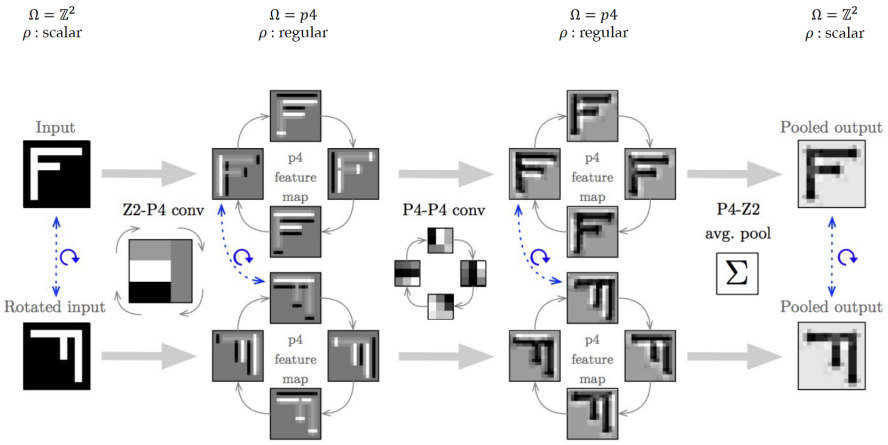


Figure 6.4: Discrete output roto-translation

signal & *translated* filter	signal & *transformed* filter
translated signal as $\rho(v)x(u)$	transformed signal as $\rho(\mathfrak{g})x(u) = x(\mathfrak{g}^{-1}u)$
Inner product $\langle x, y\rangle = \sum_{u=1}^{N}\sum_{c=1}^{\mathcal{C}} x_c(u)y_c(u)$	Inner product $\langle x, y\rangle = \int_\Omega \langle x(u), y(u)\rangle_{\mathcal{C}} d\mu(u)$
Convolution $x\star\psi(v) = \langle x, \rho(v)\psi\rangle = \sum_{u=1}^{N}\sum_{c=1}^{\mathcal{C}} x_c(u)\psi_c(u-v)$	Group Convolution $x \star \psi(\mathfrak{g}) = \langle x, \rho(\mathfrak{g})\psi\rangle = \int_\Omega \langle x(u), \psi(\mathfrak{g}^{-1}u)\rangle_{\mathcal{C}} d\mu(u)$
Equivariance $(\rho(v)x)\star\psi = \rho(v)(x\star\psi)$	Equivariance $(\rho(\mathfrak{g})x)\star\psi = \rho(\mathfrak{g})(x\star\psi)$

Table 6.1: Convolutions Analogies

product between the signal and a transformed filter across the various channels. Similarly to the normal convolution case, equivariance is verified when a transformed signal convolved with a filter is equal to the transformation of the convolution of the signal and the filter itself (i.e. the representation slides out of the convolution). Putting all the concepts together into a Definition:

Definition 6.2 (Group Convolution)**.** Given a signal $x \in \mathcal{X}(\Omega, \mathcal{C})$ and a filter $\psi : \mathfrak{G} \to \mathcal{X}(\Omega, \mathcal{C})$ we define a group convolution, denoted by $\star$ as:

$$x \star \psi(\mathfrak{g}) = \langle x, \rho(\mathfrak{g})\psi\rangle = \int_\Omega \langle x(u), \psi(\mathfrak{g}^{-1}u)\rangle_{\mathcal{C}} d\mu(u) \tag{6.1}$$

Where ρ is the representation of the group $\mathfrak{G}$. Moreover, $\star$ is equivariant if:

$$(\rho(\mathfrak{g})x)\star\psi = \rho(\mathfrak{g})(x\star\psi) \quad \forall \mathfrak{g} \in \mathfrak{G}, \forall\rho, \forall x \in \mathcal{X}(\Omega, \mathcal{C}) \tag{6.2}$$

Recall that signals are maps from the structure Ω to the channel vector space $\mathcal{C}$ seen as: $x : \Omega \to \mathcal{C}$. A valid transformation of a group action $\mathfrak{G} \times \Omega \to \Omega$, by Proposition 3.21, is of the form:

$$\rho(\mathfrak{g})x(u) = x(\mathfrak{g}^{-1}u) \tag{6.3}$$

When $\mathfrak{G} = \Omega$, so that the structure is the group itself, more can be said.

Proposition 6.3 (Group Action on Group (left version))**.**

$$\Omega = \mathfrak{G} \implies \textit{valid group action } (\mathfrak{g}, \mathfrak{h}) \to \mathfrak{g}\mathfrak{h} \tag{6.4}$$

Proof. If $\Omega = \mathfrak{G}$ then a group action is a map $\mathfrak{G} \times \mathfrak{G} \to \mathfrak{G}$. The function $(\mathfrak{g}, \mathfrak{h}) \to \mathfrak{g}\mathfrak{h}$ is of this form. Moreover, by Theorem 3.19 a sufficient condition is closure under composition, namely $\mathfrak{g}(\mathfrak{h}(\mathfrak{l})) = (\mathfrak{g}\mathfrak{h})(\mathfrak{l})$, which is verified by Definition of Group. □

With this result in mind, we recognize that the left action of $\mathfrak{G}$ on itself, which is a translation can characterize signals mapping from the group to channels.

Definition 6.4 (Regular Representation ρ of $\mathfrak{G}$ on $\mathcal{X}(\mathfrak{G}, \mathcal{C})$)**.** For a group $\mathfrak{G}$ with signals $\mathcal{X}(\mathfrak{G}, \mathcal{C})$ a regular representation is a map:

$$\rho : \mathfrak{G} \to \mathcal{C} \tag{6.5}$$

Where:

$$\rho(\mathfrak{g})x(\mathfrak{h}) = x(\mathfrak{g}^{-1}\mathfrak{h}) \quad \forall \mathfrak{g}, \mathfrak{h} \in \mathfrak{G}, \forall x \in \mathcal{X}(\mathfrak{G}, \mathcal{C}) \tag{6.6}$$

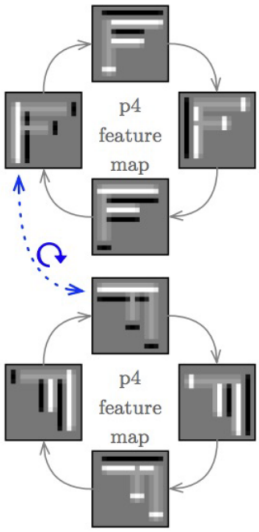


Figure 6.5: Discrete roto translation group

Hopefully, the example below will clear out doubts regarding this Definition.

Example 6.5 (Regular representation of discrete roto-translation group)**.** Consider the group that arises from $\frac{\pi}{2}$ rotations of a signal as in Figure 6.5.

Group elements $\mathfrak{g} \in \mathfrak{G}$ can be encoded in a matrix:

$$\mathfrak{g} = \begin{bmatrix} \cos(r\frac{\pi}{2}) & -\sin(r\frac{\pi}{2}) & t_x \\ \sin(r\frac{\pi}{2}) & \cos(r\frac{\pi}{2}) & t_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} R(r) & t \\ \mathbf{0} & 1 \end{bmatrix} \tag{6.7}$$

Where $r \in \{0, 1, 2, 3\}$ are the $\frac{\pi}{2}$ multiples rotations forming a rotation matrix $R(r) \in \mathbb{R}^{2\times 2}$ and $t \in \mathbb{Z}^2$ is the discrete translation vector in a plane.

We check composition and inverse to apple the definition of regular representation. Composition of two group elements is such that:

$$\begin{aligned}
\mathfrak{g}\mathfrak{h} &= \begin{bmatrix} R(r) & t \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} R(r') & t' \\ \mathbf{0} & 1 \end{bmatrix} \\
&= \begin{bmatrix} R(r)R(r') & R(r)t' + t \\ \mathbf{0} & 1 \end{bmatrix} \\
&= \begin{bmatrix} \cos(r\frac{\pi}{2})\cos(r'\frac{\pi}{2}) - \sin(r\frac{\pi}{2})\sin(r'\frac{\pi}{2}) & -\cos(r\frac{\pi}{2})\sin(r'\frac{\pi}{2}) - \sin(r\frac{\pi}{2})\cos(r'\frac{\pi}{2}) & \cos(r\frac{\pi}{2})t'_x - \sin(r\frac{\pi}{2})t'_y + t_x \\ \sin(r\frac{\pi}{2})\cos(r'\frac{\pi}{2}) + \cos(r\frac{\pi}{2})\sin(r'\frac{\pi}{2}) & -\sin(r\frac{\pi}{2})\sin(r'\frac{\pi}{2}) + \cos(r\frac{\pi}{2})\cos(r'\frac{\pi}{2}) & \sin(r\frac{\pi}{2})t'x + \cos(r\frac{\pi}{2})t'_y + t_y \\ 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} \cos((r+r')\frac{\pi}{2}) & -\sin((r+r')\frac{\pi}{2}) & \cos(r\frac{\pi}{2})t'_x - \sin(r\frac{\pi}{2})t'_y + t_x \\ \sin((r+r')\frac{\pi}{2}) & \cos((r+r')\frac{\pi}{2}) & \cos(r\frac{\pi}{2})t'_x - \sin(r\frac{\pi}{2})t'_y + t_x \\ 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} R(r+r') & R(r)t' + t \\ \mathbf{0} & 1 \end{bmatrix} \in \mathfrak{G}
\end{aligned}$$

Where we just used trigonometric identities.
For the inverse, we instead check that the candidate:

$$\begin{bmatrix} R(-r) & -R(-r)t \\ \mathbf{0} & 1 \end{bmatrix} \tag{6.8}$$

is valid (i.e. returns identity and commutes) Then:

$$\begin{aligned}
\mathfrak{g}\begin{bmatrix} R(-r) & -R(-r)t \\ \mathbf{0} & 1 \end{bmatrix} &= \begin{bmatrix} R(r) & t \\ \mathbf{0} & 1 \end{bmatrix}\begin{bmatrix} R(-r) & -R(-r)t \\ \mathbf{0} & 1 \end{bmatrix} \\
&= \begin{bmatrix} R(-r+r) & R(r)(-R(-r)t)+t \\ \mathbf{0} & 1 \end{bmatrix} && \text{applying composition found above} \\
&= \begin{bmatrix} R(0) & -R(0)t+t \\ \mathbf{0}1 & \end{bmatrix} && \text{previous calculations on } R \text{ matrices} \\
&= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} && \text{trigonometric identities} \\
&= I
\end{aligned}$$

Which is the identity of the roto translation group. Checking the opposite order $\mathfrak{g}^{-1}\mathfrak{g} = I$ follows the same steps.

Now that we know how to compute the basic operations in the roto-translation group, we aim to recover a formula for its regular representation. Denoting a group element as (r, t) and letting for simplicity $t' = 0$ (no translation in the group operation applied), we have that:

$$\rho(\mathfrak{g})x(\mathfrak{h}) = x(\mathfrak{g}^{-1}\mathfrak{h}) \tag{6.9}$$

$$\iff \rho(r', 0)x(r, t) = x((r', t)^{-1}(r, t)) \tag{6.10}$$

$$= x\left[\begin{bmatrix} R(-r') & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix}\begin{bmatrix} R(r) & t \\ \mathbf{0} & 1 \end{bmatrix}\right] \quad \text{inverse operation} \tag{6.11}$$

$$= x\left[\begin{bmatrix} R(r-r') & R(-r')t \\ \mathbf{0} & 1 \end{bmatrix}\right] \quad \text{composition operation} \tag{6.12}$$

$$= x\left[\begin{bmatrix} R(r-r') & R(-r')(t) \\ \mathbf{0} & 1 \end{bmatrix}\right] \tag{6.13}$$

$$= x\Big[(r-r')(mod\,4), R(-r')t\Big] \tag{6.14}$$

Where we use $(mod\,4)$ to make the translation cyclic and valid for $t \in \{0, 1, 2, 3\}$ as previously argued. This is where the channel cycling + rotation comes out.

As dimensions increase, channel cycling becomes more and more difficult to evaluate. In Figure 6.6, we propose the same concept for a $3D$ cube with filters on sections of the volume.

While we showed that there exists a regular representation of a group translation over orbits, we did not yet prove that such an object is convolution equivariant as per Definition 6.2.

Theorem 6.6 (Regular Representation induces Equivariant group Convolution)**.** *Assume a signal is transformed through a group symmetry via a group representation ρ_1 as:*

$$\mathcal{X}(\Omega_1) : \rho_1(\mathfrak{g})x(u) = x(\mathfrak{g}^{-1}u) \tag{6.15}$$

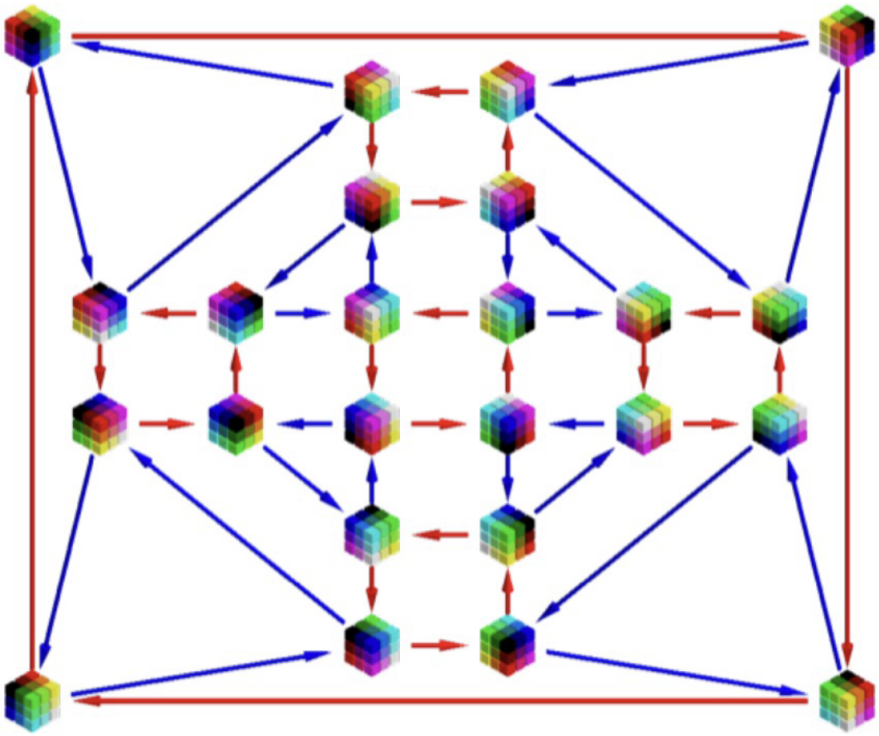


Figure 6.6: $3D$ cube filters and convolutions

There, the first convolution ψ applies a filter to an element of the signal into its orbit of signals as:

$$\cdot \star \psi : \mathcal{X}(\Omega_1) \to \mathcal{X}(\Omega_2) \quad \Omega_2 = \mathfrak{G} \tag{6.16}$$

After this first layer, a second moves around the orbit of signals with a regular representation ρ_2 as:

$$\mathcal{X}(\mathfrak{G}) : \rho_2(\mathfrak{g})x(\mathfrak{h}) = x(\mathfrak{g}^{-1}\mathfrak{h}) \tag{6.17}$$

And is convolved with a filter ψ and a map:

$$\cdot \star \psi : \mathcal{X}(\mathfrak{G}) \to \mathcal{X}(\mathfrak{G}) \tag{6.18}$$

Then:

1. $$\implies \langle x, \rho_1(\mathfrak{g})\psi \rangle = \langle \rho_1(\mathfrak{g}^{-1})x, \psi \rangle \tag{6.19}$$

2. *The an architecture for any number of second type layers, is equivariant, and can be represented with a diagram like that of Figure 6.7.*

$$\implies \Big((\rho_1(\mathfrak{h})x) \star \psi \Big)(\mathfrak{g}) = \rho_2(\mathfrak{h})(\psi \star x)(\mathfrak{g}) \tag{6.20}$$

Proof. **(Claim 1)** The formula can be interpreted as asserting that a signal matched with a transformed filter is equal to matching the inverse transformed signal with untransformed filter. An example can be seen in Figure 6.8. To prove it, we go on as follows:

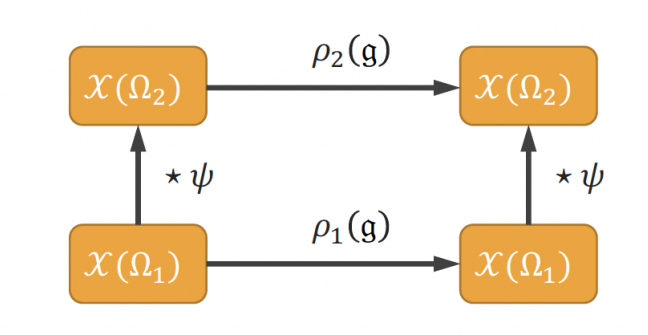


Figure 6.7: Equivariant layers with group convolutions

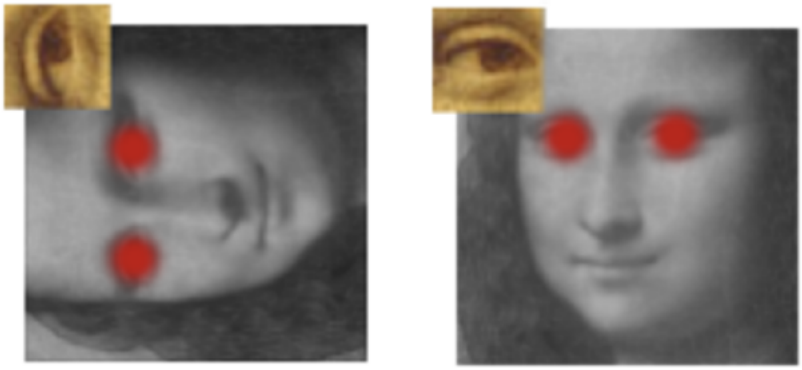


Figure 6.8: Claim 1 Theorem 6.6 visualized

$$
\begin{aligned}
\langle x, \rho_1(\mathfrak{g})\psi\rangle &= \int_\Omega \langle x(u), \psi(\mathfrak{g}^{-1}u)\rangle_{\mathcal{C}} d\mu(u) \\
&= \int_\Omega \sum_{c=1}^{\mathcal{C}} x_c(u)\psi_c(\mathfrak{g}^{-1}u) d\mu(u) \\
&= \int_\Omega \sum_{c=1}^{\mathcal{C}} \rho_1(\mathfrak{g}^{-1}) x_c\psi(u) d\mu(u) \\
&= \int_\Omega \langle \rho_1(\mathfrak{g}^{-1})x, \psi\rangle_{\mathcal{C}} d\mu(u) \\
&= \langle \rho_1(\mathfrak{g}^{-1})x, \psi\rangle
\end{aligned}
$$

Where we just notice that this is verified by the properties of inner product, namely:

$$\langle x, \beta y\rangle = \overline{\beta}\langle x, y\rangle = \langle \overline{\beta}x, y\rangle$$

(Claim 2) With the result of Claim 1 in mind, the second follows directly:

$$
\begin{aligned}
\Big((\rho_1(\mathfrak{h})x) \star \psi\Big)(\mathfrak{g}) &= \langle \rho_1(\mathfrak{h})x, \rho_1(\mathfrak{g})\psi\rangle && \text{group convolution Definition 6.2} \\
&= \langle x, \rho_1(\mathfrak{h}^{-1})\rho_1(\mathfrak{g})\psi\rangle && \text{claim 1 on } \rho_1(\mathfrak{h}) \\
&= \langle x, \rho_1(\mathfrak{h}^{-1}\mathfrak{g})\psi\rangle \\
&= \langle \rho_1(\mathfrak{g}^{-1}\mathfrak{h})x, \psi\rangle && \text{claim 1 on } \rho_1(\mathfrak{h}^{-1}\mathfrak{g}) \\
&= \langle \psi, \rho_1(\mathfrak{g}^{-1}\mathfrak{h})x\rangle && \text{symmetry of product} \\
&= \langle \psi, x(\mathfrak{h}^{-1}\mathfrak{g})\rangle && \text{group representation Definition 3.22} \\
&= \psi \star x(\mathfrak{h}^{-1}\mathfrak{g}) && \text{group convolution Definition 6.2} \\
&= \rho_2(\mathfrak{h})(\psi \star x)(\mathfrak{g}) && \text{group regular representation Definition 6.4}
\end{aligned}
$$

Where stacking second type layers the answer does not change. □

While we have proved that a group convolution is a linear equivariant map, it is also possible to assert a stronger (opposite) property of convolutions, inspected in refereces such as [KT18; CGW19; Coh21; Aro21]. In later sections, we will dive deeper into this concept.

Theorem 6.7 (Convolution is all you need, informal)**.** *Any linear equivariant map between regular representations is a group convolution*

From the first few slides the idea is simple: we consider a filter ψ, rotate them, and do a translation of convolution. To implement this, two steps are taken into consideration. Weights of filters are combined with indices that define the orientation. This gives a bigger filter bank, which stacked with Conv2D on the input feature map returns the output feature maps. A diagram of the workflow is shown in Figure 6.9. Once trained, it is not heavy computationally.

Example 6.8 (Group Convolution on the letter F)**.** As an example, consider Applying discrete roto-translations to a channel with 4 dimensions for the digital image of an F. The output will be a filter bank with 4×4 channels, shown in Figure 6.10.

Among other approaches it is worth mentioning:

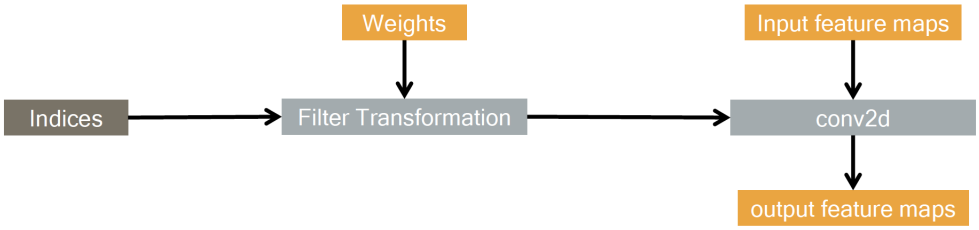


Figure 6.9: Efficient Implementation of Group convolution

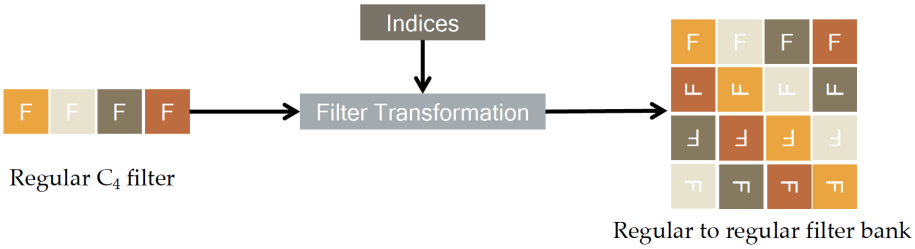


Figure 6.10: Group Convolution letter F example

- Steerable CNNs [CW16; Wei+18; WC21]
  - generalizes group convolution to handle arbitrary fields
  - computational cost does not grow as $\mathfrak{G}$ does
- B-Spline CNNs [Bek21]
- LieConv [Fin+20]
  - continuous filters parametrized by an MLP
  - live on tangent space (Lie Algebra)

6.1.1 Spherical CNNs

In the context of spherical data, CNNs are largely distorted when the image is embedded in $\mathbb{R}^2$. A solution to this problem exploits the structure of the datapoints.

This subsection is devolved to showing an example of an application of the just introduced concepts.

We assume that the input and the filter are localized on a sphere S^2.

$$x : S^2 \to \mathbb{R}^{\mathcal{C}} \quad \psi : S^2 \to \mathbb{R}^{\mathcal{C}} \tag{6.21}$$

In such a setting, the symmetry group is $SO(3)$, that of 3D rotations. We then define a Group convolution from a sphere with domain $\Omega = S^2$ to its symmetry group as:

$$x \star \psi(\mathfrak{g}) = \langle x, \rho(\mathfrak{g})\psi \rangle = \int_{S^2} \langle x(u), \psi(\mathfrak{g}^{-1}u) \rangle_{\mathcal{C}} d\mu(u) \tag{6.22}$$

Where the group convolution outputs a signal on $SO(3)$.
Similarly, for subsequent layers the operation will convolve signals of the group to signals in the

group space. The input and filters change to:

$$x : SO(3) \to \mathbb{R}^{\mathcal{C}} \quad \psi : SO(3) \to \mathbb{R}^{\mathcal{C}} \tag{6.23}$$

The domain of such operation is $\Omega = SO(3)$, and a convolution appears as:

$$x \star \psi(\mathfrak{g}) = \langle x, \rho(\mathfrak{g})\psi \rangle = \int_{SO(3)} \langle x(\mathfrak{h}), \psi(\mathfrak{g}^{-1}\mathfrak{h}) \rangle_{\mathcal{C}} d\mu(\mathfrak{h}) \tag{6.24}$$

To explore further these concepts, it is possible to take up spectral convolutions [CGW19; Est+18]. Among other types of spherical CNNs, there are many that ignore the current approach [Est+18; Per+19; KLT18].

- Spherical CNNs with isotropic filters
  - works on 2D sphere instead of 3D rotation group
  - requires isotropic filters
- Deepsphere
  - uses HEALpic grid, with much more evenly spaced sampling of points
  - connect nearby pixels to form a graph and apply a graph convolution (somehow similar of isotropic filters
  - in the limit of dense sampling it is rotation equivariant
- Spectral Spherical CNNs
  - work entirely on spectral (Fourier) domain

It is important to point out that the isotropic filters requirement imposes rotation invariance to satisfy equivariance. Later in the chapter this concept will be expanded.

6.2 General Theory of Homogeneous Group-CNNs

From the GDL blueprint of Definition 3.43 the domain of work is a possibly structured set Ω, with group actions acting as $\mathfrak{G} \times \Omega \to \Omega$. This framework can be enriched with a useful property.

Definition 6.9 (Homogeneous Space)**.** Given a set Ω and a symmetry group $\mathfrak{G}$ the set is a homogeneous space with respect to the group if the group action is transitive on it. Namely:

$$\forall u, v \in \Omega \exists \mathfrak{g} \in \mathfrak{G} \mid \mathfrak{g}u = v \implies \Omega \; Homogeneous \tag{6.25}$$

Example 6.10 (Homogeneous Spaces in practice)**.** We propose the following intuitive examples:

- A plane $\Omega = \mathbb{R}^2$ together with the translation group $\mathfrak{G}$ is a homogeneous space. A general translation for a pair of elements is shown in Figure 6.11
- A sphere $\Omega = S^2$ paired with the 3D rotation group $SO(3)$ is a homogeneous space. See Figure 6.12 for an example.
- A plane $\Omega = \mathbb{R}^2$ and the 2D rotation group $SO(2)$ do not form a homogeneous space as we can take elements belonging to two different orbits like those of Figure 6.13. However, $SO(2)$ splits $\mathbb{R}^2$ into different homogeneous spaces.

Figure 6.11: Plane and Translation form a Homogeneous Space

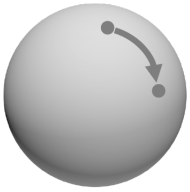


Figure 6.12: Sphere and 3D rotations form a Homogeneous Space

The guaranteed existance of a transformation for each pair of elements of the structure suggests distinguishing further between pairs that only transform and those that preserve some regularity. A stabilizer subgroup collects transformations that do not effectively change the elements considered.

Definition 6.11 (Stabilizer Subgroup $\mathfrak{H}$)**.** For a group $\mathfrak{G}$ and a set Ω homogeneous with respect to the group, a stabilizer subgroup of an element of the domain u is the set of transformations that preserve the element. Namely:

$$\mathfrak{H}_u \coloneqq \left\{ \mathfrak{g} \in \mathfrak{G} \mid \mathfrak{g}u = u \right\} \quad u \in \Omega \tag{6.26}$$

Theorem 6.12 (Stabilizer Subgroup Properties)**.** *For a Stabilizer subgroup it holds that:*

1. $\mathfrak{H}_u$ *is a subgroup*
2. $\mathfrak{H}_u$ *is the same abstract group* $\forall u \in \Omega$

Proof. **(Claim 1)** Having established by construction that $\mathfrak{H}_u \subseteq \mathfrak{G} \forall u \in \Omega$, since it is the subset of a group, it suffices to show that

$$\mathfrak{g}^{-1}\mathfrak{h} \in \mathfrak{H}_u \forall \mathfrak{g}, \mathfrak{h} \in \mathfrak{H}_u$$

We solve the claim by checking that $\mathfrak{g}^{-1}\mathfrak{h}$ satisfies Definition 6.11. It holds that:

$$(\mathfrak{g}^{-1}\mathfrak{h})(u) = \mathfrak{g}^{-1}(\mathfrak{h}(u)) = \mathfrak{g}^{-1}u = u \implies \mathfrak{g}^{-1}\mathfrak{h} \in \mathfrak{H}_u \forall u \in \Omega$$

Figure 6.13: Plane and 2D rotations do not form a Homogeneous Space

Where the last passage follows by multiplying the equatino of Definition 6.11 by $\mathfrak{g}^{-1}$.
(Claim 2**) TODO** □

A further characterization of groups is carried out by introducing Cosets.

Definition 6.13 (Left Coset $\mathfrak{gH}$)**.** Given a group $\mathfrak{G}$ and a subgroup $\mathfrak{H}$ a coset of an element $\mathfrak{g} \in \mathfrak{G}$ is the set of possible left combinations of $\mathfrak{g}$ and any element in the subgroup.

$$\mathfrak{gH} = \left\{ \mathfrak{gh} \mid \mathfrak{h} \in \mathfrak{H} \right\} \tag{6.27}$$

Where $\mathfrak{g}$ is a **coset representative**

Theorem 6.14 (Coset Properties)**.** *For a coset* $\mathfrak{gH}$*:*

1. $$\mathfrak{ghH} = \mathfrak{gH} \quad \forall \mathfrak{h} \in \mathfrak{H}, \forall \mathfrak{g} \in \mathfrak{G} \tag{6.28}$$
2. $$\mathfrak{gH} \equiv \mathfrak{g}'\mathfrak{H} \implies \exists \mathfrak{h} \in \mathfrak{H} : \mathfrak{gh} = \mathfrak{g}' \tag{6.29}$$
3. *Two cosets are either identical or disjoint, and they form a partition of* $\mathfrak{G}$*.*
$$\mathfrak{gH} \equiv \mathfrak{g}'\mathfrak{H} \ \vee \ \mathfrak{gH} \cap \mathfrak{g}'\mathfrak{H} = \emptyset \tag{6.30}$$

Proof. **(Claim** 1**)** By Definition 6.13 a requirement is $\mathfrak{H}$ being a subgroup. This implies that $\mathfrak{H}$ is closed under the group operation and:

$$\mathfrak{hh}' = \mathfrak{h}'' \in \mathfrak{H} \ \forall h, h' \in \mathfrak{H} \implies \mathfrak{ghH} = \left\{ \mathfrak{ghh}' \mid \mathfrak{h}' \in \mathfrak{H} \right\} = \left\{ \mathfrak{gh}'' \mid \mathfrak{h}'' \in \mathfrak{H} \right\}$$

Which is equal to Definition 6.13 up to the dummy index on h''.
(Claim 2**)** We start from unravelling the assumption as:

$$\mathfrak{gH} = \left\{ \mathfrak{gh} \mid \mathfrak{h} \in \mathfrak{H} \right\} \equiv \left\{ \mathfrak{g}'\mathfrak{h} \mid \mathfrak{h} \in \mathfrak{H} \right\} = \mathfrak{g}'\mathfrak{H} \tag{6.31}$$

Then $\implies \forall \mathfrak{q} = \mathfrak{gh} \in \mathfrak{gH} \exists \mathfrak{h}' \in \mathfrak{H} : \mathfrak{g}'\mathfrak{h}' = \mathfrak{q}$ and viceversa. So:

$$\mathfrak{gh} = \mathfrak{q} = \mathfrak{g}'\mathfrak{h}' \implies \mathfrak{ghh}'^{-1} = \mathfrak{gh}'' = \mathfrak{g}' \quad for \ \mathfrak{hh}' = \mathfrak{h}'' \in \mathfrak{H}$$

Where in it was sufficient to left transform by $\mathfrak{h}'^{-1}$ and recognize that $\mathfrak{hh}'^{-1} \in \mathfrak{H}$. Again, up to the dummy index, the claim is verified.
(Claim 3**)** First observe that $\forall \mathfrak{g} \in \mathfrak{G} \exists \mathfrak{gH}$, so each element of the group belong to a coset and we have that:

$$\bigcup_{\mathfrak{g} \in \mathfrak{G}} \mathfrak{gH} = \mathfrak{G}$$

To prove that cosets are disjoint, we assume by contradiction that two sets are distinct but not disjoint, namely:

$$\mathfrak{gH} \not\equiv \mathfrak{g}'\mathfrak{H} \ \wedge \ \mathfrak{gH} \cap \mathfrak{g}'\mathfrak{H} \neq \emptyset \tag{6.32}$$

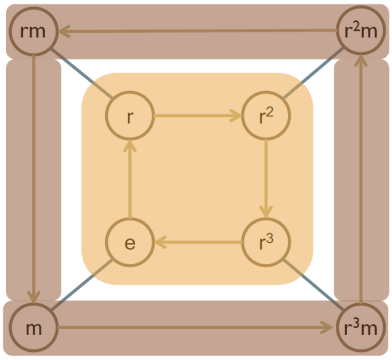


Figure 6.14: Cosets for $\mathfrak{H} = \{e, r, r^2, r^3\}$

Then $\exists \mathfrak{q}^* \in \mathfrak{g}\mathfrak{H} \cap \mathfrak{g}'\mathfrak{H}$ and we have that:

$$\implies \mathfrak{q}^* = \mathfrak{g}\mathfrak{h} = \mathfrak{g}'\mathfrak{h}' \implies \mathfrak{g}' = \mathfrak{g}\mathfrak{h}\mathfrak{h}'^{-1} = \mathfrak{g}\mathfrak{h}'' \quad \mathfrak{h}'' \in \mathfrak{H}$$

However, by Claim 1 we also have that:

$$\mathfrak{g}' \in \mathfrak{g}\mathfrak{H} \implies \mathfrak{g}'\mathfrak{H} \equiv \mathfrak{g}\mathfrak{H} \tag{6.33}$$

Which contradicts the distinction assumption $\mathfrak{g}\mathfrak{H} \not\equiv \mathfrak{g}'\mathfrak{H}$ and certifies the second partition requirement. All cosets are distinct and they cover the whole group. □

Definition 6.15 (Quotient of subgroup $\mathfrak{G}/\mathfrak{H}$)**.** For a group $\mathfrak{G}$ and a subgroup $\mathfrak{H}$ the quotient $\mathfrak{G}/\mathfrak{H}$ is the set of cosets that partition $\mathfrak{G}$.

Example 6.16 (Cosets Visualized for discrete roto-translations)**.** Consider the rotation subgroup, using Definition 6.13 and Theorem 6.14 we can analyze its cosets. Inspecting Definition 6.15 it is possible to visualize how the space is split into Quotients.

- Let $\mathfrak{H} = \{e, r, r^2, r^3\}$ be a subgroup, the cosets are denoted in Figure 6.14. Furthermore we have that:
  - The coset $e\mathfrak{H}$ is yellow
  - The coset $r\mathfrak{H}$ is brown
  - The coset $m\mathfrak{H}$ is brown
  - The quotient is $\mathfrak{G}/\mathfrak{H} = \{e\mathfrak{H}, r\mathfrak{H}\}$ shown in Figure 6.15
- If we instead consider as subgroup $\mathfrak{H} = \{e, m\}$ we can visualize the partition in Figure 6.16. Moreover, the quotient space $\mathfrak{G}/\mathfrak{H} = \{e\mathfrak{H}, r\mathfrak{H}, r^2\mathfrak{H}, r^3\mathfrak{H}\}$ is that of Figure 6.17

Example 6.17 (Cosets visualized for a Sphere and surface rotations)**.** Consider a group $\mathfrak{G} = SO(3)$ of triple axis rotations, and its subgroup $SO(2) = \mathfrak{H} = \{Z(\gamma) \mid \gamma \in [0, 2\pi]\}$ of Z axis rotations. The quotient $SO(3)/SO(2) = S^2$ a sphere parametrized by $\alpha \in [0, 2\pi], \beta \in [0, \pi]$. A figure of such kind is Figure 6.18

The final object to introduce to trace back our reasoning to Homogeneous spaces relates quotients and group elements in a specific manner.

Definition 6.18 (Action of $\mathfrak{G}$ on $\mathfrak{G}/\mathfrak{H}$)**.** For $\mathfrak{g} \in \mathfrak{G}$, $\mathfrak{H}$ a subgroup of $\mathfrak{G}$ aand $u = \mathfrak{a}\mathfrak{H} \in \mathfrak{G}/\mathfrak{H}$ (i.e. a coset), define an action of the group on the coset as:

$$(\mathfrak{g}, u) \to \mathfrak{g}(\mathfrak{a}\mathfrak{H}) = (\mathfrak{g}\mathfrak{a})\mathfrak{H} \tag{6.34}$$

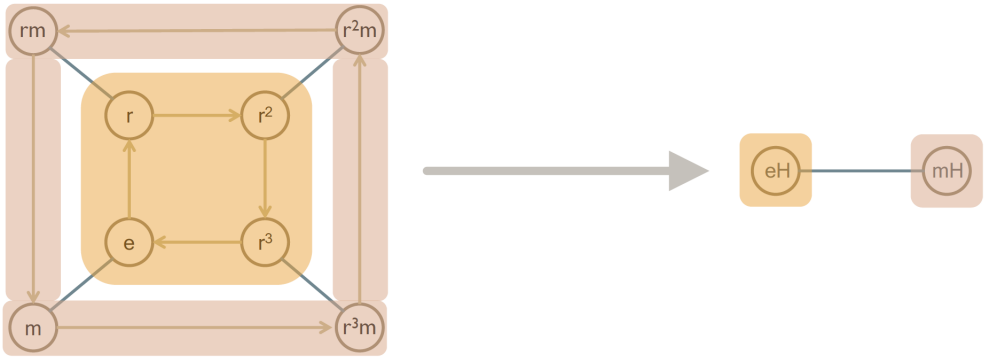


Figure 6.15: Quotients for $\mathfrak{H} = \{e, r, r^2, r^3\}$

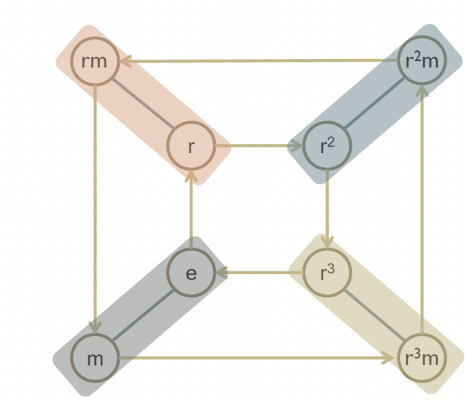


Figure 6.16: Cosets for $\mathfrak{H} = \{e, m\}$

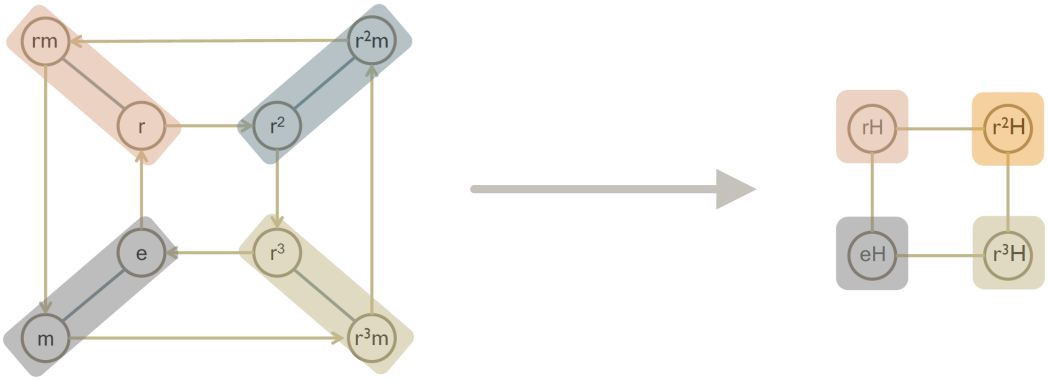


Figure 6.17: Quotient for $\mathfrak{H} = \{e, m\}$

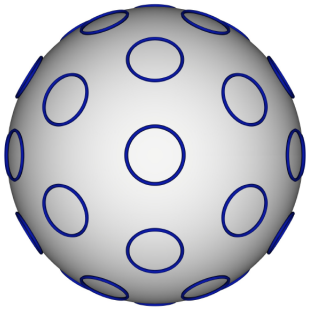


Figure 6.18: $SO(3)/SO(2)$ quotient is a sphere

Proposition 6.19 (Action on Quotient is well defined)**.**

$$\mathfrak{a}\mathfrak{H} = \mathfrak{a}'\mathfrak{H} \implies \mathfrak{g}(\mathfrak{a}\mathfrak{H}) = \mathfrak{g}(\mathfrak{a}'\mathfrak{H}) \quad \forall \mathfrak{g} \in \mathfrak{G}, \mathfrak{a}\mathfrak{H}, \mathfrak{a}'\mathfrak{H} \in \mathfrak{G}/\mathfrak{H} \tag{6.35}$$

Proof. By Theorem 6.14 cosets partition $\mathfrak{G}$. Then, if two cosets are equal, the map $\mathfrak{g}u = \mathfrak{g}v$ is the same for $u = \mathfrak{a}\mathfrak{H} = \mathfrak{a}'\mathfrak{H} = v$. □

As it turns out, Homogeneous spaces are nothing but Quotient spaces, as they partition the space into orbits that are disjoint and cover the whole group. The proof is carried out thanks to the next Lemma.

Lemma 6.20 (Quotient space is Homogeneous and its stabilizer group)**.** *Consider a group* $\mathfrak{G}$ *with a subgroup* $\mathfrak{H}$*. Its Quotient space* $\mathfrak{G}/\mathfrak{H}$ *is homogeneous with a stabilizer* $\mathfrak{H}_u \forall u \in \mathfrak{G}/\mathfrak{H}$ *.*

Proof. To show this, consider $u = \mathfrak{a}\mathfrak{H}, v = \mathfrak{b}\mathfrak{H} \in \mathfrak{G}/\mathfrak{H}$. Using Definition 6.9, it suffices to find a group element $\mathfrak{l} \in \mathfrak{G}$ that *connects* the two. This is easily found to be:

$$\mathfrak{l} = \mathfrak{b}\mathfrak{a}^{-1} \implies \mathfrak{l}u = \mathfrak{b}\mathfrak{a}^{-1}\mathfrak{a}\mathfrak{H} = \mathfrak{b}\mathfrak{H} = v \implies \mathfrak{G}/\mathfrak{H}\ Homogeneous$$

To find its stabilizer subgroup, following Definition 6.11, we aim to identify a set of transformations that keeps the element unchanged:

$$find\ \mathfrak{H}_\mathfrak{u} = \{\mathfrak{l} \in \mathfrak{G} \mid \mathfrak{l}u = u\}$$

When $u = \mathfrak{a}\mathfrak{H}$, this condition is found to be:

$$\mathfrak{l}u = \mathfrak{l}\mathfrak{a}\mathfrak{H} = \left\{\mathfrak{l}\mathfrak{a}\mathfrak{h} \mid \mathfrak{h} \in \mathfrak{H}\right\} = \left\{\mathfrak{a}\mathfrak{h} \mid \mathfrak{h} \in \mathfrak{H}\right\} = \mathfrak{a}\mathfrak{H} = u \tag{6.36}$$

If $u = e\mathfrak{H}$ then the stabilizer condition is trivially satisfied by:

$$\mathfrak{h} \in \mathfrak{H} \implies \mathfrak{h}(e\mathfrak{H}) = \mathfrak{h}\mathfrak{H} = \mathfrak{H} = e\mathfrak{H}$$

And $\mathfrak{H}$ is its stabilizer group.
For $u \neq e\mathfrak{H}$ we have that:

$$\mathfrak{l}\mathfrak{a}\mathfrak{H} = \mathfrak{a}\mathfrak{H} \iff \mathfrak{l} \in \mathfrak{a}\mathfrak{H}$$
$$\implies \mathfrak{H}_u = \mathfrak{a}\mathfrak{H}$$

Since we have to make it compose with itself to keep it closed under composition. □

Theorem 6.21 (Orbit Stabilizer Theorem)**.** *For a group $\mathfrak{G}$ and subgroup $\mathfrak{H}$ there is a bijection between the cosets of the stabilizer subgroup for u $\mathfrak{H}_u$ and the orbits of O_u as per Definition 3.27, which are the elements in which u can be transformed into through $\mathfrak{G}$. Namely, there exists a bijection τ such that:*

$$\exists \tau_u : \mathfrak{G}/\mathfrak{H}_u \to O_u \quad \tau(u) \to O_u \quad where \quad \begin{cases} u = \mathfrak{a}\mathfrak{H} \\ O_u = \mathfrak{a}u \end{cases} \tag{6.37}$$

$$\implies \mathfrak{G}/\mathfrak{H}_\mathfrak{u} \cong O_u \tag{6.38}$$

Proof. Consider for a fixed u the function

$$f : \mathfrak{G} \to \mathcal{X}(\Omega, \mathcal{C}) \quad \mathfrak{a} \to \mathfrak{a}u$$

Then the image of f is the orbit of u, denoted as $Im(f) = O_u$. Two group elements map to the same output when:

$$f(\mathfrak{a}) = f(\mathfrak{h}) \iff \mathfrak{a}u = \mathfrak{h}u \iff \mathfrak{a}^{-1}\mathfrak{h}u = u \iff \mathfrak{a}^{-1}\mathfrak{h} \in \mathfrak{H}_u \iff \mathfrak{h} \in \mathfrak{a}\mathfrak{H}_u$$

Which means that this happens if and only if $\mathfrak{h}$ is in the coset of the stabilizer subgroup of u. The $\iff$ relation implies that f induces a bijection between the set of cosets $\mathfrak{G}/\mathfrak{H}_u$ and the orbits of the u elements which are the image of f. Thus:

$$\implies \mathfrak{G}/\mathfrak{H}_u \cong O_u \tag{6.39}$$

TODO check □

Example 6.22 (Sets & Sequences)**.** Let $\Omega = \{1, \dots, n\}$ and $\mathfrak{G} = S_n$ the permutation group. The pair (Ω, S_n) is homogeneous since for each $u, v \in \Omega$ there is a group transformation (more than one actually) such that $\mathfrak{g}u = v$, just consider a bijection that sends $u \to v$. By Theorem 6.21 it is equivalent to a quotient space. The stabilizer subgroup for $1 \in \Omega$ denoted as $\mathfrak{H}_1$ is such that:

$$\mathfrak{H}_1 = \{\mathfrak{g} \in S_n \mid \mathfrak{g}1 = 1\} = \{\mathfrak{g} \in S_n \mid 1 \; fixed\}$$

And it is the set of permutations that do not impact 1 and permute the others:

$$\mathfrak{g} : 1 \to 1 \quad i \to j, \; i, j \neq 1 \tag{6.40}$$

The isomoprhism $S_n/\mathfrak{H}_1 \cong O_1 = \Omega$ implies that:

- $|S_n/\mathfrak{H}_1| = |\Omega| = n$
- The action $S_n \times S_n/\mathfrak{H}_1$ is equivalent to the action $S_n \times \Omega$ so they are both permutations!

Another useful bijection following Theorem 6.21 is between functions. Consider $f : \mathfrak{G} \to \mathcal{C}$ be a function which is right $\mathfrak{H}$ invariant, where $\mathfrak{H}$ is a subgroup of $\mathfrak{G}$ meaning:

$$f(\mathfrak{g}\mathfrak{h}) = f(\mathfrak{g}) \quad \forall \mathfrak{g} \in \mathfrak{G}, \forall \mathfrak{h} \in \mathfrak{H} \tag{6.41}$$

Then, by the isomorphism property, the function f is essentially equivalent to another function $\overline{f}$ on the quotient space:

$$\overline{f} : \mathfrak{G}/\mathfrak{H} \to \mathcal{C} \quad f(\mathfrak{g}) = \overline{f}(\mathfrak{g}\mathfrak{H}) \tag{6.42}$$

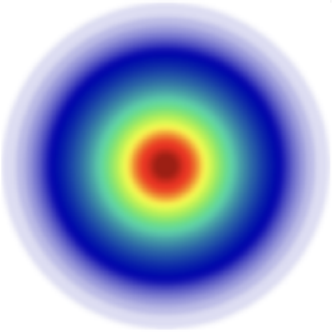


Figure 6.19: Isotropic $SO(2)$ invariant filter

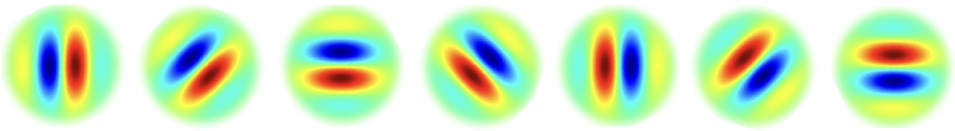


Figure 6.20: Unconstrained filters

Example 6.23 (Functions on rotation groups, functions on spheres)**.** Coming back to figure 6.18, we notice that a function on $SO(3)$ which is right $SO(2)$ invariant means that it is constant across 2D rotations. By the above argument, it will be equivalent to a function on the sphere $S^2 = SO(3)/SO(2)$, which is the quotient.

We now want to analyze spherical convolutions of the form:

$$\star : \mathcal{X}(S^2) \times \mathcal{X}(S^2) \to \mathcal{X}(S^2) \tag{6.43}$$

Which again, by Theorem 6.21 is equivalent to convolutions of the form:

$$\star : \mathcal{X}(S^2) \times \mathcal{X}(S^2) \to \mathcal{X}(SO(3)) : SO(2) \textit{ invariant} \tag{6.44}$$

Since an $SO(2)$ invariant function on $SO(3)$ is equivalent to a sphere. The same concept generalizes to any homogeneous space. To achieve this, it turns out that the filter needs to be $SO(2)$ symmetric (**isotropic**), like those of Figure 6.19. An isotropic filter is equivalent to a scalar field on $\mathfrak{G}/\mathfrak{H}$ On the contrary, it is possible to work with Regular fields on $\mathfrak{G}/\mathfrak{H}$, which are scalar on $\mathfrak{G}$ as unconstrained filters. To have an idea of what these look like, it is possible to glance at Figure 6.20. These however come at the price of increased computing overhead, resulting in potential inefficiencies. Between these two types of filters, we consider constrained filters as fields on $\mathfrak{G}/\mathfrak{H}$ which transform according to *induced representations* (not the regular one!).

Observation 6.24 (From regular to induced representation)**.** *For a regular representation of $C4$ cyclic shifts of 4 elements, it is possible to visualize an example in Figure 6.21. To go from this representation of $C4$, the subgroup $\mathfrak{H}$ which is a stabilizer, to a representation of the whole roto-translation group, it is possible to combine the rotation and translation action on the quotient space $\mathfrak{G}/\mathfrak{H}$ with the cyclic shift as in Figure 6.22.*

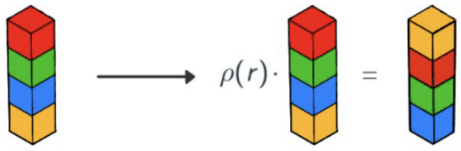


Figure 6.21: Regular representation of $C4$

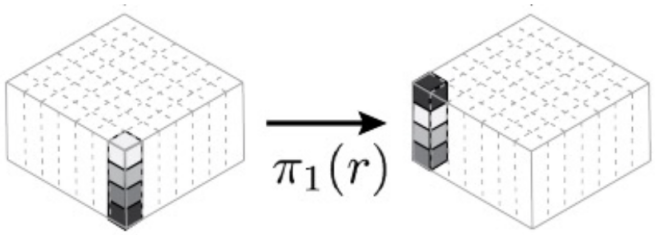


Figure 6.22: Cyclic and roto translation representation

For an RGB image, each pixel transforms trivially, but the specific channels do not. For the induced representation there will be a spatial action (e.g. rotation) and a trivial action on the channels. See Figure 6.23 for an intuition.

For an RGB image where we wish to compute the gradient, the transformation is not scalar anymore and can be seen as an operation on a vector field. The channels will undergo a rotation to compensate for the change in orientation. Figure 6.24 is an example of such situation. To visualize this, consider a vector field of the form $f(x)$ and a 90 degree rotation with $\mathfrak{g}$. If we wish to apply it, it is not enough to move each vector to the rotated position while leaving the direction of the vectors unchanged as $f(\mathfrak{g}^{-1}x)$. We also need to modify the direction, and do the following:

$$\rho(\mathfrak{g})f(\mathfrak{g}^{-1}x)$$

An visualization of the three functions $f(x), f(\mathfrak{g}^{-1}x), \rho(\mathfrak{g})f(\mathfrak{g}^{-1}x)$ is proposed in Figure 6.25.

Informally, we could describe Induced representations in general considering:

- $\mathfrak{G}$ a group and $\mathfrak{H}$ a subgroup, both *sufficiently nice*
- ρ a representation of $\mathfrak{H}$

Which are enough to build an induced by ρ representation of $\pi \in \mathfrak{G}$. This object describes how a field of ρ features transforms on $\mathfrak{G}/\mathfrak{H}$. An outline of the steps could be:

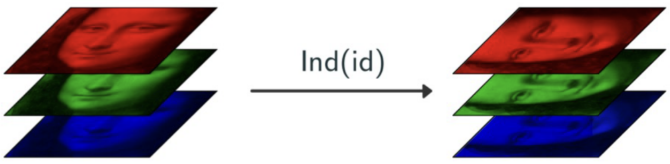


Figure 6.23: Scalar field RGB induced representation

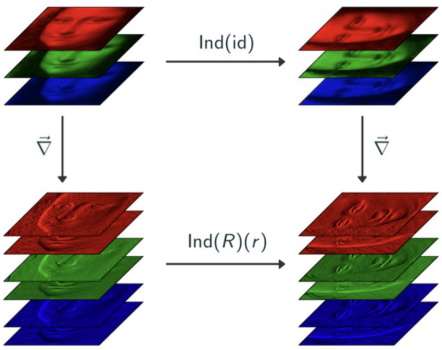


Figure 6.24: Vector Field RGB induced representation

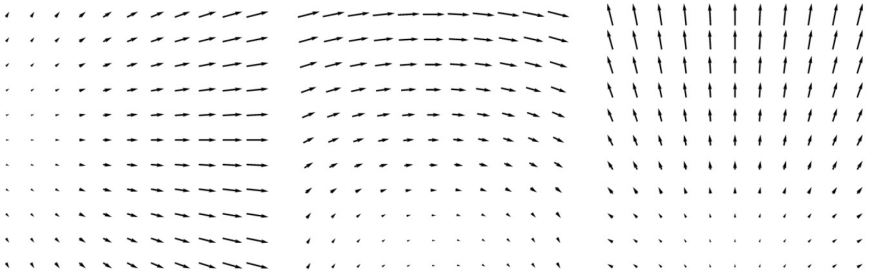


Figure 6.25: Vector Field and transformations

- move features in $\mathfrak{G}/\mathfrak{H}$ using the action on $\mathfrak{G}$
- transform them with ρ

6.3 Steerable CNNs

Neural networks that take as input feature maps that transform according to an induced representation (fields) and output the same are called Steerable CNNs. Also in this context, we find a Universality Theorem, now stated layerwise. For a more formal treatment, there are many references [KT18; CGW19; Coh21; Aro21].

Theorem 6.25 (Convolution is all you need II, informal)**.** *Any linear equivariant map between induced representations is a convolution with steerable kernels.*

Example 6.26 (Steerable Kernel and convolutions)**.** Let $\mathfrak{G} = SE(2), \mathfrak{H}_1 = \mathfrak{H}_2 = \mathfrak{H} = SO(2)$ such that: $\mathfrak{G}/\mathfrak{H} = \mathbb{R}^2$ and consider the feature field f and filter ψ:

$$f : \mathbb{R}^2 \to \mathbb{R}^{\mathcal{C}} \quad \psi : \mathbb{R}^2 \to \mathbb{R}^{K \times \mathcal{C}}$$

Then a convolution will be:

$$(f \star \psi)(u) = \int_{\mathbb{R}^2} \sum_c f_c(v)\psi_{kc}(v-u)dv$$

Where the kernel satisfies:

$$\psi(r^{-1}u) = \rho_2(r)^{-1}\psi(u)\rho_1(r)$$

G	H	G/H	ρ	Reference
$\mathbb{Z}^2$	$\{1\}$	$\mathbb{Z}^2$	regular	LeCun et al. [1990]
$p4, p4m$	C_4, D_4	$\mathbb{Z}^2$	regular	Cohen and Welling [2016],
"	"	"	"	Dieleman et al. [2016]
$p4, p4m$	C_4, D_4	$\mathbb{Z}^2$	irrep & regular	Cohen and Welling [2017]
$p6, p6m$	C_6, D_6	$\mathbb{H}^2$	regular	Hoogeboom et al. [2018]
$\mathbb{Z}^3 \rtimes H$	D_4, D_{4h}, O, O_h	$\mathbb{Z}^3$	regular	Winkels and Cohen [2018]
$\mathbb{Z}^3 \rtimes H$	V, T_4, O	$\mathbb{Z}^3$	regular	Worrall and Brostow [2018]
SE(2)	SO(2)	$\mathbb{R}^2$	regular	Weiler et al. [2018]
"	"	"	"	Zhou et al. [2017]
"	"	"	"	Bekkers et al. [2018]
SE(2)	SO(2)	$\mathbb{R}^2$	irrep	Worrall et al. [2017]
"	"	"	"	Marcos et al. [2017]
SE(3)	SO(3)	$\mathbb{R}^3$	irrep	Kondor [2018]
"	"	"	"	Thomas et al. [2018]
SO(3)	SO(2)	S^2	regular	Cohen et al. [2018]
"			trivial	Esteves et al. [2018]
$\mathbb{S}_n$	$\mathbb{S}_k \times \mathbb{S}_{n-k}$	$\mathbb{S}_n \,/\, \mathbb{S}_k \times \mathbb{S}_{n-k}$	trivial	Kondor and Trivedi [2018]

Figure 6.26: Homogeneous G-CNNs characterization

Which enforces that if we rotate the filter it must be the same as outer rotating the representations for the input and output layers.

This framework can be used to classify different G-CNNs by the group $\mathfrak{G}$, the subgroup $\mathfrak{H}$ which determines the space we are working, and what type of features ρ we are using (i.e. how they transform). A summary of methods interpretable in this uniform approach of Homogeneous G-CNNs is proposed in Figure 6.26.

To explore Tensor Field Nets, we reroute the reader to interesting sources [CW16; Wor+17; Wei+18; Tho+18; Kon18; Hy+18].

Chapter 7

Geodesics & Manifolds

This chapter is devoted to analyzing another G of the GDL blueprint that is missing: Geodesics. The domain Ω will be a Manifold, a nice mathematical object, not well explored in Computer Science.

Manifolds represent Intrinsic/Mesh CNNs and have 2 symmetry groups, Isometries $Iso(\Omega)$ and gauge symmetries $SO(s)$.

Why such a choice of domain? Manifolds are clever ways to represent 3D objects for two main reasons:

1. they are efficient as they can be used to model the 2D surface ignoring the third dimension of volume. This is useful especially when what happens in the interior of the object is not important for the model. Figure 7.1 is an example of surface vs volume complexity.
2. they are a natural representation of deformable shapes, widely implemented in $3D$ graphics and protein modelling. In these two fields, it is noticeable that the internal structure can be ignored.

As an outline of the lecture, we will explore:

- Convolutions on manifolds
- Deformation invariance
- Manifold Fourier transform
- Discretization

The concepts touched will make the reader able to

- define local *stuff* on manifolds
- move local *stuff* around
- understand global symmetry and deformation invariance as an isometry
- perform spectral analysis on manifolds

In the next chapter, the local symmetry of manifolds, formalized by gauges, will be explored.

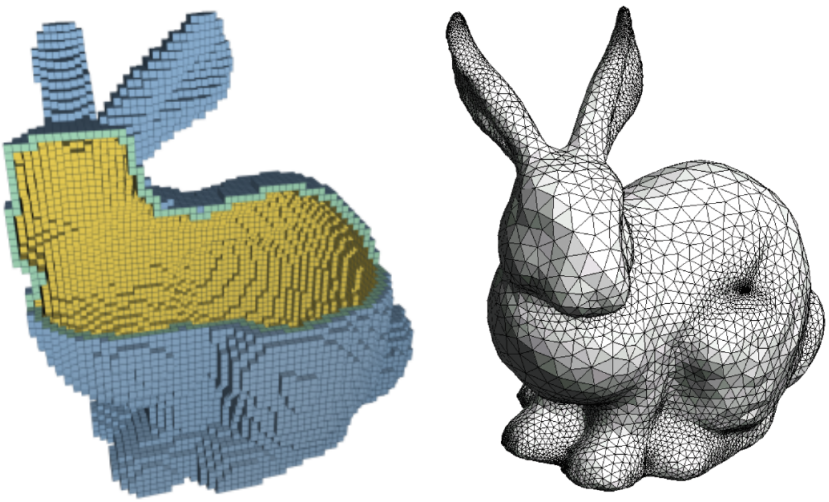


Figure 7.1: Rabbit as a volume and as a (mesh) surface

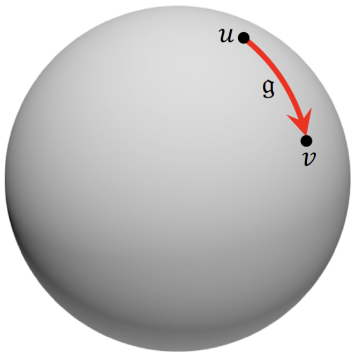


Figure 7.2: Global Symmetry group exists

7.1 A primer on Manifolds

When talking about Groups in Chapter 6, most of the analysis was based on the homogeneity property of some structures. Manifolds instead have no global symmetry, and can only be locally symmetric. For this reason, concepts related to paths are introduced.

Example 7.1 (Groups vs Manifolds)**.** In Figure 7.2 we have a global symmetry group that ensures homogeneity. In Figure 7.3 there is no global symmetry, and many types of different paths can be found for a pair of points of the domain.

Thinking about Euclidean convolutions (an example is Figure 7.4, the filter is transported around the domain with no path dependence.

On the contrary, if a filter is moved across a non Euclidean domain, the path impacts how the filter is transformed. For an s sphere rotation changes the filter orientation (the arrow) as in Figure 7.5. The ambiguity is **reflection**. For a non orientable manifold as that of Figure 7.6, the situation becomes even more complicated.

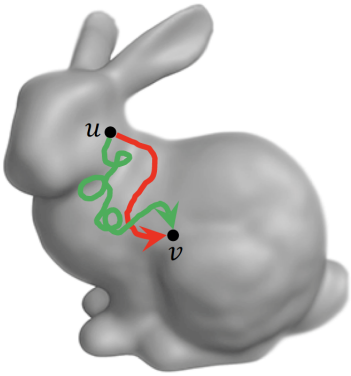


Figure 7.3: No Global Symmetry group

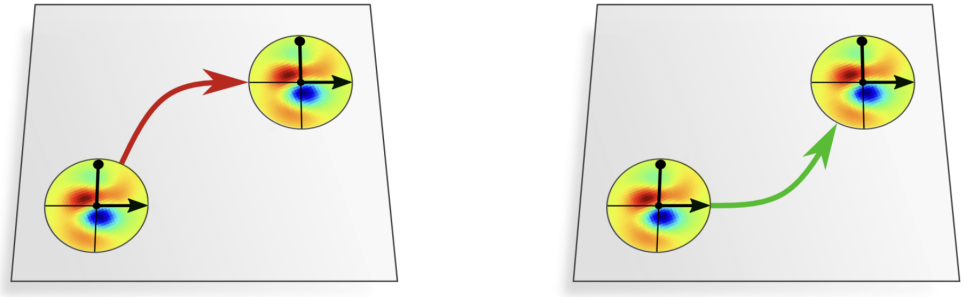


Figure 7.4: Euclidean Convolution

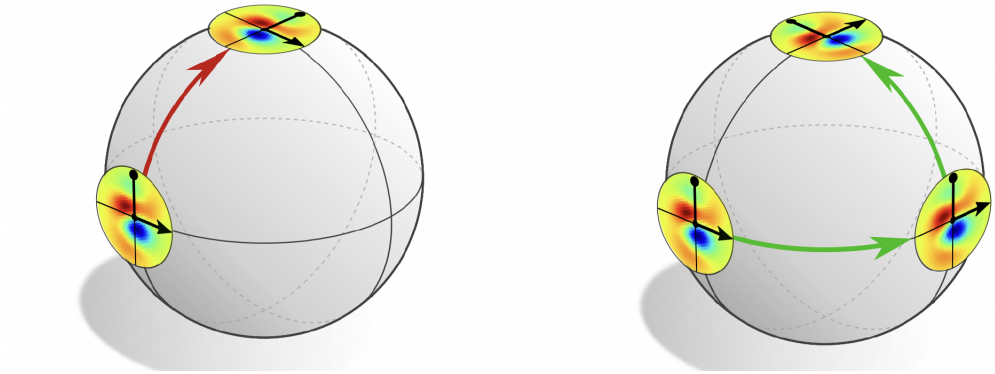


Figure 7.5: Non-Euclidean Convolution

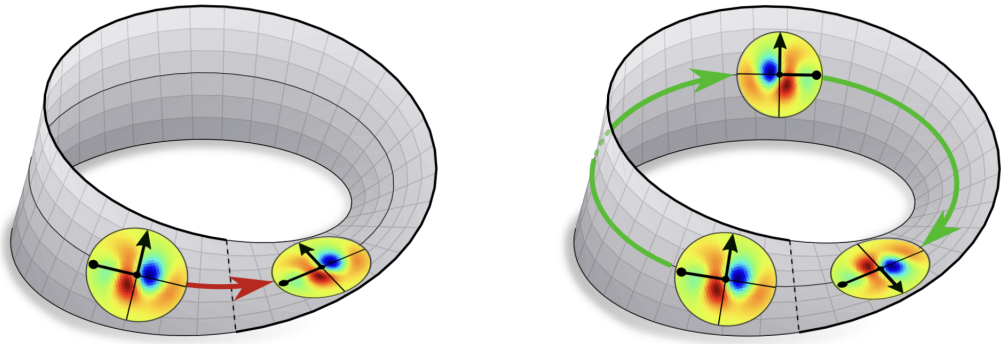


Figure 7.6: Non-Euclidean Convolution II

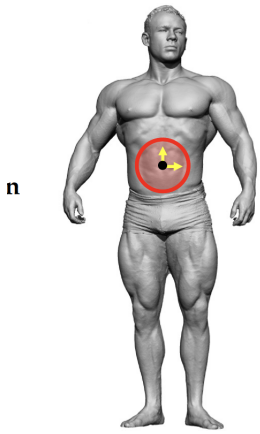


Figure 7.7: Local gauge transformation

There are many ways to tackle non Euclidean convolutions, among those we will avoid the fixed path approach and focus on:

- fixed gauge
- group pooling
- isotropic spectral filter
- gauge equivariant filter

For the types of symmetry, we can interpret how they act on the structure through visual examples:

- Local gauge transformation $SO(s)$ (Chapter 8), Figure 7.7
- global isometric deformation through invariance $Iso(\Omega)$, Figure 7.8

For the purpose of understanding better the requirements of a Manifold, we report the bare minimum of theoretical statements.

Definition 7.2 (Topological Space via Neighborhoods)**.** Let Ω be a set. Ω is a topological space if it is paired with a valid neighborhood function, where the requirements are outlined below. Let $\mathcal{N}$ be a neighborhood function such that for each element it returns a subset of Ω which is

Figure 7.8: Global deformation

the set of its neighbors:

$$\mathcal{N} : \omega \to \{Pow(\Omega)\} \quad \mathcal{N}(\omega) = \{N_1, \ldots\} : N_i \subseteq \Omega \forall i \tag{7.1}$$

Where further $\mathcal{N}$ satisfies the axioms of neighborhood topology $\forall \omega$ elements:

- element is always a neighbor in each neighborhood

$$N \in \mathcal{N}(\omega) \implies \omega \in N \tag{7.2}$$

- every superset of a neighborhood is itself a neighborhood

$$N \subset \Omega \wedge N^* \subseteq N \; N^* \in \mathcal{N}(\omega) \implies N \in \mathcal{N}(\omega) \tag{7.3}$$

- the intersection of two neighborhoods is a neighborhood

$$N, N' \in \mathcal{N}(\omega) \implies (N \cap N') \in \mathcal{N}(\omega) \tag{7.4}$$

- Neighborhood matriosk

$$N \in \mathcal{N}(\omega) \implies \exists N' \in \mathcal{N}(\omega) \mid \forall \omega' \in N' \; N \in \mathcal{N}(\omega') \tag{7.5}$$

Definition 7.3 (Homeomorphism)**.** A function $f : \Xi \to \Omega$ where Ω, Ξ are topological spaces is a homeomorphism if:

- f is a bijection
- f is continuous
- f^{-1} is continuous

As mentioned in earlier paragraphs, manifolds are general objects used to model surfaces. Informally, an s dimensional manifold is a Topological Space (Definition 7.2) with a neighborhood and no distance notion, which locally resembles $\mathbb{R}^s$. This resemblance is translated into being *locally homeomorphic* (Definition 7.3) to $\mathbb{R}^s$. Additionally, if it is *smooth*, then its transition function between the intersection of two local maps is smooth.

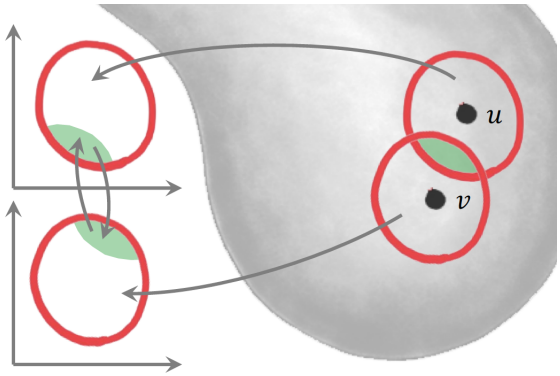


Figure 7.9: A smooth manifold

Example 7.4 (Smooth Manifold)**.** In Figure 7.9 it is possible to notice that the smoothness requirement reduces to having a smooth transition (mapping, i.e. the arrows) between the intersection of the two locally homeomorphic spaces is smooth.

Despite the absence of constraints, many highly general objects can be analyzed when dealing with manifolds. Those that will be useful for our purpose are introduced in the next pages.

For a smooth manifold Ω, we can build for any point $u \in \Omega$ a **tangent space**, as the s dimensional space which is *tangent* to Ω at u. It can be proved that:

$$T_u\Omega \cong \mathbb{R}^s \tag{7.6}$$

We denote the collection of all tangent spaces (the **tangent bundle**) as

$$T\Omega = \amalg_{u\in\Omega} T_u\Omega \tag{7.7}$$

Vectors in the tangent plane $X \in T_u\Omega$ define local displacements. Observe though that they are **abstract vectors**, without coordinates, given that direction and length arise when an additional inner product (distance notion) is provided.

The addition of a basis $\{X_1, \ldots, X_s\} \subseteq T_u\Omega$ allows to attach coordinates to X. A coordinatisation can be constructed by defining a **gauge** ω_u. Which is an invertible linear map smooth w.r.t. u such that:

$$\omega_u : \mathbb{R}^s \to T_u\Omega \tag{7.8}$$

Which induces coordinates $\mathbf{x} = \omega_u^{-1}(X)$ for an abstract vector X.
A visualization of a manifold in 2 or 3 dimensions is to be taken carefully. Having no notion of distance, any embedding is purely indicative, nor it should not be intended as an any dimensional object. In Figure 7.10 an example of s dimensional manifold, with a tangent space at u, and vector is proposed.

A very interesting fact, often presented as a joke, is that a mug is isomorphic to a donut. Both are then the same manifold, with just different $3D$ representations (see Figure 7.11). Topology is indeed very flexible and we may need some additional structure to exploit its potential. First of all, local notions need to be attached to Ω.

The local homeomorphism with $\mathbb{R}^s$ suggests exploiting the regularity of such space.

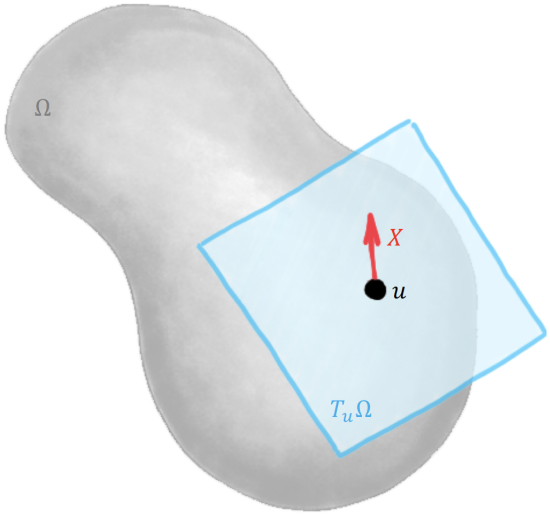


Figure 7.10: Manifold Ω, tangent plane $T_u\Omega$, tangent abstract vector X

Figure 7.11: Mug donut deformation

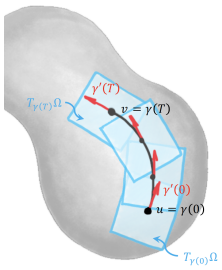


Figure 7.12: A geodesic with tangent planes

Definition 7.5 (Riemaniann Metric $g_u(\cdot,\cdot)$)**.** A Riemaniann metric $g_u(\cdot,\cdot)$ is a local inner product smooth w.r.t. u:

$$g_u(\cdot,\cdot) = \langle\cdot,\cdot\rangle_u : T_u\Omega \times T_u\Omega \to \mathbb{R} \tag{7.9}$$

Attaching a local frame to $T_u\Omega$ it can be expressed as a positive definite matrix $\mathbf{G}(u)$ storing inner products of elements of a gauge induced basis $\{X_1, \dots, X_s\} \subseteq T_u\Omega$ as:

$$\mathbf{G}(u) = \Big(\langle X_i, X_j\rangle_u\Big) \succ 0 \tag{7.10}$$

A property which solely arises from the Riemaniann Metric is said to be **intrinsic**.

Example 7.6 (Intrinsic quantities)**.** The attachment of a Riemaniann metric gives rise to the following local quantities:

- angle of vectors $\cos^{-1}(\langle X, Y\rangle_u) \quad X, Y \in T_u\Omega$
- lengths $\|X_u\| = \langle X, X\rangle_u^{\frac{1}{2}}$
- areas $g = det(\mathbf{G})$

It can be shown that a deformation that does not impact the metric (an *isometry*) will come with some properties preserved. The donut-mug case of Figure 7.11 is not of this kind, as it clearly does not satisfy the isometry requirement.

A Riemaniann metric allows us to obtain distances, up to fixation of a constraint on their shape.

Definition 7.7 (Geodesic γ)**.** A manifold Ω with a Riemaniann metric $g_u \forall u \in \Omega$ allows to define parametrized curves between points $u, v \in \Omega$. A geodesic for two points is the minimum length curve among all the possible ones, where length is computed as the integral of the velocities modules $\gamma'(t)$ defined in the tangent spaces:

$$\gamma : [0, T] \to \Omega \quad \gamma(0) = u \quad \gamma(T) = v \tag{7.11}$$

$$\gamma(t) = argmin\{\ell(\gamma)\} \tag{7.12}$$

$$\ell(\gamma) = \int_0^T \|\gamma'(t)\|_{\gamma(t)} dt = \int_0^T \langle\gamma'(t), \gamma'(t)\rangle_{\gamma(t)}^{\frac{1}{2}} dt \quad \gamma'(t) \in T_{\gamma(t)}\Omega \tag{7.13}$$

A breakdown of the creation of a geodesic is outlined in Figure 7.12, while the resulting curve alone is in Figure 7.13. Note that geodesics are **intrinsic**

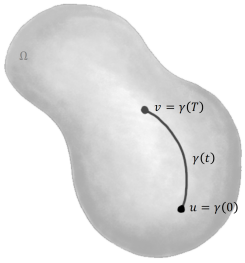


Figure 7.13: A geodesic

These objects motivate the introduction of the concepts of geodesic distance and some related results.

Lemma 7.8 (Geodesic existance for connected riemaniann manifolds)**.** *In a manifold with a riemaniann metric for any pair of points $u, v \in \Omega$ the geodesic exists.* ***TODO maybe proof***

Definition 7.9 (Complete metric space induced by geodesic distance)**.** Let

$$d_g(u,v) = min\{\ell(\gamma)\} \; s.t. \; \gamma(0) = u, \gamma(T) = v \forall u, v \in \Omega \tag{7.14}$$

Then (d_g, Ω) is a complete metric space, again **intrinsic**.

Theorem 7.10 (Hopf-Rinov Theorem)**.** *Metric completeness is equivalent to geodesic completeness* ***TODO maybe move check***

Good manifolds satisfy Theorem 7.10 and make the use of the words geodesic and metric equivalent.

Going further into the topic, a geodesic as in Definition 7.7 allows to move vectors $X \in T_u\Omega$ to $T_v\Omega$.

Definition 7.11 (Parallel (unambiguous) transport $\Gamma_{u\to v}$)**.** Given $X \in T_u\Omega$ use the geodesic:

$$\gamma : [0,T] \to \Omega \quad \gamma(0) = u, \gamma(T) = v \tag{7.15}$$

To transport X. To keep it constant across the orbit restrict $X(t) \in T_{\gamma(t)}\Omega \forall t$ to be such that:

$$\|X(t)\|_{\gamma(t)} = \|X_u\| = kost \tag{7.16}$$

$$\langle X(t), \gamma'(t) \rangle^{\frac{1}{2}}_{\gamma(t)} = \langle X_u, \gamma'(0) \rangle = kost \tag{7.17}$$

Namely constrainint the modulus and direction with respect to the geodesic to constants across the translation.
It defines a map

$$\Gamma_{u\to v} : T_u\Omega \to T_v\Omega \quad \Gamma(X) = X(T) \quad T \; torsion \tag{7.18}$$

Which amounts to a rotation of the vector and therefore is a representation of $SO(s)$.

Definition 7.11 can be recovered abstractly without assuming a Riemaniann metric through the concept of covariant derivative and a special torsion free metric compatible metric called Levi-Civita.

TODO check or move Coming back to Theorem 7.10, we give a clearer treatment of the topic.

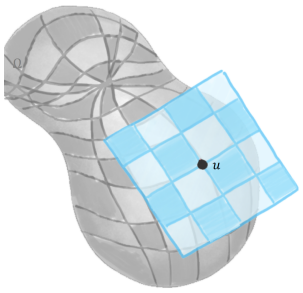


Figure 7.14: Exponential map exp_u

Definition 7.12 (Diffeomorphism)**.** Given two smooth manifolds Ω, Ξ a function $f : \Xi \to \Omega$ is a diffeomorphism if:

- f is bijective
- f is differentiable
- f^{-1} is bijective and differentiable as well

Proposition 7.13 (Diffeomophisms$\prec$ Homeomorphism)**.** *Every diffeomorphism is a homeomorphism.*

Proof. Smooth Manifolds are topological spaces as in Definition 7.2. A Diffeomorphism requires that f is differentiable, which necessarily implies that f is continuous. Thus, every diffeomorphism is also a homeomorphism. □

Consider a manifold Ω where $u \in \Omega$ and $X \in T_u\Omega$, then for the local tangent space there exists a geodesic γ_X starting from u in the X direction $\gamma'_X(0) = X$.

Definition 7.14 (Geodesically Complete Manifold)**.** If $\exists \gamma'_X(t) \forall t \geq 0, X \in T_u\Omega, u \in \Omega$ then the manifold is geodesically complete.

Definition 7.15 (Exponential Map $exp_u(\cdot)$)**.** An exponential map is the result of a unit time step from u in the direction X. While it may not be good globally, it can be localized to a radius r, where the injectivity radius is he largest radius r making the map diffeomorphic.

$$exp_u : B_r(0) \subset T_u\Omega \to \Omega \quad exp_u(X) = \gamma_X(1) \tag{7.19}$$

It is an **intrinsic** object, which naturally maps form the tangent space to the manifold. Yet geodesic completeness from Definition 7.14 does not guarantee that it is a global diffeomorphism. Again in our canonical example, we can view an exponential map in Figure 7.14.

To understand how a convolution has to be carefully defined for a manifold, we will justify why the naive approach does not have satisfying results.
Assume there is signal on the manifold $x \in \mathcal{X}(\Omega, \mathcal{C})$ and a local filter on the tangent space ψ. By construction, the filter can only be defined on the tangent space which *resembles* $\mathbb{R}^s$, and the only way to convolve the two naively is using its exponential map:

$$(x \star \psi)(u) = \int_{T_u\Omega} \psi(Y) x(exp_u(Y)) dY \tag{7.20}$$

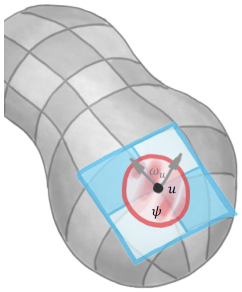


Figure 7.15: Convolution with ω_u

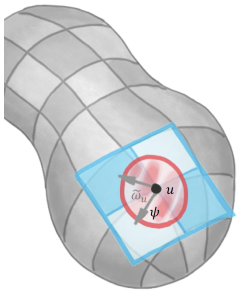


Figure 7.16: Convolution with $\widetilde{\omega_u}$

Where the vector $Y = \mathbf{v}$ lives in $\mathbb{R}^s$ and can be mapped to the tangent space through a gauge, which allows x to take up an exponentially mapped signal $exp_u(\omega_u(\mathbf{v}))$. The convolution then becomes:

$$(x \star \psi)(u) = \int_{T_u\Omega} \psi(\mathbf{v}) x[exp_u(\omega_u(\mathbf{v}))] dv \tag{7.21}$$

However, there are many gauges, and the result is dependent on which one is used. For two differently oriented gauges $\omega_u, \widetilde{\omega_u}$ the output of the convolution $(x \star \psi)_\omega(u) \neq (x \star \psi)_{\widetilde{\omega}}(u)$ will be different in Figures 7.15, 7.16. We stress that the red circle is the injectivity radius, the blue square is the exponential map and the grey arrows are the gauge orientation.

This problem was formally treated in the literature [Wei+21]. In simple words, a gauge is defined up to a gauge transformation $\mathfrak{g} : \Omega \to \mathfrak{G}$ and depends on the assumable structure. This transformation is thought of as the structure group of the manifold or the tangent bundle.

Example 7.16 (Gauge transormations)**.** Consider Figure 7.17. In a general oriented manifold the ambiguity is rotations $SO(2)$ (1^{st} image). For a non orientable surface reflection is added (2^{nd} image). For a fixed gauge it is possible to create a canonical parametrization and have no ambiguity (3^{rd} image, where the poles of the sphere do not have this property).

In practice, it is useful to assign local frames and fixed gauges. The canonical way to do this is by using an intrinsic function and taking the intrinsic gradient [Mel+19; Mon+17]. Both are concepts that will be presented. This does not guarantee a complete gauge (there are undefined points) but is sufficient for practical purposes. Indeed, theoretical conclusions assert this, but will not be covered. Some graphical results are shown in Figures 7.18, 7.19.

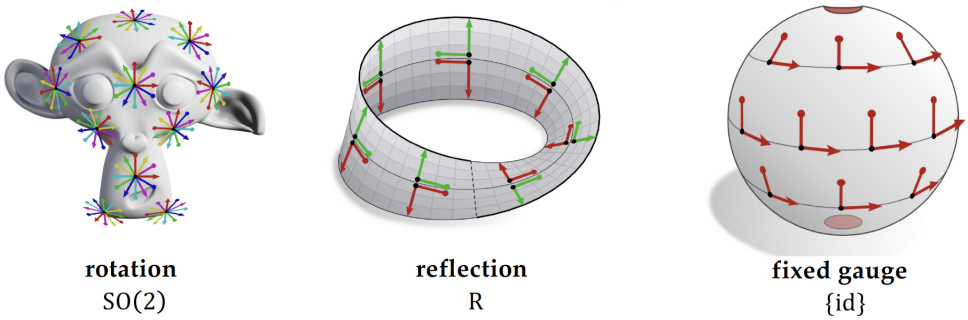


Figure 7.17: Structure groups

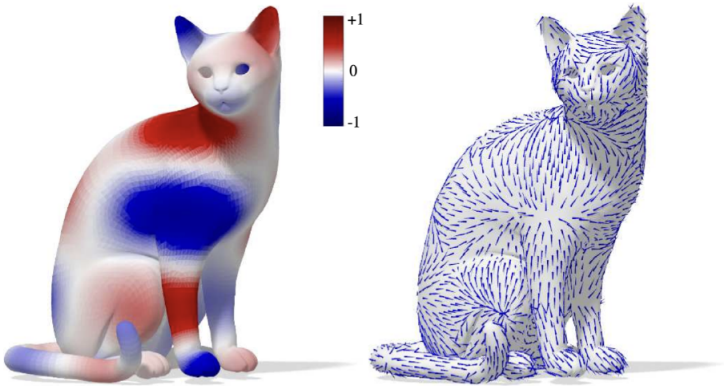


Figure 7.18: Gradient of intrinsic function

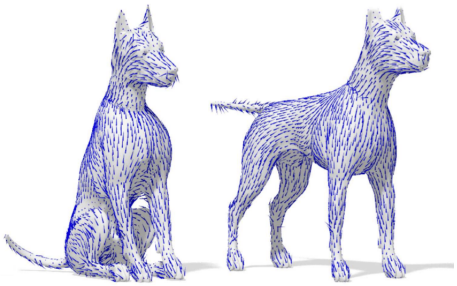


Figure 7.19: Deformation-invariant stable gauge

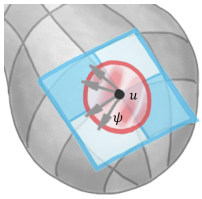


Figure 7.20: Angular Pooling technique

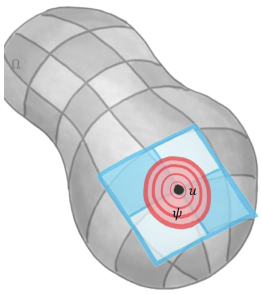


Figure 7.21: Isotropic filter on a manifold

Another procedure named *angular pooling*, which dates back to one of the first solutions to this problem, makes use of local polar coordinates in 2 dimensional manifolds [Mas+18]. Given a filter ψ expressed in polar coordinates, a rotating filter is applied as a convolution, and the maximum result is taken. The convolution will then be of the form:

$$(x \star \psi)(u) = \max_{\vartheta_0 \in [0,2\pi]} \left\{ \int_0^R \int_0^{2\pi} \psi(r,\vartheta) x(u, r, \vartheta + \vartheta_0) d\vartheta dr \right\} \tag{7.22}$$

See Figure 7.20 for a graphical intuition.

A final option is implementing isotropic filters, that depend on the radius only and have no sense of direction (e.g. in 2D, concentric circles). Figure 7.21 is an example of such filter. The convolution will again be implemented in polar coordinates and will look like:

$$(x \star \psi)(u) = \int_0^R \int_0^{2\pi} \psi(r) x(u, r\vartheta) d\vartheta dr \tag{7.23}$$

Where it is important to notice that now ψ is independent from the orientation and only has the radius r as argument. However, local rotation invariance which avoids ambiguity comes with a loss of discriminative power.

7.2 Deformation Invariance

As argued before, deformation invariance is a symmetry for manifolds. Given an instrinsic filter the aim is to have the same result for a certain class of deformations. In this section, properties and objects will be formalized better.

Definition 7.17 (Domain deformation $\eta(\cdot)$)**.** A domain deformation as in Figure 7.22 for a manifold Ω can be seen as a map to another manifold:

$$\eta : \Omega \to \widetilde{\Omega} \tag{7.24}$$

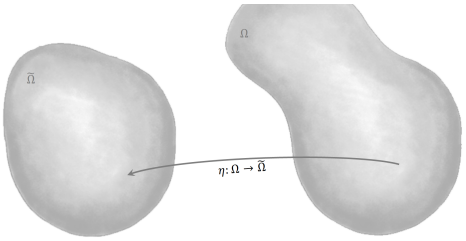


Figure 7.22: Domain deformation

A local characterization of the map η is pivotal to understand which will be the requirements. For this purpose, we resort to the easy Euclidean case. There, a function η can be linearized through the Jacobian with a Taylor series as:

$$\eta(u+x) = \eta(u) + \nabla(\eta(u))x + O(x^2) \tag{7.25}$$

While on a general manifold this is not possible, as it is not even allowed to *add* two points.

The approach will rather be local, and needs some custom objects to be presented.

Definition 7.18 (Linearization Differential $d\eta_u(x)$)**.** For a linearization of a domain deformation we define the differential as:

$$d\eta_u(x) \coloneqq \eta(u+x) - \eta(u) = \nabla(\eta(u))x + O(x^2) \tag{7.26}$$

Where $\nabla(u)x = y$. The differential is an operator that acts on the displacement.

While we cannot say something as naive as $u + x$, there are tangent spaces and this can be exploited in our favour.

Definition 7.19 (Riemaniann Isometry)**.** Assume $\eta : (\Omega, g) \to (\widetilde{\Omega}, h)$ is a diffeomorphism as in Definition 7.12 between Riemaniann manifolds with metrics $g.(\cdot,\cdot), h.(\cdot,\cdot)$ from Definition 7.5. Define a **Pushforward map** between tangent bundles of the two manifolds[1]

$$d\eta : T\Omega \to T\widetilde{\Omega} \quad d\eta_u(X \in T_u\Omega) = \widetilde{X} \in T_{\eta(u)}\widetilde{\Omega} \tag{7.27}$$

Define a **Pullback** $(\eta^* h)$ for the riemaniann metric h on $\widetilde{\Omega}$ going back to Ω as:

$$(\eta^* h)(X, Y) = h_{\eta(u)}(d\eta_u(X), d\eta_u(Y)) \quad \forall X, Y \in T_u\Omega \tag{7.28}$$

Where η is a riemaniann isometry if $(\eta^* h) = g$. Refer to Figure 7.24 for a schematic view.

Definition 7.20 (Metric Isometry)**.** For a map $\eta : (\Omega, g) \to (\widetilde{\Omega}, h)$ between metric spaces is a metric isometry if:

$$d_g = min\{\ell(\gamma)\} = d_h \circ (\eta \times \eta) \tag{7.29}$$

Where $d_h \circ (\eta \times \eta)$ for $u, v \in \Omega$ is $\{min(\ell(\eta(u), \eta(v))\}$

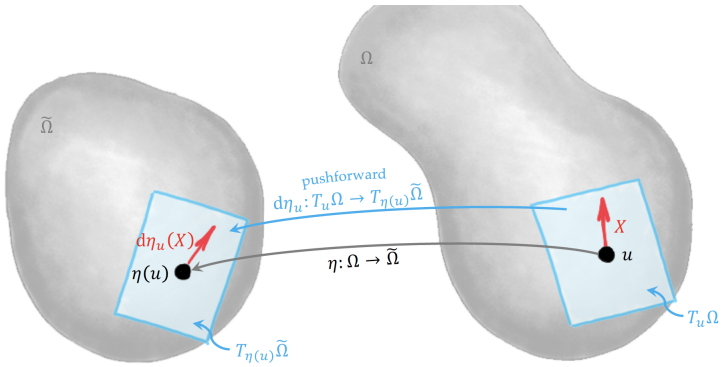


Figure 7.23: Pushforward operation

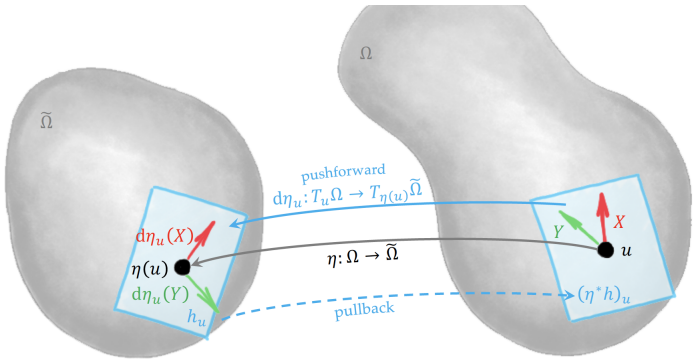


Figure 7.24: Pullback operation

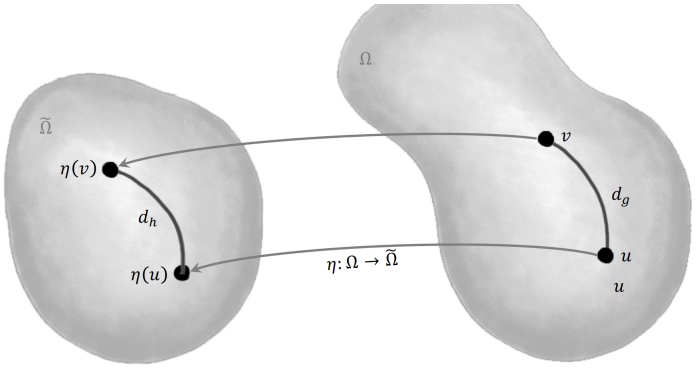


Figure 7.25: Domain deformation with tangent planes

A riemaniann metric brings guarantees for geodesics, as the next claims state. For a graphical intuition, refer to Figure 7.25

Proposition 7.21 (Riemaniann Isometry is a metric isometry)**.** *For a map η it holds that:*

$$\textit{Riemaniann isometry} \implies \textit{Metric isometry} \tag{7.30}$$

Proof. Geodesics are intrinsic, thus they are completely defined by a Riemaniann metric. Thus, a riemaniann isometry preserves the induced complete metric space. **TODO check** □

Theorem 7.22 (Myers Steenrod Theorem)**.** *For a map η nd a connected manifold Ω the opposite of Proposition 7.21 holds. Namely:*

$$\textit{Metric isometry} \implies \textit{Riemaniann isometry} \tag{7.31}$$

Proof. **TODO maybe** □

Thanks to Theorem 7.22 complete metric spaces and Riemaniann metrics are equivalent. It is also possible to define instrinsic symmetries with structure preserving automorphisms from the manifold to itself. Even though there is a whole theory on this, we will not cover it.

Recall that signal deformation invariance is of the form:

$$f(x \in \mathcal{X}(\Omega)) = f(Ax \in \mathcal{X}(\Omega) \quad A = \rho(\mathfrak{g}), \mathfrak{g} \in Gr \tag{7.32}$$

Where A is the group representation. For domain deformation, we instead have something that looks like:

$$f(x \in \mathcal{X}(\Omega)) = f(\widetilde{x} \in \mathcal{X}(\widetilde{\Omega})) \tag{7.33}$$

Where $\widetilde{\Omega}$ is an isometric deformation. Up to approximation, it is also possible to slightly enlarge the class of functions by including a distortion measure. In signal deformation, this will look like:

$$|f(x \in \mathcal{X}(\Omega) - f(Ax \in \mathcal{X}(\Omega))| \leq C\sigma(A)\|x\| \tag{7.34}$$

Where $\sigma(A)$ is a measure of dissimilarity from a group for the operator A.
In domain deformation, the condition will look similar:

$$|f(x \in \mathcal{X}(\Omega)) - f(x \in \mathcal{X}(\widetilde{\Omega})| \leq Cdis(\eta)\|x\| \tag{7.35}$$

Where $dis(\eta)$ is a measure of distortion from a isometry. The specifications of such functions are many, and will not be discussed.

Intrinsic filters are invariant under isometric deformations and it is possible to show that they are approximately invariant with approximate isometries. For the former, a good model is an unstretchable piece of paper. For the latter, rubber like elastic surface is more adequate[2].

[1] Observe that a *small* displacement $X \in T_u\Omega$ for $u \in \Omega$, causes a displacement $d\eta_u(X) \in T_{\eta(u)}\widetilde{\Omega}$ from the original point $\eta(u)$, along its tangent space. See Figure 7.23 for a visualization.

[2] There are many applications in computer graphics

7.3 Manifold Fourier Transform

In the Euclidean case, we showed that:

- convolutions commute (Theorem 5.14)
- convolutions are jointly diagonalizable (and thus have the same eigenvectors) by Lemma 5.19
- the shift operator is a convolution and its eigenvectors identify the Fourier transform (combine Theorem 5.14 and Lemma 5.19) . Thus, convolutions are diagonalized by the Fourier transform

In this section, we will extend the Fourier Transform to the general case.

Definition 7.23 (Laplacian operator Δ)**.** The Laplacian operator for a function f is defined as:

$$\Delta f = tr(\nabla^2 f) = \nabla \cdot \nabla f = \sum_{i=1}^{n} \frac{\partial^2 f}{\partial x_i^2} \tag{7.36}$$

Which is the trace of the Hessian matrix.

Definition 7.24 (Laplacian Matrix Δ)**.** In the discrete Ω case (a graph $G = (\mathcal{V}, \mathcal{E})$), the Laplace operator can be seen as a matrix such that:

$$\Delta = \begin{cases} -deg(v_i) & if\ i = j \\ 1 & if\ \exists e = (i,j) \in \mathcal{E} \end{cases} \tag{7.37}$$

Definition 7.25 (Circulant Matrix $C(\theta)$)**.** TODO

TODO circulant matrix Since the Laplacian is a circulant matrix, we can easily state by the previous observationss that:

$$\Delta e^{i\omega u} = -\omega^2 e^{i\omega u} \tag{7.38}$$

Or obtain the Fourier basis as the orthogonal basis minimizing the Dirichlet Energy[3]:

$$\varphi_{k+1} = \underset{\varphi}{argmin} \left\{ \int_{\mathbb{R}} \|\nabla(\varphi_k(u)\|^2 du \ \ s.t. \ \ \|\varphi\| = 1 \ \langle \varphi, \varphi_j \rangle = 0 \ j = 1, \dots k \right\} \tag{7.39}$$

Where we can interpret $\{\varphi_k\}$ as the smoothest (integral condition) orthogonal basis (inner product constraint), or the local difference from neighbors in the discrete case. **TODO check all of this above**

Before defining the analog of the Laplacian on manifolds it is useful to introduce additional objects.

Definition 7.26 (Scalar field)**.** A scalar field on Ω as in Figure 7.26 is a real valued function on the manifold.

$$x : \Omega \to \mathbb{R} \quad x \in \mathcal{X}(\Omega, \mathbb{R}) \tag{7.40}$$

With inner product:

$$\langle x, y \rangle = \int_{\Omega} x(u) y(u) du \tag{7.41}$$

[3]especially when computing the vectors sequentially!

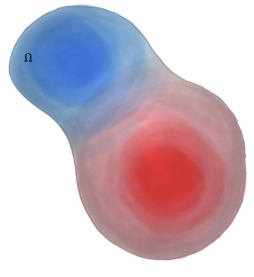


Figure 7.26: Scalar field on Ω

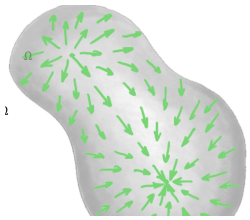


Figure 7.27: Vector field on Ω

Definition 7.27 (Vector field)**.** A vector field on Ω as in Figure 7.27 is a vector valued function on the manifold assigning for each point u a vector X in its tangent plane.

$$X : \Omega \to T\Omega \quad X \in \mathcal{X}(\Omega, T\Omega) \tag{7.42}$$

Where $u \to X(u) \in T_u\Omega$, which can be interpreted as a local flow. The inner product is the integral of the Riemaniann metric:

$$\langle X, Y \rangle = \int_\Omega \langle X(u), Y(u) \rangle_u d_u \tag{7.43}$$

If equipped with inner products, both a scalar field and a vector field become Hilbert Spaces. Observe that scalar fields are *lowercase* and vector fields are *UPPERCASE*.

We can now define the intrinsic gradient as the direction of steepest increase of a scalar field x at a point $u \in \Omega$.

Definition 7.28 (Intrinsic gradient)**.** Consider for a scalar field x a differential:

$$dx_u : T_u\Omega \to \mathbb{R} \tag{7.44}$$

Which is a linear functional acting on tangent vectors, often named *dual vector*. By the Riesz Frechet representation Theorem **TODO maybe**, any differential can be expressed as the inner product of the argument Y with the representation of the differential, the gradient ∇x. Namely:

$$dx_u(Y) = \langle \nabla x(u) Y(u) \rangle_u \tag{7.45}$$

Where ∇x is the intrinsic gradient function, mapping scalar fields into vector fields:

$$\nabla : \mathcal{X}(\Omega, \mathbb{R}) \to \mathcal{X}(\Omega, T\Omega) \tag{7.46}$$

The intrinsic gradient is thus a vector field of the steepest increase of a scalar field x. We can define another operator that does the opposite.

Definition 7.29 (Divergence operator div, ∇^*)**.** Consider a vector field X. The flow of X for an infinitesimal ball centered at $u \in \Omega$ will be a function denoted as div or ∇^* such that:

$$\nabla^* : \mathcal{X}(\Omega, T\Omega) \to \mathcal{X}(\Omega, \mathbb{R}) \tag{7.47}$$

Which maps vector fields to scalar fields.

Proposition 7.30 (Gradient and Divergence are adjoint)**.**

$$\langle X, \nabla x \rangle = \langle \nabla^*(X), x \rangle \tag{7.48}$$

Where the former an inner product on vector fields and the latter is on scalar fields.

Proof. **TODO prove** □

We can eventually define a new operator.

Definition 7.31 (Laplace Beltrami Operator Δ)**.** Given Definitions 7.26, 7.27, 7.28, 7.29, the Laplace Beltrami operator is a map from scalar fields to scalar fields:

$$\Delta : \mathcal{X}(\Omega, \mathbb{R}) \to \mathcal{X}(\Omega, \mathbb{R}) \tag{7.49}$$

Which is the divergence of the gradient of the scalar field for a point $u \in \Omega$.

$$\Delta(u) \coloneqq \nabla^*(\nabla(x(u))) \tag{7.50}$$

Observe that it is **intrinsic**.

Definition 7.32 (Self-Adjoint Operator)**.** A map $S : (\Omega, \langle \cdot, \cdot \rangle) \to (\Omega, \langle \cdot, \cdot \rangle)$ so that Ω is equipped with an inner product is self adjoint if:

$$\langle Su, v \rangle = v^\dagger S u = v^\dagger S^\dagger u = \langle u, Sv \rangle \quad \forall u, v \in \Omega \tag{7.51}$$

Which is a generalization of symmetry.

Lemma 7.33 (Self Adjoint operator eigenvalues)**.** *A self adjoint operator $S : \Omega \to \Omega$ has real eigenvalues:*

$$\langle Su, v \rangle = \langle u, Sv \rangle \quad \forall u, v \in \Omega \implies \Big(Sw = \lambda w \implies \lambda \in \mathbb{R} \Big) \tag{7.52}$$

Proof. Assume (λ, w) is an eigenpair with normalized eigenvector $\|w\| = 1$. Then:

$$\begin{aligned}
\lambda &= \lambda \langle w, w \rangle \\
&= \langle \lambda w, w \rangle && \text{inner product property} \\
&= \langle Sw, w \rangle && \text{eigenpair} \\
&= \langle w, Sw \rangle && \text{self adjointness} \\
&= \langle w, \lambda w \rangle && \text{eigenpair} \\
&= \lambda^\dagger \langle w, w \rangle && \text{inner product property} \\
&= \lambda^\dagger
\end{aligned}$$

And by $\lambda = \lambda^\dagger \iff \lambda \in \mathbb{R}$ we conclude that eigenvalues are real. □

Theorem 7.34 (Self-Adjointness of Δ)**.** *The Laplace Beltrami operator is self adjoint and by Lemma 7.33 has real eigenvalues.*

$$\langle \Delta x, x \rangle = \langle x, \Delta x \rangle \underset{Lem7.33}{\implies} eigenvalues\ \lambda \in \mathbb{R} \tag{7.53}$$

Proof. **TODO** □

Example 7.35 (Cartesian $\mathbb{R}^2$ Laplace Beltrami)**.** In $\mathbb{R}^2$ the Laplace Beltrami operator is the difference between a function at a point and its average neighbor value. and it is rotation invariant. **TODO**

The Laplace Beltrami operator, found everywhere in Physics, is fundamental for equations such as the Newton Law of cooling and waves.

Example 7.36 (Waves, standing waves, Laplace Beltrami)**.** Denote with a pedix derivatives with respect to a variable. A wave equation, by Newton's Law is of the form:

$$x_{tt} = \frac{\partial^2 x}{\partial t^2} = c\Delta x \tag{7.54}$$

Where the extension, when local, is given by the Laplacian. Chadni, a German Physicist, noticed that at certain frequencies patterns in waves arised. In a toy example, let $c = 1$. Then $x_{tt} = \Delta x$ and if we separate variables in the differential equation we enforce a solution of the form:

$$x(u,t) = \varphi(u)\tau(t) \implies x_{tt} = \varphi(u)\tau_{tt}(t) = \Delta x = \Delta\varphi(u)\tau(t) \implies \frac{\tau_{tt}(t)}{\tau(t)} = \frac{\Delta\varphi(u)}{\varphi(u)} = \lambda \forall t, u$$

This newly defined constant implies that the spatial part of the equation $\frac{\Delta\varphi(u)}{\varphi(u)}$ is such that:

$$\frac{\Delta\varphi(u)}{\varphi(u)} \forall u \iff \Delta(\varphi(u)) = \lambda\varphi(u) \tag{7.55}$$

Which is a **Laplacian Eigenfunction**. From a physical wave perspective, a standing wave identifies for φ the *vibration modes* and λ the *vibration frequencies*. The standing patterns coincide when the particles are in resonance with the oscillation.

We state informally with no proof the following claim:

If a manifold is **compact** it has a **discrete** (countable) eigendecomposition for the Laplace Beltrami Operator.

Clearly, the eigenfunctions are **intrinsic** and are **isometry invariant** , at least in a theoretical framework where we can establish a perfect isometry (up to potentially ambiguous sign flips). An example of such behavior is outlined in Figures 7.28, 7.29, where a pose modification does not impact the Eigenvectors (i.e. the colors) of the human surface (i.e. the manifold).

Remark. In practice, isometries are very difficult to establish, and still look very unstable.

As we had a Fourier transform in Definition 5.16, we can now extend it to the continous case.

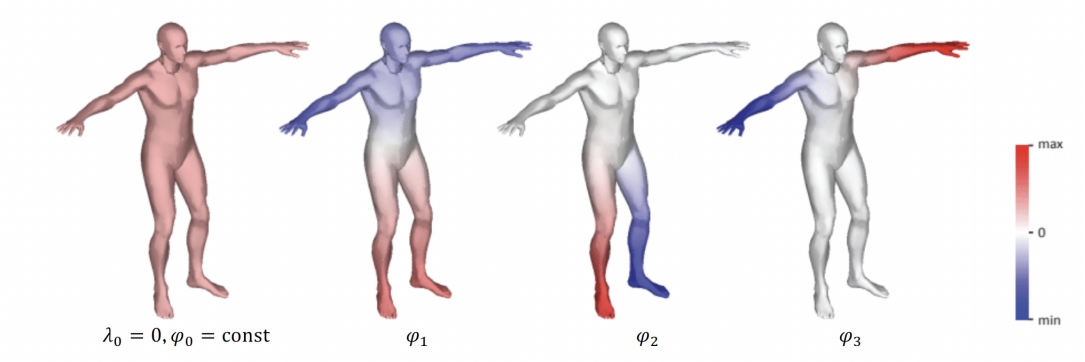


Figure 7.28: Laplacian Eigenfunctions

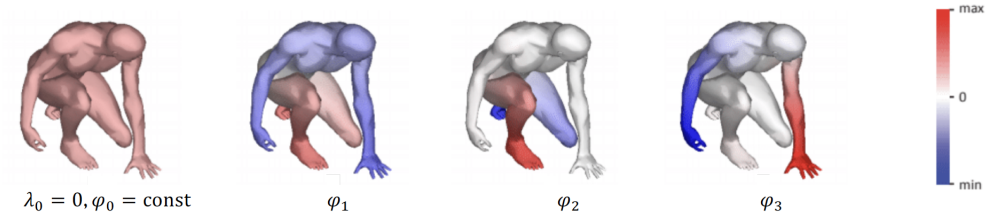


Figure 7.29: Laplacian Eigenfunctions II

Definition 7.37 (Continuous Fourier Transform via Laplacian)**.** Define the *forward* Fourier transform as:

$$\widehat{x_k} = \langle x, \varphi_k \rangle = \int_\Omega x(u)\varphi_k(u)du \tag{7.56}$$

With inverse:

$$x(u) = \sum_{k \geq 0} \langle x, \varphi_k \rangle \varphi_k(u) = \sum_{k \geq 0} \widehat{x_k}\varphi_k(u) \tag{7.57}$$

Thanks to the transformation of Definition 7.37 it is possible to decompose a scalar field into orthogonal basis and completely ignore the manifold structure. However, this comes at a cost. Few time is needed to notice that, first of all, frequencies of the Laplacian λ have no direction.

Drawing from Chapter 6, same way as we have this property that Fourier transforms map a convolution to pointwise products, we can define a convolution as pointwise products of Fourier transforms. This suggests the following definition for a Spectral Convolution.

$$(x \star \psi)(u) = \sum_{k \geq 0} (\widehat{x_k} \cdot \widehat{\varphi_k})\varphi_k(u) \tag{7.58}$$

Equation 7.58 has many drawbacks, which we will list and discuss informally.

- there is no FFT (Fast Fourier Transform), computational time is $O(n^2)$ when we discretize it into a grid of n sample points.
- typically we cannot guarantee that filters are localized
- filters are isotropic (radial symmetry)
- it is unstable under domain deformations

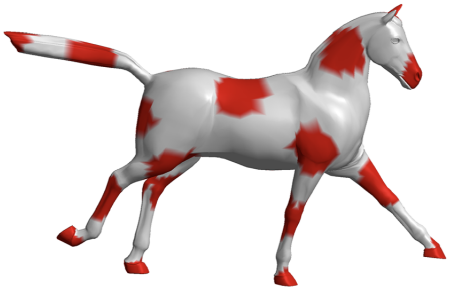


Figure 7.30: Manifold Ω with signals

Figure 7.31: Edge detection filter α applied

The last point is crucial since it is never guaranteed that we will have an isometry. To better understand the magnitude of the potential damage, an example is presented.

Example 7.38 (Spectral Convolution (in)stability)**.** Assume we have a horse (i.e. a manifold) Ω, pictured in Figure 7.30 and a filter α which filters signals as in Equation 7.58:

$$\sum_{k\geq 0} \alpha_k \langle x, \varphi_k \rangle \varphi_k(u) \tag{7.59}$$

If we define the coefficients α_k of this filter such that we perform edge detection, the result will likely resemble Figure 7.31. If the domain is deformed to $\widetilde{\Omega}$ (i.e. the horse runs), and we filter the manifold with the same coefficients and transformed laplacian & eigenvectors as:

$$\sum_{k\geq 0} \alpha_k \langle x, \widetilde{\varphi_k} \rangle \widetilde{\varphi_k}(u) \tag{7.60}$$

The result of the filter is different, even if the deformation is almost an isometry. An example is Figure 7.32.

The transformed high frequencies instability problem makes Fourier transformed rarely used in practice. What we is implemented instead is a **spectral transfer function** $\widehat{p}(\Delta)$ applied to the Laplacian (to its eigenvalues):

$$\widehat{p}(\Delta)(u) = \sum_{k\geq 0} \underbrace{\widehat{p}(\lambda_k)}_{\widehat{\psi_k}} \underbrace{\langle x, \varphi_k \rangle}_{\widehat{x_k}} \varphi_k(u) \tag{7.61}$$

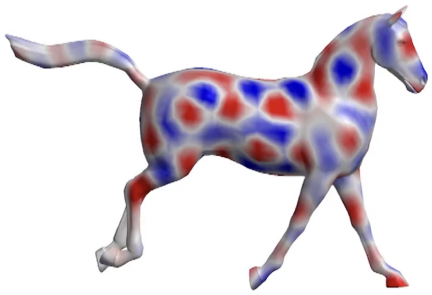


Figure 7.32: α applied on deformed manifold $\widetilde{\Omega}$

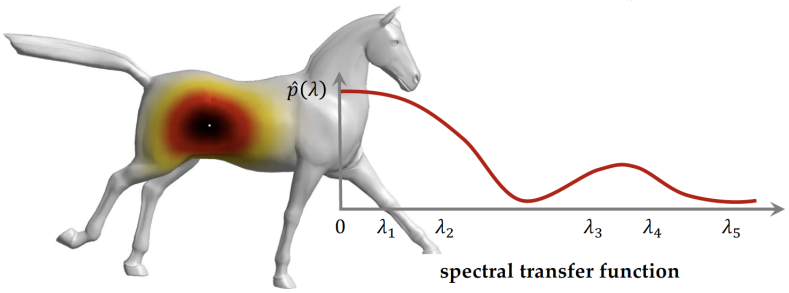


Figure 7.33: Spectral transfer function

An example of its rendering is Figure 7.33, where the graph is the spectral transfer function, customly decreasing as the frequency λ increases to avoid instabilities. It can be seen as an *equalizer* on the filter that enhances certain frequencies. Another interpretation is as a *prescription* of the filters. It can be checked that it is deformation invariant.
Yet another interpretation can be extracted by developing the inner product and exchanging the sum with the integral as:

$$\widehat{p}(\Delta)(u) = \sum_{k\geq 0} \widehat{p}(\lambda_k)\langle x, \varphi_k\rangle \varphi_k(u) \tag{7.62}$$

$$= \int_\Omega x(v) \underbrace{\sum_{k\geq 0} \widehat{p}(\lambda_k)\varphi_k(v)\varphi_k(u)}_{\psi(u,v)} \, dv \tag{7.63}$$

Where $\psi(u,\cdot)$ is a position dependent spatial kernel, which is not a convolution but is position dependent (see Figure 7.34).
Nevertheless, in the Euclidean case, it will be that $\varphi_k(u) = e^{iku}$ and $\varphi_k(v) = e^{-ikv}$ and the kernel will look like:

$$\widehat{p}(\Delta)(u) = \int_{-\pi}^{\pi} x(v) \underbrace{\sum_{k\geq 0} \widehat{p}(\lambda_k) e^{-ikv} e^{iku}}_{\psi(u-v)} \, dv \tag{7.64}$$

$$= \int_{-\pi}^{\pi} x(v)\psi(u-v)dv = (x \star \psi)(u) \tag{7.65}$$

Coming back to the classical convolution!

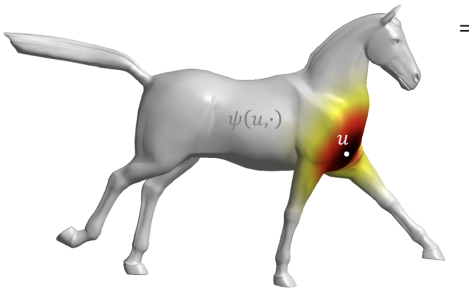


Figure 7.34: horse5

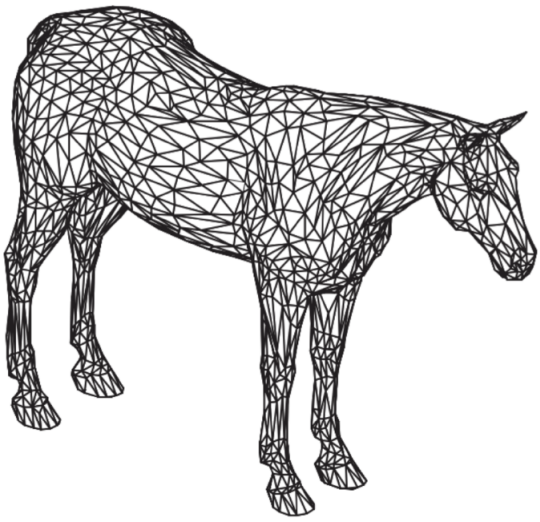


Figure 7.35: Horse mesh

7.4 Discretization

In real applications and models manifolds are replaced by discretizations. The standard way to do this operation is based on meshes, which are ways to mimic (up to approximation) the surface by meshing together geometric surfaces.

Definition 7.39 (Triangular Mesh $\mathcal{T}$)**.** A triangular mesh is a graph $\mathcal{T} = \{\mathcal{V}, \mathcal{E}, \mathcal{F}\}$ where $\mathcal{V}$ are the nodes, $\mathcal{E}$ are the edges and $\mathcal{F}$ are the faces. In particular, it is enforced that:

$$\mathcal{F} = \{triangles(u, v, q) \mid u, v, q \in \mathcal{V},\ (u, v), (v, q), (q, u) \in \mathcal{E}\} \tag{7.66}$$

Where the order of nodes in a face defines its **orientation**.

We recognize that a triangular mesh is a graph with extra structure. Yet, to guarantee manifoldness properties, an additional requirement must be attached.

Definition 7.40 (Manifold Mesh)**.** A triangular mesh is a manifold mesh if and only if:

1. each edge $e \in \mathcal{E}$ if shared, has two incident faces $F_1, F_2 \in \mathcal{F}$

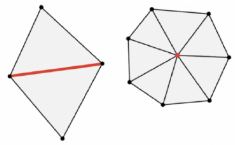


Figure 7.36: Manifold meshes

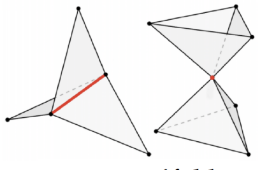


Figure 7.37: Non manifold meshes (left violates 1, right violates 2)

2. the boundary of triangles $F \in \mathcal{F}$ incident to a node $v \in \mathcal{V}$ forms a single loop

For valid examples, refer to Figure 7.36. For not valid examples, see Figure 7.37. In particular, on the right condition 1 is violated and on the left condition 2 is violated.

Thanks to the above structure, a metric can be defined on manifold meshes. The most intuitive one is based on the Euclidean distance between nodes $\ell_{uv} = \|x_u - x_v\|$. The euclidean distance naturally comes with the triangular inequality and other geometric conclusions that might end up being useful. Just like in Definition 7.5, given such a metric, any property arising solely from ℓ is **intrinsic**, and ℓ preserving mappings (deformations) are considered **isometries**.

Except some unique pathological examples, meshes are **rigid**, and hardly have non trivial isometries. Deformation stability is thus crucial for the success of such objects.

Having discretized the domain Ω, the Laplacian has to be adapted to the context.

Definition 7.41 (A Discrete Laplacian)**.** For a mesh $\mathcal{T}$, define its (discrete) Laplacian as:

$$(\Delta x)_u = \sum_{v \in \mathcal{N}_u} w_{uv}(x_u - x_v) \tag{7.67}$$

$$= \underbrace{x_u \sum_{v \in \mathcal{N}_u} w_{uv}}_{\propto value\ at\ u} - \underbrace{\sum_{v \in \mathcal{N}_u} w_{uv} x_v}_{\propto average\ value\ neighbors} \tag{7.68}$$

Where w_{uv} is the weight of the (u, v) edge connection and $x.$ is the feature of the node. The second interpretation comes with a $\propto$ sign since they are not exactly equal but have the same magnitude with a sum over neighbors.
Depending on how the weight is defined, a specific type of Laplacian will arise[4].
Figure 7.38 shows one of the pairs involved in the computation. Given the discretization, it can also be expressed as a matrix $\Delta \in \mathbb{R}^{n \times n}$ where $|\mathcal{V}| = n$ and:

$$\Delta = D - W \quad D = Id \ : \forall u \ d_u = \sum_{v \in \mathcal{N}_u} w_{uv} \ \ W\ weights \tag{7.69}$$

[4]It can actually be proved that in no discrete characterization the resulting Laplacian will satisfy all the properties of the continuous one.

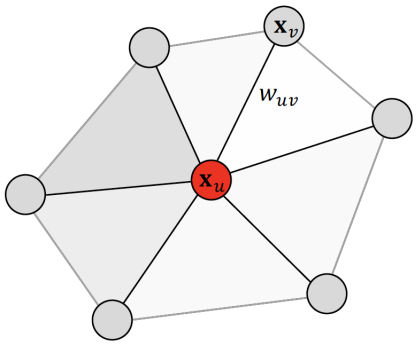


Figure 7.38: Discrete Laplacian elements

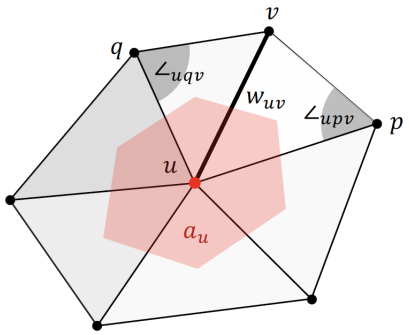


Figure 7.39: Mesh (cotangent) Laplacian

TODO check what are the weights etc.

Out of the different types of Laplacian, one comes with theoretical guarantees at the limit [PP93; War+08; Mey+03; Mac04].

Definition 7.42 (Mesh (cotangent) Laplacian)**.** Let the weights w_{uv} for Definition 7.41 be:

$$w_{uv} = \frac{cot(\angle_{uqv}) + cot(\angle_{upv})}{2a_u} \tag{7.70}$$

As in Figure 7.39, where $\angle_{\dots}$ indicates the angle inscribed by three specified points and a_u is a *local area element* such as "the area of the polygon constructed upon the barycenters of the triangles (u, p, q) sharing the node u and given by $a_u = \sum_{(v,q):(v,q,u)\in\mathcal{F}} a_{vqu}$".

While the intrinsicness of the cotangent Laplacian does not look obvious, with some computations it is possible to work out a completely induced by ℓ formula.

Proposition 7.43 (Cotangent Laplacian is intrinsic)**.** *See Figure 7.40 for the context of the objects.*

$$w_{uv} = \frac{-\ell_{uv}^2 + \ell_{vq}^2 + \ell_{uq}^2}{8a_{uvq}} + \frac{-\ell_{uv}^2 + \ell_{vp}^2 + \ell_{up}^2}{8a_{uvp}} \tag{7.71}$$

$$a_{uvq} = \sqrt{s_{uvq}(s_{uvq} - \ell_{uv})(s_{uvq} - \ell_{vq})(s_{uvq} - \ell_{uq})} \quad \textit{by Heron's semiperimeter s... formula} \tag{7.72}$$

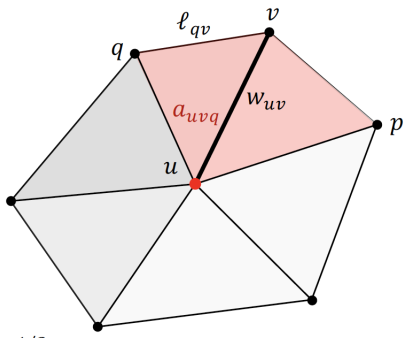


Figure 7.40: Elements required in intrinsic cotangent Laplacian

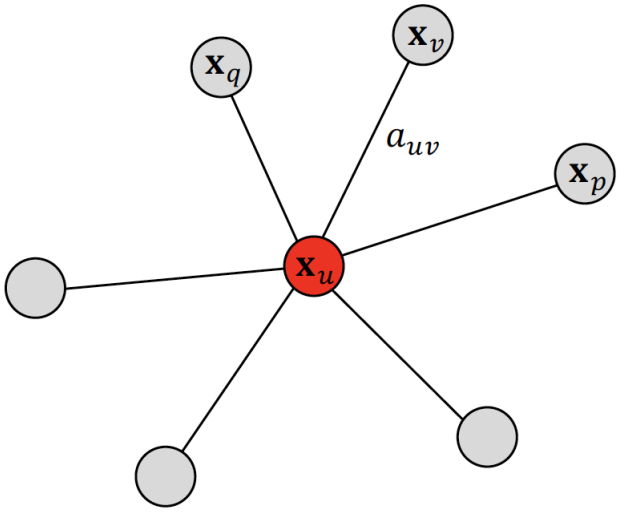


Figure 7.41: Laplacian6

Up to now, there is nothing specific to the mesh. To get to the point, for a Laplacian we usually use the adjacency matrix (Figure 7.41) for weights and write it as:

$$(\Delta x)_u = \sum_{v \in \mathcal{N}_u} a_{uw}(x_u - x_v) = d_u x_u - \sum_{v \in \mathcal{N}_u} a_{uv} x_v = \phi(x_u, X_{\mathcal{N}_u}) \tag{7.73}$$

Where $\phi(\cdot, \cdot)$ is a permutation invariant aggregator as in Definition 3.28, which guarantees that the function $F(X) = \Delta X$ will be permutation equivariant as in Definition 3.29 by Theorem 3.42.

In the much older field of Spectral analysis on graphs some of these representations were already studied years ago, also in the context of GNNs. We will briefly touch on the ChebNet architecture [DBV17].

Definition 7.44 (Graph Fourier Transform)**.** Given an undirected graph G, and its Laplacian Δ, which is symmetric[5], let $\Delta = \Phi \Lambda \Phi^T$ where $\Phi \Phi^T = I$ be its decomposition. Then:

- Graph Fourier transform $\widehat{x} = \Phi^T x$
- inverse transform $x = \Phi \widehat{x}$

[5] and thus has an orthogonal decomposition

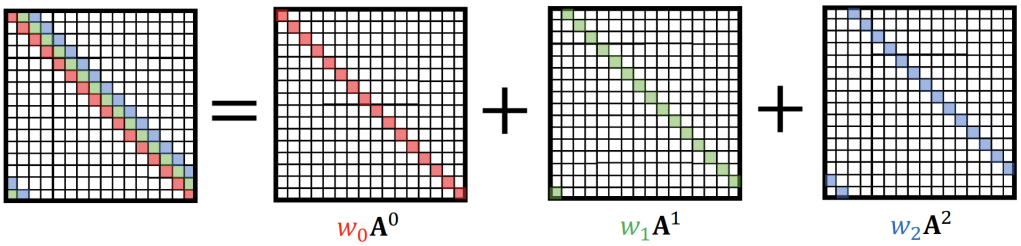


Figure 7.42: Convolution revisited

- spectral convolutions:

$$x \star \Psi = \Phi\Big((\Phi^T x) \cdot (\Phi^T \Psi)\Big) = \Phi diag(\widehat{\Psi})\widehat{x} \tag{7.74}$$

Where their computation has $O(n^2)$ complexity (no FFT).

ChebNet implements a polynomial spectral filter, a convenient form. The polynomial is applied to the eigenvalues as before:

$$\widehat{p}(\lambda) = \sum_{l=0}^{r} \alpha_l \lambda^l \tag{7.75}$$

Which is applied to the graph Laplacian as:

$$\widehat{p}(\Delta) = \Phi\widehat{p}(\Delta)\Phi^T = \sum_{l=0}^{r} \alpha_l \Phi\Lambda^l\Phi^T = \sum_{l=0}^{r} \alpha^l \Delta^l \tag{7.76}$$

Where:

- there is no need for an explicit eigendecomposition
- simply amounts to applying powers of the graph Laplacian, which is likely sparse and thus requires $O(r\mathcal{E})$ computations.
- $\widehat{p}(\Delta)$ is stable under deformations
- since the Laplacian is local, then the filters are
- "Spectral" filter boils down to simple local averaging, i.e. the *convolutional flavor* of GNN

Another extension of this framework is a side interpretation of convolution [SM13]. As we saw before, the shift operator can be seen as the adjacency matrix of the ring (directed graph) of a set Ω. It is then possible to state that a convolution $p(A)$ is a linear combinations of the powers of the adjacency matrix.

Example 7.45 (Convolutions and Adjacency matrix powers)**.** For Figure 7.42 the polynomial is:

$$p(A) = w_0 I + w_1 A^1 + w_2 A^2 \tag{7.77}$$

TODO maybe expand

Graph Convolution Network (GCN) are the simplest method of convolution. If we arrange the nodes

$$Y = softmax(AXW) \tag{7.78}$$

$$Y = softmax(A\, ReLu(AXW_1)W_2) \tag{7.79}$$

$$Y = softmax(AAXW_1W_2) \tag{7.80}$$

$$Y = softmax(A^2XW) \tag{7.81}$$

Chapter 8

Gauges

As briefly mentioned before in Chapter 7, gauge theory is fundamental to understand vector fields. This Chapter is devoted to giving an intuitive level introduction, with some mathematical flavour. Given the highly complex field of study, it will be presented in narrative form rather than logical.

The outline of the Chapter is as follows:

- Motivations, why gauge theory?
  - feature fields in real data
  - the problem of oritenting the kernel for manifolds' convolution
- gauge equivariant convolution on manifolds & meshes
- theoretical background
  - bundles and fibers
  - gauges and gauge transformations (internal vs external)
  - gauge symmetry and equivariance

Before going to the motivation, the concepts will be restated for the sake of clearness. For a manifold M, some vector spaces are attached (e.g. tangent space). For each of these there is a large number of frames (bases of ordered vectors). A gauge is essentially a choice of frame for each region U_i of the manifold. Tipycally, they are local. Any vector can be represented by a gauge, where different gauges give different representations for the same object.

A **change** of gauge reference is a **gauge transformation**.

Observation 8.1 (Coordinates vs Gauges)**.** *While coordinates provide a position of points, gauges are orientations and do not necessarily align with the coordinate axes. See Figure 8.1*

8.1 Why Gauges?

In many scientific domains data as a vector field is studied. As an example, consider methereology (e.g. wind direction), or tensor fields diffusion in a brain model, where at each of the 3 dimensional points the diffusion in every direction is stored in a 3×3 matrix.

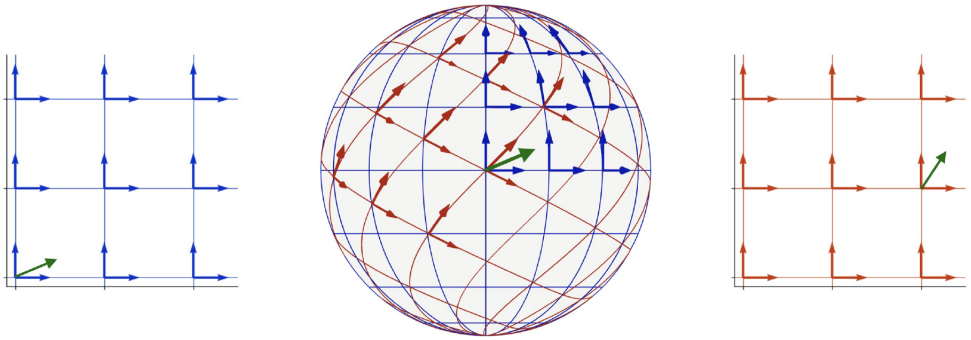


Figure 8.1: Two Gauges, one Manifold

In principle, gauges provide increased representation power with respect to the consistent xyz reference axes for local properties of a manifold.

Another reason for feature fields as in Definition 7.27 comes by the necessity to analyze scalar fields of Definition 7.26. Consider the Spherical CNN case of Section 6.1.1. With a scalar field, convolution is expensive in the $SO(3)$ group. On the other hand, if we wish to stay in the homogeneous space it is necessary to resort to isotropic filters, which are agnostic to orientation and lose representative power. Feature fields are a middle ground between the two.

Recall that a function is a mapping of the form:

$$u \in \Omega \to x(u) \in \mathcal{C} \tag{8.1}$$

Similarly, but not equivalently, a field (*a section of a bundle*) maps domain elements to element specific sets (*fibers*) as:

$$u \in \Omega \to x(u) \in \mathcal{C}_u \tag{8.2}$$

Where the different $\mathcal{C}_u$ are isomorphic but not canonically isomorphic[1]. This difference implies that to make feature fields into functions we need a canonical fiber, as an *alignment*. This alignment is guaranteed by the choice of a linear invertible map (*gauge*):

$$\omega_u : \mathbb{R}^C \to \mathcal{C}_u \tag{8.3}$$

Where $\mathcal{C} = \mathbb{R}^C$ is the canonical fiber.

Example 8.2 (Fibers and Vector Fields)**.** A fiber $\mathcal{C}_u$ could be a tangent plane $T_u\Omega$ for a point $u \in \Omega$, as in Figure 8.2.
A vector field could be the collection of vectors in tangent planes $x(u) \in T_u\Omega$ assigned for each point of the domain $u \in \Omega$, as in Figure 8.3

Example 8.3 (Easy and hard to find Gauges)**.** In some instances, defining a gauge is natural. See for example the classic cartesian reference system for an $\mathbb{R}^2$ plane of Figure 8.4. In other cases, as for the sphere S^2 of Figure 8.5, it needs more thought.

Since gauges impose a representation of information, gauge equivariance is crucial to avoid differences in equivalent depictions of the same data. In addition to this requirement, as briefly mentioned in Chapter 7, we remind that in some instances a smooth (globally continuous) gauge may also not exist.

[1]The intuitive difference is that those fibers will be isomorphic **up to the choice** of a basis, but canonically is sometimes judged as a colloquial term.

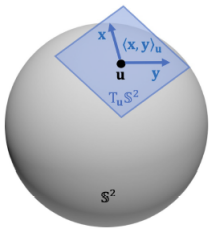


Figure 8.2: Tangent plane fiber $\mathcal{C}_u = T_u\Omega$

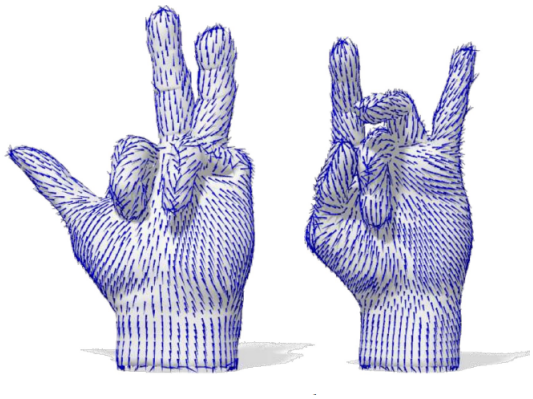


Figure 8.3: Vector Field of tangent vectors

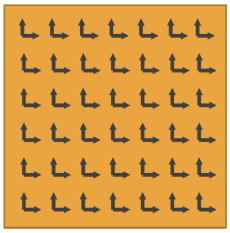


Figure 8.4: Cartesian Gauge, easy

Figure 8.5: Spherical Gauge, hard

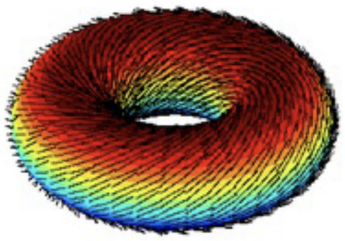


Figure 8.6: Toroid, smooth gauge, parallelizable

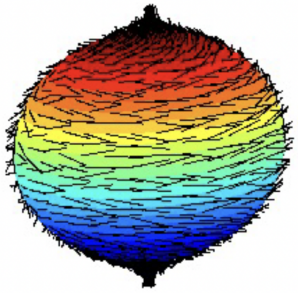


Figure 8.7: Spin, non-smooth gauge, non parallelizable

Example 8.4 (A toroid and a spin)**.** For the case of the toroid, it is possible to construct a gauge that does not present singularities (non-smooth points). An example is Figure 8.6. For a spin it is not feasible. In Figure 8.7, singularities arise at the top and at the bottom.

Given the previous arguments, it can be concluded that:

- Geometric data can only be represented relative to a gauge
- no preferred choice for this orientation, all are equivalent
- for a continuous field, gauges in multiple charts are able to represent a field as a continuous function

The need for equivariance suggests attaching to gauges a stable transformation property.

Definition 8.5 (Gauge Transformation)**.** A gauge transformation is a function $\mathfrak{g}$ defined on a subset of the domain that has an action on signals:

$$\mathfrak{g} : U \subset \Omega \to \mathfrak{G} \tag{8.4}$$

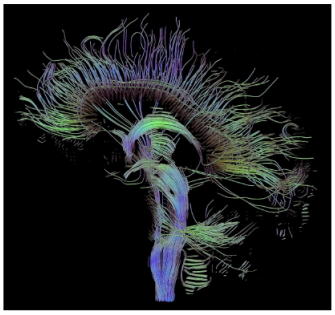
$$x(u) \to \rho(\mathfrak{g}(u))x(u) \tag{8.5}$$

Where $\mathfrak{G}$ is the structure group of what we wish to preserve across transformations. In Figure 8.8, we notice a simple change of axes through $\mathfrak{g}$.

Example 8.6 (Diffusion Tensor Images of the brain)**.** A Diffusion Tensor Image (DTI) signal is a function $x : \mathbb{R}^3 \to \mathbb{R}^{3\times3}$, with representation:

$$\rho(R)M = RMR^T \tag{8.6}$$

Figure 8.8: $\mathbb{R}^2$ domain Gauge Transformation



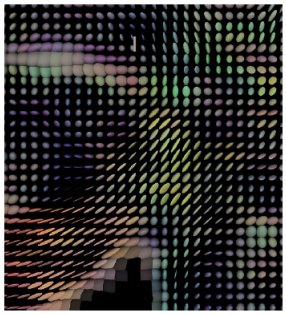


Figure 8.9: Brain MRI and Diffusion Tensor

Where R is a rotation. Since the brain has no evident rotation symmetry, we wish to keep this only for local directions. The derived assumption is that orthogonal frames are all equivalent, and thus the symmetry group becomes $\mathfrak{G} = O(3)$. The gauge transformations, following Definition 8.5, and the action on the signal will then be:

$$\mathfrak{g} : \mathbb{R}^3 \to O(3) \tag{8.7}$$

$$x(u) \to \rho(\mathfrak{g}(u))x(u) \tag{8.8}$$

For an example of Tensor and brain MRI, refer to Figure 8.9.

It is useful to stress that by gauge equivariance we mean that the underlying vector field does not change up to a $\mathfrak{g}$ transformation. This idea is perfectly represented in Figure 8.10, which is similar to diagrams seen in previous Chapters.

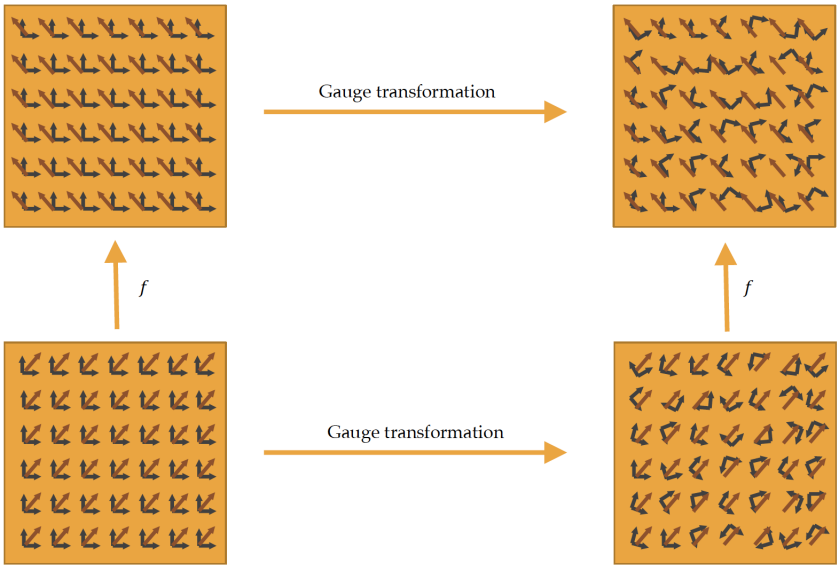


Figure 8.10: Gauge Equivariant function f

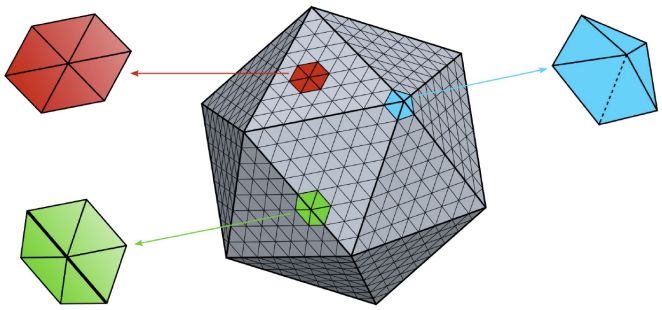


Figure 8.11: Icosahedron, not homogeneous

Example 8.7 (Internal Gauge)**.** For an RGB image, signals are maps form a grid to the three color channels $x : \mathbb{R}^2 \to \mathbb{R}^3$. Two kinds of symmetries arise in such objects:

- if the channels (R, G, B) are equivalent then $\mathfrak{G} = S_3$ permutations are the structure group
- if hue values are equivalent then $\mathfrak{G} = SO(2)$ hue rotations are the structure group

Notice that a gauge transformation will change every pixel $u \in \Omega$ independently. Thus, if we assume the gauge is a symmetry $\mathfrak{g} \in \mathfrak{G}$, a constraint on the convolution (i.e. evry color is equivalent, or every hue value is equivalent) will arise.

Another reason why Gauges are powerful objects is their deep link with convolution on manifolds. While convolutions on manifolds can be implemented from homogeneous group convolutions where a kernel is moved around the manidfold by global symmetries (e.g. rotations over the sphere). While it is well defined for a homogeneous space, for a Icosahedron as in Figure 8.11. In this case a symmetry will not guarantee weight sharing between non homogeneous regions (colored in Figure 8.11).

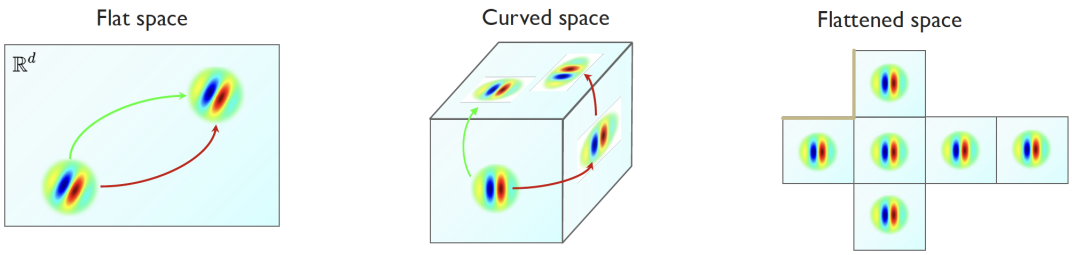


Figure 8.12: Filters on flat and curved shape

Another option could be parallel transport, but we saw in Chapter 7 that it ends up being path dependent, with filters' orientation being impacted by how they were moved.

The arbitrariness of choice (see Figure 8.12) is not even solved by flattening a curved space, as it leads to discountinuities over not crossing edges in the flattened manifold with a fixed gauge. In the 3^{rd} image, adjacent faces of the cube have $\frac{\pi}{2}$ rotated filters when the flattened cube is closed again.

Yet another option could be using spectral and graph convolutions. The former makes use of the eigenfunctions of the Laplacian, which is rotation invariant. The former, as already discussed, uses isotropic filters (kernels). Both present decreased representative power.

Geodesic convolution instead, finds the best orientation across possible inner products of rotated copies of a filter (scalar to regular layer), and finds the maximum over ϑ_0 orientations (regular to scalar layer). It is the *Angular Pooling* method discussed in Chapter 7, Equation 7.22. After max pooling, no orientation will be kept as well, losing the potential of such information. Yet, for some problems, this is crucial.

All of these rather suboptimal methods suggest proposing Gauge Equivariant Networks:

- For each layer, define a feature space
  - $\forall u \in \Omega$ attach a fiber $\mathcal{C}_u$
  - $\mathcal{C}_u$ has a ρ group representation, which chooses how to transform
  - the feature space becomes a space of ρ-fields over Ω (*sections of the associated bundle*)
- seek a convolution layer that is gauge equivariant
  - up to a gauge transformation the input and output coefficients of the layer must *equivariantly* change across group elements.

TODO check these two proofs

Definition 8.8 (Gauge Convolution on scalars)**.** Let Ω be a manifold with dimension s. Then, signals are of the form $x : \Omega \to \mathbb{R}$, and filters $\psi : \mathbb{R}^s \to \mathbb{R}$. A gauge will link the canonical fiber $\mathcal{C} = \mathbb{R}^s$ to different tangent spaces. For a point $p \in \Omega$ the gauge is:

$$\omega_p : \mathbb{R}^s \to T_p\Omega \tag{8.9}$$

We require ω_p to map to isomorphisms (not canonical) and possibly to be smooth in p. If it is not smooth, which as we saw is very likely, the operations will be localized.
Additionally, recover the exponential map of Definition 7.15, which in this case for a point $p \in \Omega$

is:

$$exp_p : T_p\Omega \to \Omega \tag{8.10}$$

Then a scalar to scalar gauge convolution can be constructed as:

$$f = (\psi \star x)(p) = \int_{\mathbb{R}^s} \psi(\mathbf{v}) x(exp_u(\omega_u(\mathbf{v})))d\mathbf{v} \tag{8.11}$$

Where we can check that all the mappings have valid argument and $\mathbf{v} \in \mathbb{R}^s$ is denoted in bold for clearness.

Proposition 8.9 (Equivariance for scalar Gauge convolution)**.** *Consider the operation $f = (\psi \star x)(p)$ of Definition 8.8, then:*

$$f \; equiv \iff \psi(\mathfrak{g}\boldsymbol{v}) = \psi(\boldsymbol{v}) \forall \mathfrak{g} \in \mathfrak{G}, \forall \boldsymbol{v} \in \mathbb{R}^s \tag{8.12}$$

Proof. A function $f : \mathcal{X}(\Omega, \mathbb{R}) \to \mathcal{X}(\Omega, \mathbb{R})$ is equivariant iff a transformation on the input is the same as a transformation on the output. Namely:

$$f \; equiv \iff f(\mathfrak{g}x) = f(x) \iff \rho(\mathfrak{g}) = 1 \tag{8.13}$$

Which trivially implies that a transformed filter has to not be impacted by the action of the group:

$$f \; equiv \iff \psi(\mathfrak{g}\mathbf{v}) = \psi(\mathbf{v}) \tag{8.14}$$

TODO check □

Definition 8.10 (Gauge Convolution on vectors)**.** Let Ω be a manifold with dimension s. Then, signals are of the form $x : \Omega \to \mathbb{R}^{C_{in}}$ for the input, and filters $\psi : \mathbb{R}^s \to \mathbb{R}^{C_{in} \times C_{out}}$. A gauge will link the canonical fiber $\mathcal{C} = \mathbb{R}^s$ to different tangent spaces. For a point $p \in \Omega$ the gauge is:

$$\omega_p : \mathbb{R}^s \to T_p\Omega \tag{8.15}$$

We require ω_p to map to isomorphisms (not canonical) and possibly to be smooth in p. If it is not smooth, which as we saw is very likely, the operations will be localized.
Let $\mathfrak{G}$ be the structure group, where $\omega_p^{-1}\omega_p' \in \mathfrak{G}$, and ρ_{in}, ρ_{out} be the representations of dimensions C_{in}, C_{out} Additionally, recover the exponential map of Definition 7.15, which in this case for a point $p \in \Omega$ is:

$$exp_p : T_p\Omega \to \Omega \tag{8.16}$$

And denote as $\mathfrak{g}_{\mathbf{v} \to p} \in \mathfrak{G}$ denote the parallel transport from $exp_u(\omega_u(\mathbf{v}))$ to p Then a global gauge convolution $(\rho_{in} \to \rho_{out})$ can be constructed as:

$$f = (\psi \star x)(p) = \int_{\mathbb{R}^s} \psi(\mathbf{v}) \rho_{in}(\mathfrak{g}_{\mathbf{v} \to p} x(exp_u(\omega_u(\mathbf{v})))d\mathbf{v} \tag{8.17}$$

Where we can check that all the mappings have valid argument and $\mathbf{v} \in \mathbb{R}^s$ is denoted in bold for clearness. The parallel transport requirement is implemented since the two objects now live in different fibers. For an example of such vector gauge, see Figure 8.13.

Proposition 8.11 (Equivariance for vector Gauge convolution)**.** *Consider the operation $f = (\psi \star x)(p)$ of Definition 8.10. Then:*

$$f \; equiv \iff \psi(\mathfrak{g}\boldsymbol{v}) = \rho_{out}(\mathfrak{g}) \psi(\boldsymbol{v}) \rho_{in}(\mathfrak{g})^{-1} \tag{8.18}$$

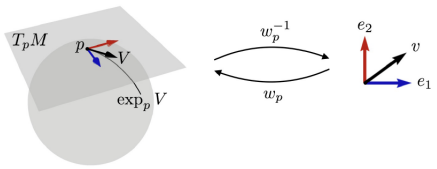


Figure 8.13: Vector Gauge Transformation

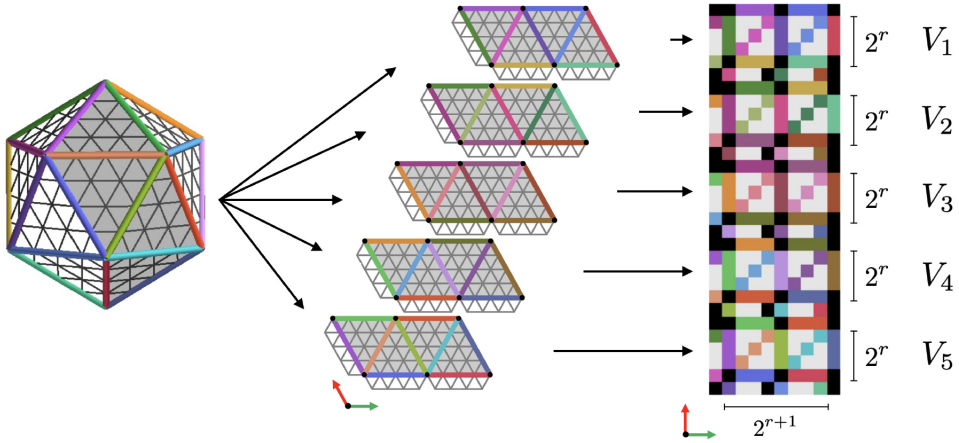


Figure 8.14: Charts and

Proof. We have that a function is equivariant for general inputs and outputs representations of dimensions C_{in} and C_{out} when it satisfies the conditions:

$$f \text{ equiv} \iff f \circ \rho_{in}(\mathfrak{g}) = \rho_{out}(\mathfrak{g}) \circ f \iff f = \rho_{out}^{-1}(\mathfrak{g}) \circ f \circ \rho_{in}(\mathfrak{g}) \tag{8.19}$$

Which if we see the filter ψ counterpart, implies that:

$$f \text{ equiv} \iff \psi(\mathbf{v}) = \rho_{out}(\mathfrak{g})\psi(\mathbf{v})\rho_{in}(\mathfrak{g})^{-1} \tag{8.20}$$

TODO □

Proposition 8.11 highlights a parallelism with group convolution nets (with steerable kernel) of Example 6.26, Equation 6.26. It is also possible to use a local gauge right away, but this method is not covered.

We now move to an *informal* example of how this procedure can be carried out.

Example 8.12 (Icosahedron)**.** An icosahedron as in Figure 8.11 is a good approximation of a spherical signals. For the purpose of the analysis, it is cut in charts of square pixel grids as in the left drawing of Figure 8.14 By construction, some pixels appear in multiple charts, and this will be crucial to define a gauge equivariant convolution. For each pixel, a canonical chart is chosen. From this canonical chart, their value will be copied before convolving. This method is often called $\mathfrak{G}$-padding. In particular, as shown in Figure 8.15 there are $k = 6$ rotations of angle $\frac{2\pi}{6}$ which make the structure group $\mathfrak{G} = C_6$ that of rotations. If a standard Conv2D was applied to the various faces, some blacked out neighbors would impact it. To alleviate this problem,

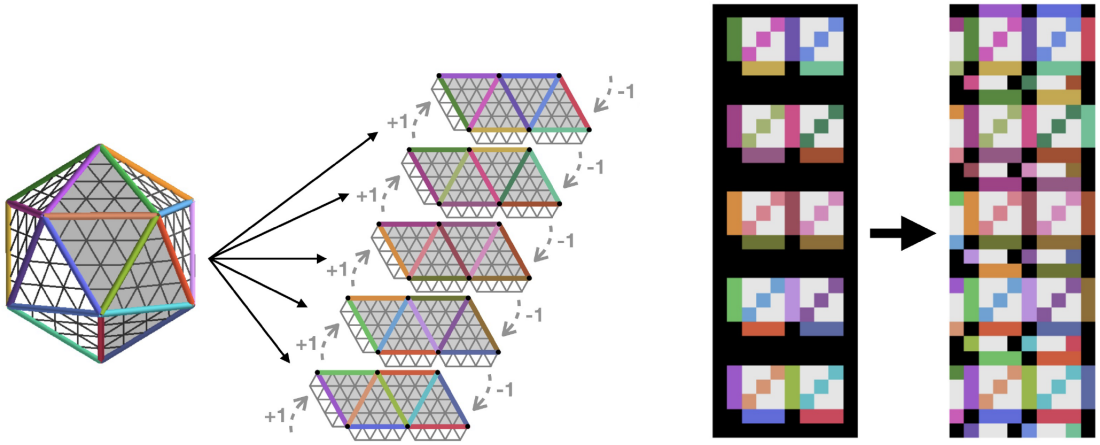


Figure 8.15: $\mathfrak{G}$-padding

some of those values are copied over (Figure 8.15, left).
Another problem is the kernel constraint, but this is not covered in the lecture and we reroute the reader to another source [Hoo+18].

While we have quickly presented one method, we stress that this is not the only path available. Another interesting approach is that of Gauge Equivariant Mesh CNNs, where the gauge is essentially interpreted as a reference neighbor with enforced equivariance [Haa+21]. On the theoretical side, some worthful mentions are also present [Wei+21; Ger+21; Aro21; Coh21].

8.2 General Theory of Equivariant CNNs Sketch

In this section, quick theory facts for Equivariant CNNs are proposed. The theory of Fiber Bundels is vast and much more complicated than what is reported in this document.

The main goal of such field has been that of being able to deal with convolutional networks on many domains and symmetries, to obtain a generic theorem with universality results for linear equivariant layers.

Definition 8.13 (Fiber Bundles, terminology)**.** Let $\Omega = B$ be the base space. Fibers are denoted as F, and are typically isomorphic. The base space and fibers together form the total space E. See Figure 8.16 for a graphical example.

Definition 8.14 (Bundles π)**.** A bundle is a continuous/smooth map between topological/smooth spaces:

$$\pi : E \to \Omega \tag{8.21}$$

It essentially projects elements of the toatl space to their belonging origin on the base space.

Definition 8.15 (Fiber F)**.** For a given element $u \in \Omega$ and a Bundle as in Definition 8.14, a Fiber is a the inverse image of the bundle:

$$F_u = E_u = \pi^{-1}(u) \quad u \in \Omega \tag{8.22}$$

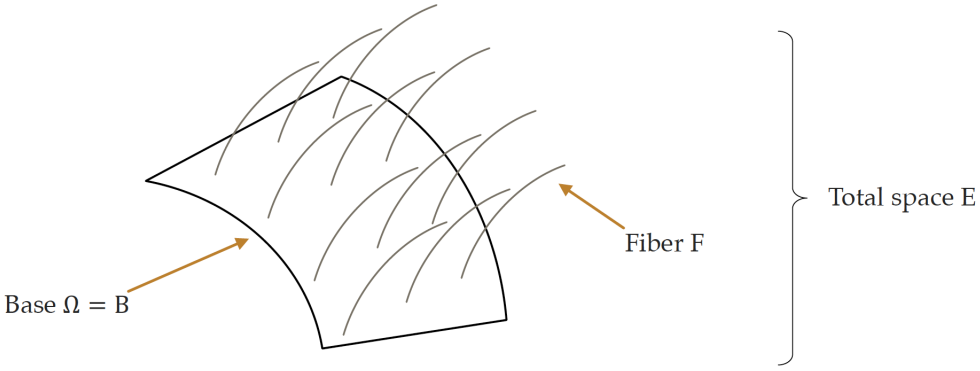


Figure 8.16: Base, Fibers, Total Space

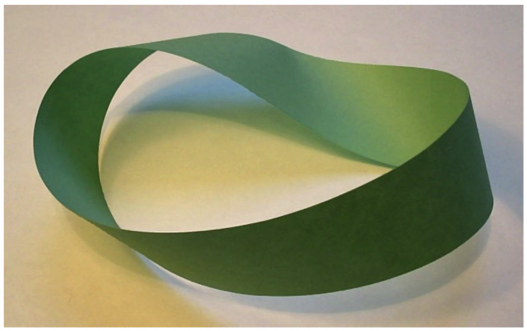


Figure 8.17: Non trivial bundle, the Möbius strip

For Fiber bundles the assumptions provide clarity and strucutre for further results.

Assumption 8.16 (On Fiber Bundles)**.** It is required that:

- fibers are isomorphic
- bundles π are *locally* trivial, so locally $E \cong Omega \times F$. Not necessarily globally.

For an example of a non globally trivial bundle, consider as base space a circle and as fibers line segments. A globally trivial bundle would be a cylinder. On the contrary, a non trivial bundle would be Figure 8.17.

Definition 8.17 (Section σ)**.** A section is a function from the base space to the fiber, where the codomain E depends on the input Ω. Namely:

$$\sigma : \Omega \to E \quad s.t. \quad \pi \circ \sigma = id_\Omega \tag{8.23}$$

It is an important notion, which for each fiber returns any input arising from u, constrained to getting the identity in the base space once composed with the bundle π. Consider as an example the σ map returning the base of each of the yellow point in the fibers of Figure 8.18.

Two kinds of bundles are *informally* considered:

- **Principal Bundle**, a bundle of frames where:

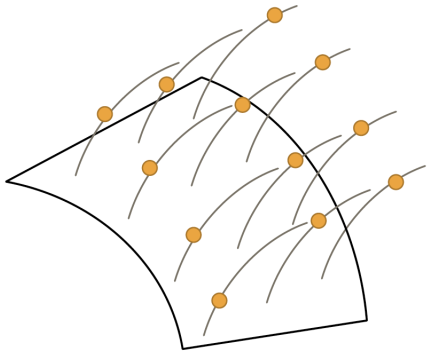


Figure 8.18: Sections σ, yellow to base

  - frames (references) are related by $!\mathfrak{g} \in \mathfrak{G}$ where $\mathfrak{G}$ is the structure group
  - sections are gauges, as choosing a section (a map element$\rightarrow$ element of the fiber) is equivalent to choosing a gauge.
- **Associated Bundle**, a vector bundle where there are vector spaces replacing fibers and:
  - fibers transform according to ρ the representation of $\mathfrak{G}$
  - correspondence at a vector space level is a change of frame (reference) of the vector space
  - sections are fields.

Definition 8.18 (Principal Bundle)**.** A principal bundle (bundle of frames) is a bundle P as in Definition 8.14 where additionally:

- fibers are preserved up to transformation

$$\pi(\mathfrak{g}p) = \pi(p) \tag{8.24}$$

- transitivity across fibers

$$\forall p, p' \in P_u \; fiber \exists \mathfrak{g} \mid \mathfrak{g}p = p' \tag{8.25}$$

- fixed point free

$$p = \mathfrak{g}p \implies \mathfrak{g} = e \tag{8.26}$$

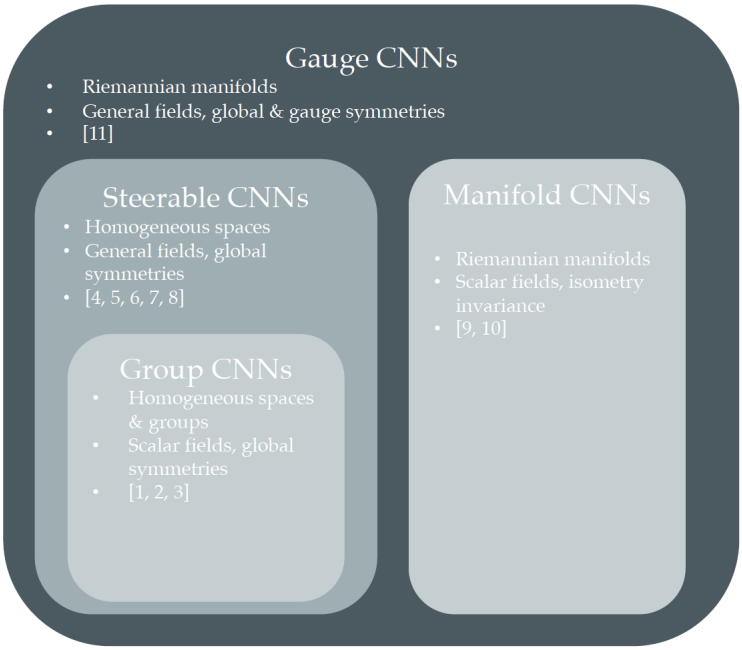
Where the fibers of P are such that $P_u \cong \mathfrak{G}$, except that there is no preferred origin on the fibers.

Example 8.19 (Principal Bundles in practice)**.** The easiest possible examples could be:

- Consider the frame bundle of a manifold as its tangent spaces
- See $\mathfrak{H}$ as the principal $\mathfrak{G}$-bundle over $\mathfrak{G}/\mathfrak{H}$ (e.g. SO(2) as a principal SO(3) bundle over S^2) of Figure 6.18.

All of the discussed notions of symmetry, from global to gauge to manifold can be described as principal bundle automorphisms, through some symmetry. Thanks to this a universality result for generalized CNNs can be proved, which guarantees that equivariant linear maps can be expressed as convolution-like integrals with a constrained to invariance kernel [Coh21].

The general theory of Gauge CNNs encapsulates many CNN architectures discussed in previous Chapters, as Figure 8.19 shows



1. Cohen & Welling, Group Equivariant Convolutional Networks, ICML 2016.
2. Cohen, Geiger, Koehler & Welling, Spherical CNNs, ICLR 2018.
3. Kondor & Trivedi, On the Generalization of Equivariance and Convolution in Neural Networks to the Action of Compact Groups, ICML 2018.
4. Cohen & Welling, Steerable CNNs, ICLR 2017.
5. Worrall, Garbin, Turmukhambetov, & Brostow, Harmonic Networks: Deep Translation and Rotation Equivariance, *CVPR 2017*.
6. Thomas, Smidt, Kearnes, Yang, Li, Kohlhoff, Riley. *Tensor Field Networks: Rotation- and Translation-Equivariant Neural Networks for 3D Point Clouds, 2018.*
7. *Cohen, Geiger, Weiler, Intertwiners between Induced Representations, 2018*
8. *Cohen, Geiger, Weiler, A General Theory of Equivariant CNNs on Homogeneous Spaces, 2018*
9. Boscaini, D., Masci, J., Melzi, S., Bronstein, M. M., Castellani, U., and Vandergheynst, P. Learning class-specific descriptors for deformable shapes using localized spectral convolutional networks. Computer Graphics Forum, 2015
10. Masci, Boscaini, Bronstein & Vandergheynst. Geodesic convolutional neural networks on Riemannian manifolds, ICCVW, 2015
11. Cohen, Weiler, Kicanaoglu & Welling, Gauge Equivariant Convolutional Networks and the Icosahedral CNN, ICML 2019.

Figure 8.19: The power of Gauge CNNs

Chapter 9

Sequences & Time-Warping

In this Chapter we will deal with the problem of processing data as sequences, where the concept of time warping leads naturally to Recurrent Neural Networks (RNNs). We will show that in this fashion, a popular class of RNNs arises from invariance.

In previous Chapters, we explained how from the GDL Blueprint of Definition 3.43 it is possible to derive several architectures. Though conceptually more complicated, sequential data obeys the same rules.

So far, we assumed some sort of *spatial* geometry (e.g. grids, graphs,...), but what about a *sequential* geometry? Video, audio, text and speech are cases of interest for Machine Learning.

9.1 Sequential Problem Setup

For the purpose of formalizing, we will assume that now information comes as a time governed sequence where signals belong to time dependent sets $X^{(t)} \in \mathcal{X}(\Omega^{(t)})$ on a discrete input space. By construction, time t only goes in one direction, and seems inherently asymmetric for this reason. The aim is identifying symmetries (if any) without counting those arising at a fixed t (i.e. symmetries of *snapshots* of signals). As to lighten up the discussion, one important assumption will be made throughout the analysis.

Assumption 9.1 (Fixed Domain)**.** The domain of signals is static over time.

$$\Omega^{(t)} = \Omega \quad \forall t \tag{9.1}$$

While Assumption 9.1 might seem to imply an inductive bias, it is worth noticing that for most problems it is an almost perfect approximation, which loses power only as the time scale gets significantly big. For a treatment of fully dynamic problems, the field of *Temporal Graph Networks* is exciting and promising [Ros+20].

Example 9.2 (Road traffic Domain)**.** For the problem of road traffic estimation, it is fairly reasonable to consider $\Omega^{(t)} = \Omega$ over time, even though it is necessarily true that roads are built every day. In the event in which the time window was enormous, such an assumption would be flawed by inconsistency of the domain and the data observed many years apart(e.g. roads in 1000 AD and in 2022).

Figure 9.1: Sequential inputs, traffic data

The need to *abstract* away *snapshot* symmetries on Ω is satisfied thanks to the distinction of learning at fixed time and learning over time functions.

Definition 9.3 (Sequential result $\mathbf{z}^{(t)}$)**.** For a sequential task, learn an encoder function over signals:

$$f : \mathcal{X}(\Omega) \to \mathbb{R}^k \tag{9.2}$$

Which, depending on Ω is in line with the invariance & equivariance requirements of the Blueprint of Definition 3.43. In particular, it outputs k dimensional vectors:

$$\mathbf{z}^{(t)} = f(X^{(t)}) \tag{9.3}$$

Instead, if f represents an intermediate or non purposely invariant layer, the output will be a matrix $Z^{(t)}$, where f is only equivariant (i.e. not aggregated).

Given $\mathbf{z}^{(t)}$ from Definition 9.3, the domain of interest is clearly a 1D (oriented) grid, which tracks the evolution of the response $\mathbf{z}^{(t)}$ over the arrow of time. Equivariance is thus interpreted as **translation equivariance**. Being a vector, it can be processed by any of the previously introduced architectures on Grids such as CNNs.

However, the objective to mimic the arrow of time, *online* and *efficiently*, is not achieved by such architectures. We will resort to another technique!

Definition 9.4 (RNN Architecture)**.** Consider an **update function** R where:

$$R : \mathbb{R}^k \times \mathbb{R}^m \to \mathbb{R}^m \tag{9.4}$$

Which computes **summaries** $\mathbf{h}^{(t)} \in \mathbb{R}^m$ as:

$$\mathbf{h}^{(t)} = R(\mathbf{z}^{(t)}, \mathbf{h}^{(t-1)}) \tag{9.5}$$

Where the standard starting point is $\mathbf{h}^{(0)} = \mathbf{0}$.
Update sequentially layers with R taking inputs over time, and learn its coefficients.

Example 9.5 (Road Traffic RNN)**.** Consider a sequential set of inputs as in Figure 9.1. Apply a function f to obtain sequential results $\{\mathbf{z}^{(t)}\}_t$ as in Figure 9.2. For each t iteration, build the next summary sequentially by feeding an update function R with inputs $\mathbf{z}^{(t)}, \mathbf{h}^{(t-1)}$ as in Figure 9.3.

A special choice of $\mathbf{h}^{(0)}$ can impact considerably RNNs. Nevertheless, it is not sufficient to guarantee translation equivariance.

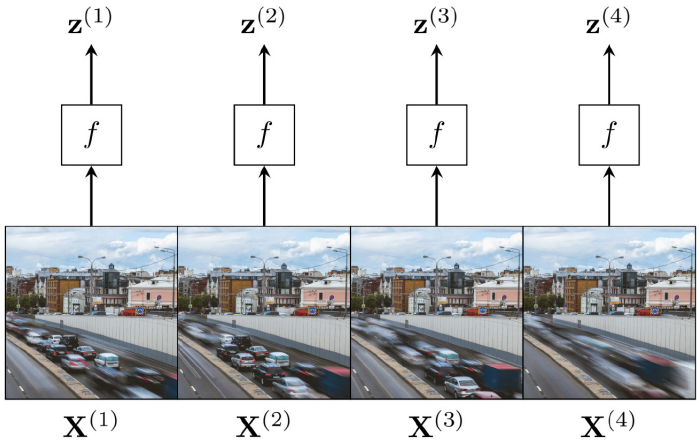


Figure 9.2: Sequential results $\{\mathbf{z}^{(t)}\}_t$ computation

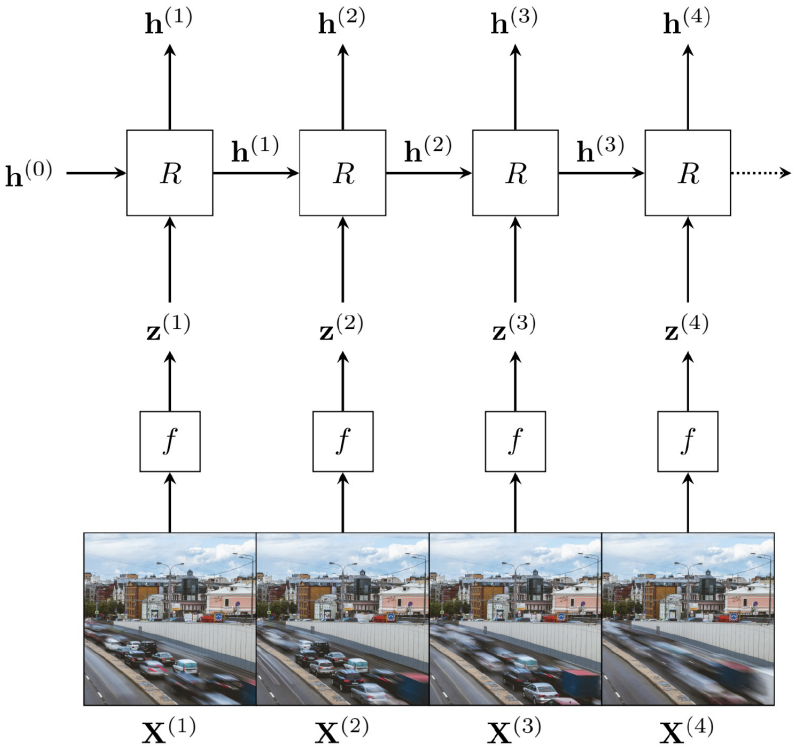


Figure 9.3: Summaries $\{\mathbf{h}^{(t)}\}_t$ computation

Proposition 9.6 (RNNs are not translation equivariant)**.** *Consider a sequential result* $\boldsymbol{z}^{(t)}$ *and a left shift transformation* $\boldsymbol{z}'^{(t)}$*. Then:*

$$\boldsymbol{z}^{(t+1)} = \boldsymbol{z}'^{(t)} \implies \boldsymbol{h}^{(t+1)} = \boldsymbol{h}'^{(t)} \tag{9.6}$$

At least in a ***feasible*** *manner.*

Proof. We have that for $t = 1$:

$$\begin{aligned} \mathbf{h}'^{(1)} = R(\mathbf{z}'^{(1)}, \mathbf{h}'^{(0)}) = R(\mathbf{z}'^{(1)}, \mathbf{h}^{(0)}) \\ \mathbf{h}^{(2)} = R(\mathbf{z}^{(2)}, \mathbf{h}^{(1)}) \end{aligned}$$

Imposing equivariance $\mathbf{h}^{(t+1)} = \mathbf{h}'^{(t)}$ and using $\mathbf{z}^{(t+1)} = \mathbf{z}'^{(t)}$:

$$\begin{aligned} \mathbf{h}'^{(1)} = \mathbf{h}^{(2)} &\iff R(\mathbf{z}'^{(1)}, \mathbf{h}'^{(0)}) = R(\mathbf{z}^{(2)}, \mathbf{h}^{(1)}) = R(\mathbf{z}'^{(1)}, \mathbf{h}^{(1)}) \\ &\iff \mathbf{h}'^{(0)} = \mathbf{h}^{(0)} = \mathbf{h}^{(1)} = R(\mathbf{z}^{(1)}, \mathbf{h}^{(0)}) \end{aligned}$$

Where the second implication is by setting the second arguments of the R function equal. We conclude that the requirement is:

$$\mathbf{h}^{(0)} = R(\mathbf{z}^{(1)}, \mathbf{h}^{(0)}) \tag{9.7}$$

Which is **not feasible** as it is a quantity decided *a priori* and not determined by $\mathbf{z}^{(1)}$, which gets deleted by the left shift (condition $\mathbf{z}^{(t+1)} = \mathbf{z}'^{(t)}$). Another option is defining a left padded sequence by t' steps:

$$\overline{\mathbf{z}}^{(t)} = \begin{cases} \mathbf{0} & t \leq t' \\ \mathbf{z}^{(t-t')} & t > t' \end{cases} \tag{9.8}$$

It is possible to shift $\overline{\mathbf{z}}^{(t)}$ by at most t' without losing its values. Yet by imposing the same condition:

$$\mathbf{h}'^{(1)} = R(\overline{\mathbf{z}'}^{(1)}, \mathbf{h'}^{(0)}) = R(\overline{\mathbf{z}}^{(2)}, \mathbf{h}^{(0)}) = R\Big(\overline{\mathbf{z}}^{(2)}, R(\overline{\mathbf{z}}^{(1)}, \mathbf{h}^{(0)})\Big) = R(\overline{\mathbf{z}}^{(2)}, \mathbf{h}^{(1)}) = \mathbf{h}^{(2)}$$

Which looking at the central equations (second arguments) implies that:

$$\iff \mathbf{h}^{(0)} = R(\overline{z}^{(1)}, \mathbf{h}^{(0)}) \tag{9.9}$$

Which for $t' > 1$ is such of the form:

$$\iff \mathbf{h}^{(0)} = R(0, \mathbf{h}^{(0)}) \tag{9.10}$$

Implying that the R function must return the second input conditioned on having a 0 in the first input. **TODO check**
Again, while it might be sufficient for $t = 0$, it is not a valid trick when $t > t'$ and information starts to get lost, provided that the memory size of such array is not infinite. □

Time sequences do not make RNNs equivariant to translations as naturally as CNNs. The question moves to understanding the strengths of such method apart from its *online* capabilities.

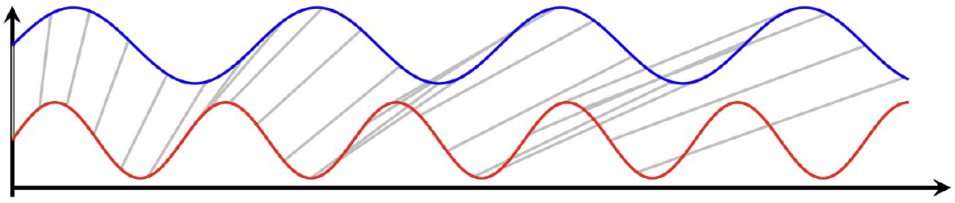


Figure 9.4: Time scales, original (red), warped (blue)

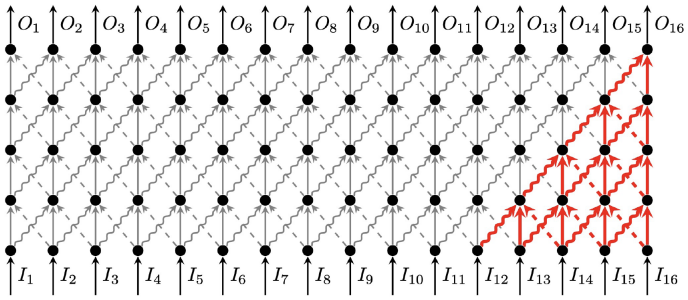


Figure 9.5: CNN with 4 layers

If setup properly, RNNs support a useful kind of symmetry. Data as a series can be interpreted as a sample from a continuous signal. While in some cases the rate is fixed, this is not guaranteed in general. Controlling the sampling rate *dynamically* and analyzing how the model reacts is crucial to understand the hidden strengths of RNNs.

The action of changing the sampling rate is basically a change of units of measurement. The concept of time warping formalizes such idea.

Definition 9.7 (Time warping function $\tau(\cdot)$)**.** Let time t be a continuous variable. A time warping function is a **monotically increasing and differentiable** transformation of the form:

$$\tau : \mathbb{R}^+ \to \mathbb{R}^+ \tag{9.11}$$

It can also be seen as an **automorphism**.

Example 9.8 (Easiest possible time warping: Time rescaling)**.** Consider $\tau(t) = \frac{7}{10}t$ as a function. In Figure 9.4 the original function is in red and the warped wave is in blue.

What we ultimately wish is finding an invariant to time warping class of functions, such that a change of unit of measurements does not impact the output of a sequence.

Observation 9.9 (CNNs are **not** warping invariant!)**.** *Consider a CNN with kernel size* 3, $T = 16$ *steps and* 4 *layers. The output at* O_{16} *sees inputs* $\{I_{12}, \dots, I_{16}\}$*, as Figure 9.5 shows. With time warping, such discretization is destroyed by the zeros added between inputs (e.g. imagine dilating time intervals).*

An option to solve such issue is Dilated CNNs where as layers group kernels are made more far apart (see Figure 9.6. Such a trick allows O_{16} *to have access to the whole sequence of inputs.*

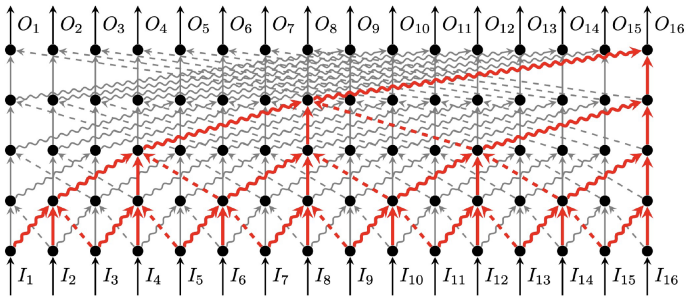


Figure 9.6: Dilated CNN

However, this exponentially increasing receptive field is still fixed and thus not immune to a very big dilation that would completely break the input sequence apart.

While we have shown that CNNs do not offer the right tools, it is still practically possible to use a Transformer. Yet, it comes with no theoretical guarantees. A transformer operates as a fully connected graph, with no order, where implicit features are added to cope with this deficiency. Also, the arising symmetry is permutation, not translation. It is easy to notice that this is not the framework we wish to work on, and even though results can be good, they are not justified by theory.

9.2 Neural Networks & Time Warping

The basis of this section is a paper which shed a light on the necessary pieces of architecture for a Neural Network to behave well in presence of time warping [TO18]. Under certain constraints, invariance to time warping is achieved.

Since warping is well defined for continuous data and more convoluted for discrete data, we start from the easy case, and imagine that also our RNN is somewhat adapted to continuity.

Lemma 9.10 (Continuous RNN warp invariance Condition)**.** *For $z(t), h(t)$ continuous a continuous RNN is invariant if and only if (consider dimesion 1 for simplicity):*

$$\frac{dh(\tau(t))}{dt} = \frac{d\tau(t)}{dt} R\left[z\Big(\tau(t+1)\Big), h\Big(\tau(t)\Big)\right] - \frac{d\tau(t)}{dt} h(\tau(t)) \tag{9.12}$$

Proof. Consider the summary function $h(t)$ and its Taylor expansion:

$$h(t+\delta) \approx h(t) + \delta \frac{dh(t)}{dt} \tag{9.13}$$

If we set $\delta = 1$ we obtain the 1 step approximation between $h(t)$ and $h(t+1)$ which will be useful for the discrete case. Moreover, it is coincident to the argument-result distance for the R function where $h(t+1) = R[z(t+1), h(t)]$.
Ignoring the approximation term we get an ODE that must be satisfied for the RNN:

$$h(t+1) = h(t) + \frac{dh(t)}{dt} = R[z(t+1), h(t)] \implies \frac{dh(t)}{dt} = R[z(t+1), h(t)] - h(t) \tag{9.14}$$

If time is warped by $\tau(t)$ the ODE becomes:

$$\frac{dh(\tau(t))}{d\tau(t)} = R\Big[z\Big(\tau(t+1)\Big), h\Big(\tau(t)\Big)\Big] - h\Big(\tau(t)\Big) \tag{9.15}$$

Which imposes the condition $h\Big(\tau(t+1)\Big) = R\Big[z\Big(\tau(t+1)\Big), h\Big(\tau(t)\Big)\Big]$. Applying the chain rule on the warped ODE it is possible to express the condition in terms of the original time scale:

$$\frac{dh(\tau(t))}{dt} = \frac{dh(\tau(t))}{d\tau(t)}\frac{d\tau(t)}{dt} \tag{9.16}$$

Which substituted into Equation 9.15 becomes:

$$\frac{dh(\tau(t))}{dt} = \frac{d\tau(t)}{dt}R\Big[z\Big(\tau(t+1)\Big), h\Big(\tau(t)\Big)\Big] - \frac{d\tau(t)}{dt}h\Big(\tau(t)\Big) \tag{9.17}$$

Which is the result of the claim. □

Having an idea of what the continuous version could be, a similar approach is used to derive the result for a discretized RNN.

Theorem 9.11 (Discrete RNN warp invariance Condition)**.** *The necessary update for a time warping invariant RNN is:*

$$\boldsymbol{h}^{(t+1)} = \frac{d\tau(t)}{dt}R\Big(\boldsymbol{z}^{(t+1)}, \boldsymbol{h}^{(t)}\Big) + \Big(1 - \frac{d\tau(t)}{dt}\Big)\boldsymbol{h}^{(t)} \tag{9.18}$$

Proof. Consider the Taylor expansion of the warped summary where we let $\delta = 1$:

$$h(\tau(t+\delta)) \approx h(\tau(t)) + \delta\frac{d\tau(t)}{dt}) \approx h(\tau(t)) + \delta\frac{d\tau(t)}{dt}\frac{dh\tau(t)}{d\tau(t)} = h(\tau(t)) + \delta\frac{dh(\tau(t))}{dt} \tag{9.19}$$

$$\overset{\delta=1}{\Longrightarrow} \; h(\tau(t+1)) \approx h(\tau(t)) + \frac{dh(\tau(t))}{dt} \tag{9.20}$$

Notice that a discrete RNN is such that $\mathbf{z}^{(t)} = z(\tau(t))$ and $\mathbf{h}^{(t)} = h(\tau(t))$. Then invariance is guaranteed by

$$\mathbf{h}^{(t+1)} = h(\tau(t+1)) = R\Big(z(\tau(t+1)), h(\tau(t))\Big) = R\Big(\mathbf{z}^{(t+1)}, \mathbf{h}^{(t)}\Big)$$

Consider the two central results. Substituting $h\Big(\tau(t+1)\Big)$ with the result of Equation 9.19 we get that:

$$h\Big(\tau(t+1)\Big) = h\Big(\tau(t)\Big) + \frac{dh(\tau(t))}{dt} \tag{9.21}$$

$$= h\Big(\tau(t)\Big) + \frac{d\tau(t)}{dt}R\Big[z\Big(\tau(t+1)\Big), h\Big(\tau(t)\Big)\Big] - \frac{d\tau(t)}{dt}h\Big(\tau(t)\Big) \quad \text{Lemma 9.10} \tag{9.22}$$

$$= \frac{d\tau(t)}{dt}R\Big[z\Big(\tau(t+1)\Big), h\Big(\tau(t)\Big)\Big] + \Big(1 - \frac{d\tau(t)}{dt})h\Big(\tau(t)\Big) \quad \text{reordering} \tag{9.23}$$

$$\mathbf{h}^{(t+1)} = \frac{d\tau(t)}{dt}R\Big(\mathbf{z}^{(t+1)}, \mathbf{h}^{(t)}\Big) + \Big(1 - \frac{d\tau(t)}{dt}\Big)\mathbf{h}^{(t)} \quad \text{bold version} \tag{9.24}$$

And the claim is proved. □

A possible interpretation of such result is that the warping derivative with respect to time $\frac{d\tau(t)}{dt}$ bound how much of $\mathbf{h}^{(t)}$ is rewritten at each step.

The result of Theorem 9.11 is helpful to discern between theoretically justified models and those that are not.

Simple RNNs [Jor97; Elm90] implement an MLP to estimate summaries. Let **U,W,b** be learnable parameters and σ an activation function:

$$\mathbf{h}^{(t+1)} = \sigma\left(\mathbf{W}\mathbf{z}^{(t+1)} + \mathbf{U}\mathbf{h}^{(t)} + \mathbf{b}\right) \tag{9.25}$$

Where no explicit coefficient bounds possible warpings affecting $\mathbf{h}^{(t)}$. Thus, the model implicitly assumes $\frac{d\tau(t)}{dt} = 1$ and is not warping invariant.

Starting from this result, since the warping derivative $\frac{d\tau(t)}{dt}$ is unknown *a priori*, it is possible to attach to the architecture of *Simple RNNs* a function approximating the warping derivative.

Definition 9.12 (Gated RNN)**.** Consider a derivative estimator matrix:

$$\Gamma : \mathbb{R}^k \times \mathbb{R}^m \to \mathbb{R} \quad \Gamma(\mathbf{z}^{(t+1)}, \mathbf{h}^{(t)}) \tag{9.26}$$

Where the inputs are $\mathbf{z}^{(t+1)}, \mathbf{h}^{(t)}$, basically all the knowledge available at time t to estimate $t+1$. Define a time warping invariant CNN as one that builds new summaries following:

$$\mathbf{h}^{(t+1)} = \Gamma(\mathbf{z}^{(t+1)}, \mathbf{h}^{(t)})R(\mathbf{z}^{(t+1)}, \mathbf{h}^{(t)}) + (1 - \Gamma(\mathbf{z}^{(t+1)}, \mathbf{h}^{(t)}))\mathbf{h}^{(t)} \tag{9.27}$$

Where the Γ function is constrained such that: $\frac{d\tau(t)}{dt} \preceq 1$ as to have a *not so big* Taylor approximation error and avoid *overcontractions* of time. By the monotonically increasing assumption of Definition 9.7, the final constraint on Γ is:

$$0 < \Gamma(\mathbf{z}^{(t+1)}, \mathbf{h}^{(t)}) < 1 \tag{9.28}$$

Which is equivalent to the requirements of a **gating mechanism**! Those are usually estimated as:

$$\Gamma(\mathbf{z}^{(t+1)}, \mathbf{h}^{(t)}) = logit\left(\mathbf{W}_\Gamma \mathbf{z}^{(t+1)}, \mathbf{U}_\Gamma \mathbf{h}^{(t)} + \mathbf{b}_\Gamma\right) \tag{9.29}$$

Where $\mathbf{W}_\Gamma, \mathbf{U}_\Gamma, \mathbf{b}_\Gamma$ are learnable parameters.

Observation 9.13 (On the Γ of Definition 9.12)**.** *It is also possible to make* Γ *output a vector by a multinomial logistic function and update different entries of* $\boldsymbol{h}^{(t+1)}$ *separately! This increases expressivity by separate dimension updates.*

Gated RNNs are widely used, and dominant in the subfield of RNNs. Models following this fashion include:

- Long Short Term Memory (LSTM) [HS97]

- Gated Recurrent Unit (GRU) [Cho+14]

In this Chapter, we will brielfy overview LSTM. Such model is the oldest and most popular. It was originally proposed to fix the vaninshing gradient problem of many activation functions in Machine Learning. It decides how much to forget based on data (it is a *data driven method*). It uses a persistent **memory cell** $\mathbf{c}^{(t)}$, and uses gating to make the following update choices:

- how much of the new candidate to transfer to the cell $\mathbf{i}^{(t)}$
- how much of the previous to forget $\mathbf{f}^{(t)}$
- how much of the output to allow out $\mathbf{o}^{(t)}$

The memory cell has a state $\mathbf{c}^{(t)}$ and the workflow on its updates is as follows:

1. Compute candidate features vector for the cell:

$$\widetilde{\mathbf{c}}^{(t)} = \tanh(\mathbf{W}_c\mathbf{z}^{(t)} + \mathbf{U}_c\mathbf{h}^{(t-1)} + \mathbf{b}_c) \tag{9.30}$$

2. Compute the input/output/forget gates:

$$\mathbf{i}^{(t)} = logit(\mathbf{W}_i\mathbf{z}^{(t)} + \mathbf{U}_i\mathbf{h}^{(t-1)} + \mathbf{b}_i) \tag{9.31}$$
$$\mathbf{f}^{(t)} = logit(\mathbf{W}_f\mathbf{z}^{(t)} + \mathbf{U}_f\mathbf{h}^{(t-1)} + \mathbf{b}_f) \tag{9.32}$$
$$\mathbf{o}^{(t)} = logit(\mathbf{W}_o\mathbf{z}^{(t)} + \mathbf{U}_o\mathbf{h}^{(t-1)} + \mathbf{b}_o) \tag{9.33}$$

3. Derive the new cell state and update the summary:

$$\mathbf{c}^{(t)} = \mathbf{i}^{(t)} \bigodot \widetilde{\mathbf{c}}^{(t)} + \mathbf{f}^{(t)} \bigodot \mathbf{c}^{(t-1)} \tag{9.34}$$
$$\mathbf{h}^{(t)} = \mathbf{o}^{(t)} \bigodot \tanh(\mathbf{c}^{(t)}) \tag{9.35}$$

An example of such architecture is found in Figure 9.7

While LSTM is a special case of a the wider set of *Gated RNNs*, we have justified what the gating mechanism theoretically guarantees: invariance to non contractive time warpings. Specifically, the aim of a gating mechanism is mimicking the condition of Theorem 9.11, as to guarantee this symmetry property for sequential data.

This set of results implies a slightly different kind of invariance: **class invariance**. We are indeed showing that if an RNN produces $h(t)$ from $z(t)$ then another one, with possibly different coefficients, but the **same architecture**, will be able to produce $h(\tau(t))$ from $z(\tau(t))$ for any τ.

More interestingly, a case of **zero shot transfer** is when an easy rescaling as in Example 9.8 from τ_1 to τ_2 is implemented. In this context, it is possible to assert that:

$$\tau_1(t) = \alpha\tau_2(t) \implies \Gamma_2(\mathbf{z}^{(t+1)}, \mathbf{h}^{(t)}) = \alpha\Gamma_1(\mathbf{z}^{(t+1)}, \mathbf{h}^{(t)}) \tag{9.36}$$

To match the derivative change!

Thus, gated RNNs support translation equivariance by setting invariance to time warping. Now that we have covered the most important architectures arising from GDL, we can eventually say that we checked all the cells in Table 1.1!

The next Chapter will present applications and possible developments of GDL.

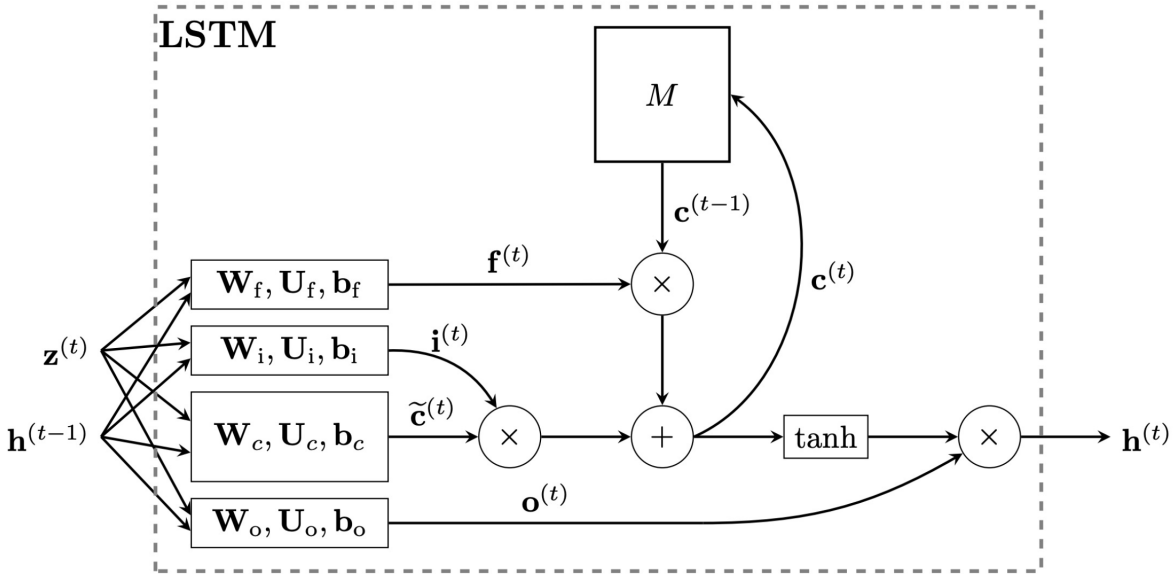


Figure 9.7: LSTM Architecture

Chapter 10

Applications & Conclusions

In this set of lectures, we saw how the key concept unifying deep learning is **symmetry**. We have seen instances on the 5Gs (Figure 10.1) and derived popular architectures from first principles. A summary of the Neural Networks encountered is Figure 10.2.

A good question could be:

What is next?

Indeed, the quality of a theory can only be judged by its predictive power. This Chapter is roughly based on a blog post by the main author of the course [Bro20]. We will explore potential advancements in the field and applications where GNNs proved to be well versed. Most of the content will just redirect to current research without much narrative. It shoudl rather be taken as an inspiration on where to look for new/interesting content.

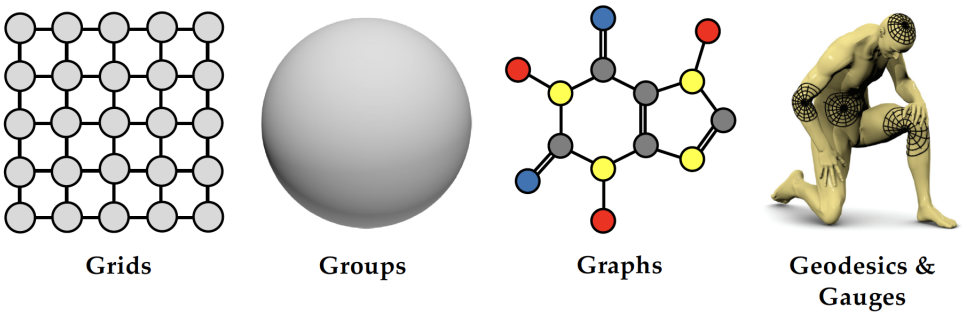


Figure 10.1: Five Gs of Geometric Deep Learning

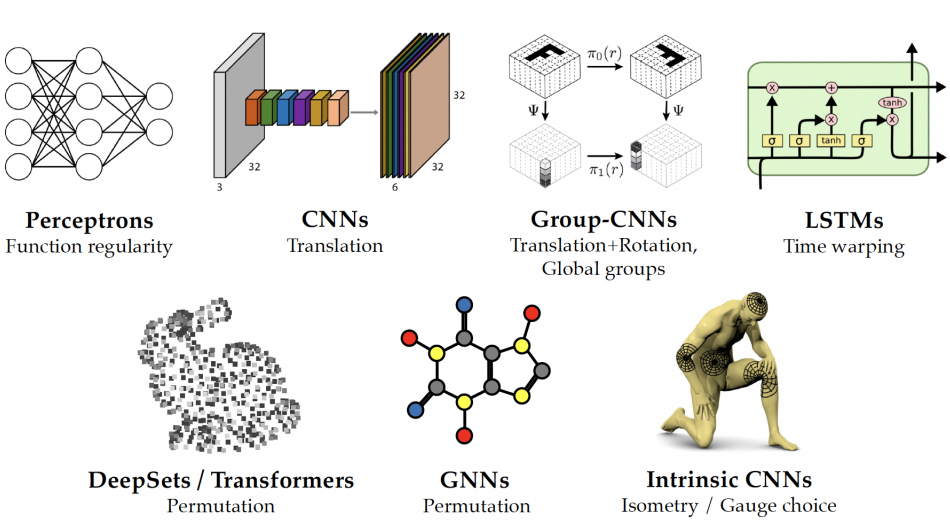


Figure 10.2: GDL architectures and symmetry groups

10.1 Advancing GDL

> *2020 saw the field of Graph ML come to terms with the fundamental limitations of the message-passing paradigm.*
>
> Will Hamilton, Mila

This quote by a great researcher highlights an intrinsic problem in both the world of Graph Machine Learning and message-passing. There are several papers that propose Weisfeler Lehman like (Algorithm 4.2.3) schemes for simplicial complexes, which themselves generalize meshes, implementing a discrete topology with many interesting theoretical results, including strict over-peformance of MPNNS[1] [Bod+21].

Another interesting direction is *Explainable GNNs*. Instead of a generic function, a symbolic function regressed on the results of a small MLP as message passing can be used [Cra+20].

Such context is also deeply related with Algorithmic Reasoning. The idea is the following: we have a very established way to work with graphs through algorithmic procedure (e.g. Djikstra's for Shortest paths). Algorithms are explainable but not generalize. On the contrary, Neural Networks are *not always* explainable and often generalize well across tasks. What algorithmic reasoning aims at doing is to take the advantages of both worlds. For further discussion, refer to [VB21].

Causal Inference is another field of interest. Causal graphs such as gene regulatory networks (Figure 10.3) can be learnt by a Neural Network [Bel+21].
On a similar note, knowledge graphs (e.g. 10.4), as heterogeneous graphs highlighting relations between objects could be a hot topic.

[1] Message-Passing Neural Networks

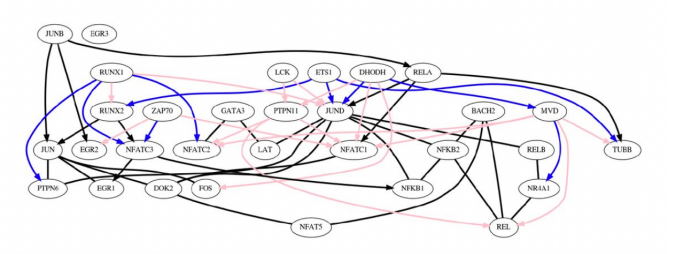


Figure 10.3: A Gene Regulatory Network

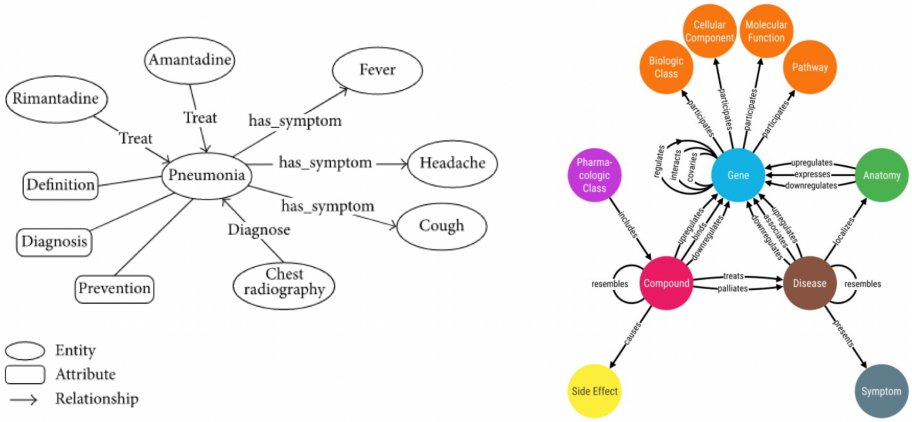


Figure 10.4: Two Knowledge Graphs

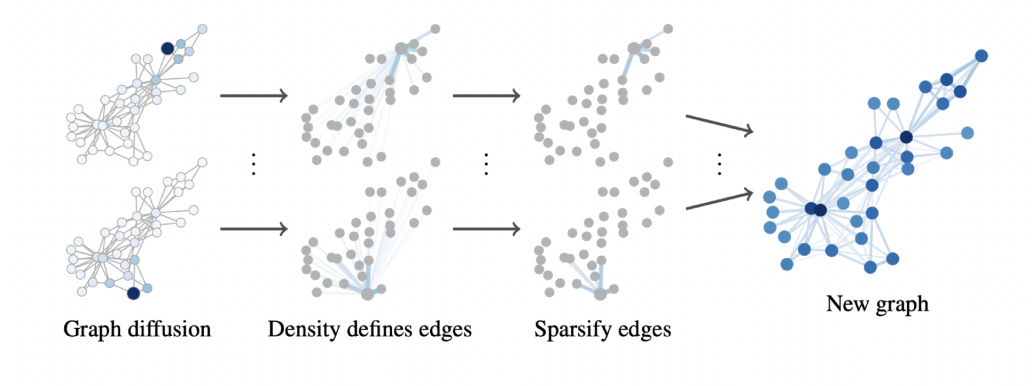


Figure 10.5: Graph Rewiring

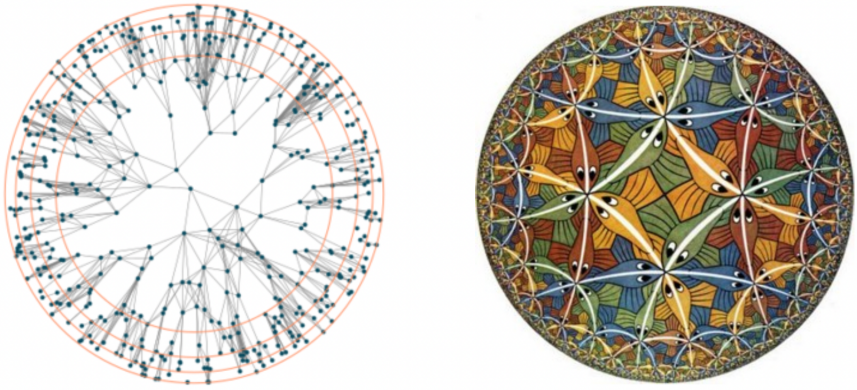


Figure 10.6: Network Geometry

Another great example of theoretical development is *Graph Rewiring*. Essentially, a graph is embedded with a diffusion model and *rewired* guaranteeing better performance [GWG22]. Figure 10.5 is an outline of the procedure.

Considering graph as discretizations of a continous space is the basis of Network Geometry (a visual example is Figure 10.6), which joins Differential Geometry and Graph Theory. There are available reviews of research directions in the field [Bog+21].

Coming back to diffusion, Neural Networks are often seen as discretizations of diffusion equations (Neural PDEs). A Network eventually becomes a specific step of Euler solvers, where layers are discretizations of the diffusion time. Nice questions could be understanding how to discretize a diffusion derivative and determining which graph would arise [Cha+21].

Linked to problems and Assumptions made in Chapter 9, what if the domain $\Omega^{(t)}$ is not fixed over time? In such a context, it is possible to study dynamic graphs, which are non static and have different change frequencies [Ros+22].

Very famous recently, generative models such as GANS have experienced a steep improvement in their results. Figure 10.7 is a proof of this claim. Despite promising performance for images, they are still very imprecise with graphs. The issue is believed to be the absence of a canonical order of nodes and features, which is inherent in the definition of a graphical object. This leads

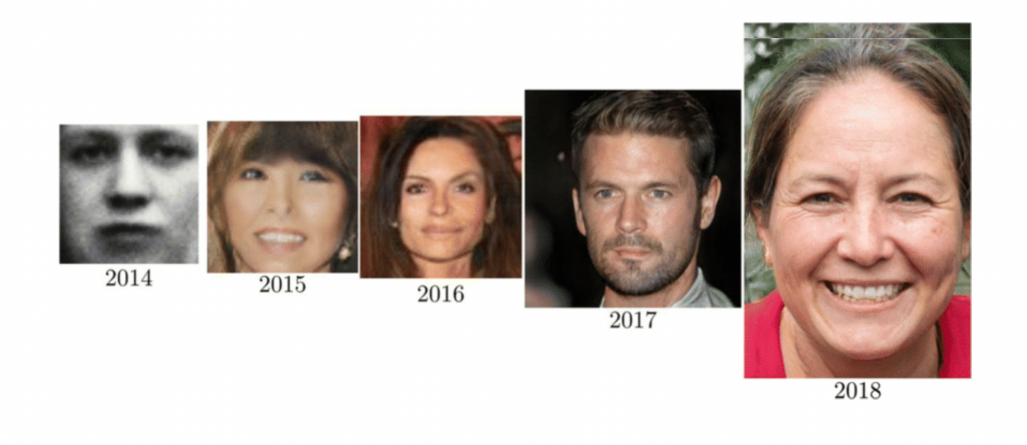


Figure 10.7: GANs Improvement over time

to having a not well defined loss (by the absence of a well defined similarity measure). Some attempts are already present in research, with promising outputs for molecular graphs [JBJ19].

Lastly, pivotal for the understanding of methods, standardized benchmarks drive research forward. While there are already available sources such as the Open Graph Benchmark (OGB), more data would help having a more refined idea of the potential of a method.

10.2 Applications

There are multiple industrial applications worth mentioning. In this section, we will briefly present some of them, attempting to cover most of the fields where Graph Neural Networks have been proposed.

Semiconductors Google optimized semiconductors chips design, a known combinatorially exploding in size issue. The method proposed is a combination of GNNs and Reinforcement Learning, which is known to be pretty efficient when dealing with NP-Hard problems.

Hardware The success of CNNs was also driven by the appearence of GPUs, which are well versed for their computations. Some companies are researching how to build a processing unit that is graph friendly. GraphCore(UK) has recently proposed their IPU architecture.

Social Networks Recommender systems are crucial for Companies' profits. Correctly identifying what a used would like makes a difference between revenue and no revenue. Pinterest implemented GraphSage for its platform, and built PinSage. UberEats is notably using GNNs to recomment food delivery options for customers. Aligraph, by Alibaba, performs edge prediction to determine the likelihood of a certain action of registered users.
Also Malicious content detection is a possibly expanding field. It was noticed that fake news

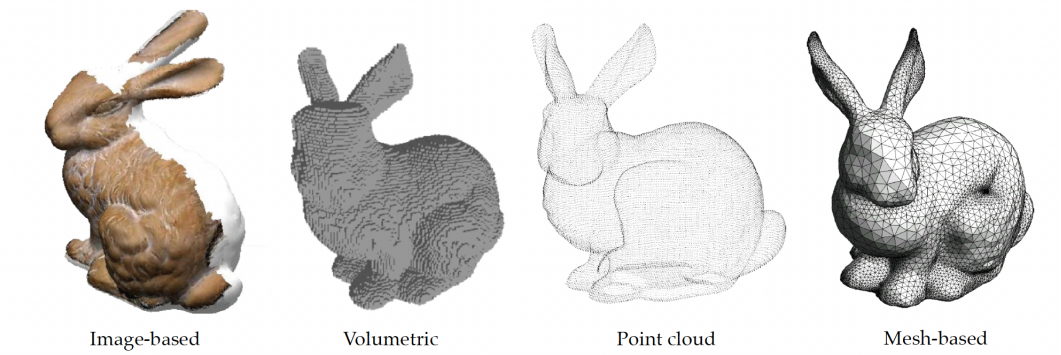


Figure 10.8: 3D Geometric Data Models

spread differently. Thus, a Neural Network that learns the pattern could definitely be useful for such a task. *Fabula.AI*, acquired by Twitter in 2019, had this mission.

3D Vision & Graphics A model for a solid object can be of many types. Some are shown in Figure 10.8. For example cars' sensors build point cloud representations of reality. Uber used GNNs to predict other cars' directions in the context of self driving [Cui+21]. Apple's products come with motion capture applications. A nice evolving field is that of 3D avatars design. Some examples include

- Humans' mesh models with Variational Auto Encoders for shape completion [Lit+18]
- face modeling [Bah+21]
- 2D to 3D hand estimation [Kul+20]

In the industry, **Ariel.AI**, acquired by Snapchat, developed a Neural Network that enhances AR effects with efficient hand recognition. Ultimately, face reconstruction from DNA, currently explored by Peter Claes at KU Leuven, could support forensic investigations and many other fields in the future.

Physics

> *It has been amazing to see how in the last two years Graph ML has become very popular in the field of physics.*
>
> Kyle Cranmer, NYU

Large and expensive equipment is necessary for particle accelerators. In this context, graphs naturally represent collisions. Readings from detectors and graphs combined proved to be able to reconstruct the origin of the particle [Ser+20]. Also recent research for the IceCube detector improved the exploration of links of neutrinos and astronomic events [Cho+18], where a GNN was exploited for classification. In addition to this, GNNs are also implemented for fluid dynamics simulation [Pfa+21].

Biology

In 2020, exciting progress has been made in protein structure prediction, a key problem in bioinformatics. Yet, ultimately the chemical and geometric pattern displayed at the surface of these molecules are critical for protein function.

Bruno Correira, EPFL

Proteins are imporatnt since there is no knowlege of life **not based** on proteins. These biological structures are made of long sequences of nucleotides of DNA. Each nucleotide tuplet is an encoding of an amminoacid, where only 20 combinations make proteins that can be synthetized. These long biological molecules fold depending on forces in act. The protein folding problem is that of determining the structure of a given sequence. Christian Anfinsen, 1972 Nobel Laureate in Chemistry, conjectured that their structure is solely determined by the aminoacids. It was not until 50 years later, when AlphaFold came into play, that it became possible to correctly predict proteins' 3D appearence [Jum+21]. This was a major breakthrough, which makes uses of many adjustments, including historical knowledge of precedent proteins and invariant points attention, a special structure [Jum+21]. The field is fast moving. Recently, Recurrent Geometric Networks, which underperformed AlphaFold, were improved. The authors claim to have single sequence prediction with no evolution knowledge.
Protein functions are everywhere in nature, some examples are shown in Figure 10.9. The majority of drugs make use of proteins in their interactions. In simple terms, they are molecules that disable protein interactions causing diseases. The Immunotheraphy 2018 Nobel prize is an example of such methods: cancer cells would be killed by T-cells, but they produce proteins that inhibit the immunitary system. The idea is blocking these proteins through a protein that acts on those that are produced by cancer cells with the *right* shape. This problem can be seen as an inverse folding problem. Given a shape, we wish to determine the sequence that generates it. Ignoring edge cases, a sequence defines a strucutre that has a specific function. Knowledge of this process is crucial to tackle diseases at a microscopic level. On similar grounds, MaSIF is a protein function prediction network that aims at determining particular binding properties of proteins *De Novo* [Gai+20].

Chemistry, Drug Design Drug discovery is yet another exciting subject, where long test times suggest exploring the space of available molecules (10^60 vs 10^3 laboratory capacity) using GNNs. These networks are indeed good for property prediction. It was recently found out that the daunting problem of antibiotic bacterias' resistance may be solved by custom designed drugs. A GNN was able to produce an instance of antibiotic with these characteristics [Sto+20].
Also drug repositioning and combinatorial therapy are being studied. Often the simultaneous combination of treatments can avoid side effects while guaranteeing a non linear effect of the therapy. Drug to protein (& mixed) levels of interactions rangining from binding effects to molecular can be predicted by a GNN [ZAL18].
A last exotic research direction is Food Chemistry studies. Understanding the benefits of foods

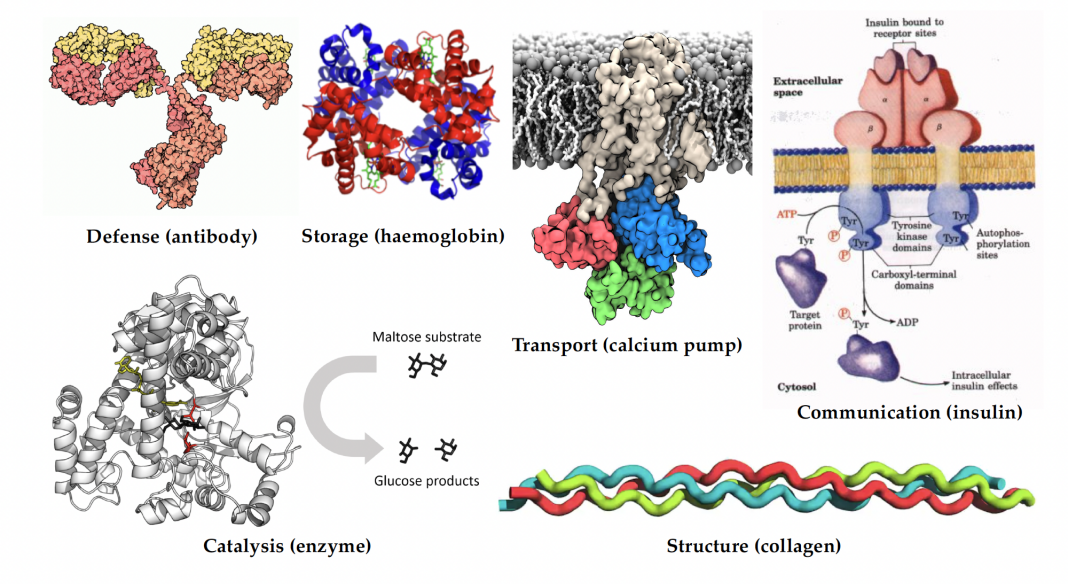


Figure 10.9: Some Protein Functions

and how their combinations interact is crucial to design thoughtful compounds. *Hyperfoods* is one of the first examples of the field, vastly using GNNs for its purposes [Ves+19].

Bibliography

[Vel21] Petar Veličković. *Geometric Deep Learning - Grids, Groups, Graphs, Geodesics, and Gauges*. 2021. URL: https://geometricdeeplearning.com/geometricdeeplearning.com/ (visited on 07/12/2022).

[Bro+21] Michael M. Bronstein et al. *Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges*. Number: arXiv:2104.13478. May 2, 2021. arXiv: 2104.13478[cs,stat]. URL: http://arxiv.org/abs/2104.13478 (visited on 07/11/2022).

[Bro+17] Michael M. Bronstein et al. "Geometric deep learning: going beyond Euclidean data". In: *IEEE Signal Processing Magazine* 34.4 (July 2017), pp. 18–42. ISSN: 1053-5888, 1558-0792. DOI: 10.1109/MSP.2017.2693418. arXiv: 1611.08097[cs]. URL: http://arxiv.org/abs/1611.08097 (visited on 07/11/2022).

[LeC+89] Y. LeCun et al. "Backpropagation Applied to Handwritten Zip Code Recognition". In: *Neural Computation* 1.4 (Dec. 1989), pp. 541–551. ISSN: 0899-7667, 1530-888X. DOI: 10.1162/neco.1989.1.4.541. URL: https://direct.mit.edu/neco/article/1/4/541-551/5515 (visited on 07/11/2022).

[Wei+21] Maurice Weiler et al. *Coordinate Independent Convolutional Networks – Isometry and Gauge Equivariant Convolutions on Riemannian Manifolds*. Number: arXiv:2106.06020. June 10, 2021. arXiv: 2106.06020[cs,stat]. URL: http://arxiv.org/abs/2106.06020 (visited on 07/11/2022).

[BB07] Léon Bottou and Olivier Bousquet. "The Tradeoffs of Large Scale Learning". In: *Advances in Neural Information Processing Systems*. Vol. 20. Curran Associates, Inc., 2007. URL: https://papers.nips.cc/paper/2007/hash/0d3180d672e08b4c5312dcdafdf6ef36-Abstract.html (visited on 07/11/2022).

[HSW89] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators". In: *Neural Networks* 2.5 (Jan. 1989), pp. 359–366. ISSN: 08936080. DOI: 10.1016/0893-6080(89)90020-8. URL: https://linkinghub.elsevier.com/retrieve/pii/0893608089900208 (visited on 07/11/2022).

[Pin99] Allan Pinkus. "Approximation theory of the MLP model in neural networks". In: *Acta Numerica* 8 (Jan. 1999), pp. 143–195. ISSN: 0962-4929, 1474-0508. DOI: 10.1017/S0962492900002919. URL: https://www.cambridge.org/core/product/identifier/S0962492900002919/type/journal_article (visited on 07/11/2022).

[Cyb89] G Cybenkot. "Approximation by superpositions of a sigmoidal function". In: (1989), p. 12.

[Bar93] A.R. Barron. "Universal approximation bounds for superpositions of a sigmoidal function". In: *IEEE Transactions on Information Theory* 39.3 (May 1993), pp. 930–945. ISSN: 0018-9448, 1557-9654. DOI: 10.1109/18.256500. URL: https://ieeexplore.ieee.org/document/256500/ (visited on 07/11/2022).

[Mai99] V. E Maiorov. "On Best Approximation by Ridge Functions". In: *Journal of Approximation Theory* 99.1 (July 1, 1999), pp. 68–94. ISSN: 0021-9045. DOI: 10.1006/jath.1998.3304. URL: https://www.sciencedirect.com/science/article/pii/S0021904598933044 (visited on 07/11/2022).

[Bar94] Andrew R. Barron. "Approximation and estimation bounds for artificial neural networks". In: *Machine Learning* 14.1 (Jan. 1, 1994), pp. 115–133. ISSN: 1573-0565. DOI: 10.1007/BF00993164. URL: https://doi.org/10.1007/BF00993164 (visited on 07/11/2022).

[JNJ17] Chi Jin, Praneeth Netrapalli, and Michael I. Jordan. *Accelerated Gradient Descent Escapes Saddle Points Faster than Gradient Descent*. Number: arXiv:1711.10456. Nov. 28, 2017. arXiv: 1711.10456[cs,math,stat]. URL: http://arxiv.org/abs/1711.10456 (visited on 07/11/2022).

[LV15] Karel Lenc and Andrea Vedaldi. *Understanding image representations by measuring their equivariance and equivalence*. Number: arXiv:1411.5908. June 20, 2015. arXiv: 1411.5908[cs]. URL: http://arxiv.org/abs/1411.5908 (visited on 07/11/2022).

[MMM21] Song Mei, Theodor Misiakiewicz, and Andrea Montanari. *Learning with invariances in random features and kernel models*. Number: arXiv:2102.13219. Feb. 25, 2021. arXiv: 2102.13219[cs,math,stat]. URL: http://arxiv.org/abs/2102.13219 (visited on 07/11/2022).

[BVB21] Alberto Bietti, Luca Venturi, and Joan Bruna. *On the Sample Complexity of Learning under Invariance and Geometric Stability*. Number: arXiv:2106.07148. Nov. 4, 2021. arXiv: 2106.07148[cs,stat]. URL: http://arxiv.org/abs/2106.07148 (visited on 07/11/2022).

[LB03] Ulrike von Luxburg and Olivier Bousquet. "Distance-Based Classification with Lipschitz Functions". In: *Learning Theory and Kernel Machines*. Ed. by Bernhard Schölkopf and Manfred K. Warmuth. Red. by Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen. Vol. 2777. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 314–328. ISBN: 978-3-540-40720-1 978-3-540-45167-9. DOI: 10.1007/978-3-540-45167-9_24. URL: http://link.springer.com/10.1007/978-3-540-45167-9_24 (visited on 07/11/2022).

[FCW21] Alessandro Favero, Francesco Cagnetta, and Matthieu Wyart. *Locality defeats the curse of dimensionality in convolutional teacher-student scenarios*. Number: arXiv:2106.08619. Nov. 12, 2021. arXiv: 2106.08619[cond-mat,stat]. URL: http://arxiv.org/abs/2106.08619 (visited on 07/11/2022).

[Zah+18] Manzil Zaheer et al. *Deep Sets*. Number: arXiv:1703.06114. Apr. 14, 2018. arXiv: 1703.06114[cs,stat]. URL: http://arxiv.org/abs/1703.06114 (visited on 07/11/2022).

[DBV17] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. *Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering*. Number: arXiv:1606.09375. Feb. 5, 2017. arXiv: 1606.09375[cs,stat]. URL: http://arxiv.org/abs/1606.09375 (visited on 07/11/2022).

[KW17] Thomas N. Kipf and Max Welling. *Semi-Supervised Classification with Graph Convolutional Networks*. Number: arXiv:1609.02907. Feb. 22, 2017. arXiv: 1609.02907[cs,stat]. URL: http://arxiv.org/abs/1609.02907 (visited on 07/11/2022).

[Wu+19] Felix Wu et al. *Simplifying Graph Convolutional Networks*. Number: arXiv:1902.07153. June 20, 2019. arXiv: 1902.07153[cs,stat]. URL: http://arxiv.org/abs/1902.07153 (visited on 07/11/2022).

[Mon+17] Federico Monti et al. "Geometric Deep Learning on Graphs and Manifolds Using Mixture Model CNNs". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Honolulu, HI: IEEE, July 2017, pp. 5425–5434. ISBN: 978-1-5386-0457-1. DOI: 10.1109/CVPR.2017.576. URL: http://ieeexplore.ieee.org/document/8100059/ (visited on 07/11/2022).

[Vel+18] Petar Veličković et al. *Graph Attention Networks*. Number: arXiv:1710.10903. Feb. 4, 2018. arXiv: 1710.10903[cs,stat]. URL: http://arxiv.org/abs/1710.10903 (visited on 07/11/2022).

[BAY22] Shaked Brody, Uri Alon, and Eran Yahav. *How Attentive are Graph Attention Networks?* Number: arXiv:2105.14491. Jan. 31, 2022. arXiv: 2105.14491[cs]. URL: http://arxiv.org/abs/2105.14491 (visited on 07/11/2022).

[Bat+16] Peter W. Battaglia et al. *Interaction Networks for Learning about Objects, Relations and Physics*. Number: arXiv:1612.00222. Dec. 1, 2016. arXiv: 1612.00222[cs]. URL: http://arxiv.org/abs/1612.00222 (visited on 07/11/2022).

[Gil+17] Justin Gilmer et al. *Neural Message Passing for Quantum Chemistry*. Number: arXiv:1704.01212. June 12, 2017. arXiv: 1704.01212[cs]. URL: http://arxiv.org/abs/1704.01212 (visited on 07/11/2022).

[Bat+18] Peter W. Battaglia et al. *Relational inductive biases, deep learning, and graph networks*. Number: arXiv:1806.01261. Oct. 17, 2018. arXiv: 1806.01261[cs,stat]. URL: http://arxiv.org/abs/1806.01261 (visited on 07/11/2022).

[Jos20] Chaytania Joshi. *Transformers are Graph Neural Networks*. The Gradient. Sept. 12, 2020. URL: https://thegradient.pub/transformers-are-graph-neural-networks/ (visited on 07/11/2022).

[Kip+18] Thomas Kipf et al. *Neural Relational Inference for Interacting Systems*. Number: arXiv:1802.04687. June 6, 2018. arXiv: 1802.04687[cs,stat]. URL: http://arxiv.org/abs/1802.04687 (visited on 07/11/2022).

[Ada13] Ryan Adams. *The Gumbel-Max Trick for Discrete Distributions | Laboratory for Intelligent Probabilistic Systems*. 2013. URL: https://lips.cs.princeton.edu/the-gumbel-max-trick-for-discrete-distributions/ (visited on 07/12/2022).

[Wan+19] Yue Wang et al. *Dynamic Graph CNN for Learning on Point Clouds*. Number: arXiv:1801.07829. June 11, 2019. arXiv: 1801.07829[cs]. URL: http://arxiv.org/abs/1801.07829 (visited on 07/11/2022).

[Kaz+22] Anees Kazi et al. "Differentiable Graph Module (DGM) for Graph Convolutional Networks". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022), pp. 1–1. ISSN: 0162-8828, 2160-9292, 1939-3539. DOI: 10.1109/TPAMI.2022.3170249. arXiv: 2002.04999[cs,stat]. URL: http://arxiv.org/abs/2002.04999 (visited on 07/11/2022).

[Vel+20] Petar Veličković et al. *Pointer Graph Networks*. Number: arXiv:2006.06380. Oct. 18, 2020. arXiv: 2006.06380[cs,stat]. URL: http://arxiv.org/abs/2006.06380 (visited on 07/11/2022).

[SYK21] Ryoma Sato, Makoto Yamada, and Hisashi Kashima. *Random Features Strengthen Graph Neural Networks*. Number: arXiv:2002.03155. Jan. 18, 2021. arXiv: 2002.03155[cs,stat]. URL: http://arxiv.org/abs/2002.03155 (visited on 07/11/2022).

[Cor+20] Gabriele Corso et al. *Principal Neighbourhood Aggregation for Graph Nets*. Number: arXiv:2004.05718. Dec. 31, 2020. arXiv: 2004.05718[cs,stat]. URL: http://arxiv.org/abs/2004.05718 (visited on 07/11/2022).

[Mal09] Stephane Mallat. *A Wavelet Tour of Signal Processing*. Elsevier, 2009. ISBN: 978-0-12-374370-1. DOI: 10.1016/B978-0-12-374370-1.X0001-8. URL: https://linkinghub.elsevier.com/retrieve/pii/B9780123743701X00018 (visited on 07/12/2022).

[Mal12] Stéphane Mallat. *Group Invariant Scattering*. Number: arXiv:1101.2286. Apr. 15, 2012. arXiv: 1101.2286[cs,math]. URL: http://arxiv.org/abs/1101.2286 (visited on 07/11/2022).

[Wal17] Irene Waldspurger. "Phase retrieval for wavelet transforms". In: *IEEE Transactions on Information Theory* (2017), pp. 1–1. ISSN: 0018-9448, 1557-9654. DOI: 10.1109/TIT.2017.2672727. URL: http://ieeexplore.ieee.org/document/7862873/ (visited on 07/11/2022).

[Oya+19] Edouard Oyallon et al. "Scattering Networks for Hybrid Representation Learning". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41.9 (Sept. 1, 2019), pp. 2208–2221. ISSN: 0162-8828, 2160-9292, 1939-3539. DOI: 10.1109/TPAMI.2018.2855738. arXiv: 1809.06367[cs,stat]. URL: http://arxiv.org/abs/1809.06367 (visited on 07/11/2022).

[Zar+20] John Zarka et al. *Deep Network Classification by Scattering and Homotopy Dictionary Learning*. Number: arXiv:1910.03561. Feb. 20, 2020. arXiv: 1910.03561[cs,stat]. URL: http://arxiv.org/abs/1910.03561 (visited on 07/11/2022).

[BM19] Joan Bruna and Stephane Mallat. *Multiscale Sparse Microcanonical Models*. Number: arXiv:1801.02013. May 9, 2019. arXiv: 1801.02013[math-ph,stat]. URL: http://arxiv.org/abs/1801.02013 (visited on 07/11/2022).

[Che+20] Sihao Cheng et al. "A new approach to observational cosmology using the scattering transform". In: *Monthly Notices of the Royal Astronomical Society* 499.4 (Nov. 7, 2020), pp. 5902–5914. ISSN: 0035-8711, 1365-2966. DOI: 10.1093/mnras/staa3165. arXiv: 2006.08561[astro-ph]. URL: http://arxiv.org/abs/2006.08561 (visited on 07/11/2022).

[Eic+18] Michael Eickenberg et al. "Solid harmonic wavelet scattering for predictions of molecule properties". In: *The Journal of Chemical Physics* 148.24 (June 28, 2018), p. 241732. ISSN: 0021-9606, 1089-7690. DOI: 10.1063/1.5023798. URL: http://aip.scitation.org/doi/10.1063/1.5023798 (visited on 07/11/2022).

[KT18] Risi Kondor and Shubhendu Trivedi. *On the Generalization of Equivariance and Convolution in Neural Networks to the Action of Compact Groups*. Number: arXiv:1802.03690. Nov. 10, 2018. arXiv: 1802.03690[cs,stat]. URL: http://arxiv.org/abs/1802.03690 (visited on 07/11/2022).

[CGW19] Taco S Cohen, Mario Geiger, and Maurice Weiler. "A General Theory of Equivariant CNNs on Homogeneous Spaces". In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc., 2019. URL: https://papers.nips.cc/paper/2019/hash/b9cfe8b6042cf759dc4c0cccb27a6737-Abstract.html (visited on 07/11/2022).

[Coh21] Taco S Cohen. "Equivariant Convolutional Networks". In: *Thesis, Chapter 9*. 2021. URL: https://pure.uva.nl/ws/files/60770359/Thesis.pdf (visited on 07/11/2022).

[Aro21] Jimmy Aronsson. *Homogeneous vector bundles and G-equivariant convolutional neural networks*. Number: arXiv:2105.05400. May 11, 2021. arXiv: 2105.05400[cs,math,stat]. URL: http://arxiv.org/abs/2105.05400 (visited on 07/11/2022).

[CW16] Taco S. Cohen and Max Welling. *Steerable CNNs*. Number: arXiv:1612.08498 version: 1. Dec. 26, 2016. arXiv: 1612.08498[cs,stat]. URL: http://arxiv.org/abs/1612.08498 (visited on 07/11/2022).

[Wei+18] Maurice Weiler et al. "3D Steerable CNNs: Learning Rotationally Equivariant Features in Volumetric Data". In: *Advances in Neural Information Processing Systems*. Vol. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper/2018/hash/488e4104520c6aab692863cc1dba45af-Abstract.html (visited on 07/11/2022).

[WC21] Maurice Weiler and Gabriele Cesa. *General $E(2)$-Equivariant Steerable CNNs*. Number: arXiv:1911.08251. Apr. 6, 2021. arXiv: 1911.08251[cs,eess]. URL: http://arxiv.org/abs/1911.08251 (visited on 07/11/2022).

[Bek21] Erik J. Bekkers. *B-Spline CNNs on Lie Groups*. Number: arXiv:1909.12057. Mar. 22, 2021. arXiv: 1909.12057[cs,stat]. URL: http://arxiv.org/abs/1909.12057 (visited on 07/11/2022).

[Fin+20] Marc Finzi et al. *Generalizing Convolutional Neural Networks for Equivariance to Lie Groups on Arbitrary Continuous Data*. Number: arXiv:2002.12880. Sept. 24, 2020. arXiv: 2002.12880[cs,stat]. URL: http://arxiv.org/abs/2002.12880 (visited on 07/11/2022).

[Est+18] Carlos Esteves et al. *Learning SO(3) Equivariant Representations with Spherical CNNs*. Number: arXiv:1711.06721. Sept. 27, 2018. DOI: 10.48550/arXiv.1711.06721. arXiv: 1711.06721[cs]. URL: http://arxiv.org/abs/1711.06721 (visited on 07/11/2022).

[Per+19] N. Perraudin et al. “DeepSphere: Efficient spherical convolutional neural network with HEALPix sampling for cosmological applications”. In: *Astronomy and Computing* 27 (Apr. 1, 2019), pp. 130–146. ISSN: 2213-1337. DOI: 10.1016/j.ascom.2019.03.004. URL: https://www.sciencedirect.com/science/article/pii/S2213133718301392 (visited on 07/11/2022).

[KLT18] Risi Kondor, Zhen Lin, and Shubhendu Trivedi. *Clebsch-Gordan Nets: a Fully Fourier Space Spherical Convolutional Neural Network*. Number: arXiv:1806.09231. Nov. 10, 2018. arXiv: 1806.09231[cs,stat]. URL: http://arxiv.org/abs/1806.09231 (visited on 07/11/2022).

[Wor+17] Daniel E. Worrall et al. *Harmonic Networks: Deep Translation and Rotation Equivariance*. Number: arXiv:1612.04642. Apr. 11, 2017. arXiv: 1612.04642[cs,stat]. URL: http://arxiv.org/abs/1612.04642 (visited on 07/11/2022).

[Tho+18] Nathaniel Thomas et al. *Tensor field networks: Rotation- and translation-equivariant neural networks for 3D point clouds*. Number: arXiv:1802.08219. May 18, 2018. arXiv: 1802.08219[cs]. URL: http://arxiv.org/abs/1802.08219 (visited on 07/11/2022).

[Kon18] Risi Kondor. *N-body Networks: a Covariant Hierarchical Neural Network Architecture for Learning Atomic Potentials*. Number: arXiv:1803.01588. Mar. 5, 2018. arXiv: 1803.01588[cs]. URL: http://arxiv.org/abs/1803.01588 (visited on 07/11/2022).

[Hy+18] Truong Son Hy et al. “Predicting molecular properties with covariant compositional networks”. In: *The Journal of Chemical Physics* 148.24 (June 27, 2018). Publisher: AIP Publishing LLCAIP Publishing, p. 241745. ISSN: 0021-9606. DOI: 10.1063/1.5024797. URL: https://aip.scitation.org/doi/abs/10.1063/1.5024797 (visited on 07/11/2022).

[Mel+19] Simone Melzi et al. “GFrames: Gradient-Based Local Reference Frame for 3D Shape Matching”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Long Beach, CA, USA: IEEE, June 2019, pp. 4624–4633. ISBN: 978-1-72813-293-8. DOI: 10.1109/CVPR.2019.00476. URL: https://ieeexplore.ieee.org/document/8953995/ (visited on 07/11/2022).

[Mas+18] Jonathan Masci et al. *Geodesic convolutional neural networks on Riemannian manifolds*. Number: arXiv:1501.06297. June 8, 2018. arXiv: 1501.06297[cs]. URL: http://arxiv.org/abs/1501.06297 (visited on 07/11/2022).

[PP93] Ulrich Pinkall and Konrad Polthier. “Computing Discrete Minimal Surfaces and Their Conjugates”. In: *Experimental Mathematics* 2.1 (Jan. 1993), pp. 15–36. ISSN: 1058-6458, 1944-950X. DOI: 10.1080/10586458.1993.10504266. URL: http://www.tandfonline.com/doi/abs/10.1080/10586458.1993.10504266 (visited on 07/11/2022).

[War+08] Max Wardetzky et al. “Discrete Laplace operators: no free lunch”. In: *ACM SIGGRAPH ASIA 2008 courses on - SIGGRAPH Asia '08*. ACM SIGGRAPH ASIA 2008 courses. Singapore: ACM Press, 2008, pp. 1–5. DOI: 10.1145/1508044.1508063. URL: http://portal.acm.org/citation.cfm?doid=1508044.1508063 (visited on 07/11/2022).

[Mey+03] Mark Meyer et al. “Discrete Differential-Geometry Operators for Triangulated 2-Manifolds”. In: *Visualization and Mathematics III*. Ed. by Hans-Christian Hege and Konrad Polthier. Red. by Gerald Farin et al. Series Title: Mathematics and Visualization. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 35–57. ISBN: 978-3-642-05682-6 978-3-662-05105-4. DOI: 10.1007/978-3-662-05105-4_2. URL: http://link.springer.com/10.1007/978-3-662-05105-4_2 (visited on 07/11/2022).

[Mac04] Richard H. MacNeal. “The solution of partial differential equations by means of electrical networks”. Medium: PDF Version Number: Final. PhD thesis. California Institute of Technology, Apr. 29, 2004. DOI: 10.7907/PZ04-5290. URL: https://resolver.caltech.edu/CaltechETD:etd-04282004-143609 (visited on 07/12/2022).

[SM13] Aliaksei Sandryhaila and Jose M. F. Moura. *Discrete Signal Processing on Graphs: Frequency Analysis*. Number: arXiv:1307.0468. Nov. 18, 2013. arXiv: 1307.0468[cs, math]. URL: http://arxiv.org/abs/1307.0468 (visited on 07/11/2022).

[Hoo+18] Emiel Hoogeboom et al. *HexaConv*. Number: arXiv:1803.02108. Mar. 6, 2018. arXiv: 1803.02108[cs,stat]. URL: http://arxiv.org/abs/1803.02108 (visited on 07/11/2022).

[Haa+21] Pim de Haan et al. *Gauge Equivariant Mesh CNNs: Anisotropic convolutions on geometric graphs*. Number: arXiv:2003.05425. Nov. 19, 2021. arXiv: 2003.05425[cs, stat]. URL: http://arxiv.org/abs/2003.05425 (visited on 07/11/2022).

[Ger+21] Jan E. Gerken et al. *Geometric Deep Learning and Equivariant Neural Networks*. Number: arXiv:2105.13926. May 28, 2021. arXiv: 2105.13926[hep-th]. URL: http://arxiv.org/abs/2105.13926 (visited on 07/11/2022).

[Ros+20] Emanuele Rossi et al. *Temporal Graph Networks for Deep Learning on Dynamic Graphs*. Number: arXiv:2006.10637. Oct. 9, 2020. arXiv: 2006.10637[cs,stat]. URL: http://arxiv.org/abs/2006.10637 (visited on 07/12/2022).

[TO18] Corentin Tallec and Yann Ollivier. *Can recurrent neural networks warp time?* Number: arXiv:1804.11188. Mar. 23, 2018. arXiv: 1804.11188[cs,stat]. URL: http://arxiv.org/abs/1804.11188 (visited on 07/12/2022).

[Jor97] Michael I. Jordan. “Chapter 25 - Serial Order: A Parallel Distributed Processing Approach”. In: *Advances in Psychology*. Ed. by John W. Donahoe and Vivian Packard Dorsel. Vol. 121. Neural-Network Models of Cognition. North-Holland, Jan. 1, 1997, pp. 471–495. DOI: 10.1016/S0166-4115(97)80111-2. URL: https://www.sciencedirect.com/science/article/pii/S0166411597801112 (visited on 07/12/2022).

[Elm90] Jeffrey L. Elman. “Finding structure in time”. In: *Cognitive Science* 14.2 (Apr. 1, 1990), pp. 179–211. ISSN: 0364-0213. DOI: 10.1016/0364-0213(90)90002-E. URL: https://www.sciencedirect.com/science/article/pii/036402139090002E (visited on 07/12/2022).

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory”. In: *Neural Computation* 9.8 (Nov. 1, 1997), pp. 1735–1780. ISSN: 0899-7667, 1530-888X. DOI: 10.1162/neco.1997.9.8.1735. URL: https://direct.mit.edu/neco/article/9/8/1735-1780/6109 (visited on 07/11/2022).

[Cho+14] Kyunghyun Cho et al. *On the Properties of Neural Machine Translation: Encoder-Decoder Approaches*. Number: arXiv:1409.1259. Oct. 7, 2014. arXiv: 1409.1259[cs, stat]. URL: http://arxiv.org/abs/1409.1259 (visited on 07/11/2022).

[Bro20] M. M. Bronstein. *What does 2021 hold for Graph ML?* 2020. URL: https://towardsdatascience.com/predictions-and-hopes-for-graph-ml-in-2021-6af2121c3e3d (visited on 07/11/2022).

[Bod+21] Cristian Bodnar et al. *Weisfeiler and Lehman Go Topological: Message Passing Simplicial Networks*. Number: arXiv:2103.03212. June 14, 2021. arXiv: 2103.03212[cs]. URL: http://arxiv.org/abs/2103.03212 (visited on 07/11/2022).

[Cra+20] Miles Cranmer et al. *Discovering Symbolic Models from Deep Learning with Inductive Biases*. Number: arXiv:2006.11287. Nov. 17, 2020. arXiv: 2006.11287[astro-ph,physics:physics,stat]. URL: http://arxiv.org/abs/2006.11287 (visited on 07/11/2022).

[VB21] Petar Veličković and Charles Blundell. "Neural Algorithmic Reasoning". In: *Patterns* 2.7 (July 2021), p. 100273. ISSN: 26663899. DOI: 10.1016/j.patter.2021.100273. arXiv: 2105.02761[cs,math,stat]. URL: http://arxiv.org/abs/2105.02761 (visited on 07/11/2022).

[Bel+21] Anastasiya Belyaeva et al. "Causal network models of SARS-CoV-2 expression and aging to identify candidates for drug repurposing". In: *Nature Communications* 12.1 (Dec. 2021), p. 1024. ISSN: 2041-1723. DOI: 10.1038/s41467-021-21056-z. URL: http://www.nature.com/articles/s41467-021-21056-z (visited on 07/11/2022).

[GWG22] Johannes Gasteiger, Stefan Weißenberger, and Stephan Günnemann. *Diffusion Improves Graph Learning*. Number: arXiv:1911.05485. Apr. 5, 2022. arXiv: 1911.05485[cs,stat]. URL: http://arxiv.org/abs/1911.05485 (visited on 07/11/2022).

[Bog+21] Marian Boguna et al. "Network Geometry". In: *Nature Reviews Physics* 3.2 (Feb. 2021), pp. 114–135. ISSN: 2522-5820. DOI: 10.1038/s42254-020-00264-4. arXiv: 2001.03241[cond-mat,physics:physics]. URL: http://arxiv.org/abs/2001.03241 (visited on 07/11/2022).

[Cha+21] Ben Chamberlain et al. "GRAND: Graph Neural Diffusion". In: *Proceedings of the 38th International Conference on Machine Learning*. International Conference on Machine Learning. ISSN: 2640-3498. PMLR, July 1, 2021, pp. 1407–1418. URL: https://proceedings.mlr.press/v139/chamberlain21a.html (visited on 07/11/2022).

[Ros+22] Emanuele Rossi et al. *On the Unreasonable Effectiveness of Feature propagation in Learning on Graphs with Missing Node Features*. Number: arXiv:2111.12128. May 23, 2022. arXiv: 2111.12128[cs]. URL: http://arxiv.org/abs/2111.12128 (visited on 07/11/2022).

[JBJ19] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. *Junction Tree Variational Autoencoder for Molecular Graph Generation*. Number: arXiv:1802.04364. Mar. 29, 2019. arXiv: 1802.04364[cs,stat]. URL: http://arxiv.org/abs/1802.04364 (visited on 07/11/2022).

[Cui+21] Alexander Cui et al. *LookOut: Diverse Multi-Future Prediction and Planning for Self-Driving*. Number: arXiv:2101.06547. May 7, 2021. arXiv: 2101.06547[cs]. URL: http://arxiv.org/abs/2101.06547 (visited on 07/11/2022).

[Lit+18] Or Litany et al. *Deformable Shape Completion with Graph Convolutional Autoencoders*. Number: arXiv:1712.00268. Apr. 3, 2018. arXiv: 1712.00268[cs]. URL: http://arxiv.org/abs/1712.00268 (visited on 07/11/2022).

[Bah+21] Mehdi Bahri et al. *Shape My Face: Registering 3D Face Scans by Surface-to-Surface Translation*. Number: arXiv:2012.09235. Mar. 10, 2021. arXiv: 2012.09235[cs]. URL: http://arxiv.org/abs/2012.09235 (visited on 07/11/2022).

[Kul+20] Dominik Kulon et al. *Weakly-Supervised Mesh-Convolutional Hand Reconstruction in the Wild*. Number: arXiv:2004.01946. Apr. 4, 2020. arXiv: 2004.01946[cs]. URL: http://arxiv.org/abs/2004.01946 (visited on 07/11/2022).

[Ser+20] Hadar Serviansky et al. "Set2Graph: Learning Graphs From Sets". In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., 2020, pp. 22080–22091. URL: https://proceedings.neurips.cc/paper/2020/hash/fb4ab556bc42d6f0ee0f9e24ec4d1af0-Abstract.html (visited on 07/11/2022).

[Cho+18] Nicholas Choma et al. *Graph Neural Networks for IceCube Signal Classification*. Number: arXiv:1809.06166. Sept. 17, 2018. arXiv: 1809.06166[astro-ph,stat]. URL: http://arxiv.org/abs/1809.06166 (visited on 07/11/2022).

[Pfa+21] Tobias Pfaff et al. *Learning Mesh-Based Simulation with Graph Networks*. Number: arXiv:2010.03409. June 18, 2021. arXiv: 2010.03409[cs]. URL: http://arxiv.org/abs/2010.03409 (visited on 07/11/2022).

[Jum+21] John Jumper et al. "Highly accurate protein structure prediction with AlphaFold". In: *Nature* 596.7873 (Aug. 2021). Number: 7873 Publisher: Nature Publishing Group, pp. 583–589. ISSN: 1476-4687. DOI: 10.1038/s41586-021-03819-2. URL: https://www.nature.com/articles/s41586-021-03819-2 (visited on 07/11/2022).

[Gai+20] P. Gainza et al. "Deciphering interaction fingerprints from protein molecular surfaces using geometric deep learning". In: *Nature Methods* 17.2 (Feb. 2020). Number: 2 Publisher: Nature Publishing Group, pp. 184–192. ISSN: 1548-7105. DOI: 10.1038/s41592-019-0666-6. URL: https://www.nature.com/articles/s41592-019-0666-6 (visited on 07/11/2022).

[Sto+20] Jonathan M. Stokes et al. "A Deep Learning Approach to Antibiotic Discovery". In: *Cell* 180.4 (Feb. 2020), 688–702.e13. ISSN: 00928674. DOI: 10.1016/j.cell.2020.01.021. URL: https://linkinghub.elsevier.com/retrieve/pii/S0092867420301021 (visited on 07/11/2022).

[ZAL18] Marinka Zitnik, Monica Agrawal, and Jure Leskovec. "Modeling polypharmacy side effects with graph convolutional networks". In: *Bioinformatics* 34.13 (July 1, 2018), pp. i457–i466. ISSN: 1367-4803, 1460-2059. DOI: 10.1093/bioinformatics/bty294. arXiv: 1802.00543[cs,q-bio,stat]. URL: http://arxiv.org/abs/1802.00543 (visited on 07/11/2022).

[Ves+19] Kirill Veselkov et al. "HyperFoods: Machine intelligent mapping of cancer-beating molecules in foods". In: *Scientific Reports* 9.1 (July 3, 2019). Number: 1 Publisher: Nature Publishing Group, p. 9237. ISSN: 2045-2322. DOI: 10.1038/s41598-019-45349-y. URL: https://www.nature.com/articles/s41598-019-45349-y (visited on 07/11/2022).