

**Geometric Deep Learning**  
**Grids, Groups, Graphs,**  
**Geodesics, and Gauges**  
几何深度学习  
格网, 群, 图, 测地线, 量规

Michael M. Bronstein<sup>1</sup>, Joan Bruna<sup>2</sup>, Taco Cohen<sup>3</sup>, Petar Veličković<sup>4</sup>  
翻译: 子仁<sup>5</sup>, 昕晧<sup>6</sup>

2021 年 5 月 13 日

<sup>1</sup>Imperial College London / USI IDSIA / Twitter

<sup>2</sup>New York University

<sup>3</sup>Qualcomm AI Research. Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc.

<sup>4</sup>DeepMind

<sup>5</sup>NUDT, luojunren17@nudt.edu.cn

<sup>6</sup>WHU, yarongluo@whu.edu.cn



# 目录

前言	1
注记	3
1 引言	4
2 高维学习	5
2.1 函数正则化归纳偏置	6
2.2 维数诅咒	7
3 Geometric Priors	10
3.1 Symmetries, Representations, and Invariance	11
3.2 Isomorphisms and Automorphisms	18
3.3 Deformation Stability	21
3.4 Scale Separation	25
3.5 The Blueprint of Geometric Deep Learning	30
4 几何域：五个世代	36
4.1 图与集合	36

4.2	格网与欧几里得空间 . . . . .	42
4.3	群与齐次空间 . . . . .	47
4.4	测地线与流形 . . . . .	52
4.5	量规与丛 . . . . .	68
4.6	几何图与网格 . . . . .	73
5	几何深度学习模型 . . . . .	83
5.1	卷积神经网络 . . . . .	84
5.2	群等变卷积神经网络 . . . . .	88
5.3	图神经网络 . . . . .	92
5.4	Deep Sets, Transformers, 潜在图推理 . . . . .	94
5.5	等变消息传递网络 . . . . .	96
5.6	内在网格卷积神经网络 . . . . .	99
5.7	递归神经网络 . . . . .	102
5.8	长短期记忆网络 . . . . .	108
6	问题与应用 . . . . .	115
6.1	化学与药物设计 . . . . .	115
6.2	药物重新定位 . . . . .	116
6.3	蛋白质生物学 . . . . .	116
6.4	推荐系统与社交网络 . . . . .	117
6.5	交通预测 . . . . .	118

6.6	物体识别 . . . . .	119
6.7	博弈对抗 . . . . .	120
6.8	文本与语音合成 . . . . .	121
6.9	医疗保健 . . . . .	122
6.10	粒子物理和天体物理 . . . . .	124
6.11	虚拟与增强现实 . . . . .	125
7	历史视角 . . . . .	127
7.1	数学与物理中的对称 . . . . .	127
7.2	机器学习中早期使用对称 . . . . .	128
7.3	图神经网络 . . . . .	129
7.4	计算化学 . . . . .	130
7.5	节点嵌入 . . . . .	130
7.6	概率图模型 . . . . .	131
7.7	Weisfeiler-Lehman 形式化 . . . . .	132
7.8	高阶方法 . . . . .	133
7.9	信号处理与调和分析 . . . . .	134
7.10	图与网格上信号处理 . . . . .	135
7.11	计算机图形学与几何处理 . . . . .	135
7.12	算法推理 . . . . .	137
7.13	几何深度学习 . . . . .	138



## 前言

至欧几里得《几何原本》以来的近两千年, 单词“几何”与“欧几里得几何”是同义词, 因为没有几何类型的几何存在. 欧几里得几何的垄断在 19 世纪被终结, 其中有由洛巴切夫斯基, 博利亚伊, 高斯和黎曼构建的非欧几里得几何. 到 19 世纪末, 这些研究已分散到各个领域, 数学家和哲学家就这些几何的有效性以及它们之间的关系以及“真实几何”的本质进行了辩论.

年轻的数学家费利克斯·克莱因 (Felix Klein) 指出了解决这一难题的方法, 他于 1872 年被任命为小规模巴伐利亚州埃尔兰根大学的教授在一份作为埃尔兰根纲领进入数学编年史的研究计划书中, 克莱因提出将几何作为不变量的研究, 即在某种变换下不变的性质, 称为几何的对称性. 这种方法做出了一些澄清, 表明当时已知的各种几何形状可以通过适当选择对称变换来定义, 并使用群论的语言来形式化. 例如, 欧几里得几何与长度和角度有关, 因为这些属性由欧几里得变换 (旋转和平移) 保留, 而仿射几何研究平行性, 由仿射变换保留. 当考虑各自的群时, 这些几何之间的关系是显而易见, 因为欧几里得群是仿射群的子群, 而仿射群又是射影变换群的子群.

埃尔兰根纲领对几何学的影响非常深远. 此外, 它蔓延到其他领域, 特别是物理学, 在那里对称原理允许从对称的第一性原理 (一个被称为诺瑟定理的惊人结论) 中导出守恒定律, 甚至允许将基本粒子分类为对称群的不可约表示. 用范畴理论 (*Category theory*) 的创立者萨缪尔·艾伦伯和桑德斯·麦克兰恩的话来说, 现在在纯数学中普遍存在的范畴理论可以被视为克莱因“埃尔兰根纲领”的延续, 在这个意义上, 一个几何空间及其变换群被推广到一个范畴及其映射代数”.

在写作时, 深度学习领域的状态有点让人想起十九世纪的几何领域. 对于各种各样的数据, 有一个名副其实的神经网络体系结构动物园, 但是很少有统一的原则. 与过去一样, 这使得很难理解各种方法之间的关系, 不可避免地导致不同应用领域中相同概念的重新发明和品牌化. 对于一个试图学习该领域的新手来说, 去吸收大量多余的想法是一场真正的噩梦.



根据一种普遍的看法, “埃尔兰根纲领 (Erlangen Programme)” 于 1872 年 10 月在克莱因的就职演说中发表. 克莱因确实做了这样的演讲 (尽管是在同年 12 月 7 日), 但这是针对非数学观众的, 并且主要关注他关于数学教育的想法. 如今被称作“埃尔兰根纲领”实际是他为他的教授任命而准备的研究计划书 *em Vergleichende Betrachtungen über neuere geometrische Forschungen* (“几何学最新研究比较综述”). 见 [Tobies \(2019\)](#). 见 [Marquis \(2009\)](#).

在本文中, 我们尝试将“埃尔兰根纲领”的思维模式应用到深度学习领域, 最终目标是实现该领域的系统化和“点与点之间的联系”。我们将这种几何化的尝试称为“几何深度学习”, 并忠实于费利克斯·克莱因的精神, 提出从对称和不变性的第一原则中导出不同的归纳偏置和实现它们的网络架构。特别是, 我们专注于一大类神经网络, 用于分析非结构化集合、格网、图形和流形, 并表明它们可以以统一的方式理解为保留这些域的结构和对称性的方法。

我们相信, 这篇文章将吸引广大深入学习的研究人员、实践者和爱好者。新手可以将它作为几何深度学习的概述和介绍。一个经验丰富的深度学习专家可能会发现从基本原则和一些令人惊讶的联系中获得熟悉的体系结构的新方法。从业者可能会获得如何解决各自领域问题的新见解。

在现代机器学习这样一个快节奏的领域, 写这样一个文本的风险是, 它在天亮之前就变得过时和不相关了。专注于基础, 我们希望我们讨论的关键概念将超越它们的具体实现, 正如 Claude Adrien Helvétius 所说,“对某些原则的认识很容易取代对某些事实的认识”。

“对某些原则的了解很容易弥补对某些事实了解的不足。”  
([Helvétius, 1759](#))



## 注记

$\Omega, u$	域, 域上点
$x(u) \in \mathcal{X}(\Omega, \mathcal{C})$	域上信号 $x : \Omega \rightarrow \mathcal{C}$
$f(x) \in \mathcal{F}(\mathcal{X}(\Omega))$	域上信号的函数 $f : \mathcal{X}(\Omega) \rightarrow \mathcal{Y}$
$\mathfrak{G}, \mathfrak{g}$	群, 群元素
$\mathfrak{g}.u, \rho(\mathfrak{g})$	群作用, 群表示
$\mathbf{X} \in \mathcal{C}^{ \Omega  \times s}$	离散域上信号的矩阵表示
$\mathbf{x}_u \in \mathcal{C}^s$	离散信号 $\mathbf{X}$ 在元素 $u \in \Omega$ 处向量表示
$x_{uj} \in \mathcal{C}$	离散信号 $\mathbf{X}$ 在元素 $u \in \Omega$ 处第 $j$ 个分量的标量表示
$\mathbf{F}(\mathbf{X})$	离散信号上的函数, 返回矩阵形式的离散信号
$\tau : \Omega \rightarrow \Omega$	域自同构
$\eta : \Omega \rightarrow \Omega'$	两个不同域之间同构
$\sigma : \mathcal{C} \rightarrow \mathcal{C}'$	激活函数 (逐点非线性)
$G = (\mathcal{V}, \mathcal{E})$	带点 $\mathcal{V}$ 和边 $\mathcal{E}$ 的图
$\mathcal{T} = (\mathcal{V}, \mathcal{E}, \mathcal{F})$	带点 $\mathcal{V}$ , 边 $\mathcal{E}$ , 面 $\mathcal{F}$ 的网格
$x \star \theta$	与滤波器 $\theta$ 的卷积
$S_v$	变换算子
$\varphi_i$	基函数
$T_u\Omega, T\Omega$	在 $u$ 处的切空间, 切丛
$X \in T_u\Omega$	切向量
$g_u(X, Y) = \langle X, Y \rangle_u$	黎曼测度
$\ell(\gamma), \ell_{uv}$	曲线 $\gamma$ 的长度, 边 $(u, v)$ 上的离散测度

## 1 引言

过去十年见证了数据科学和机器学习的实验革命, 其缩影是深度学习方法. 事实上, 许多以前被认为遥不可及的高维学习任务——如计算机视觉、围棋或蛋白质折叠——在适当的计算规模下实际上是可行的. 值得注意的是, 深度学习的本质是建立在两个简单的算法原则之上的: 第一, 表示或特征学习 (*feature learning*) 的概念, 其中自适应的, 通常是分层的, 特征为每个任务捕获适当的规律性概念, 第二, 通过局部梯度下降的学习, 通常被实现为反向传播 (*backpropagation*).

虽然学习高维度的一般函数是一个讨厌的估计问题, 但大多数感兴趣的任务都不是一般的, 并且带有来自底层物理世界的低维度和结构的基本预定义规则. 本文通过统一的几何原理来揭示这些规律, 这些原理可以应用于广泛的应用领域.

利用一个大系统的已知对称性是对抗维度诅咒的一种强有力的经典疗法, 并且构成了大多数物理理论的基础. 深度学习系统也不例外, 从早期开始, 研究人员就采用神经网络来利用物理测量产生的低维几何, 例如图像中的网格、时间序列中的序列或分子中的位置和动量, 以及它们相关的对称性, 例如平移或旋转. 在我们的整个论述中, 我们将把这些模型, 以及许多其他模型, 描述为相同的基本几何规则的自然实例.

这种以“埃尔兰根纲领”为精神的“几何统一”努力有双重目的: 一方面, 它提供了一个通用的数学框架来研究最成功的神经网络架构, 如 CNNs、RNNs、GNNs 和 Transformers. 另一方面, 它给出了一个建设性的过程, 将先前的物理知识结合到神经体系结构中, 并为构建未来尚未发明的体系结构提供了有原则的方法.

在继续之前, 值得注意的是, 我们的工作涉及表示学习架构 (*representation learning architectures*) 和利用其中的数据对称性. 许多令人兴奋的可以使用这种表示的途径 (*pipelines*)(如自我监督学习、生成模型或强化学习) 并不是我们关注的中心. 因此, 我们不会深入综述有影响的神经网络途径, 如

这同样适用于用于优化 (*optimising*) 或规范 (*regularising*) 我们的架构的技术, 例如 Adam (Kingma and Ba, 2014), dropout (Srivastava et al., 2014) 和批归一化 (batch normalisation) (Ioffe and Szegedy, 2015).

变分自动编码器 (variational autoencoders (Kingma and Welling, 2013))、生成对抗网络 (generative adversarial networks (Goodfellow et al., 2014))、标准化流 (normalising flows (Rezende and Mohamed, 2015))、深度 Q 网络 (deep Q-networks (Mnih et al., 2015))、近端策略优化 (proximal policy optimisation (Schulman et al., 2017)) 或深度互信息最大化 (deep mutual information maximisation (Hjelm et al., 2019)). 尽管如此, 我们认为, 我们将侧重的原则在所有这些领域都非常重要.

此外, 虽然我们试图撒下一张相当宽的网来说明我们几何蓝图的力量, 但我们的工作并没有试图准确总结几何深度学习的全部现有研究成果. 相反, 我们深入研究了几个著名的架构, 以展示这些原则, 并在现有的研究中对它们进行基础研究, 希望我们为读者留下足够的参考资料, 以便有意义地将这些原则应用到他们遇到或设计的任何未来的几何深度架构中.

## 2 高维学习

监督机器学习, 在其最简单的形式中, 考虑一组来自定义在  $\mathcal{X} \times \mathcal{Y}$  上的服从数据分布  $P$  的  $N$  个独立同分布 (*i.i.d.*) 观察值  $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ , 其中  $\mathcal{X}$  和  $\mathcal{Y}$  分别是数据和标签域. 这种设置的定义特征是  $\mathcal{X}$  是一个高维空间: 人们通常假设  $\mathcal{X} = \mathbb{R}^d$  是一个  $d$  维的欧几里得空间.

让我们进一步假设标签  $y$  由未知函数  $f$  生成, 满足  $y_i = f(x_i)$ , 并且学习问题简化为使用参数化函数类  $\mathcal{F} = \{f_{\theta \in \Theta}\}$  来估计函数  $f$ . 神经网络是这种参数函数类的常见实现, 在这种情况下,  $\theta \in \Theta$  对应于网络权重. 在这种理想的设置中, 标签中没有噪声, 现代深度学习系统通常在所谓的插值机制 (*interpolating regime*) 下运行, 其中对于所有  $i = 1, \dots, N$  估计的  $\tilde{f} \in \mathcal{F}$  满足  $\tilde{f}(x_i) = f(x_i)$ . 学习算法的性能是根据从  $P$  中抽取新样本的期望性能 (*expected performance*), 使用一些损失  $L(\cdot, \cdot)$  来衡量的

$$\mathcal{R}(\tilde{f}) := \mathbb{E}_P L(\tilde{f}(x), f(x)),$$

其中平方损失  $L(y, y') = \frac{1}{2}|y - y'|^2$  是最常用的.

统计学习理论关注的是基于集中不等式的更精确的广义概念; 我们将在未来的工作中回顾其中的一些内容.

因此, 一个成功的学习方案需要对  $f$  的正则性 (regularity) 或归纳偏置 (inductive bias) 概念进行编码, 通过函数类  $\mathcal{F}$  的构造和正则化的使用来施加. 我们将在下一节简要介绍这些概念.

## 2.1 函数正则化归纳偏置

现代机器学习使用大型、高质量的数据集, 这些数据集与适当的计算资源一起, 激发了丰富的函数类  $\mathcal{F}$  的设计, 这些函数类  $\mathcal{F}$  具有插值如此大的数据的能力. 这种思维模式很适合神经网络, 因为即使是最简单的架构选择也会产生稠密的 (dense) 函数类. 逼近几乎任意函数的能力是各种通用逼近定理 (Universal Approximation Theorems) 的主题; 几个这样的结果在 1990 年代被应用数学家和计算机科学家证明和推广 (例如, 见 [Cybenko \(1989\)](#); [Hornik \(1991\)](#); [Barron \(1993\)](#); [Leshno et al. \(1993\)](#); [Maierov \(1999\)](#); [Pinkus \(1999\)](#)).

集合  $\mathcal{A} \subset \mathcal{X}$  的闭集如果满足

$$\mathcal{A} \cup \left\{ \lim_{i \rightarrow \infty} a_i : a_i \in \mathcal{A} \right\} = \mathcal{X}.$$

则集合被称为稠密的 (dense).

这意味着  $\mathcal{X}$  中的任何一点都任意靠近  $\mathcal{A}$  中的一点. 一个典型的通用近似结果表明, 例

如由两层感知器

$f(\mathbf{x}) = \mathbf{c}^\top \text{sign}(\mathbf{A}\mathbf{x} + \mathbf{b})$  表示的函数类在  $\mathbb{R}^d$  上的连续函数空间中是稠密的.

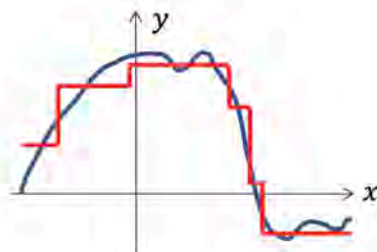
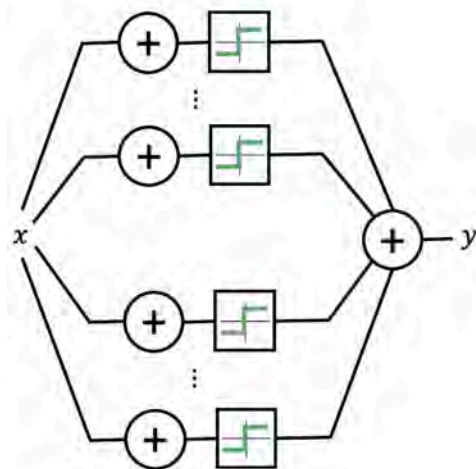


图 1: 多层感知器 ([Rosenblatt, 1958](#)) 是最简单的前馈神经网络, 是通用的逼近器: 只有一个隐藏层, 它们可以表示阶跃函数的组合, 允许以任意精度逼近任何连续函数.

然而, 通用近似并不意味着没有归纳偏置. 给定一个具有泛逼近性的假设空间  $\mathcal{F}$ , 我们可以定义一个复杂度度量  $c: \mathcal{F} \rightarrow \mathbb{R}_+$  并将我们的插值问题重新

定义为

$$\tilde{f} \in \arg \min_{g \in \mathcal{F}} c(g) \quad \text{s.t.} \quad g(x_i) = f(x_i) \quad \text{for } i = 1, \dots, N,$$

即, 我们在我们的假设类中寻找最正则 (regular) 的函数. 对于标准函数空间, 这种复杂性度量可以被定义为一个范数 (*norm*), 使  $\mathcal{F}$  成为一个巴拿赫空间 (*Banach space*), 并允许在泛函分析中利用过多的理论结果. 在低维中, 样条是函数逼近的主要工具. 它们是可以由上面的公式表示, 用一个范数捕捉光滑性的经典概念, 比如三次样条的二阶导数的平方范数  $\int_{-\infty}^{+\infty} |f''(x)|^2 dx$ .

非正式地说, 一个范数  $\|x\|$  可以看作是向量  $x$  的一个“长度”. 巴拿赫空间是一个配有范数的完备向量空间.

在神经网络的情况下, 复杂度度量  $c$  可以用网络权重来表示, 即  $c(f_{\theta}) = c(\theta)$ . 网络权重的  $L_2$  范数, 即权重衰减 (*weight decay*), 或所谓的路径范数 (*path-norm*, (Neyshabur et al., 2015)) 是深度学习文献中的流行选择. 从贝叶斯的角度来看, 这种复杂性度量也可以被解释为感兴趣函数的先验的负对数. 更一般地说, 这种复杂性可以通过将其纳入经验损失 (导致所谓的结构风险最小化) 来显式计算, 或者通过某种优化方案来隐式计算. 例如, 众所周知, 欠定最小二乘目标上的梯度下降将选择具有最小  $L_2$  范数的插值解. 这种隐式正则化结果对现代神经网络的扩展是当前研究的主题 (例如, 参见Blanc et al. (2020); Shamir and Vardi (2020); Razin and Cohen (2020); Gunasekar et al. (2017)). 总之, 一个自然的问题出现了: 如何定义有效的先验来捕捉现实世界预测任务的期望正则性 (expected regularities) 和复杂性?

## 2.2 维数诅咒

虽然低维 ( $d = 1, 2$  或  $3$ ) 插值是一项经典的信号处理任务, 使用日益复杂的正则类 (如样条插值、小波、曲波或脊波) 对估计误差进行非常精确的数学控制, 但高维问题的情况完全不同.

为了传达这一思想的本质, 让我们考虑一个经典的正则性 (regularity) 概念, 它可以很容易地扩展到高维: 1-李普希茨函数  $f : \mathcal{X} \rightarrow \mathbb{R}$ , 即对于所有  $x, x' \in \mathcal{X}$ , 满足  $|f(x) - f(x')| \leq \|x - x'\|$  的函数. 这一假设只要求目标函数是局部 (*locally*) 光滑的, 即如果我们稍微扰动输入  $x$  (用范数  $\|x - x'\|$  衡

量), 输出  $f(x)$  不允许有太大的变化. 如果我们对目标函数  $f$  的唯一了解是它是 1-李普希茨, 那么我们期望需要多少次观测才能确保我们的估计  $\tilde{f}$  接近  $f$ ? 图2显示, 一般的答案在  $d$  维上必然是指数的, 这表明随着输入维的增加, 李普希茨类的增长“太快”: 在许多即使是适度的  $d$  维的应用中, 样本的数量将大于宇宙中的原子数量. 如果用一个全局光滑性假设来代替李普希茨类, 比如 Sobolev 类  $\mathcal{H}^s(\Omega_d)$ , 情况不会更好. 事实上, 经典的结果 (Tsybakov, 2008) 为阶数为  $\epsilon^{-d/s}$  的 Sobolev 类建立了逼近和学习的极大极小率, 表明  $f$  上的额外平滑度假设仅在  $s \propto d$  时改善统计图像, 这在实践中是不现实的假设.

如果  $f \in L^2(\Omega_d)$  且广义的  $s$  阶导数是平方可积的  $\int |\omega|^{2s+1} |\hat{f}(\omega)|^2 d\omega < \infty$ , 则函数  $f$  属于 Sobolev 类  $\mathcal{H}^s(\Omega_d)$ , 其中  $\hat{f}$  是  $f$  的傅里叶变换; 参见第 4.2 节.

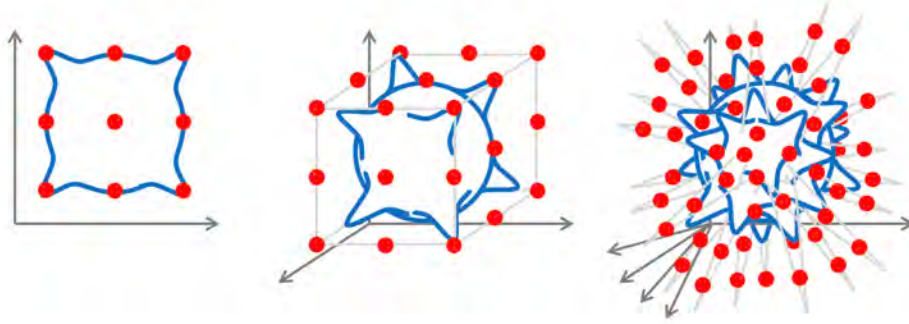


图 2: 我们考虑一个李普希茨函数  $f(x) = \sum_{j=1}^{2^d} z_j \phi(x - x_j)$ , 其中  $z_j = \pm 1$ ,  $x_j \in \mathbb{R}^d$  位于每个象限,  $\phi$  是一个局部支持的李普希茨“凸块”. 除非我们在第  $2^d$  个象限的大部分中观察到这个函数, 否则我们在预测它时会不断出错. 这个简单的几何论证可以通过最大差异 (Maximum Discrepancy) 的概念 (von Luxburg and Bousquet, 2004) 来形式化, 为李普希茨类定义为  $\kappa(d) = \mathbb{E}_{x, x'} \sup_{f \in \text{Lip}(1)} \left| \frac{1}{N} \sum_l f(x_l) - \frac{1}{N} \sum_l f(x'_l) \right| \simeq N^{-1/d}$ , 衡量两个独立  $N$  样本预期之间的最大预期差异. 确保  $\kappa(d) \simeq \epsilon$  需要  $N = \Theta(\epsilon^{-d})$ ; 相应的示例  $\{x_l\}_l$  定义了一个域的  $\epsilon$ -网络. 对于直径为 1 的  $d$  维欧氏域, 其大小呈指数增长为  $\epsilon^{-d}$ .

全连接神经网络定义了函数空间, 这些空间允许更灵活的正则性 (regularity) 概念, 这是通过考虑复杂性函数  $c$  的权重而获得的. 特别是, 通过选择稀疏促进正则化 (regularisation), 他们有能力打破这种维数灾难 (Bach, 2017). 然

而, 这是以对目标函数  $f$  的性质做出强有力的假设为代价的, 例如  $f$  依赖于输入的低维投影的集合 (见图3). 在大多数现实世界的应用中 (如计算机视觉、语音分析、物理或化学), 感兴趣的函数往往表现出复杂的长期相关性, 无法用低维投影来表达 (图3), 这使得这一假设不现实. 因此, 有必要通过利用物理域的空间结构和  $f$  的几何先验来定义正则性 (regularity) 的另一个来源, 正如我们在下一节3中所描述的.

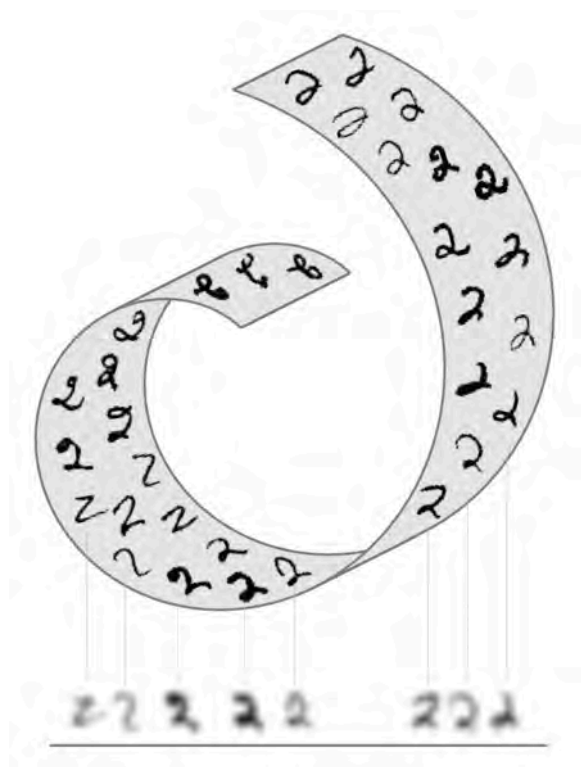


图 3: 如果对于某个未知的  $\mathbf{A} \in \mathbb{R}^{k \times d}, k \ll d$ , 假设未知函数  $f$  很好地近似为  $f(\mathbf{x}) \approx g(\mathbf{Ax})$  那么浅层神经网络可以捕捉这种归纳偏置, 例如Bach (2017). 在典型的应用中, 这种对低维投影的依赖是不现实的, 如这个例子所示: 低通滤波器将输入图像投影到低维子空间; 虽然它传达了大部分语义, 但大量信息丢失了.



### 3 Geometric Priors

Modern data analysis is synonymous with high-dimensional learning. While the simple arguments of Section 2.1 reveal the impossibility of learning from generic high-dimensional data as a result of the curse of dimensionality, there is hope for physically-structured data, where we can employ two fundamental principles: *symmetry* and *scale separation*. In the settings considered in this text, this additional structure will usually come from the structure of the domain underlying the input signals: we will assume that our machine learning system operates on *signals* (functions) on some domain  $\Omega$ . While in many cases linear combinations of points on  $\Omega$  is not well-defined, we can linearly combine signals on it, i.e., the space of signals forms a vector space. Moreover, since we can define an inner product between signals, this space is a *Hilbert space*.

$\Omega$  must be a vector space in order for an expression  $\alpha u + \beta v$  to make sense.

When  $\Omega$  has some additional structure, we may further restrict the kinds of signals in  $\mathcal{X}(\Omega, \mathcal{C})$ . For example, when  $\Omega$  is a smooth manifold, we may require the signals to be smooth. Whenever possible, we will omit the range  $\mathcal{C}$  for brevity.

When the domain  $\Omega$  is discrete,  $\mu$  can be chosen as the *counting measure*, in which case the integral becomes a sum. In the following, we will omit the measure and use  $du$  for brevity.

The space of  $\mathcal{C}$ -valued signals on  $\Omega$  (for  $\Omega$  a set, possibly with additional structure, and  $\mathcal{C}$  a vector space, whose dimensions are called *channels*)

$$\mathcal{X}(\Omega, \mathcal{C}) = \{x : \Omega \rightarrow \mathcal{C}\} \quad (1)$$

is a function space that has a vector space structure. Addition and scalar multiplication of signals is defined as:

$$(\alpha x + \beta y)(u) = \alpha x(u) + \beta y(u) \quad \text{for all } u \in \Omega,$$

with real scalars  $\alpha, \beta$ . Given an inner product  $\langle v, w \rangle_{\mathcal{C}}$  on  $\mathcal{C}$  and a measure  $\mu$  on  $\Omega$  (with respect to which we can define an integral), we can define an inner product on  $\mathcal{X}(\Omega, \mathcal{C})$  as

$$\langle x, y \rangle = \int_{\Omega} \langle x(u), y(u) \rangle_{\mathcal{C}} d\mu(u). \quad (2)$$

As a typical illustration, take  $\Omega = \mathbb{Z}_n \times \mathbb{Z}_n$  to be a two-dimensional  $n \times$



$n$  grid,  $x$  an RGB image (i.e. a signal  $x : \Omega \rightarrow \mathbb{R}^3$ ), and  $f$  a function (such as a single-layer Perceptron) operating on  $3n^2$ -dimensional inputs. As we will see in the following with greater detail, the domain  $\Omega$  is usually endowed with certain geometric structure and symmetries. Scale separation results from our ability to preserve important characteristics of the signal when transferring it onto a coarser version of the domain (in our example, subsampling the image by coarsening the underlying grid).

We will show that both principles, to which we will generically refer as *geometric priors*, are prominent in most modern deep learning architectures. In the case of images considered above, geometric priors are built into Convolutional Neural Networks (CNNs) in the form of convolutional filters with shared weights (exploiting translational symmetry) and pooling (exploiting scale separation). Extending these ideas to other domains such as graphs and manifolds and showing how geometric priors emerge from fundamental principles is the main goal of Geometric Deep Learning and the *leitmotif* of our text.

### 3.1 Symmetries, Representations, and Invariance

Informally, a *symmetry* of an object or system is a transformation that leaves a certain property of said object or system unchanged or *invariant*. Such transformations may be either smooth, continuous, or discrete. Symmetries are ubiquitous in many machine learning tasks. For example, in computer vision the object category is unchanged by shifts, so shifts are symmetries in the problem of visual object classification. In computational chemistry, the task of predicting properties of molecules independently of their orientation in space requires *rotational invariance*. Discrete symmetries emerge naturally when describing particle systems where particles do not have canonical ordering and thus can be arbitrarily permuted, as well as in many dynamical systems, via the time-reversal symmetry (such as systems in detailed balance

or the Newton's second law of motion). As we will see in Section 4.1, permutation symmetries are also central to the analysis of graph-structured data.

**Symmetry groups** The set of symmetries of an object satisfies a number of properties. First, symmetries may be combined to obtain new symmetries: if  $\mathfrak{g}$  and  $\mathfrak{h}$  are two symmetries, then their compositions  $\mathfrak{g} \circ \mathfrak{h}$  and  $\mathfrak{h} \circ \mathfrak{g}$  are also symmetries. The reason is that if both transformations leave the object invariant, then so does the composition of transformations, and hence the composition is also a symmetry. Furthermore, symmetries are always invertible, and the inverse is also a symmetry. This shows that the collection of all symmetries form an algebraic object known as a *group*. Since these objects will be a centerpiece of the mathematical model of Geometric Deep Learning, they deserve a formal definition and detailed discussion:

We will follow the juxtaposition notation convention used in group theory,  $\mathfrak{g} \circ \mathfrak{h} = \mathfrak{gh}$ , which should be read right-to-left: we first apply  $\mathfrak{h}$  and then  $\mathfrak{g}$ . The order is important, as in many cases symmetries are non-commutative. Readers familiar with Lie groups might be disturbed by our choice to use the Fraktur font to denote group elements, as it is a common notation of Lie algebras.

A *group* is a set  $\mathfrak{G}$  along with a binary operation  $\circ : \mathfrak{G} \times \mathfrak{G} \rightarrow \mathfrak{G}$  called *composition* (for brevity, denoted by juxtaposition  $\mathfrak{g} \circ \mathfrak{h} = \mathfrak{gh}$ ) satisfying the following axioms:

*Associativity:*  $(\mathfrak{gh})\mathfrak{k} = \mathfrak{g}(\mathfrak{h}\mathfrak{k})$  for all  $\mathfrak{g}, \mathfrak{h}, \mathfrak{k} \in \mathfrak{G}$ .

*Identity:* there exists a unique  $\mathfrak{e} \in \mathfrak{G}$  satisfying  $\mathfrak{eg} = \mathfrak{ge} = \mathfrak{g}$  for all  $\mathfrak{g} \in \mathfrak{G}$ .

*Inverse:* For each  $\mathfrak{g} \in \mathfrak{G}$  there is a unique inverse  $\mathfrak{g}^{-1} \in \mathfrak{G}$  such that  $\mathfrak{gg}^{-1} = \mathfrak{g}^{-1}\mathfrak{g} = \mathfrak{e}$ .

*Closure:* The group is closed under composition, i.e., for every  $\mathfrak{g}, \mathfrak{h} \in \mathfrak{G}$ , we have  $\mathfrak{gh} \in \mathfrak{G}$ .

Note that *commutativity* is not part of this definition, i.e. we may have  $\mathfrak{gh} \neq \mathfrak{hg}$ . Groups for which  $\mathfrak{gh} = \mathfrak{hg}$  for all  $\mathfrak{g}, \mathfrak{h} \in \mathfrak{G}$  are called commutative or *Abelian*.

After the Norwegian mathematician Niels Henrik Abel (1802–1829).

Though some groups can be very large and even infinite, they often arise from compositions of just a few elements, called *group generators*. Formally,  $\mathfrak{G}$  is said to be *generated* by a subset  $S \subseteq \mathfrak{G}$  (called the *group generator*) if every element  $\mathfrak{g} \in \mathfrak{G}$  can be written as a finite composition of the elements of  $S$  and their inverses. For instance, the symmetry group of an equilateral triangle (dihedral group  $D_3$ ) is generated by a  $60^\circ$  rotation and a reflection (Figure 4). The 1D *translation group*, which we will discuss in detail in the following, is generated by infinitesimal displacements; this is an example of a *Lie group* of differentiable symmetries.

Note that here we have defined a group as an abstract object, without saying what the group elements *are* (e.g. transformations of some domain), only how they *compose*. Hence, very different kinds of objects may have the same symmetry group. For instance, the aforementioned group of rotational and reflection symmetries of a triangle is the same as the group of permutations of a sequence of three elements (we can permute the corners in the triangle in any way using a rotation and reflection – see Figure 4).

**Group Actions and Group Representations** Rather than considering groups as abstract entities, we are mostly interested in how groups *act* on data. Since we assumed that there is some domain  $\Omega$  underlying our data, we will study how the group acts on  $\Omega$  (e.g. translation of points of the plane), and from there obtain actions of the same group on the space of signals  $\mathcal{X}(\Omega)$  (e.g. translations of planar images and feature maps).

A *group action* of  $\mathfrak{G}$  on a set  $\Omega$  is defined as a mapping  $(\mathfrak{g}, u) \mapsto \mathfrak{g}.u$  associating a group element  $\mathfrak{g} \in \mathfrak{G}$  and a point  $u \in \Omega$  with some other point on  $\Omega$  in a way that is compatible with the group operations, i.e.,  $\mathfrak{g}.(\mathfrak{h}.u) = (\mathfrak{g}\mathfrak{h}).u$  for all  $\mathfrak{g}, \mathfrak{h} \in \mathfrak{G}$  and  $u \in \Omega$ . We shall see numerous instances of group actions in the following sections. For example, in the plane the *Euclidean group*  $E(2)$  is the group of transformations of  $\mathbb{R}^2$  that preserves Euclidean distances,

Lie groups have a differentiable manifold structure. One such example that we will study in Section 4.3 is the special orthogonal group  $SO(3)$ , which is a 3-dimensional manifold.

The diagram shown in Figure 4 (where each node is associated with a group element, and each arrow with a generator), is known as the *Cayley diagram*.

Technically, what we define here is a *left* group action.

Distance-preserving transformations are called *isometries*. According to Klein’s Erlangen Programme, the classical Euclidean geometry arises from this group.

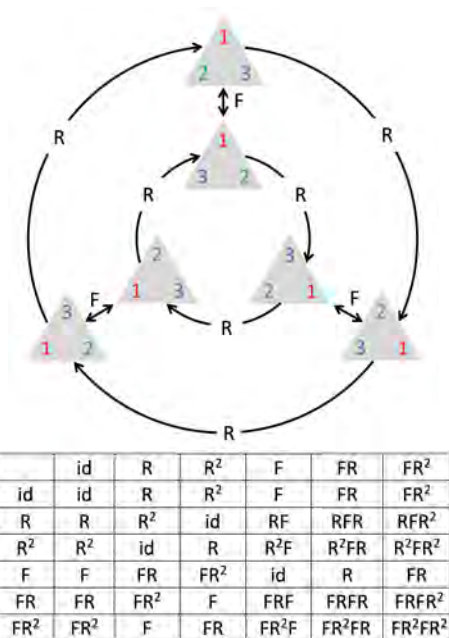


图 4: Left: an equilateral triangle with corners labelled by 1, 2, 3, and all possible rotations and reflections of the triangle. The group  $D_3$  of rotation/reflection symmetries of the triangle is generated by only two elements (rotation by  $60^\circ$   $R$  and reflection  $F$ ) and is the same as the group  $\Sigma_3$  of permutations of three elements. Right: the multiplication table of the group  $D_3$ . The element in the row  $\mathfrak{g}$  and column  $\mathfrak{h}$  corresponds to the element  $\mathfrak{gh}$ .

and consists of translations, rotations, and reflections. The same group, however, can also act on the space of *images* on the plane (by translating, rotating and flipping the grid of pixels), as well as on the representation spaces learned by a neural network. More precisely, if we have a group  $\mathfrak{G}$  acting on  $\Omega$ , we automatically obtain an action of  $\mathfrak{G}$  on the space  $\mathcal{X}(\Omega)$ :

$$(\mathfrak{g}.x)(u) = x(\mathfrak{g}^{-1}u). \quad (3)$$

Due to the inverse on  $\mathfrak{g}$ , this is indeed a valid group action, in that we have  $(\mathfrak{g}.\mathfrak{h}.x)(u) = ((\mathfrak{g}\mathfrak{h}).x)(u)$ .

The most important kind of group actions, which we will encounter repeatedly throughout this text, are *linear* group actions, also known as *group representations*. The action on signals in equation (3) is indeed linear, in the sense that

$$\mathfrak{g}.\alpha x + \beta x' = \alpha(\mathfrak{g}.x) + \beta(\mathfrak{g}.x')$$

for any scalars  $\alpha, \beta$  and signals  $x, x' \in \mathcal{X}(\Omega)$ . We can describe linear actions either as maps  $(\mathfrak{g}, x) \mapsto \mathfrak{g}.x$  that are linear in  $x$ , or equivalently, by currying, as a map  $\rho : \mathfrak{G} \rightarrow \mathbb{R}^{n \times n}$  that assigns to each group element  $\mathfrak{g}$  an (invertible) matrix  $\rho(\mathfrak{g})$ . The dimension  $n$  of the matrix is in general arbitrary and not necessarily related to the dimensionality of the group or the dimensionality of  $\Omega$ , but in applications to deep learning  $n$  will usually be the dimensionality of the feature space on which the group acts. For instance, we may have the group of 2D translations acting on a space of images with  $n$  pixels.

When  $\Omega$  is infinite, the space of signals  $\mathcal{X}(\Omega)$  is infinite dimensional, in which case  $\rho(\mathfrak{g})$  is a linear operator on this space, rather than a finite dimensional matrix. In practice, one must always discretise to a finite grid, though.

As with a general group action, the assignment of matrices to group elements should be compatible with the group action. More specifically, the matrix representing a composite group element  $\mathfrak{g}\mathfrak{h}$  should equal the matrix product of the representation of  $\mathfrak{g}$  and  $\mathfrak{h}$ :

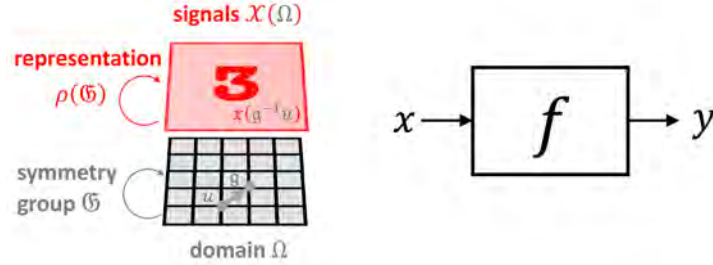


图 5: Three spaces of interest in Geometric Deep Learning: the (physical) domain  $\Omega$ , the space of signals  $\mathcal{X}(\Omega)$ , and the hypothesis class  $\mathcal{F}(\mathcal{X}(\Omega))$ . Symmetries of the domain  $\Omega$  (captured by the group  $\mathfrak{G}$ ) act on signals  $x \in \mathcal{X}(\Omega)$  through group representations  $\rho(\mathfrak{g})$ , imposing structure on the functions  $f \in \mathcal{F}(\mathcal{X}(\Omega))$  acting on such signals.

Similarly, a complex representation is a map  $\rho : \mathfrak{G} \rightarrow \mathbb{C}^{n \times n}$  satisfying the same equation.

A  $n$ -dimensional real *representation* of a group  $\mathfrak{G}$  is a map  $\rho : \mathfrak{G} \rightarrow \mathbb{R}^{n \times n}$ , assigning to each  $\mathfrak{g} \in \mathfrak{G}$  an *invertible* matrix  $\rho(\mathfrak{g})$ , and satisfying the condition  $\rho(\mathfrak{g}\mathfrak{h}) = \rho(\mathfrak{g})\rho(\mathfrak{h})$  for all  $\mathfrak{g}, \mathfrak{h} \in \mathfrak{G}$ . A representation is called *unitary* or *orthogonal* if the matrix  $\rho(\mathfrak{g})$  is unitary or orthogonal for all  $\mathfrak{g} \in \mathfrak{G}$ .

Written in the language of group representations, the action of  $\mathfrak{G}$  on signals  $x \in \mathcal{X}(\Omega)$  is defined as  $\rho(\mathfrak{g})x(u) = x(\mathfrak{g}^{-1}u)$ . We again verify that

$$(\rho(\mathfrak{g})(\rho(\mathfrak{h})x))(u) = (\rho(\mathfrak{g}\mathfrak{h})x)(u).$$

**Invariant and Equivariant functions** The symmetry of the domain  $\Omega$  underlying the signals  $\mathcal{X}(\Omega)$  imposes structure on the function  $f$  defined on such signals. It turns out to be a powerful inductive bias, improving learning efficiency by reducing the space of possible interpolants,  $\mathcal{F}(\mathcal{X}(\Omega))$ , to those which satisfy the symmetry priors. Two important cases we will be exploring in this text are *invariant* and *equivariant* functions.

In general,  $f$  depends both on the signal and the domain, i.e.,  $\mathcal{F}(\mathcal{X}(\Omega), \Omega)$ . We will often omit the latter dependency for brevity.

A function  $f : \mathcal{X}(\Omega) \rightarrow \mathcal{Y}$  is  $\mathfrak{G}$ -invariant if  $f(\rho(\mathfrak{g})x) = f(x)$  for all  $\mathfrak{g} \in \mathfrak{G}$  and  $x \in \mathcal{X}(\Omega)$ , i.e., its output is unaffected by the group action on the input.

A classical example of invariance is *shift-invariance*, arising in computer vision and pattern recognition applications such as image classification. The function  $f$  in this case (typically implemented as a Convolutional Neural Network) inputs an image and outputs the probability of the image to contain an object from a certain class (e.g. cat or dog). It is often reasonably assumed that the classification result should not be affected by the position of the object in the image, i.e., the function  $f$  must be shift-invariant. Multi-layer Perceptrons, which can approximate any smooth function, do not have this property – one of the reasons why early attempts to apply these architectures to problems of pattern recognition in the 1970s failed. The development of neural network architectures with local weight sharing, as epitomised by Convolutional Neural Networks, was, among other reasons, motivated by the need for shift-invariant object classification.

Note that signal processing books routinely use the term ‘shift-invariance’ referring to shift-equivariance, e.g. Linear Shift-invariant Systems.

If we however take a closer look at the convolutional layers of CNNs, we will find that they are not shift-invariant but *shift-equivariant*: in other words, a shift of the input to a convolutional layer produces a shift in the output feature maps by the same amount.

A function  $f : \mathcal{X}(\Omega) \rightarrow \mathcal{X}(\Omega)$  is  $\mathfrak{G}$ -equivariant if  $f(\rho(\mathfrak{g})x) = \rho(\mathfrak{g})f(x)$  for all  $\mathfrak{g} \in \mathfrak{G}$ , i.e., group action on the input affects the output in the same way.

Resorting again to computer vision, a prototypical application requiring shift-equivariance is image segmentation, where the output of  $f$  is a pixel-wise image mask. Obviously, the segmentation mask must follow shifts in the input image. In this example, the domains of the input and output are

More generally, we might have  $f : \mathcal{X}(\Omega) \rightarrow \mathcal{X}(\Omega')$  with input and output spaces having different domains  $\Omega, \Omega'$  and representations  $\rho, \rho'$  of the same group  $\mathfrak{G}$ . In this case, equivariance is defined as  $f(\rho(\mathfrak{g})x) = \rho'(\mathfrak{g})f(x)$ .

the same, but since the input has three color channels while the output has one channel per class, the representations  $(\rho, \mathcal{X}(\Omega, \mathcal{C}))$  and  $(\rho', \mathcal{X}(\Omega, \mathcal{C}'))$  are somewhat different.

However, even the previous use case of image classification is usually implemented as a sequence of convolutional (shift-equivariant) layers, followed by global pooling (which is shift-invariant). As we will see in Section 3.5, this is a general blueprint of a majority of deep learning architectures, including CNNs and Graph Neural Networks (GNNs).

### 3.2 Isomorphisms and Automorphisms

Invertible and structure-preserving maps between different objects often go under the generic name of *isomorphisms* (Greek for ‘equal shape’). An isomorphism from an object to itself is called an *automorphism*, or symmetry.

**Subgroups and Levels of structure** As mentioned before, a symmetry is a transformation that preserves some property or structure, and the set of all such transformations for a given structure forms a symmetry group. It happens often that there is not one but multiple structures of interest, and so we can consider several *levels of structure* on our domain  $\Omega$ . Hence, what counts as a symmetry depends on the structure under consideration, but in all cases a symmetry is an invertible map that respects this structure.

For a finite set, the cardinality is the number of elements (‘size’) of the set, and for infinite sets the cardinality indicates different kinds of infinities, such as the countable infinity of the natural numbers, or the uncountable infinity of the continuum  $\mathbb{R}$ .

On the most basic level, the domain  $\Omega$  is a *set*, which has a minimal amount of structure: all we can say is that the set has some *cardinality*. Self-maps that preserve this structure are *bijections* (invertible maps), which we may consider as set-level symmetries. One can easily verify that this is a group by checking the axioms: a compositions of two bijections is also a bijection (closure), the associativity stems from the associativity of the function composition, the map  $\tau(u) = u$  is the identity element, and for every  $\tau$  the inverse exists by definition, satisfying  $(\tau \circ \tau^{-1})(u) = (\tau^{-1} \circ \tau)(u) = u$ .

Depending on the application, there may be further levels of structure. For instance, if  $\Omega$  is a topological space, we can consider maps that preserve



*continuity*: such maps are called *homeomorphisms* and in addition to simple bijections between sets, are also continuous and have continuous inverse. Intuitively, continuous functions are well-behaved and map points in a neighbourhood (open set) around a point  $u$  to a neighbourhood around  $\tau(u)$ .

One can further demand that the map and its inverse are (continuously) *differentiable*, i.e., the map and its inverse have a derivative at every point (and the derivative is also continuous). This requires further differentiable structure that comes with differentiable manifolds, where such maps are called *diffeomorphisms* and denoted by  $\text{Diff}(\Omega)$ . Additional examples of structures we will encounter include *distances* or *metrics* (maps preserving them are called *isometries*) or *orientation* (to the best of our knowledge, orientation-preserving maps do not have a common Greek name).

Every differentiable function is continuous. If the map is continuously differentiable ‘sufficiently many times’, it is said to be *smooth*.

A *metric* or *distance* is a function  $d : \Omega \times \Omega \rightarrow [0, \infty)$  satisfying for all  $u, v, w \in \Omega$ :

*Identity of indiscernibles*:  $d(u, v) = 0$  iff  $u = v$ .

*Symmetry*:  $d(u, v) = d(v, u)$ .

*Triangle inequality*:  $d(u, v) \leq d(u, w) + d(w, v)$ .

A space equipped with a metric  $(\Omega, d)$  is called a *metric space*.

The right level of structure to consider depends on the problem. For example, when segmenting histopathology slide images, we may wish to consider flipped versions of an image as equivalent (as the sample can be flipped when put under the microscope), but if we are trying to classify road signs, we would only want to consider orientation-preserving transformations as symmetries (since reflections could change the meaning of the sign).

As we add levels of structure to be preserved, the symmetry group will get

smaller. Indeed, adding structure is equivalent to selecting a *subgroup*, which is a subset of the larger group that satisfies the axioms of a group by itself:

Let  $(\mathfrak{G}, \circ)$  be a group and  $\mathfrak{H} \subseteq \mathfrak{G}$  a subset.  $\mathfrak{H}$  is said to be a *subgroup* of  $\mathfrak{G}$  if  $(\mathfrak{H}, \circ)$  constitutes a group with the same operation.

For instance, the group of Euclidean isometries  $E(2)$  is a subgroup of the group of planar diffeomorphisms  $\text{Diff}(2)$ , and in turn the group of orientation-preserving isometries  $SE(2)$  is a subgroup of  $E(2)$ . This hierarchy of structure follows the Erlangen Programme philosophy outlined in the Preface: in Klein's construction, the Projective, Affine, and Euclidean geometries have increasingly more invariants and correspond to progressively smaller groups.

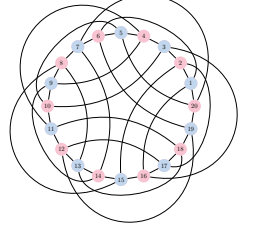
**Isomorphisms and Automorphisms** We have described symmetries as structure preserving and invertible maps *from an object to itself*. Such maps are also known as *automorphisms*, and describe a way in which an object is equivalent to itself. However, an equally important class of maps are the so-called *isomorphisms*, which exhibit an equivalence between two non-identical objects. These concepts are often conflated, but distinguishing them is necessary to create clarity for our following discussion.

To understand the difference, consider a set  $\Omega = \{0, 1, 2\}$ . An automorphism of the set  $\Omega$  is a bijection  $\tau : \Omega \rightarrow \Omega$  such as a cyclic shift  $\tau(u) = u+1 \pmod 3$ . Such a map preserves the cardinality property, and maps  $\Omega$  onto itself. If we have another set  $\Omega' = \{a, b, c\}$  with the same number of elements, then a bijection  $\eta : \Omega \rightarrow \Omega'$  such as  $\eta(0) = a, \eta(1) = b, \eta(2) = c$  is a *set isomorphism*.

As we will see in Section 4.1 for graphs, the notion of structure includes not just the number of nodes, but also the connectivity. An isomorphism  $\eta : \mathcal{V} \rightarrow \mathcal{V}'$  between two graphs  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  and  $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$  is thus a

bijection between the nodes that maps pairs of connected nodes to pairs of connected nodes, and likewise for pairs of non-connected nodes. Two isomorphic graphs are thus structurally identical, and differ only in the way their nodes are ordered. On the other hand, a graph automorphism or symmetry is a map  $\tau : \mathcal{V} \rightarrow \mathcal{V}$  maps the nodes of the graph back to itself, while preserving the connectivity. A graph with a non-trivial automorphism (i.e.,  $\tau \neq \text{id}$ ) presents symmetries.

I.e.,  $(\eta(u), \eta(v)) \in \mathcal{V}'$  iff  $(u, v) \in \mathcal{V}$ .



### 3.3 Deformation Stability

The symmetry formalism introduced in Sections 3.1–3.2 captures an idealised world where we know exactly which transformations are to be considered as symmetries, and we want to respect these symmetries *exactly*. For instance in computer vision, we might assume that planar translations are exact symmetries. However, the real world is noisy and this model falls short in two ways.

The *Folkman graph* (Folkman, 1967) is a beautiful example of a graph with 3840 automorphisms, exemplified by the many symmetric ways to draw it.

Firstly, while these simple groups provide a way to understand *global* symmetries of the domain  $\Omega$  (and by extension, of signals on it,  $\mathcal{X}(\Omega)$ ), they do not capture *local* symmetries well. For instance, consider a video scene with several objects, each moving along its own different direction. At subsequent frames, the resulting scene will contain approximately the same semantic information, yet no global translation explains the transformation from one frame to another. In other cases, such as a deformable 3D object viewed by a camera, it is simply very hard to describe the group of transformations that preserve the object identity. These examples illustrate that in reality we are more interested in a far larger set of transformations where global, exact invariance is replaced by a local, inexact one. In our discussion, we will distinguish between two scenarios: the setting where the domain  $\Omega$  is fixed, and signals  $x \in \mathcal{X}(\Omega)$  are undergoing deformations, and the setting where the domain  $\Omega$  itself may be deformed.



Two objects moving at different velocities in a video define a transformation outside the translation group.

E.g., the composition of two  $\epsilon$ -isometries is a  $2\epsilon$ -isometry, violating the closure property.

**Stability to signal deformations** In many applications, we know a priori that a small deformation of the signal  $x$  should not change the output of  $f(x)$ , so it is tempting to consider such deformations as symmetries. For instance, we could view small diffeomorphisms  $\tau \in \text{Diff}(\Omega)$ , or even small bijections, as symmetries. However, small deformations can be composed to form large deformations, so “small deformations” do not form a group, and we cannot ask for invariance or equivariance to small deformations only. Since large deformations can actually materially change the semantic content of the input, it is not a good idea to use the full group  $\text{Diff}(\Omega)$  as symmetry group either.

A better approach is to quantify how “far” a given  $\tau \in \text{Diff}(\Omega)$  is from a given symmetry subgroup  $\mathfrak{G} \subset \text{Diff}(\Omega)$  (e.g. translations) with a complexity measure  $c(\tau)$ , so that  $c(\tau) = 0$  whenever  $\tau \in \mathfrak{G}$ . We can now replace our previous definition of exact invariance and equivariance under group actions with a ‘softer’ notion of *deformation stability* (or *approximate invariance*):

$$\|f(\rho(\tau)x) - f(x)\| \leq Cc(\tau)\|x\|, \quad , \quad \forall x \in \mathcal{X}(\Omega) \quad (4)$$

where  $\rho(\tau)x(u) = x(\tau^{-1}u)$  as before, and where  $C$  is some constant independent of the signal  $x$ . A function  $f \in \mathcal{F}(\mathcal{X}(\Omega))$  satisfying the above equation is said to be *geometrically stable*. We will see examples of such functions in the next Section 3.4.

Since  $c(\tau) = 0$  for  $\tau \in \mathfrak{G}$ , this definition generalises the  $\mathfrak{G}$ -invariance property defined above. Its utility in applications depends on introducing an appropriate deformation cost. In the case of images defined over a continuous Euclidean plane, a popular choice is  $c^2(\tau) := \int_{\Omega} \|\nabla \tau(u)\|^2 du$ , which measures the ‘elasticity’ of  $\tau$ , i.e., how different it is from the displacement by a constant vector field. This deformation cost is in fact a norm often called the *Dirichlet energy*, and can be used to quantify how far  $\tau$  is from the translation group.

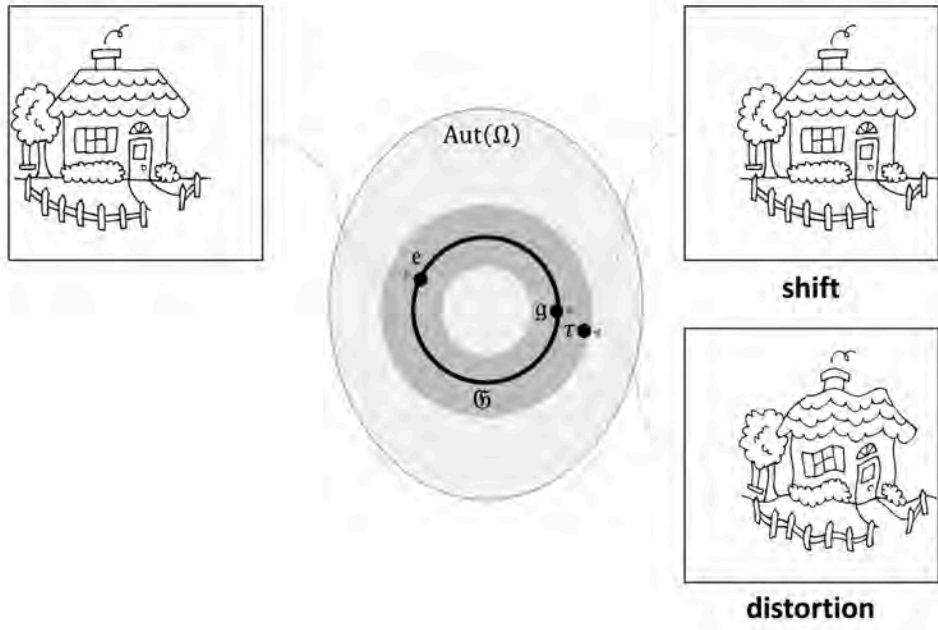


图 6: The set of all bijective mappings from  $\Omega$  into itself forms the *set automorphism group*  $\text{Aut}(\Omega)$ , of which a symmetry group  $\mathfrak{G}$  (shown as a circle) is a subgroup. Geometric Stability extends the notion of  $\mathfrak{G}$ -invariance and equivariance to ‘transformations around  $\mathfrak{G}$ ’ (shown as gray ring), quantified in the sense of some metric between transformations. In this example, a smooth distortion of the image is close to a shift.

**Stability to domain deformations** In many applications, the object being deformed is not the signal, but the geometric domain  $\Omega$  itself. Canonical instances of this are applications dealing with graphs and manifolds: a graph can model a social network at different instance of time containing slightly different social relations (follow graph), or a manifold can model a 3D object undergoing non-rigid deformations. This deformation can be quantified as follows. If  $\mathcal{D}$  denotes the space of all possible variable domains (such as the space of all graphs, or the space of Riemannian manifolds), one can define for  $\Omega, \tilde{\Omega} \in \mathcal{D}$  an appropriate metric (‘distance’)  $d(\Omega, \tilde{\Omega})$  satisfying  $d(\Omega, \tilde{\Omega}) = 0$  if  $\Omega$  and  $\tilde{\Omega}$  are equivalent in some sense: for example, the graph edit distance vanishes when the graphs are isomorphic, and the Gromov-Hausdorff distance between Riemannian manifolds equipped with geodesic distances vanishes when two manifolds are isometric.

The graph edit distance measures the minimal cost of making two graphs isomorphic by a sequences of graph edit operations. The Gromov-Hausdorff distance measures the smallest possible metric distortion of a correspondence between two metric spaces, see [Gromov \(1981\)](#).

A common construction of such distances between domains relies on some family of invertible mapping  $\eta : \Omega \rightarrow \tilde{\Omega}$  that try to ‘align’ the domains in a way that the corresponding structures are best preserved. For example, in the case of graphs or Riemannian manifolds (regarded as metric spaces with the geodesic distance), this alignment can compare pair-wise adjacency or distance structures ( $d$  and  $\tilde{d}$ , respectively),

$$d_{\mathcal{D}}(\Omega, \tilde{\Omega}) = \inf_{\eta \in \mathfrak{G}} \|d - \tilde{d} \circ (\eta \times \eta)\|$$

where  $\mathfrak{G}$  is the group of isomorphisms such as bijections or isometries, and the norm is defined over the product space  $\Omega \times \Omega$ . In other words, a distance between elements of  $\Omega, \tilde{\Omega}$  is ‘lifted’ to a distance between the domains themselves, by accounting for all the possible alignments that preserve the internal structure. Given a signal  $x \in \mathcal{X}(\Omega)$  and a deformed domain  $\tilde{\Omega}$ , one can then consider the deformed signal  $\tilde{x} = x \circ \eta^{-1} \in \mathcal{X}(\tilde{\Omega})$ .

Two graphs can be aligned by the Quadratic Assignment Problem (QAP), which considers in its simplest form two graphs  $G, \tilde{G}$  of the same size  $n$ , and solves

$$\min_{\mathbf{P} \in \Sigma_n} \text{trace}(\mathbf{A}\mathbf{P}\mathbf{A}^{\top}),$$

where  $\mathbf{A}, \tilde{\mathbf{A}}$  are the respective adjacency matrices and  $\Sigma_n$  is the group of  $n \times n$  permutation matrices. The graph edit distance can be associated

By slightly abusing the notation, we define  $\mathcal{X}(\mathcal{D}) = \{(\mathcal{X}(\Omega), \Omega) : \Omega \in \mathcal{D}\}$  as the ensemble of possible input signals defined over a varying domain. A

function  $f : \mathcal{X}(\mathcal{D}) \rightarrow \mathcal{Y}$  is stable to domain deformations if

$$\|f(x, \Omega) - f(\tilde{x}, \tilde{\Omega})\| \leq C\|x\|d_{\mathcal{D}}(\Omega, \tilde{\Omega}) \quad (5)$$

for all  $\Omega, \tilde{\Omega} \in \mathcal{D}$ , and  $x \in \mathcal{X}(\Omega)$ . We will discuss this notion of stability in the context of manifolds in Sections 4.4–4.6, where isometric deformations play a crucial role. Furthermore, it can be shown that the stability to domain deformations is a natural generalisation of the stability to signal deformations, by viewing the latter in terms of deformations of the volume form Gama et al. (2019).

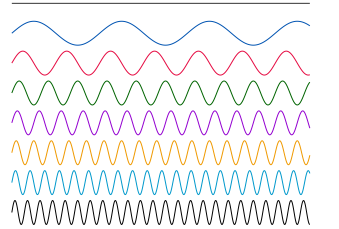
### 3.4 Scale Separation

While deformation stability substantially strengthens the global symmetry priors, it is not sufficient in itself to overcome the curse of dimensionality, in the sense that, informally speaking, there are still “too many” functions that respect (4) as the size of the domain grows. A key insight to overcome this curse is to exploit the multiscale structure of physical tasks. Before describing multiscale representations, we need to introduce the main elements of Fourier transforms, which rely on frequency rather than scale.

**Fourier Transform and Global invariants** Arguably the most famous signal decomposition is the *Fourier transform*, the cornerstone of harmonic analysis. The classical one-dimensional Fourier transform

$$\hat{x}(\xi) = \int_{-\infty}^{+\infty} x(u)e^{-i\xi u} du$$

expresses the function  $x(u) \in L^2(\Omega)$  on the domain  $\Omega = \mathbb{R}$  as a linear combination of orthogonal oscillating *basis functions*  $\varphi_{\xi}(u) = e^{i\xi u}$ , indexed by their rate of oscillation (or *frequency*)  $\xi$ . Such an organisation into frequencies reveals important information about the signal, e.g. its smoothness and



Fourier basis functions have global support. As a result, local signals produce energy across all frequencies.

localisation. The Fourier basis itself has a deep geometric foundation and can be interpreted as the natural vibrations of the domain, related to its geometric structure (see e.g. [Berger \(2012\)](#)).

In the following, we will use convolution and (cross-)correlation. The Fourier transform plays a crucial role in signal processing as it offers a dual formulation of *convolution*,

$$(x \star \theta)(u) = \int_{-\infty}^{+\infty} x(v)\theta(u+v)dv$$

$$(x \star \theta)(u) = \int_{-\infty}^{+\infty} x(v)\theta(u-v)dv$$

interchangeably, as it is a standard model of linear signal filtering (here and in the following,  $x$  denotes the signal and  $\theta$  the filter). As we will show in the following, the convolution operator is diagonalised in the Fourier basis, making it possible to express convolution as the product of the respective Fourier transforms,

$$\widehat{(x \star \theta)}(\xi) = \hat{x}(\xi) \cdot \hat{\theta}(\xi),$$

purely notational. a fact known in signal processing as the Convolution Theorem.

As it turns out, many fundamental differential operators such as the Laplacian are described as convolutions on Euclidean domains. Since such differential operators can be defined intrinsically over very general geometries, this provides a formal procedure to extend Fourier transforms beyond Euclidean domains, including graphs, groups and manifolds. We will discuss this in detail in Section 4.4.

An essential aspect of Fourier transforms is that they reveal *global* properties of the signal and the domain, such as smoothness or conductance. Such global behavior is convenient in presence of global symmetries of the domain such as translation, but not to study more general diffeomorphisms. This requires a representation that trades off spatial and frequential localisation, as we see next.

**Multiscale representations** The notion of local invariance can be articulated by switching from a Fourier frequency-based representation to a *scale-*



*based* representation, the cornerstone of multi-scale decomposition methods such as *wavelets*. The essential insight of multi-scale methods is to decompose functions defined over the domain  $\Omega$  into elementary functions that are localised *both in space and frequency*. In the case of wavelets, this is achieved by correlating a translated and dilated filter (*mother wavelet*)  $\psi$ , producing a combined spatio-frequency representation called a *continuous wavelet transform*

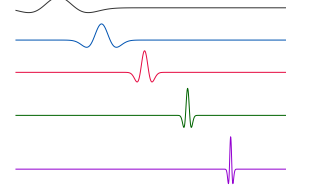
$$(W_\psi x)(u, \xi) = \xi^{-1/2} \int_{-\infty}^{+\infty} \psi\left(\frac{v-u}{\xi}\right) x(v) dv.$$

The translated and dilated filters are called *wavelet atoms*; their spatial position and dilation correspond to the coordinates  $u$  and  $\xi$  of the wavelet transform. These coordinates are usually sampled dyadically ( $\xi = 2^{-j}$  and  $u = 2^{-j}k$ ), with  $j$  referred to as *scale*. Multi-scale signal representations bring important benefits in terms of capturing regularity properties beyond global smoothness, such as piece-wise smoothness, which made them a popular tool in signal and image processing and numerical analysis in the 90s.

**Deformation stability of Multiscale representations:** The benefit of multiscale localised wavelet decompositions over Fourier decompositions is revealed when considering the effect of small deformations ‘nearby’ the underlying symmetry group. Let us illustrate this important concept in the Euclidean domain and the translation group. Since the Fourier representation diagonalises the shift operator (which can be thought of as convolution, as we will see in more detail in Section 4.2), it is an efficient representation for translation transformations. However, Fourier decompositions are unstable under high-frequency deformations. In contrast, wavelet decompositions offer a stable representation in such cases.

Indeed, let us consider  $\tau \in \text{Aut}(\Omega)$  and its associated linear representation  $\rho(\tau)$ . When  $\tau(u) = u - v$  is a shift, as we will verify in Section 4.2, the operator  $\rho(\tau) = S_v$  is a *shift operator* that commutes with convolution.

See [Mallat \(1999\)](#) for a comprehensive introduction.



Contrary to Fourier, wavelet atoms are localised and multi-scale, allowing to capture fine details of the signal with atoms having small spatial support and coarse details with atoms having large spatial support. The term *atom* here is synonymous with ‘basis element’ in Fourier analysis, with the caveat that wavelets are redundant (over-complete).

Since convolution operators are diagonalised by the Fourier transform, the action of shift in the frequency domain amounts to shifting the complex phase of the Fourier transform,

$$(\widehat{S_v x})(\xi) = e^{-i\xi v} \hat{x}(\xi).$$

Thus, the *Fourier modulus*  $f(x) = |\hat{x}|$  removing the complex phase is a simple shift-invariant function,  $f(S_v x) = f(x)$ . However, if we have only approximate translation,  $\tau(u) = u - \tilde{\tau}(u)$  with  $\|\nabla \tau\|_\infty = \sup_{u \in \Omega} \|\nabla \tilde{\tau}(u)\| \leq \epsilon$ , the situation is entirely different: it is possible to show that

$$\frac{\|f(\rho(\tau)x) - f(x)\|}{\|x\|} = \mathcal{O}(1)$$

irrespective of how small  $\epsilon$  is (i.e., how close is  $\tau$  to being a shift). Consequently, such Fourier representation is *unstable under deformations*, however small. This unstability is manifested in general domains and non-rigid transformations; we will see another instance of this unstability in the analysis of 3d shapes using the natural extension of Fourier transforms described in Section 4.4.

Wavelets offer a remedy to this problem that also reveals the power of multi-scale representations. In the above example, we can show (Mallat, 2012) that the wavelet decomposition  $W_\psi x$  is *approximately equivariant* to deformations,

$$\frac{\|\rho(\tau)(W_\psi x) - W_\psi(\rho(\tau)x)\|}{\|x\|} = \mathcal{O}(\epsilon).$$

This notation implies that

$\rho(\tau)$  acts on the spatial

coordinate of  $(W_\psi x)(u, \xi)$ .

In other words, decomposing the signal information into scales using localised filters rather than frequencies turns a global unstable representation into a family of locally stable features. Importantly, such measurements at different scales are not yet invariant, and need to be progressively processed towards the low frequencies, hinting at the deep compositional nature of modern neural networks, and captured in our Blueprint for Geometric Deep Learning, presented next.

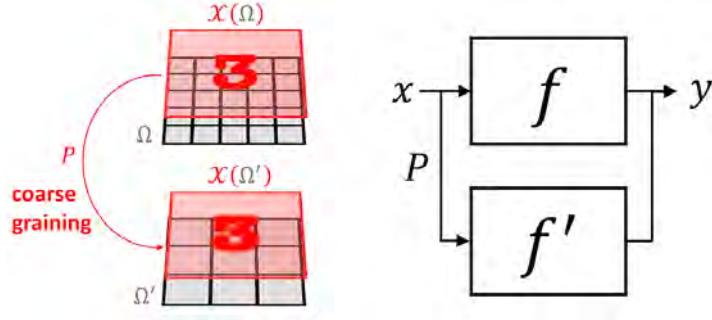


图 7: Illustration of Scale Separation for image classification tasks. The classifier  $f'$  defined on signals on the coarse grid  $\mathcal{X}(\Omega')$  should satisfy  $f \approx f' \circ P$ , where  $P : \mathcal{X}(\Omega) \rightarrow \mathcal{X}(\Omega')$ .

**Scale Separation Prior:** We can build from this insight by considering a multiscale coarsening of the data domain  $\Omega$  into a hierarchy  $\Omega_1, \dots, \Omega_J$ . As it turns out, such coarsening can be defined on very general domains, including grids, graphs, and manifolds. Informally, a coarsening assimilates nearby points  $u, u' \in \Omega$  together, and thus only requires an appropriate notion of *metric* in the domain. If  $\mathcal{X}_j(\Omega_j, \mathcal{C}_j) := \{x_j : \Omega_j \rightarrow \mathcal{C}_j\}$  denotes signals defined over the coarsened domain  $\Omega_j$ , we informally say that a function  $f : \mathcal{X}(\Omega) \rightarrow \mathcal{Y}$  is *locally stable* at scale  $j$  if it admits a factorisation of the form  $f \approx f_j \circ P_j$ , where  $P_j : \mathcal{X}(\Omega) \rightarrow \mathcal{X}_j(\Omega_j)$  is a non-linear *coarse graining* and  $f_j : \mathcal{X}_j(\Omega_j) \rightarrow \mathcal{Y}$ . In other words, while the target function  $f$  might depend on complex long-range interactions between features over the whole domain, in locally-stable functions it is possible to *separate* the interactions across scales, by first focusing on localised interactions that are then propagated towards the coarse scales.

Such principles are of fundamental importance in many areas of physics and mathematics, as manifested for instance in statistical physics in the so-called renormalisation group, or leveraged in important numerical algorithms such as the Fast Multipole Method. In machine learning, multiscale represen-

Fast Multipole Method (FMM) is a numerical technique originally developed to speed up the calculation of long-ranged forces in  $n$ -body problems. FMM groups sources that lie close together and treats them as a single source.

tations and local invariance are the fundamental mathematical principles underpinning the efficiency of Convolutional Neural Networks and Graph Neural Networks and are typically implemented in the form of *local pooling*. In future work, we will further develop tools from computational harmonic analysis that unify these principles across our geometric domains and will shed light onto the statistical learning benefits of scale separation.

### 3.5 The Blueprint of Geometric Deep Learning

The geometric principles of Symmetry, Geometric Stability, and Scale Separation discussed in Sections 3.1–3.4 can be combined to provide a universal blueprint for learning stable representations of high-dimensional data. These representations will be produced by functions  $f$  operating on signals  $\mathcal{X}(\Omega, \mathcal{C})$  defined on the domain  $\Omega$ , which is endowed with a symmetry group  $\mathfrak{G}$ .

The geometric priors we have described so far do not prescribe a specific *architecture* for building such representation, but rather a series of necessary conditions. However, they hint at an axiomatic construction that provably satisfies these geometric priors, while ensuring a highly expressive representation that can approximate any target function satisfying such priors.

A simple initial observation is that, in order to obtain a highly expressive representation, we are required to introduce a non-linear element, since if  $f$

Here,  $\mu(\mathfrak{g})$  is known as the *Haar measure* of the group  $\mathfrak{G}$ , and the integral is performed over the entire group.

is linear and  $\mathfrak{G}$ -invariant, then for all  $x \in \mathcal{X}(\Omega)$ ,

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

which indicates that  $F$  only depends on  $x$  through the  $\mathfrak{G}$ -average  $Ax = \frac{1}{\mu(\mathfrak{G})} \int_{\mathfrak{G}} (\mathfrak{g}.x) d\mu(\mathfrak{g})$ . In the case of images and translation, this would entail using only the average RGB color of the input!

While this reasoning shows that the family of *linear invariants* is not a very rich object, the family of *linear equivariants* provides a much more

powerful tool, since it enables the construction of rich and stable features by composition with appropriate non-linear maps, as we will now explain. Indeed, if  $B : \mathcal{X}(\Omega, \mathcal{C}) \rightarrow \mathcal{X}(\Omega, \mathcal{C}')$  is  $\mathfrak{G}$ -equivariant satisfying  $B(\mathfrak{g}.x) = \mathfrak{g}.B(x)$  for all  $x \in \mathcal{X}$  and  $\mathfrak{g} \in \mathfrak{G}$ , and  $\sigma : \mathcal{C}' \rightarrow \mathcal{C}''$  is an arbitrary (non-linear) map, then we easily verify that the composition  $U := (\sigma \circ B) : \mathcal{X}(\Omega, \mathcal{C}) \rightarrow \mathcal{X}(\Omega, \mathcal{C}'')$  is also  $\mathfrak{G}$ -equivariant, where  $\sigma : \mathcal{X}(\Omega, \mathcal{C}') \rightarrow \mathcal{X}(\Omega, \mathcal{C}'')$  is the element-wise instantiation of  $\sigma$  given as  $(\sigma(x))(u) := \sigma(x(u))$ .

This simple property allows us to define a very general family of  $\mathfrak{G}$ -invariants, by composing  $U$  with the group averages  $A \circ U : \mathcal{X}(\Omega, \mathcal{C}) \rightarrow \mathcal{C}''$ . A natural question is thus whether any  $\mathfrak{G}$ -invariant function can be approximated at arbitrary precision by such a model, for appropriate choices of  $B$  and  $\sigma$ . It is not hard to adapt the standard Universal Approximation Theorems from unstructured vector inputs to show that shallow ‘geometric’ networks are also universal approximators, by properly generalising the group average to a general non-linear invariant. However, as already described in the case of Fourier versus Wavelet invariants, there is a fundamental tension between shallow global invariance and deformation stability. This motivates an alternative representation, which considers instead *localised* equivariant maps. Assuming that  $\Omega$  is further equipped with a distance metric  $d$ , we call an equivariant map  $U$  localised if  $(Ux)(u)$  depends only on the values of  $x(v)$  for  $\mathcal{N}_u = \{v : d(u, v) \leq r\}$ , for some small radius  $r$ ; the latter set  $\mathcal{N}_u$  is called the *receptive field*.

A single layer of local equivariant map  $U$  cannot approximate functions with long-range interactions, but a composition of several local equivariant maps  $U_J \circ U_{J-1} \cdots \circ U_1$  increases the receptive field while preserving the stability properties of local equivariants. The receptive field is further increased by interleaving downsampling operators that coarsen the domain (again assuming a metric structure), completing the parallel with Multiresolution Analysis (MRA, see e.g. [Mallat \(1999\)](#)).

Such proofs have been demonstrated, for example, for the Deep Sets model by [Zaheer et al. \(2017\)](#).

Meaningful metrics can be defined on grids, graphs, manifolds, and groups. A notable exception are sets, where there is no predefined notion of metric.

The term ‘receptive field’ originated in the neuroscience literature, referring to the spatial domain that affects the output of a given neuron.

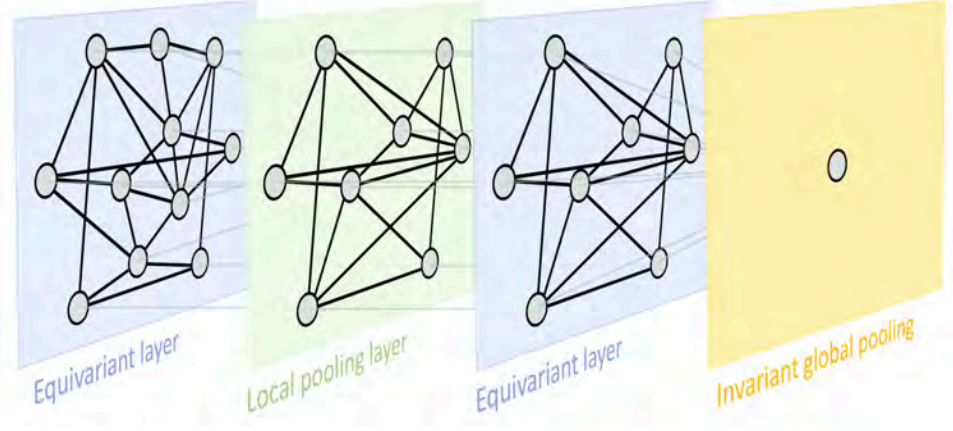


图 8: Geometric Deep Learning blueprint, exemplified on a graph. A typical Graph Neural Network architecture may contain permutation equivariant layers (computing node-wise features), local pooling (graph coarsening), and a permutation-invariant global pooling layer (readout layer).

In summary, the geometry of the input domain, with knowledge of an underlying symmetry group, provides three key building blocks: (i) a local equivariant map, (ii) a global invariant map, and (iii) a coarsening operator. These building blocks provide a rich function approximation space with prescribed invariance and stability properties by combining them together in a scheme we refer to as the *Geometric Deep Learning Blueprint* (Figure 8).

### Geometric Deep Learning Blueprint

Let  $\Omega$  and  $\Omega'$  be domains,  $\mathfrak{G}$  a symmetry group over  $\Omega$ , and write  $\Omega' \subseteq \Omega$  if  $\Omega'$  can be considered a compact version of  $\Omega$ .

We define the following building blocks:

*Linear  $\mathfrak{G}$ -equivariant layer*  $B : \mathcal{X}(\Omega, \mathcal{C}) \rightarrow \mathcal{X}(\Omega', \mathcal{C}')$  satisfying  $B(\mathfrak{g}.x) = \mathfrak{g}.B(x)$  for all  $\mathfrak{g} \in \mathfrak{G}$  and  $x \in \mathcal{X}(\Omega, \mathcal{C})$ .

*Nonlinearity*  $\sigma : \mathcal{C} \rightarrow \mathcal{C}'$  applied element-wise as  $(\sigma(x))(u) = \sigma(x(u))$ .

*Local pooling (coarsening)*  $P : \mathcal{X}(\Omega, \mathcal{C}) \rightarrow \mathcal{X}(\Omega', \mathcal{C})$ , such that  $\Omega' \subseteq \Omega$ .

*$\mathfrak{G}$ -invariant layer (global pooling)*  $A : \mathcal{X}(\Omega, \mathcal{C}) \rightarrow \mathcal{Y}$  satisfying  $A(\mathfrak{g}.x) = A(x)$  for all  $\mathfrak{g} \in \mathfrak{G}$  and  $x \in \mathcal{X}(\Omega, \mathcal{C})$ .

Using these blocks allows constructing  $\mathfrak{G}$ -invariant functions  $f : \mathcal{X}(\Omega, \mathcal{C}) \rightarrow \mathcal{Y}$  of the form

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

where the blocks are selected such that the output space of each block matches the input space of the next one. Different blocks may exploit different choices of symmetry groups  $\mathfrak{G}$ .

**Different settings of Geometric Deep Learning** One can make an important distinction between the setting when the domain  $\Omega$  is assumed to be *fixed* and one is only interested in varying input signals defined on that domain,

or the domain is part of the input as *varies* together with signals defined on it. A classical instance of the former case is encountered in computer vision applications, where images are assumed to be defined on a fixed domain (grid). Graph classification is an example of the latter setting, where both the structure of the graph as well as the signal defined on it (e.g. node features) are important. In the case of varying domain, geometric stability (in the sense of insensitivity to the deformation of  $\Omega$ ) plays a crucial role in Geometric Deep Learning architectures.

This blueprint has the right level of generality to be used across a wide range of geometric domains. Different Geometric Deep Learning methods thus differ in their choice of the domain, symmetry group, and the specific implementation details of the aforementioned building blocks. As we will see in the following, a large class of deep learning architectures currently in use fall into this scheme and can thus be derived from common geometric principles.

In the following sections (4.1–4.6) we will describe the various geometric domains focusing on the ‘5G’, and in Sections 5.1–5.8 the specific implementations of Geometric Deep Learning on these domains.



Architecture	Domain $\Omega$	Symmetry group $\mathfrak{G}$
<i>CNN</i>	Grid	Translation
<i>Spherical CNN</i>	Sphere / $\text{SO}(3)$	Rotation $\text{SO}(3)$
<i>Intrinsic / Mesh CNN</i>	Manifold	Isometry $\text{Iso}(\Omega)$ / Gauge symmetry $\text{SO}(2)$
<i>GNN</i>	Graph	Permutation $\Sigma_n$
<i>Deep Sets</i>	Set	Permutation $\Sigma_n$
<i>Transformer</i>	Complete Graph	Permutation $\Sigma_n$
<i>LSTM</i>	1D Grid	Time warping

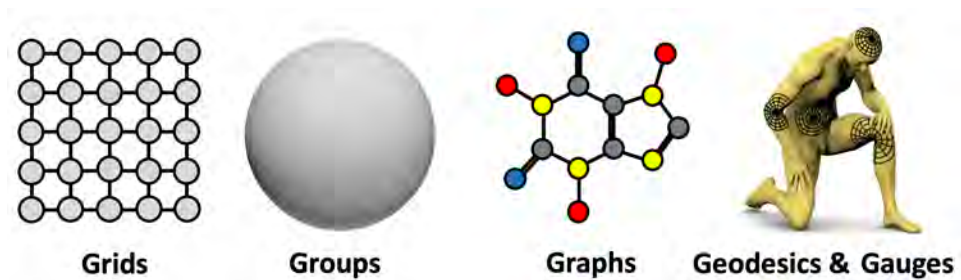


图 9: The 5G of Geometric Deep Learning: grids, groups & homogeneous spaces with global symmetry, graphs, geodesics & metrics on manifolds, and gauges (frames for tangent or feature spaces).

## 4 几何域：五个世代

The main focus of our text will be on graphs, grids, groups, geodesics, and gauges. In this context, by ‘groups’ we mean global symmetry transformations in homogeneous space, by ‘geodesics’ metric structures on manifolds, and by ‘gauges’ local reference frames defined on tangent bundles (and vector bundles in general). These notions will be explained in more detail later. In the next sections, we will discuss in detail the main elements in common and the key distinguishing features between these structures and describe the symmetry groups associated with them. Our exposition is not in the order of generality – in fact, grids are particular cases of graphs – but a way to highlight important concepts underlying our Geometric Deep Learning blueprint.

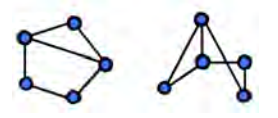
### 4.1 图与集合

In multiple branches of science, from sociology to particle physics, graphs are used as models of systems of relations and interactions. From our perspective, graphs give rise to a very basic type of invariance modelled by the

group of permutations. Furthermore, other objects of interest to us, such as grids and sets, can be obtained as a particular case of graphs.

A *graph*  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  is a collection of *nodes*  $\mathcal{V}$  and *edges*  $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$  between pairs of nodes. For the purpose of the following discussion, we will further assume the nodes to be endowed with  $s$ -dimensional *node features*, denoted by  $\mathbf{x}_u$  for all  $u \in \mathcal{V}$ . Social networks are perhaps among the most commonly studied examples of graphs, where nodes represent users, edges correspond to friendship relations between them, and node features model user properties such as age, profile picture, etc. It is also often possible to endow the edges, or entire graphs, with features; but as this does not alter the main findings of this section, we will defer discussing it to future work.

Depending on the application field, nodes may also be called *vertices*, and edges are often referred to as *links* or *relations*. We will use these terms interchangeably.



The key structural property of graphs is that the nodes in  $\mathcal{V}$  are usually not assumed to be provided in any particular order, and thus any operations performed on graphs should not depend on the ordering of nodes. The desirable property that functions acting on graphs should satisfy is thus *permutation invariance*, and it implies that for any two *isomorphic* graphs, the outcomes of these functions are identical. We can see this as a particular setting of our blueprint, where the domain  $\Omega = \mathcal{G}$  and the space  $\mathcal{X}(\mathcal{G}, \mathbb{R}^d)$  is that of  $d$ -dimensional node-wise signals. The symmetry we consider is given by the *permutation group*  $\mathfrak{G} = \Sigma_n$ , whose elements are all the possible orderings of the set of node indices  $\{1, \dots, n\}$ .

*Isomorphism* is an edge-preserving bijection between two graphs. Two isomorphic graphs shown here are identical up to reordering of their nodes.

Let us first illustrate the concept of permutation invariance on *sets*, a special case of graphs without edges (i.e.,  $\mathcal{E} = \emptyset$ ). By stacking the node features as rows of the  $n \times d$  matrix  $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_n)^\top$ , we do effectively specify an ordering of the nodes. The action of the permutation  $\mathbf{g} \in \Sigma_n$  on the set of nodes amounts to the reordering of the rows of  $\mathbf{X}$ , which can be represented as an  $n \times n$  *permutation matrix*  $\rho(\mathbf{g}) = \mathbf{P}$ , where each row and column contains exactly one 1 and all the other entries are zeros.

There are exactly  $n!$  such permutations, so  $\Sigma_n$  is, even for modest  $n$ , a very large group.

A function  $f$  operating on this set is then said to be *permutation invariant* if, for any such permutation matrix  $\mathbf{P}$ , it holds that  $f(\mathbf{PX}) = f(\mathbf{X})$ . One simple such function is

$$f(\mathbf{X}) = \phi \left( \sum_{u \in \mathcal{V}} \psi(\mathbf{x}_u) \right), \quad (6)$$

where the function  $\psi$  is independently applied to every node's features, and  $\phi$  is applied on its *sum-aggregated* outputs: as sum is independent of the order in which its inputs are provided, such a function is invariant with respect to the permutation of the node set, and is hence guaranteed to always return the same output, no matter how the nodes are permuted.

Functions like the above provide a ‘global’ graph-wise output, but very often, we will be interested in functions that act ‘locally’, in a node-wise manner. For example, we may want to apply some function to *update* the features in every node, obtaining the set of *latent* node features. If we stack these latent features into a matrix  $\mathbf{H} = \mathbf{F}(\mathbf{X})$  is no longer permutation invariant: the order of the rows of  $\mathbf{H}$  should be *tied* to the order of the rows of  $\mathbf{X}$ , so that we know which output node feature corresponds to which input node.

We use the bold notation for our function  $\mathbf{F}(\mathbf{X})$  to emphasise it outputs node-wise vector features and is hence a matrix-valued function.

We need instead a more fine-grained notion of *permutation equivariance*, stating that, once we “commit” to a permutation of inputs, it consistently permutes the resulting objects. Formally,  $\mathbf{F}(\mathbf{X})$  is a *permutation equivariant* function if, for any permutation matrix  $\mathbf{P}$ , it holds that  $\mathbf{F}(\mathbf{PX}) = \mathbf{PF}(\mathbf{X})$ . A shared node-wise linear transform

$$\mathbf{F}(\mathbf{X}) = \mathbf{X} \quad (7)$$

specified by a weight matrix  $\Theta \in \mathbb{R}^{d \times d'}$ , is one possible construction of such a permutation equivariant function, producing in our example latent features of the form  $\mathbf{h}_u = \Theta^\top \mathbf{x}_u$ .

This construction arises naturally from our Geometric Deep Learning blueprint. We can first attempt to characterise *linear equivariants* (functions of the form

$\mathbf{F}\mathbf{P}\mathbf{X} = \mathbf{P}\mathbf{F}\mathbf{X}$ ), for which it is easy to verify that any such map can be written as a linear combination of two *generators*, the identity  $\mathbf{F}_1\mathbf{X} = \mathbf{X}$  and the average  $\mathbf{F}_2\mathbf{X} = \frac{1}{n}\mathbf{1}\mathbf{1}^\top\mathbf{X} = \frac{1}{n}\sum_{u=1}^n \mathbf{x}_u$ . As will be described in Section 5.4, the popular Deep Sets (Zaheer et al., 2017) architecture follows precisely this blueprint.

We can now generalise the notions of permutation invariance and equivariance from sets to graphs. In the generic setting  $\mathcal{E} \neq \emptyset$ , the graph connectivity can be represented by the  $n \times n$  *adjacency matrix*  $\mathbf{A}$ , defined as

$$a_{uv} = \begin{cases} 1 & (u, v) \in \mathcal{E} \\ 0 & \text{otherwise.} \end{cases} \quad (8)$$

When the graph is *undirected*, i.e.  $(u, v) \in \mathcal{E}$  iff  $(v, u) \in \mathcal{E}$ , the adjacency matrix is *symmetric*,  $\mathbf{A} = \mathbf{A}^\top$ .

Note that now the adjacency and feature matrices  $\mathbf{A}$  and  $\mathbf{X}$  are “synchronised”, in the sense that  $a_{uv}$  specifies the adjacency information between the nodes described by the  $u$ th and  $v$ th rows of  $\mathbf{X}$ . Therefore, applying a permutation matrix  $\mathbf{P}$  to the node features  $\mathbf{X}$  automatically implies applying it to  $\mathbf{A}$ ’s rows and columns,  $\mathbf{P}\mathbf{A}\mathbf{P}^\top$ . We say that (a graph-wise function)  $f$  is *permutation invariant* if

$$f(\mathbf{P}\mathbf{X}, \mathbf{P}\mathbf{A}\mathbf{P}^\top) = f(\mathbf{X}, \mathbf{A}) \quad (9)$$

As a way to emphasise the fact that our functions operating over graphs now need to take into account the adjacency information, we use the notation  $f(\mathbf{X}, \mathbf{A})$ .

and (a node-wise function)  $\mathbf{F}$  is *permutation equivariant* if

$$\mathbf{F}(\mathbf{P}\mathbf{X}, \mathbf{P}\mathbf{A}\mathbf{P}^\top) = \mathbf{P}\mathbf{F}(\mathbf{X}, \mathbf{A}) \quad (10)$$

for any permutation matrix  $\mathbf{P}$ .

Here again, we can first characterise linear equivariant functions. As observed by Maron et al. (2018), any linear  $\mathbf{F}$  satisfying equation (10) can be expressed as a linear combination of fifteen linear generators; remarkably, this family of generators is *independent of  $n$* . Amongst these generators, our blueprint specifically advocates for those that are also *local*, i.e., whereby the output on node  $u$  directly depends on its neighbouring nodes in the graph.

This corresponds to the *Bell number*  $B_4$ , which counts the number of ways to partition a set of 4 elements, in this case given by the 4-indices  $(u, v), (u', v')$  indexing a linear map acting on the adjacency matrix.

We can formalise this constraint explicitly in our model construction, by defining what it means for a node to be neighbouring another.

A (undirected) *neighbourhood* of node  $u$ , sometimes also called *1-hop*, is defined as

$$\mathcal{N}_u = \{v : (u, v) \in \mathcal{E} \text{ or } (v, u) \in \mathcal{E}\} \quad (11)$$

and the *neighbourhood features* as the multiset

$$\mathbf{X}_{\mathcal{N}_u} = \{\{\mathbf{x}_v : v \in \mathcal{N}_u\}\}. \quad (12)$$

A *multiset*, denoted  $\{\{\dots\}\}$ , is a set where the same element can appear more than once. This is the case here because the features of different nodes can be equal.

Operating on 1-hop neighbourhoods aligns well with the *locality* aspect of our blueprint: namely, defining our metric over graphs as the *shortest path distance* between nodes using edges in  $\mathcal{E}$ .

The GDL blueprint thus yields a general recipe for constructing permutation equivariant functions on graphs, by specifying a *local* function  $\phi$  that operates over the features of a node and its neighbourhood,  $\phi(\mathbf{x}_u, \mathbf{X}_{\mathcal{N}_u})$ . Then, a permutation equivariant function  $\mathbf{F}$  can be constructed by applying  $\phi$  to every node's neighbourhood in isolation (see Figure 10):

$$\mathbf{F}(\mathbf{X}, \mathbf{A}) = \begin{bmatrix} \text{---} & \phi(\mathbf{x}_1, \mathbf{X}_{\mathcal{N}_1}) & \text{---} \\ \text{---} & \phi(\mathbf{x}_2, \mathbf{X}_{\mathcal{N}_2}) & \text{---} \\ & \vdots & \\ \text{---} & \phi(\mathbf{x}_n, \mathbf{X}_{\mathcal{N}_n}) & \text{---} \end{bmatrix} \quad (13)$$

As  $\mathbf{F}$  is constructed by applying a shared function  $\phi$  to each node locally, its permutation equivariance rests on  $\phi$ 's output being independent on the ordering of the nodes in  $\mathcal{N}_u$ . Thus, if  $\phi$  is built to be permutation invariant, then this property is satisfied. As we will see in future work, the choice of  $\phi$  plays a crucial role in the expressive power of such a scheme. When  $\phi$  is injective, it is equivalent to one step of the *Weisfeiler-Lehman graph isomorphism test*, a classical algorithm in graph theory providing a necessary condition for two graphs to be isomorphic by an iterative color refinement procedure.

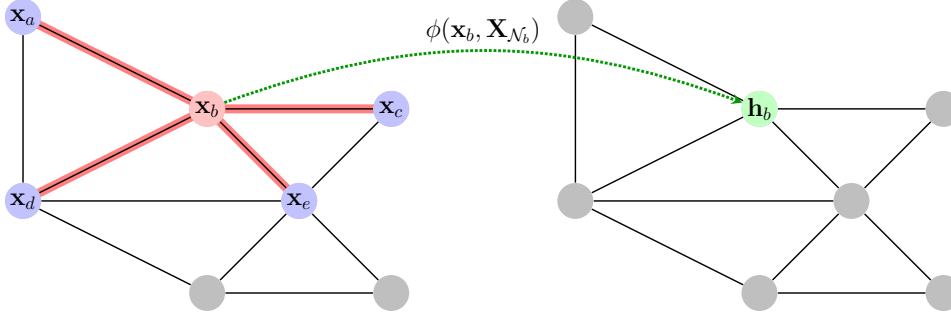


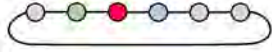
图 10: An illustration of constructing permutation-equivariant functions over graphs, by applying a permutation-invariant function  $\phi$  to every neighbourhood. In this case,  $\phi$  is applied to the features  $\mathbf{x}_b$  of node  $b$  as well as the multiset of its neighbourhood features,  $\mathbf{X}_{\mathcal{N}_b} = \{\{\mathbf{x}_a, \mathbf{x}_b, \mathbf{x}_c, \mathbf{x}_d, \mathbf{x}_e\}\}$ . Applying  $\phi$  in this manner to every node's neighbourhood recovers the rows of the resulting matrix of latent features  $\mathbf{H} = \mathbf{F}(\mathbf{X}, \mathbf{A})$ .

It is also worth noticing that the difference between functions defined on sets and more general graphs in this example is that in the latter case we need to explicitly account for the structure of the domain. As a consequence, graphs stand apart in the sense that the domain becomes *part of the input* in machine learning problems, whereas when dealing with sets and grids (both particular cases of graphs) we can specify only the features and assume the domain to be *fixed*. This distinction will be a recurring motif in our discussion. As a result, the notion of geometric stability (invariance to domain deformation) is crucial in most problems of learning on graphs. It straightforwardly follows from our construction that permutation invariant and equivariant functions produce identical outputs on isomorphic (topologically-equivalent) graphs. These results can be generalised to approximately isomorphic graphs, and several results on stability under graph perturbations exist (Levie et al., 2018). We will return to this important point in our discussion on manifolds, which we will use as an vehicle to study such invariance in further detail.

More precisely, we cannot define a non-trivial coarsening assuming set structure alone. There exist established approaches that infer topological structure from unordered sets, and those can admit non-trivial coarsening.

## 4.2 格网与欧几里得空间

The second type of objects we consider are grids. It is fair to say that the impact of deep learning was particularly dramatic in computer vision, natural language processing, and speech recognition. These applications all share a geometric common denominator: an underlying grid structure. As already mentioned, grids are a particular case of graphs with special adjacency. However, since the order of nodes in a grid is fixed, machine learning models for signals defined on grids are no longer required to account for permutation invariance, and have a stronger geometric prior: translation invariance.



**循环矩阵与卷积** Let us dwell on this point in more detail. Assuming for simplicity periodic boundary conditions, we can think of a one-dimensional grid as a *ring graph* with nodes indexed by  $0, 1, \dots, n-1$  modulo  $n$  (which we will omit for notation brevity) and the adjacency matrix with elements  $a_{u, u+1 \bmod n} = 1$  and zero otherwise. There are two main differences from the general graph case we have discussed before. First, each node  $u$  has identical connectivity, to its neighbours  $u-1$  and  $u+1$ , and thus structure-wise indistinguishable from the others. Second and more importantly, since the nodes of the grid have a fixed ordering, we also have a fixed ordering of the *neighbours*: we can call  $u-1$  the ‘left neighbour’ and  $u+1$  the ‘right neighbour’. If we use our previous recipe for designing an equivariant function  $\mathbf{F}$  using a local aggregation function  $\phi$ , we now have  $\mathbf{f}(\mathbf{x}_u) = \phi(\mathbf{x}_{u-1}, \mathbf{x}_u, \mathbf{x}_{u+1})$  at every node of the grid:  $\phi$  does not need to be permutation invariant anymore. For a particular choice of a linear transformation  $\phi(\mathbf{x}_{u-1}, \mathbf{x}_u, \mathbf{x}_{u+1}) =$

As we will see later, this makes the grid a homogeneous space.



$\theta_{-1}\mathbf{x}_{u-1} + \theta_0\mathbf{x}_u + \theta_1\mathbf{x}_{u+1}$ , we can write  $\mathbf{F}(\mathbf{X})$  as a matrix product,

$$\mathbf{F}(\mathbf{X}) = \begin{bmatrix} \theta_0 & \theta_1 & & & \theta_{-1} \\ \theta_{-1} & \theta_0 & \theta_1 & & \\ & \ddots & \ddots & \ddots & \\ & & \theta_{-1} & \theta_0 & \theta_1 \\ \theta_1 & & & \theta_{-1} & \theta_0 \end{bmatrix} \begin{bmatrix} \text{---} & \mathbf{x}_0 & \text{---} \\ \text{---} & \mathbf{x}_1 & \text{---} \\ & \vdots & \\ \text{---} & \mathbf{x}_{n-2} & \text{---} \\ \text{---} & \mathbf{x}_{n-1} & \text{---} \end{bmatrix}$$

Note this very special multi-diagonal structure with one element repeated along each diagonal, sometimes referred to as “weight sharing” in the machine learning literature.

More generally, given a vector  $\boldsymbol{\theta} = (\theta_0, \dots, \theta_{n-1})$ , a *circulant matrix*  $\mathbf{C}(\boldsymbol{\theta}) = (\theta_{u-v \bmod n})$  is obtained by appending circularly shifted versions of the vector  $\boldsymbol{\theta}$ . Circulant matrices are synonymous with discrete convolutions,

$$(\mathbf{x} \star \boldsymbol{\theta})_u = \sum_{v=0}^{n-1} x_{v \bmod n} \theta_{u-v \bmod n}$$

as one has  $\mathbf{C}(\boldsymbol{\theta})\mathbf{x} = \mathbf{x} \star \boldsymbol{\theta}$ . A particular choice of  $\boldsymbol{\theta} = (0, 1, 0, \dots, 0)^\top$  yields a special circulant matrix that shifts vectors to the right by one position. This matrix is called the (right) *shift* or *translation operator* and denoted by  $\mathbf{S}$ .

Circulant matrices can be characterised by their *commutativity* property: the product of circulant matrices is commutative, i.e.  $\mathbf{C}(\boldsymbol{\theta})\mathbf{C}(\boldsymbol{\eta}) = \mathbf{C}(\boldsymbol{\eta})\mathbf{C}(\boldsymbol{\theta})$  for any  $\boldsymbol{\theta}$  and  $\boldsymbol{\eta}$ . Since the shift is a circulant matrix, we get the familiar *translation* or *shift equivariance* of the convolution operator,

$$\mathbf{S}\mathbf{C}(\boldsymbol{\theta})\mathbf{x} = \mathbf{C}(\boldsymbol{\theta})\mathbf{S}\mathbf{x}.$$

Such commutativity property should not be surprising, since the underlying symmetry group (the translation group) is Abelian. Moreover, the opposite direction appears to be true as well, i.e. a matrix is circulant iff it commutes with shift. This, in turn, allows us to *define* convolution as a translation

Because of the periodic boundary conditions, it is a *circular* or *cyclic convolution*. In signal processing,  $\boldsymbol{\theta}$  is often referred to as the “filter,” and in CNNs, its coefficients are learnable.

The left shift operator is given by  $\mathbf{S}^\top$ . Obviously, shifting left and then right (or vice versa) does not do anything, which means  $\mathbf{S}$  is *orthogonal*:  $\mathbf{S}^\top \mathbf{S} = \mathbf{S}\mathbf{S}^\top = \mathbf{I}$ .

equivariant linear operation, and is a nice illustration of the power of geometric priors and the overall philosophy of Geometric ML: convolution emerges from the first principle of translational symmetry.

Note that unlike the situation on sets and graphs, the number of linearly independent shift-equivariant functions (convolutions) *grows* with the size of the domain (since we have one degree of freedom in each diagonal of a circulant matrix). However, the scale separation prior guarantees filters can be *local*, resulting in the same  $\Theta(1)$ -parameter complexity per layer, as we will verify in Section 5.1 when discussing the use of these principles in the implementation of Convolutional Neural Network architectures.

**Derivation of the discrete Fourier transform** We have already mentioned the Fourier transform and its connection to convolution: the fact that the Fourier transform diagonalises the convolution operation is an important property used in signal processing to perform convolution in the frequency domain as an element-wise product of the Fourier transforms. However, textbooks usually only state this fact, rarely explaining *where* the Fourier transform comes from and what is so *special* about the Fourier basis. Here we can show it, demonstrating once more how foundational are the basic principles of symmetry.

We must additionally assume distinct eigenvalues, otherwise there might be multiple possible diagonalisations. This assumption is satisfied with our choice of  $\mathbf{S}$ .

For this purpose, recall a fact from linear algebra that (diagonalisable) matrices are *jointly diagonalisable* iff they mutually commute. In other words, there exists a common eigenbasis for all the circulant matrices, in which they differ only by their eigenvalues. We can therefore pick one circulant matrix and compute its eigenvectors—we are assured that these will be the eigenvectors of all other circulant matrices as well. It is convenient to pick the shift operator, for which the eigenvectors happen to be the discrete Fourier

basis

$$\varphi_k = \frac{1}{\sqrt{n}} \left( 1, e^{\frac{2\pi i k}{n}}, e^{\frac{4\pi i k}{n}}, \dots, e^{\frac{2\pi i (n-1)k}{n}} \right)^\top, \quad k = 0, 1, \dots, n-1,$$

which we can arrange into an  $n \times n$  Fourier matrix  $\Phi = (\varphi_0, \dots, \varphi_{n-1})$ .

Multiplication by  $\Phi^*$  gives the Discrete Fourier Transform (DFT), and by  $\Phi$  the inverse DFT,

$$\hat{x}_k = \frac{1}{\sqrt{n}} \sum_{u=0}^{n-1} x_u e^{-\frac{2\pi i k u}{n}} \quad x_u = \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} \hat{x}_k e^{+\frac{2\pi i k u}{n}}.$$

Since all circulant matrices are jointly diagonalisable, they are also diagonalised by the Fourier transform and differ only in their eigenvalues. Since the eigenvalues of the circulant matrix  $\mathbf{C}(\theta)$  are the Fourier transform of the filter (see e.g. [Bamieh \(2018\)](#)),  $\hat{\theta} = \Phi^* \theta$ , we obtain the Convolution Theorem:

$$\mathbf{C}(\theta) \mathbf{x} = \Phi \begin{bmatrix} \hat{\theta}_0 & & \\ & \ddots & \\ & & \hat{\theta}_{n-1} \end{bmatrix} \Phi^* \mathbf{x} = \Phi(\hat{\theta} \odot \hat{\mathbf{x}})$$

Because the Fourier matrix  $\Phi$  has a special algebraic structure, the products  $\Phi^* \mathbf{x}$  and  $\Phi \mathbf{x}$  can be computed with  $\mathcal{O}(n \log n)$  complexity using a Fast Fourier Transform (FFT) algorithm. This is one of the reasons why frequency-domain filtering is so popular in signal processing; furthermore, the filter is typically designed directly in the frequency domain, so the Fourier transform  $\hat{\theta}$  is never explicitly computed.

Besides the didactic value of the derivation of the Fourier transform and convolution we have done here, it provides a scheme to generalise these concepts to graphs. Realising that the adjacency matrix of the ring graph is exactly the shift operator, one can develop the graph Fourier transform and an analogy of the convolution operator by computing the eigenvectors of the adjacency matrix (see e.g. [Sandryhaila and Moura \(2013\)](#)). Early attempts

$\mathbf{S}$  is orthogonal but non-symmetric, hence, its eigenvectors are orthogonal but the eigenvalues are complex (roots of unity).

Note that the eigenvectors are complex, so we need to take complex conjugation when transposing  $\Phi$ .

Since the Fourier transform is an orthogonal matrix ( $\Phi^* \Phi = \mathbf{I}$ ), geometrically it acts as a change of the system of coordinates that amounts to an  $n$ -dimensional rotation. In this system of coordinates (“Fourier domain”), the action of a circulant  $\mathbf{C}$  matrix becomes element-wise product.

to develop graph neural networks by analogy to CNNs, sometimes termed ‘spectral GNNs’, exploited this exact blueprint. We will see in Sections 4.4–4.6 that this analogy has some important limitations. The first limitation comes from the fact that a grid is fixed, and hence all signals on it can be represented in the same Fourier basis. In contrast, on general graphs, the Fourier basis depends on the structure of the graph. Hence, we cannot directly compare Fourier transforms on two different graphs — a problem that translated into a lack of generalisation in machine learning problems. Secondly, multi-dimensional grids, which are constructed as tensor products of one-dimensional grids, retain the underlying structure: the Fourier basis elements and the corresponding frequencies (eigenvalues) can be organised in multiple dimensions. In images, for example, we can naturally talk about horizontal and vertical frequency and filters have a notion of *direction*. On graphs, the structure of the Fourier domain is one-dimensional, as we can only organise the Fourier basis functions by the magnitude of the corresponding frequencies. As a result, graph filters are oblivious of direction or *isotropic*.

In graph signal processing, the eigenvectors of the *graph Laplacian* are often used as an alternative of the adjacency matrix to construct the graph Fourier transform, see Shuman et al. (2013). On grids, both matrices have joint eigenvectors, but on graphs they results in somewhat different though related constructions.

**Derivation of the continuous Fourier transform** For the sake of completeness, and as a segway for the next discussion, we repeat our analysis in the continuous setting. Like in Section 3.4, consider functions defined on  $\Omega = \mathbb{R}$  and the translation operator  $(S_v f)(u) = f(u - v)$  shifting  $f$  by some position  $v$ . Applying  $S_v$  to the Fourier basis functions  $\varphi_\xi(u) = e^{i\xi u}$  yields, by associativity of the exponent,

$$S_v e^{i\xi u} = e^{i\xi(u-v)} = e^{-i\xi v} e^{i\xi u},$$

i.e.,  $\varphi_{u\xi}(u)$  is the complex eigenvector of  $S_v$  with the complex eigenvalue  $e^{-i\xi v}$  — exactly mirroring the situation we had in the discrete setting. Since  $S_v$  is a unitary operator (i.e.,  $\|S_v x\|_p = \|x\|_p$  for any  $p$  and  $x \in L_p(\mathbb{R})$ ), any eigenvalue  $\lambda$  must satisfy  $|\lambda| = 1$ , which corresponds precisely to the

eigenvalues  $e^{-i\xi v}$  found above. Moreover, the spectrum of the translation operator is *simple*, meaning that two functions sharing the same eigenvalue must necessarily be collinear. Indeed, suppose that  $S_v f = e^{-i\xi_0 v} f$  for some  $\xi_0$ . Taking the Fourier transform in both sides, we obtain

$$\forall \xi, \quad e^{-i\xi v} \hat{f}(\xi) = e^{-i\xi_0 v} \hat{f}(\xi),$$

which implies that  $\hat{f}(\xi) = 0$  for  $\xi \neq \xi_0$ , thus  $f = \alpha \varphi_{\xi_0}$ .

For a general linear operator  $C$  that is translation equivariant ( $S_v C = C S_v$ ), we have

$$S_v C e^{i\xi u} = C S_v e^{i\xi u} = e^{-i\xi v} C e^{i\xi u},$$

implying that  $C e^{i\xi u}$  is also an eigenfunction of  $S_v$  with eigenvalue  $e^{-i\xi v}$ , from where it follows from the simplicity of spectrum that  $C e^{i\xi u} = \beta \varphi_\xi(u)$ ; in other words, the Fourier basis is the eigenbasis of all translation equivariant operators. As a result,  $C$  is *diagonal* in the Fourier domain and can be expressed as  $C e^{i\xi u} = \hat{p}_C(\xi) e^{i\xi u}$ , where  $\hat{p}_C(\xi)$  is a *transfer function* acting on different frequencies  $\xi$ . Finally, for an arbitrary function  $x(u)$ , by linearity,

$$\begin{aligned} (Cx)(u) &= C \int_{-\infty}^{+\infty} \hat{x}(\xi) e^{i\xi u} d\xi = \int_{-\infty}^{+\infty} \hat{x}(\xi) \hat{p}_C(\xi) e^{i\xi u} d\xi \\ &= \int_{-\infty}^{+\infty} p_C(v) x(u-v) dv = (x \star p_C)(u), \end{aligned}$$

where  $p_C(u)$  is the inverse Fourier transform of  $\hat{p}_C(\xi)$ . It thus follows that every linear translation equivariant operator is a convolution.

### 4.3 群与齐次空间

Our discussion of grids highlighted how shifts and convolutions are intimately connected: convolutions are linear shift-equivariant operations, and vice versa, any shift-equivariant linear operator is a convolution. Furthermore, shift operators can be jointly diagonalised by the Fourier transform.

*Eigenfunction* is synonymous with ‘eigenvector’ and is used when referring to eigenvectors of continuous operators.

The spectral characterisation of the translation group is a particular case of a more general result in Functional Analysis, the *Stone’s Theorem*, which derives an equivalent characterisation for any one-parameter unitary group.

Technically, we need the group to be *locally compact*, so that there exists a left-invariant Haar measure. Integrating with respect to this measure, we can “shift” the integrand by any group element and obtain the same result, just as how we have

$$\int_{-\infty}^{+\infty} x(u) du = \int_{-\infty}^{+\infty} x(u-v) du$$

for functions  $x : \mathbb{R} \rightarrow \mathbb{R}$

As it turns out, this is part of a far larger story: both convolution and the Fourier transform can be defined *for any group of symmetries* that we can sum or integrate over.

Consider the Euclidean domain  $\Omega = \mathbb{R}$ . We can understand the convolution as a pattern matching operation: we match shifted copies of a filter  $\theta(u)$  with an input signal  $x(u)$ . The value of the convolution  $(x \star \theta)(u)$  at a point  $u$  is the inner product of the signal  $x$  with the filter *shifted by*  $u$ ,

$$(x \star \theta)(u) = \langle x, S_u \theta \rangle = \int_{\mathbb{R}} x(v) \theta(u + v) dv.$$

Note that what we define here is not convolution but *cross-correlation*, which is tacitly used in deep learning under the name ‘convolution’. We do it for consistency with the following discussion, since in

our notation

$$\begin{aligned} (\rho(\mathfrak{g})x)(u) &= x(u - v) \text{ and} \\ (\rho(\mathfrak{g}^{-1})x)(u) &= x(u + v). \end{aligned}$$

Note that in this case  $u$  is both *a point on the domain*  $\Omega = \mathbb{R}$  and also *an element of the translation group*, which we can identify with the domain itself,  $\mathfrak{G} = \mathbb{R}$ . We will now show how to generalise this construction, by simply replacing the translation group by another group  $\mathfrak{G}$  acting on  $\Omega$ .

**Group convolution** As discussed in Section 3, the action of the group  $\mathfrak{G}$  on the domain  $\Omega$  induces a representation  $\rho$  of  $\mathfrak{G}$  on the space of signals  $\mathcal{X}(\Omega)$  via  $\rho(\mathfrak{g})x(u) = x(\mathfrak{g}^{-1}u)$ . In the above example,  $\mathfrak{G}$  is the translation group whose elements act by shifting the coordinates,  $u + v$ , whereas  $\rho(\mathfrak{g})$  is the shift operator acting on signals as  $(S_v x)(u) = x(u - v)$ . Finally, in order to apply a filter to the signal, we invoke our assumption of  $\mathcal{X}(\Omega)$  being a Hilbert space, with an inner product

$$\langle x, \theta \rangle = \int_{\Omega} x(u) \theta(u) du,$$

The integration is done w.r.t. an invariant measure  $\mu$  on  $\Omega$ . In case  $\mu$  is discrete, this means summing over  $\Omega$ .

where we assumed, for the sake of simplicity, scalar-valued signals,  $\mathcal{X}(\Omega, \mathbb{R})$ ; in general the inner product has the form of equation (2).

Having thus defined how to transform signals and match them with filters, we can define the *group convolution* for signals on  $\Omega$ ,

$$(x \star \theta)(\mathfrak{g}) = \langle x, \rho(\mathfrak{g}) \theta \rangle = \int_{\Omega} x(u) \theta(\mathfrak{g}^{-1}u) du. \quad (14)$$

Note that  $x \star \theta$  takes values on the *elements*  $\mathfrak{g}$  of our group  $\mathfrak{G}$  rather than points on the domain  $\Omega$ . Hence, the next layer, which takes  $x \star \theta$  as input, should act on signals defined *on to the group*  $\mathfrak{G}$ , a point we will return to shortly.

Just like how the traditional Euclidean convolution is shift-equivariant, the more general group convolution is  $\mathfrak{G}$ -equivariant. The key observation is that matching the signal  $x$  with a  $\mathfrak{g}$ -transformed filter  $\rho(\mathfrak{g})\theta$  is the same as matching the inverse transformed signal  $\rho(\mathfrak{g}^{-1})x$  with the untransformed filter  $\theta$ . Mathematically, this can be expressed as  $\langle x, \rho(\mathfrak{g})\theta \rangle = \langle \rho(\mathfrak{g}^{-1})x, \theta \rangle$ . With this insight,  $\mathfrak{G}$ -equivariance of the group convolution (14) follows immediately from its definition and the defining property  $\rho(\mathfrak{h}^{-1})\rho(\mathfrak{g}) = \rho(\mathfrak{h}^{-1}\mathfrak{g})$  of group representations,

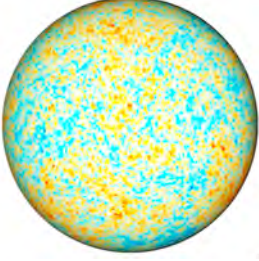
$$(\rho(\mathfrak{h})x \star \theta)(\mathfrak{g}) = \langle \rho(\mathfrak{h})x, \rho(\mathfrak{g})\theta \rangle = \langle x, \rho(\mathfrak{h}^{-1}\mathfrak{g})\theta \rangle = \rho(\mathfrak{h})(x \star \theta)(\mathfrak{g}).$$

Let us look at some examples. The case of one-dimensional grid we have studied above is obtained with the choice  $\Omega = \mathbb{Z}_n = \{0, \dots, n-1\}$  and the cyclic shift group  $\mathfrak{G} = \mathbb{Z}_n$ . The group elements in this case are cyclic shifts of indices, i.e., an element  $\mathfrak{g} \in \mathfrak{G}$  can be identified with some  $u = 0, \dots, n-1$  such that  $\mathfrak{g}.v = v - u \bmod n$ , whereas the inverse element is  $\mathfrak{g}^{-1}.v = v + u \bmod n$ . Importantly, in this example the elements of the *group* (shifts) are also elements of the *domain* (indices). We thus can, with some abuse of notation, identify the two structures (i.e.,  $\Omega = \mathfrak{G}$ ); our expression for the group convolution in this case

$$(x \star \theta)(\mathfrak{g}) = \sum_{v=0}^{n-1} x_v \theta_{\mathfrak{g}^{-1}v},$$

leads to the familiar convolution  $(x \star \theta)_u = \sum_{v=0}^{n-1} x_v \theta_{v+u \bmod n}$ .

Actually here again, this is cross-correlation.

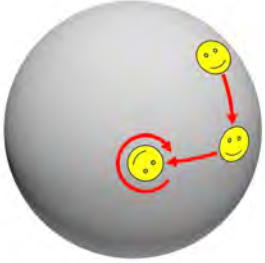


Cosmic microwave background radiation, captured by the Planck space observatory, is a signal on  $\mathbb{S}^2$ .

**Spherical convolution** Now consider the two-dimensional sphere  $\Omega = \mathbb{S}^2$  with the group of rotations, the *special orthogonal group*  $\mathfrak{G} = \text{SO}(3)$ . While chosen for pedagogical reason, this example is actually very practical and arises in numerous applications. In astrophysics, for example, observational data often naturally has spherical geometry. Furthermore, spherical symmetries are very important in applications in chemistry when modeling molecules and trying to predict their properties, e.g. for the purpose of virtual drug screening.

Representing a point on the sphere as a three-dimensional unit vector  $\mathbf{u}$  :  $\|\mathbf{u}\| = 1$ , the action of the group can be represented as a  $3 \times 3$  orthogonal matrix  $\mathbf{R}$  with  $\det(\mathbf{R}) = 1$ . The spherical convolution can thus be written as the inner product between the signal and the rotated filter,

$$(x \star \theta)(\mathbf{R}) = \int_{\mathbb{S}^2} x(\mathbf{u})\theta(\mathbf{R}^{-1}\mathbf{u})d\mathbf{u}.$$



The action of  $\text{SO}(3)$  group on  $\mathbb{S}^2$ . Note that three types of rotation are possible; the  $\text{SO}(3)$  is a three-dimensional manifold.

The first thing to note is that now the group is not identical to the domain: the group  $\text{SO}(3)$  is a Lie group that is in fact a three-dimensional manifold, whereas  $\mathbb{S}^2$  is a two-dimensional one. Consequently, in this case, unlike the previous example, the convolution is a function *on*  $\text{SO}(3)$  *rather than on*  $\Omega$ .

This has important practical consequences: in our Geometric Deep Learning blueprint, we concatenate multiple equivariant maps (“layers” in deep learning jargon) by applying a subsequent operator to the output of the previous one. In the case of translations, we can apply multiple convolutions in sequence, since their outputs are all defined on the same domain  $\Omega$ . In the general setting, since  $x \star \theta$  is a function on  $\mathfrak{G}$  rather than on  $\Omega$ , we cannot use exactly the same operation subsequently—it means that the next operation has to deal with *signals on*  $\mathfrak{G}$ , i.e.  $x \in \mathcal{X}(\mathfrak{G})$ . Our definition of group convolution allows this case: we take as domain  $\Omega = \mathfrak{G}$  acted on by  $\mathfrak{G}$  itself via the group action  $(\mathbf{g}, \mathbf{h}) \mapsto \mathbf{gh}$  defined by the composition operation of  $\mathfrak{G}$ . This yields the representation  $\rho(\mathbf{g})$  acting on  $x \in \mathcal{X}(\mathfrak{G})$  by



$(\rho(\mathfrak{g})x)(\mathfrak{h}) = x(\mathfrak{g}^{-1}\mathfrak{h})$ . Just like before, the inner product is defined by integrating the point-wise product of the signal and the filter over the domain, which now equals  $\Omega = \mathfrak{G}$ . In our example of spherical convolution, a second layer of convolution would thus have the form

$$((x \star \theta) \star \phi)(\mathbf{R}) = \int_{\text{SO}(3)} (x \star \theta)(\mathbf{Q}) \phi(\mathbf{R}^{-1}\mathbf{Q}) d\mathbf{Q}.$$

The representation of  $\mathfrak{G}$  acting on functions defined on  $\mathfrak{G}$  itself is called the *regular representation* of  $\mathfrak{G}$ .

Since convolution involves inner product that in turn requires integrating over the domain  $\Omega$ , we can only use it on domains  $\Omega$  that are small (in the discrete case) or low-dimensional (in the continuous case). For instance, we can use convolutions on the plane  $\mathbb{R}^2$  (two dimensional) or special orthogonal group  $\text{SE}(3)$  (three dimensional), or on the finite set of nodes of a graph ( $n$ -dimensional), but we cannot in practice perform convolution on the group of permutations  $\Sigma_n$ , which has  $n!$  elements. Likewise, integrating over higher-dimensional groups like the affine group (containing translations, rotations, shearing and scaling, for a total of 6 dimensions) is not feasible in practice. Nevertheless, as we have seen in Section 5.3, we can still build equivariant convolutions for large groups  $\mathfrak{G}$  by working with signals defined on low-dimensional spaces  $\Omega$  on which  $\mathfrak{G}$  acts. Indeed, it is possible to show that any equivariant linear map  $f : \mathcal{X}(\Omega) \rightarrow \mathcal{X}(\Omega')$  between two domains  $\Omega, \Omega'$  can be written as a generalised convolution similar to the group convolution discussed here.

Second, we note that the Fourier transform we derived in the previous section from the shift-equivariance property of the convolution can also be extended to a more general case by projecting the signal onto the matrix elements of irreducible representations of the symmetry group. We will discuss this in future work. In the case of  $\text{SO}(3)$  studied here, this gives rise to the *spherical harmonics* and *Wigner D-functions*, which find wide applications in quantum mechanics and chemistry.

Finally, we point to the assumption that has so far underpinned our discus-

sion in this section: whether  $\Omega$  was a grid, plane, or the sphere, we could transform every point into any other point, intuitively meaning that all the points on the domain “look the same.” A domain  $\Omega$  with such property is called a *homogeneous space*, where for any  $u, v \in \Omega$  there exists  $\mathbf{g} \in \mathfrak{G}$  such

The additional properties, that  $\mathbf{g}.u = v$ . In the next section we will try to relax this assumption.  
 $\mathbf{e}.u = u$  and  $\mathbf{g}(\mathbf{h}.u) = (\mathbf{gh}).u$   
 are tacitly assumed here.

#### 4.4 测地线与流形

As well as the group of rotations  $\text{SO}(3)$ , by virtue of it being a *Lie group*. In our last example, the sphere  $\mathbb{S}^2$  was a *manifold*, albeit a special one with a global symmetry group due to its homogeneous structure. Unfortunately, this is not the case for the majority of manifolds, which typically do not have global symmetries. In this case, we cannot straightforwardly define an action of  $\mathfrak{G}$  on the space of signals on  $\Omega$  and use it to ‘slide’ filters around in order to define a convolution as a direct generalisation of the classical construction. Nevertheless, manifolds do have two types of invariance that we will explore in this section: transformations preserving metric structure and local reference frame change.

While for many machine learning readers manifolds might appear as somewhat exotic objects, they are in fact very common in various scientific domains. In physics, manifolds play a central role as the model of our Universe — according to Einstein’s General Relativity Theory, gravity arises from the curvature of the space-time, modeled as a pseudo-Riemannian manifold. In more ‘prosaic’ fields such as computer graphics and vision, manifolds are a common mathematical model of 3D shapes. The broad spectrum of applications of such models ranges from virtual and augmented reality and special effects obtained by means of ‘motion capture’ to structural biology dealing with protein interactions that stick together (‘bind’ in chemical jargon) like pieces of 3D puzzle. The common denominator of these applications is the use of a manifold to represent the boundary surface of some 3D object.

The term ‘3D’ is somewhat misleading and refers to the *embedding space*. The shapes themselves are 2D manifolds (surfaces).

There are several reasons why such models are convenient. First, they offer a compact description of the 3D object, eliminating the need to allocate memory to ‘empty space’ as is required in grid-based representations. Second, they allow to ignore the internal structure of the object. This is a handy property for example in structural biology where the internal folding of a protein molecule is often irrelevant for interactions that happen on the molecular surface. Third and most importantly, one often needs to deal with *deformable objects* that undergo non-rigid deformations. Our own body is one such example, and many applications in computer graphics and vision, such as the aforementioned motion capture and virtual avatars, require *deformation invariance*. Such deformations can be modelled very well as transformations that preserve the intrinsic structure of a (Riemannian) manifold, namely the distances between points measured *along* the manifold, without regard to the way the manifold is embedded in the ambient space.



The human body is an example of a non-rigid object deforming in a nearly-isometric way.

We should emphasise that manifolds fall under the setting of *varying domains* in our Geometric Deep Learning blueprint, and in this sense are similar to graphs. We will highlight the importance of the notion of invariance to domain deformations – what we called ‘geometric stability’ in Section 3.3. Since differential geometry is perhaps less familiar to the machine learning audience, we will introduce the basic concepts required for our discussion and refer the reader to Penrose (2005) for their detailed exposition.

**Riemannian manifolds** Since the formal definition of a manifold is somewhat involved, we prefer to provide an intuitive picture at the expense of some precision. In this context, we can think of a (differentiable or smooth) manifold as a smooth multidimensional curved surface that is *locally Euclidean*, in the sense that any small neighbourhood around any point it can be deformed to a neighbourhood of  $\mathbb{R}^s$ ; in this case the manifold is said

By ‘smooth’ we mean differentiable sufficient number of times, which is tacitly assumed for convenience. ‘Deformed’ here means *diffeomorphic*, i.e., we can map between the two neighbourhoods using a smooth and invertible map with smooth inverse.

to be *s-dimensional*. This allows us to locally approximate the manifold around point  $u$  through the *tangent space*  $T_u\Omega$ . The latter can be visualised by thinking of a prototypical two-dimensional manifold, the sphere, and attaching a plane to it at a point: with sufficient zoom, the spherical surface will seem planar (Figure 11). The collection of all tangent spaces is called the *tangent bundle*, denoted  $T\Omega$ ; we will dwell on the concept of bundles in more detail in Section 4.5.

Formally, the tangent bundle is the *disjoint union*

$$T\Omega = \bigsqcup_{u \in \Omega} T_u\Omega.$$

A *tangent vector*, which we denote by  $X \in T_u\Omega$ , can be thought of as a local displacement from point  $u$ . In order to measure the *lengths* of tangent vectors and *angles* between them, we need to equip the tangent space with additional structure, expressed as a positive-definite bilinear function  $g_u : T_u\Omega \times T_u\Omega \rightarrow \mathbb{R}$  depending smoothly on  $u$ . Such a function is called a *Riemannian metric*, in honour of Bernhardt Riemann who introduced the concept in 1856, and can be thought of as an inner product on the tangent space,  $\langle X, Y \rangle_u = g_u(X, Y)$ , which is an expression of the angle between any two tangent vectors  $X, Y \in T_u\Omega$ . The metric also induces a norm  $\|X\|_u = g_u^{1/2}(X, X)$  allowing to locally measure lengths of vectors.

A bilinear function  $g$  is said to be *positive-definite* if  $g(X, X) > 0$  for any non-zero vector  $X \neq 0$ . If  $g$  is expressed as a matrix  $\mathbf{G}$ , it means  $\mathbf{G} \succ 0$ . The determinant  $|\mathbf{G}|^{1/2}$  provides a local volume element, which does not depend on the choice of the basis.

We must stress that tangent vectors are abstract geometric entities that exists in their own right and are *coordinate-free*. If we are to express a tangent vector  $X$  numerically as an array of numbers, we can only represent it as a list of coordinates  $\mathbf{x} = (x_1, \dots, x_s)$  *relative to some local basis*  $\{X_1, \dots, X_s\} \subseteq T_u\Omega$ . Similarly, the metric can be expressed as an  $s \times s$  matrix  $\mathbf{G}$  with elements  $g_{ij} = g_u(X_i, X_j)$  in that basis. We will return to this point in Section 4.5.

Unfortunately, too often vectors are identified with their coordinates. To emphasise this important difference, we use  $X$  to denote a tangent vector and  $\mathbf{x}$  to denote its coordinates.

A manifold equipped with a metric is called a *Riemannian manifold* and properties that can be expressed entirely in terms of the metric are said to be *intrinsic*. This is a crucial notion for our discussion, as according to our template, we will be seeking to construct functions acting on signals defined on  $\Omega$  that are invariant to metric-preserving transformations called *isome-*

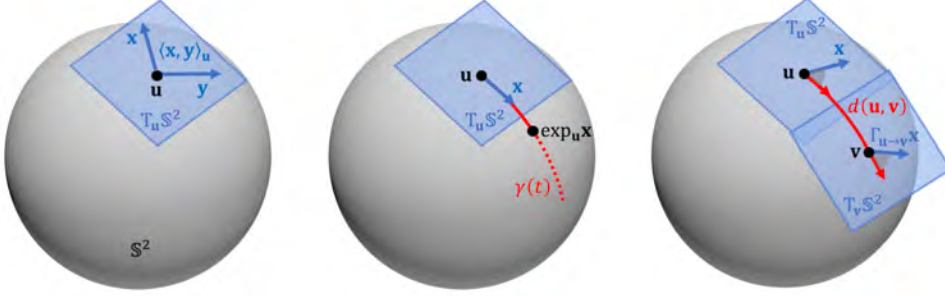


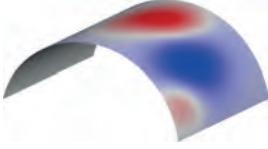
图 11: Basic notions of Riemannian geometry illustrated on the example of the two-dimensional sphere  $\mathbb{S}^2 = \{\mathbf{u} \in \mathbb{R}^3 : \|\mathbf{u}\| = 1\}$ , realised a subset (sub-manifold) of  $\mathbb{R}^3$ . The tangent space to the sphere is given as  $T_{\mathbf{u}}\mathbb{S}^2 = \{\mathbf{x} \in \mathbb{R}^3 : \mathbf{x}^\top \mathbf{u} = 0\}$  and is a 2D plane – hence this is a 2-dimensional manifold. The Riemannian metric is simply the Euclidean inner product restricted to the tangent plane,  $\langle \mathbf{x}, \mathbf{y} \rangle_{\mathbf{u}} = \mathbf{x}^\top \mathbf{y}$  for any  $\mathbf{x}, \mathbf{y} \in T_{\mathbf{u}}\mathbb{S}^2$ . The exponential map is given by  $\exp_{\mathbf{u}}(\mathbf{x}) = \cos(\|\mathbf{x}\|)\mathbf{u} + \frac{\sin(\|\mathbf{x}\|)}{\|\mathbf{x}\|}\mathbf{x}$ , for  $\mathbf{x} \in T_{\mathbf{u}}\mathbb{S}^2$ . Geodesics are great arcs of length  $d(\mathbf{u}, \mathbf{v}) = \cos^{-1}(\mathbf{u}^\top \mathbf{v})$ .

tries that deform the manifold without affecting its local structure. If such functions can be expressed in terms of intrinsic quantities, they are automatically guaranteed to be isometry-invariant and thus unaffected by isometric deformations. These results can be further extended to dealing with approximate isometries; this is thus an instance of the geometric stability (domain deformation) discussed in our blueprint.

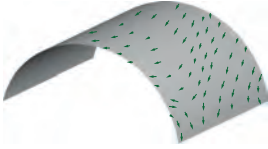
While, as we noted, the definition of a Riemannian manifold does not require a geometric realisation in any space, it turns out that any smooth Riemannian manifold can be realised as a subset of a Euclidean space of sufficiently high dimension (in which case it is said to be ‘embedded’ in that space) by using the structure of the Euclidean space to induce a Riemannian metric. Such an embedding is however not necessarily unique – as we will see, two different isometric realisations of a Riemannian metric are possible.



This result is known as the *Embedding Theorem*, due to Nash (1956). The art of origami is a manifestation of different isometric embeddings of the planar surface in  $\mathbb{R}^3$  (Figure: Shutterstock/300 librarians).



Example of a scalar field.



Example of a vector field.

The fields are typically assumed to be of the same regularity class (smoothness) as the manifold itself.

**Scalar and Vector fields** Since we are interested in signals defined on  $\Omega$ , we need to provide the proper notion of scalar- and vector-valued functions on manifolds. A (smooth) *scalar field* is a function of the form  $x : \Omega \rightarrow \mathbb{R}$ . Scalar fields form a vector space  $\mathcal{X}(\Omega, \mathbb{R})$  that can be equipped with the inner product

$$\langle x, y \rangle = \int_{\Omega} x(u)y(u)du, \quad (15)$$

where  $du$  is the volume element induced by the Riemannian metric. A (smooth) *tangent vector field* is a function of the form  $X : \Omega \rightarrow T\Omega$  assigning to each point a tangent vector in the respective tangent space,  $u \mapsto X(u) \in T_u\Omega$ . Vector fields also form a vector space  $\mathcal{X}(\Omega, T\Omega)$  with the inner product defined through the Riemannian metric,

$$\langle X, Y \rangle = \int_{\Omega} g_u(X(u), Y(u))du. \quad (16)$$

**Intrinsic gradient** Another way to think of (and actually *define*) vector fields is as a generalised notion of derivative. In classical calculus, one can locally linearise a (smooth) function through the *differential*  $dx(u) = x(u + du) - x(u)$ , which provides the change of the value of the function  $x$  at point  $u$  as a result of an infinitesimal displacement  $du$ . However, in our case the naïve use of this definition is impossible, since expressions of the form “ $u + du$ ” are meaningless on manifolds due to the lack of a global vector space structure.

The solution is to use tangent vectors as a model of local infinitesimal displacement. Given a smooth scalar field  $x \in \mathcal{X}(\Omega, \mathbb{R})$ , we can think of a (smooth) vector field as a linear map  $Y : \mathcal{X}(\Omega, \mathbb{R}) \rightarrow \mathcal{X}(\Omega, \mathbb{R})$  satisfying the properties of a *derivation*:  $Y(c) = 0$  for any constant  $c$  (corresponding to the intuition that constant functions have vanishing derivatives),  $Y(x + z) = Y(x) + Y(z)$  (linearity), and  $Y(xz) = Y(x)z + xY(z)$  (*product* or *Leibniz rule*), for any smooth scalar fields  $x, z \in \mathcal{X}(\Omega, \mathbb{R})$ . It can be shown that one can use these properties to define vector fields axiomatically.

The differential  $dx(Y) = Y(x)$  can be viewed as an operator  $(u, Y) \mapsto Y(x)$  and interpreted as follows: the change of  $x$  as the result of displacement  $Y \in T_u\Omega$  at point  $u$  is given by  $d_u x(Y)$ . It is thus an extension of the classical notion of *directional derivative*.

Importantly, this construction *does not use the Riemannian metric whatsoever* and can thus be extended to a more general construction of bundles discussed in the Section 4.5.

Alternatively, at each point  $u$  the differential can be regarded as a *linear functional*  $dx_u : T_u\Omega \rightarrow \mathbb{R}$  acting on tangent vectors  $X \in T_u\Omega$ . Linear functionals on a vector space are called *dual vectors* or *covectors*; if in addition we are given an inner product (Riemannian metric), a dual vector can always be represented as

$$dx_u(X) = g_u(\nabla x(u), X).$$

This is a consequence of the Riesz-Fréchet

The representation of the differential at point  $u$  is a tangent vector  $\nabla x(u) \in T_u\Omega$  called the (intrinsic) *gradient* of  $x$ ; similarly to the gradient in classical calculus, it can be thought of as the direction of the steepest increase of  $x$ . The gradient considered as an *operator*  $\nabla : \mathcal{X}(\Omega, \mathbb{R}) \rightarrow \mathcal{X}(\Omega, T\Omega)$  assigns at each point  $x(u) \mapsto \nabla x(u) \in T_u\Omega$ ; thus, the gradient of a scalar field  $x$  is a vector field  $\nabla x$ .

Representation Theorem, by which every dual vector can be expressed as an inner product with a vector.

**Geodesics** Now consider a smooth curve  $\gamma : [0, T] \rightarrow \Omega$  on the manifold with endpoints  $u = \gamma(0)$  and  $v = \gamma(T)$ . The derivative of the curve at point  $t$  is a tangent vector  $\gamma'(t) \in T_{\gamma(t)}\Omega$  called the *velocity vector*. Among all the curves connecting points  $u$  and  $v$ , we are interested in those of *minimum length*, i.e., we are seeking  $\gamma$  minimising the length functional

It is tacitly assumed that curves are given in *arclength parametrisation*, such that  $\|\gamma'\| = 1$  (constant velocity).

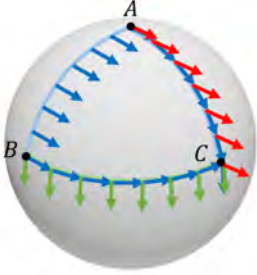
$$\ell(\gamma) = \int_0^T \|\gamma'(t)\|_{\gamma(t)} dt = \int_0^T g_{\gamma(t)}^{1/2}(\gamma'(t), \gamma'(t)) dt.$$

Such curves are called *geodesics* (from the Greek γεωδαισία, literally ‘division of Earth’) and they play important role in differential geometry. Crucially to our discussion, the way we defined geodesics is intrinsic, as they depend solely on the Riemannian metric (through the length functional).

Readers familiar with differential geometry might recall that geodesics are a more general concept and their definition in fact does not necessarily require a Riemannian metric but a *connection* (also called a *covariant derivative*, as it generalises the notion of derivative to vector and tensor fields), which is defined axiomatically, similarly to our construction of the differential. Given a Riemannian metric, there exists a unique special connection called the *Levi-Civita connection* which is often tacitly assumed in Riemannian geometry. Geodesics arising from this connection are the length-minimising curves we have defined above.

The Levi-Civita connection is torsion-free and compatible with the metric. The Fundamental Theorem of Riemannian geometry guarantees its existence and uniqueness.

We will show next how to use geodesics to define a way to transport tangent vectors on the manifold (parallel transport), create local intrinsic maps from the manifold to the tangent space (exponential map), and define distances (geodesic metric). This will allow us to construct convolution-like operations by applying a filter locally in the tangent space.



Euclidean transport of a vector from A to C makes no sense on the sphere, as the resulting vectors (red) are not in the tangent plane. Parallel transport from A to C (blue) rotates the vector along the path. It is path dependent: going along the path BC and ABC produces different results.

**Parallel transport** One issue we have already encountered when dealing with manifolds is that we cannot directly add or subtract two points  $u, v \in \Omega$ . The same problem arises when trying to compare tangent vectors at different points: though they have the same dimension, they belong to *different spaces*, e.g.  $X \in T_u\Omega$  and  $Y \in T_v\Omega$ , and thus not directly comparable. Geodesics provide a mechanism to move vectors from one point to another, in the following way: let  $\gamma$  be a geodesic connecting points  $u = \gamma(0)$  and  $v = \gamma(T)$  and let  $X \in T_u\Omega$ . We can define a new set of tangent vectors along the geodesic,  $X(t) \in T_{\gamma(t)}\Omega$  such that the length of  $X(t)$  and the angle (expressed through the Riemannian metric) between it and the velocity vector of the curve is constant,

$$g_{\gamma(t)}(X(t), \gamma'(t)) = g_u(X, \gamma'(0)) = \text{const}, \quad \|X(t)\|_{\gamma(t)} = \|X\|_u = \text{const}.$$

As a result, we get a unique vector  $X(T) \in T_v\Omega$  at the end point  $v$ .



The map  $\Gamma_{u \rightarrow v}(X) : T_u\Omega \rightarrow T_v\Omega$  and  $T_v\Omega$  defined as  $\Gamma_{u \rightarrow v}(X) = X(T)$  using the above notation is called *parallel transport* or *connection*; the latter term implying it is a mechanism to ‘connect’ between the tangent spaces  $T_u\Omega$  and  $T_v\Omega$ . Due to the angle and length preservation conditions, parallel transport amounts to only rotation of the vector, so it can be associated with an element of the special orthogonal group  $\text{SO}(s)$  (called the *structure group* of the tangent bundle), which we will denote by  $\mathfrak{g}_{u \rightarrow v}$  and discuss in further detail in Section 4.5.

Assuming that the manifold is orientable, otherwise  $\text{O}(s)$ .

As we mentioned before, a connection can be defined axiomatically independently of the Riemannian metric, providing thus an abstract notion of parallel transport along any smooth curve. The result of such transport, however, depends on the path taken.

**Exponential map** Locally around a point  $u$ , it is always possible to define a unique geodesic in a given direction  $X \in T_u\Omega$ , i.e. such that  $\gamma(0) = u$  and  $\gamma'(0) = X$ . When  $\gamma_X(t)$  is defined for all  $t \geq 0$  (that is, we can shoot the geodesic from a point  $u$  for as long as we like), the manifold is said to be *geodesically complete* and the exponential map is defined on the whole tangent space. Since compact manifolds are geodesically complete, we can tacitly assume this convenient property.

This definition of geodesic provided a point and a direction gives a natural mapping from (a subset of) the tangent space  $T_u\Omega$  to  $\Omega$  called the *exponential map*  $\exp : B_r(0) \subset T_u\Omega \rightarrow \Omega$ , which is defined by taking a unit step along the geodesic in the direction  $X$ , i.e.,  $\exp_u(X) = \gamma_X(1)$ . The exponential map  $\exp_u$  is a local diffeomorphism, as it deforms the neighbourhood  $B_r(0)$  (a ball or radius  $r$ ) of the origin on  $T_u\Omega$  into a neighbourhood of  $u$ . Conversely, one can also regard the exponential map as an intrinsic local deformation (‘flattening’) of the manifold into the tangent space.

Note that geodesic completeness does not necessarily guarantee that  $\exp$  is a global diffeomorphism – the largest radius  $r$  about  $u$  for which  $\exp_u(B_r(0) \subseteq T_u\Omega)$  is mapped diffeomorphically is called the *injectivity radius*.

Hopf-Rinow Theorem thus establishes the equivalence between geodesic and metric completeness, the latter meaning every Cauchy sequence converges in the geodesic distance metric.

**Geodesic distances** A result known as the Hopf-Rinow Theorem guarantees that geodesically complete manifolds are also *complete metric spaces*, in which one can realise a distance (called the *geodesic distance* or *metric*) between any pair of points  $u, v$  as the length of the shortest path between them

$$d_g(u, v) = \min_{\gamma} \ell(\gamma) \quad \text{s.t.} \quad \gamma(0) = u, \gamma(T) = v,$$

Note that the term ‘metric’ is used in two senses:

Riemannian metric  $g$  and distance  $d$ . To avoid confusion, we will use the term ‘distance’ referring to the latter. Our notation  $d_g$  makes the distance depend on the Riemannian metric  $g$ , though the definition of geodesic length  $L$ .

which exists (i.e., the minimum is attained).

**Isometries** Consider now a deformation of our manifold  $\Omega$  into another manifold  $\tilde{\Omega}$  with a Riemannian metric  $h$ , which we assume to be a diffeomorphism  $\eta : (\Omega, g) \rightarrow (\tilde{\Omega}, h)$  between the manifolds. Its differential  $d\eta : T\Omega \rightarrow T\tilde{\Omega}$  defines a map between the respective tangent bundles (referred to as *pushforward*), such that at a point  $u$ , we have  $d\eta_u : T_u\Omega \rightarrow T_{\eta(u)}\tilde{\Omega}$ , interpreted as before: if we make a small displacement from point  $u$  by tangent vector  $X \in T_u\Omega$ , the map  $\eta$  will be displaced from point  $\eta(u)$  by tangent vector  $d\eta_u(X) \in T_{\eta(u)}\tilde{\Omega}$ .

Pushforward and pullback are adjoint operators  $\langle \eta^* \alpha, X \rangle = \langle \alpha, \eta_* X \rangle$  where  $\alpha \in T^*\Omega$  is a *dual vector field*, defined at each point as a linear functional acting on  $T_u\Omega$  and the inner products are defined respectively on vector and dual vector fields.

Since the pushforward provides a mechanism to associate tangent vectors on the two manifolds, it allows to *pullback* the metric  $h$  from  $\tilde{\Omega}$  to  $\Omega$ ,

$$(\eta^* h)_u(X, Y) = h_{\eta(u)}(d\eta_u(X), d\eta_u(Y))$$

If the pullback metric coincides at every point with that of  $\Omega$ , i.e.,  $g = \eta^* h$ , the map  $\eta$  is called (a Riemannian) *isometry*. For two-dimensional manifolds (surfaces), isometries can be intuitively understood as inelastic deformations that deform the manifold without ‘stretching’ or ‘tearing’ it.

By virtue of their definition, isometries preserve intrinsic structures such as geodesic distances, which are expressed entirely in terms of the Riemannian metric. Therefore, we can also understand isometries from the position of metric geometry, as distance-preserving maps (‘metric isometries’) *between*

*metric spaces*  $\eta : (\Omega, d_g) \rightarrow (\tilde{\Omega}, d_h)$ , in the sense that

$$d_g(u, v) = d_h(\eta(u), \eta(v))$$

for all  $u, v \in \Omega$ , or more compactly,  $d_g = d_h \circ (\eta \times \eta)$ . In other words, Riemannian isometries are also metric isometries. On *connected* manifolds, the converse is also true: every metric isometry is also a Riemannian isometry.

This result is known as the Myers-Steenrod Theorem.

In our Geometric Deep Learning blueprint,  $\eta$  is a model of domain deformations. When  $\eta$  is an isometry, any intrinsic quantities are unaffected by such deformations. One can generalise exact (metric) isometries through the notions of *metric dilation*

We tacitly assume our manifolds to be connected.

$$\text{dil}(\eta) = \sup_{u \neq v \in \Omega} \frac{d_h(\eta(u), \eta(v))}{d_g(u, v)}$$

or *metric distortion*

$$\text{dis}(\eta) = \sup_{u, v \in \Omega} |d_h(\eta(u), \eta(v)) - d_g(u, v)|,$$

The Gromov-Hausdorff distance between metric spaces, which we mentioned in Section 3.2, can be expressed as the smallest possible metric distortion.

which capture the relative and absolute change of the geodesic distances under  $\eta$ , respectively. The condition (5) for the stability of a function  $f \in \mathcal{F}(\mathcal{X}(\Omega))$  under domain deformation can be rewritten in this case as

$$\|f(x, \Omega) - f(x \circ \eta^{-1}, \tilde{\Omega})\| \leq C\|x\|\text{dis}(\eta).$$

**Intrinsic symmetries** A particular case of the above is a diffeomorphism of the domain itself (what we termed *automorphism* in Section 3.2), which we will denote by  $\tau \in \text{Diff}(\Omega)$ . We will call it a Riemannian (self-)isometry if the pullback metric satisfies  $\tau^*g = g$ , or a metric (self-)isometry if  $d_g = d_g \circ (\tau \times \tau)$ . Not surprisingly, isometries form a group with the composition operator denoted by  $\text{Iso}(\Omega)$  and called the *isometry group*; the identity element is the map  $\tau(u) = u$  and the inverse always exists (by definition of  $\tau$  as a diffeomorphism). Self-isometries are thus *intrinsic symmetries* of manifolds.

Continuous symmetries on manifolds are infinitesimally generated by special tangent vector fields called *Killing fields*, named after Wilhelm Killing.

**Fourier analysis on Manifolds** We will now show how to construct intrinsic convolution-like operations on manifolds, which, by construction, will be invariant to isometric deformations. For this purpose, we have two options: One is to use an analogy of the Fourier transform, and define the convolution as a product in the Fourier domain. The other is to define the convolution spatially, by correlating a filter locally with the signal. Let us discuss the spectral approach first.

We remind that in the Euclidean domain the Fourier transform is obtained as the eigenvectors of circulant matrices, which are jointly diagonalisable due to their commutativity. Thus, any circulant matrix and in particular, differential operator, can be used to define an analogy of the Fourier transform on general domains. In Riemannian geometry, it is common to use the orthogonal eigenbasis of the Laplacian operator, which we will define here.

For this purpose, recall our definition of the intrinsic gradient operator  $\nabla : \mathcal{X}(\Omega, \mathbb{R}) \rightarrow \mathcal{X}(\Omega, T\Omega)$ , producing a tangent vector field that indicates the local direction of steepest increase of a scalar field on the manifold. In a similar manner, we can define the *divergence operator*  $\nabla^* : \mathcal{X}(\Omega, T\Omega) \rightarrow \mathcal{X}(\Omega, \mathbb{R})$ . If we think of a tangent vector field as a flow on the manifold, the divergence measures the net flow of a field at a point, allowing to distinguish between field ‘sources’ and ‘sinks’. We use the notation  $\nabla^*$  (as opposed to the common  $\text{div}$ ) to emphasise that the two operators are adjoint,

$$\langle X, \nabla x \rangle = \langle \nabla^* X, x \rangle,$$

where we use the inner products (15) and (16) between scalar and vector fields.

The *Laplacian* (also known as the *Laplace-Beltrami operator* in differential geometry) is an operator on  $\mathcal{X}(\Omega)$  defined as  $\Delta = \nabla^* \nabla$ , which can be interpreted as the difference between the average of a function on an infinitesimal sphere around a point and the value of the function at the point itself. It

From this interpretation it is also clear that the Laplacian is isotropic. We will see in Section 4.6 that it is possible to define *anisotropic Laplacians* (see (Andreux et al., 2014; Boscaini et al., 2016b)) of the form  $\nabla^*(A(u)\nabla)$ , where  $A(u)$  is a position-dependent tensor determining local direction.

is one of the most important operators in mathematical physics, used to describe phenomena as diverse as heat diffusion, quantum oscillations, and wave propagation. Importantly in our context, the Laplacian is intrinsic, and thus invariant under isometries of  $\Omega$ .

It is easy to see that the Laplacian is self-adjoint ('symmetric'),

$$\langle \nabla x, \nabla x \rangle = \langle x, \Delta x \rangle = \langle \Delta x, x \rangle.$$

The quadratic form on the left in the above expression is actually the already familiar Dirichlet energy,

$$c^2(x) = \|\nabla x\|^2 = \langle \nabla x, \nabla x \rangle = \int_{\Omega} \|\nabla x(u)\|_u^2 du = \int_{\Omega} g_u(\nabla x(u), \nabla x(u)) du$$

measuring the smoothness of  $x$ .

The Laplacian operator admits an eigedecomposition

$$\Delta \varphi_k = \lambda_k \varphi_k, \quad k = 0, 1, \dots$$

with countable spectrum if the manifold is compact (which we tacitly assume), and orthogonal eigenfunctions,  $\langle \varphi_k, \varphi_l \rangle = \delta_{kl}$ , due to the self-adjointness of  $\Delta$ . The Laplacian eigenbasis can also be constructed as a set of orthogonal minimisers of the Dirichlet energy,

$$\varphi_{k+1} = \arg \min_{\varphi} \|\nabla \varphi\|^2 \quad \text{s.t.} \quad \|\varphi\| = 1 \text{ and } \langle \varphi, \varphi_j \rangle = 0$$

for  $j = 0, \dots, k$ , allowing to interpret it as the smoothest orthogonal basis on  $\Omega$ . The eigenfunctions  $\varphi_0, \varphi_1, \dots$  and the corresponding eigenvalues  $0 = \lambda_0 \leq \lambda_1 \leq \dots$  can be interpreted as the analogy of the atoms and frequencies in the classical Fourier transform.

In fact  $e^{i\xi u}$  are the eigenfunctions of the Euclidean Laplacian  $\frac{d^2}{du^2}$ .

This orthogonal basis allows to expand square-integrable functions on  $\Omega$  into *Fourier series*

$$x(u) = \sum_{k \geq 0} \langle x, \varphi_k \rangle \varphi_k(u)$$

where  $\hat{x}_k = \langle x, \varphi_k \rangle$  are referred to as the *Fourier coefficient* or the (generalised) Fourier transform of  $x$ . Truncating the Fourier series results in an approximation error that can be bounded (Aflalo and Kimmel, 2013) by

Note that this Fourier transform has a discrete index, since the spectrum is discrete due to the compactness of  $\Omega$ .

$$\left\| x - \sum_{k=0}^N \langle x, \varphi_k \rangle \varphi_k \right\|^2 \leq \frac{\|\nabla x\|^2}{\lambda_{N+1}}.$$

Aflalo et al. (2015) further showed that no other basis attains a better error, making the Laplacian eigenbasis *optimal* for representing smooth signals on manifolds.

**Spectral Convolution on Manifolds** *Spectral convolution* can be defined as the product of Fourier transforms of the signal  $x$  and the filter  $\theta$ ,

$$(x \star \theta)(u) = \sum_{k \geq 0} (\hat{x}_k \cdot \hat{\theta}_k) \varphi_k(u). \quad (17)$$

Note that here we use what is a *property* of the classical Fourier transform (the Convolution Theorem) as a way to *define* a non-Euclidean convolution. By virtue of its construction, the spectral convolution is intrinsic and thus isometry-invariant. Furthermore, since the Laplacian operator is isotropic, it has no sense of direction; in this sense, the situation is similar to that we had on graphs in Section 4.1 due to permutation invariance of neighbour aggregation.

In practice, a direct computation of (17) appears to be prohibitively expensive due to the need to diagonalise the Laplacian. Even worse, it turns out geometrically unstable: the higher-frequency eigenfunctions of the Laplacian can change dramatically as a result of even small near-isometric perturbations of the domain  $\Omega$  (see Figure 12). A more stable solution is provided



图 12: Instability of spectral filters under domain perturbation. Left: a signal  $\mathbf{x}$  on the mesh  $\Omega$ . Middle: result of spectral filtering in the eigenbasis of the Laplacian  $\Delta$  on  $\Omega$ . Right: the same spectral filter applied to the eigenvectors of the Laplacian  $\tilde{\Delta}$  of a nearly-isometrically perturbed domain  $\tilde{\Omega}$  produces a very different result.

by realising the filter as a *spectral transfer function* of the form  $\hat{p}(\Delta)$ ,

$$(\hat{p}(\Delta)x)(u) = \sum_{k \geq 0} \hat{p}(\lambda_k) \langle x, \varphi_k \rangle \varphi_k(u) \quad (18)$$

$$= \int_{\Omega} x(v) \sum_{k \geq 0} \hat{p}(\lambda_k) \varphi_k(v) \varphi_k(u) \, dv \quad (19)$$

which can be interpreted in two manners: either as a spectral filter (18), where we identify  $\hat{\theta}_k = \hat{p}(\lambda_k)$ , or as a spatial filter (19) with a position-dependent kernel  $\theta(u, v) = \sum_{k \geq 0} \hat{p}(\lambda_k) \varphi_k(v) \varphi_k(u)$ . The advantage of this formulation is that  $\hat{p}(\lambda)$  can be parametrised by a small number of coefficients, and choosing parametric functions such as polynomials  $\hat{p}(\lambda) = \sum_{l=0}^r \alpha_l \lambda^l$  allows for efficiently computing the filter as

$$(\hat{p}(\Delta)x)(u) = \sum_{k \geq 0} \sum_{l=0}^r \alpha_l \lambda_k^l \langle x, \varphi_k \rangle \varphi_k(u) = \sum_{l=0}^r \alpha_l (\Delta^l x)(u),$$

avoiding the spectral decomposition altogether. We will discuss this construction in further detail in Section 4.6.

**Spatial Convolution on Manifolds** A second alternative is to attempt defining convolution on manifolds is by matching a filter at different points, like

Geometric Deep Learning methods based on spectral convolution expressed through the Fourier transform are often referred to as ‘spectral’ and opposed to ‘spatial’ methods we have seen before in the context of graphs. We see here that these two views may be equivalent, so this dichotomy is somewhat artificial and not completely appropriate.

we did in formula (14),

$$(x \star \theta)(u) = \int_{T_u \Omega} x(\exp_u Y) \theta_u(Y) dY, \quad (20)$$

where we now have to use the exponential map to access the values of the scalar field  $x$  from the tangent space, and the filter  $\theta_u$  is defined in the tangent space at each point and hence position-dependent. If one defines the filter intrinsically, such a convolution would be isometry-invariant, a property we mentioned as crucial in many computer vision and graphics applications.

We need, however, to note several substantial differences from our previous construction in Sections 4.2–4.3. First, because a manifold is generally not a homogeneous space, we do not have anymore a global group structure allowing us have a shared filter (i.e., the same  $\theta$  at every  $u$  rather than  $\theta_u$  in expression (20)) defined at one point and then move it around. An analogy of this operation on the manifold would require parallel transport, allowing to apply a shared  $\theta$ , defined as a function on  $T_u \Omega$ , at some other  $T_v \Omega$ . However, as we have seen, this in general will depend on the path between  $u$  and  $v$ , so *the way we move the filter around matters*. Third, since we can use the exponential map only locally, the filter must be *local*, with support bounded by the injectivity radius. Fourth and most crucially, we cannot work with  $\theta(X)$ , as  $X$  is an abstract geometric object: in order for it to be used for computations, we must represent it *relative to some local basis*  $\omega_u : \mathbb{R}^s \rightarrow T_u \Omega$ , as an  $s$ -dimensional array of coordinates  $\mathbf{x} = \omega_u^{-1}(X)$ . This allows us to rewrite the convolution (20) as

$$(x \star \theta)(u) = \int_{[0,1]^s} x(\exp_u(\omega_u \mathbf{y})) \theta(\mathbf{y}) d\mathbf{y}, \quad (21)$$

with the filter defined on the unit cube. Since the exponential map is intrinsic (through the definition of geodesic), the resulting convolution is isometry-invariant.



Yet, this tacitly assumed we can carry the frame  $\omega_u$  along to another manifold, i.e.  $\omega'_u = d\eta_u \circ \omega_u$ . Obtaining such a frame (or *gauge*, in physics terminology) given only a the manifold  $\Omega$  in a consistent manner is however fraught with difficulty. First, a smooth global gauge may not exist: this is the situation on manifolds that are not *parallelisable*, in which case one cannot define a smooth non-vanishing tangent vector field. Second, we do not have a canonical gauge on manifolds, so this choice is arbitrary; since our convolution depends on  $\omega$ , if one chose a different one, we would obtain different results.

We should note that this is a case where practice diverges from theory: in practice, it is possible to build frames that are mostly smooth, with a limited number of singularities, e.g. by taking the intrinsic gradient of some intrinsic scalar field on the manifold. Moreover, such constructions are stable, i.e., the frames constructed this way will be identical on isometric manifolds and similar on approximately isometric ones. Such approaches were in fact employed in the early works on deep learning on manifolds (Masci et al., 2015; Monti et al., 2017).

Nevertheless, this solution is not entirely satisfactory because near singularities, the filter orientation (being defined in a fixed manner relative to the gauge) will vary wildly, leading to a non-smooth feature map even if the input signal and filter are smooth. Moreover, there is no clear reason why a given direction at some point  $u$  should be considered equivalent to another direction at an altogether different point  $v$ . Thus, despite *practical* alternatives, we will look next for a more *theoretically* well-founded approach that would be altogether independent on the choice of gauge.

The sphere  $S^2$  is an example of a non-parallelisable manifold, a result of the *Poincaré-Hopf Theorem*, which is colloquially stated as ‘one cannot comb a hairy ball without creating a cowlick.’



Example of stable gauges constructed on nearly-isometric manifolds (only one axis is shown) using the GFrames algorithm of Melzi et al. (2019).

## 4.5 量规与丛

Historically, fibre bundles arose first in modern differential geometry of Élie Cartan (who however did not define them explicitly), and were then further developed as a standalone object in the field of topology in the 1930s.

The notion of gauge, which we have defined as a frame for the tangent space, is quite a bit more general in physics: it can refer to a frame for any *vector bundle*, not just the tangent bundle. Informally, a vector bundle describes a family of vector spaces parametrised by another space and consists of a *base space*  $\Omega$  with an identical vector space  $\mathbb{V}$  (called the *fibre*) attached to each position  $u \in \Omega$  (for the tangent bundle these are the tangent spaces  $T_u\Omega$ ). Roughly speaking, a bundle looks as a product  $\Omega \times \mathbb{V}$  locally around  $u$ , but globally might be ‘twisted’ and have an overall different structure. In Geometric Deep Learning, fibres can be used to model the feature spaces at each point in the manifold  $\Omega$ , with the dimension of the fibre being equal to the number of feature channels. In this context, a new and fascinating kind of symmetry, called *gauge symmetry* may present itself.

Let us consider again an  $s$ -dimensional manifold  $\Omega$  with its tangent bundle  $T\Omega$ , and a vector field  $X : \Omega \rightarrow T\Omega$  (which in this terminology is referred to as a *section* on the tangent bundle). Relative to a gauge  $\omega$  for the tangent bundle,  $X$  is represented as a function  $\mathbf{x} : \Omega \rightarrow \mathbb{R}^s$ . However it is important to realise that what we are really interested in is the underlying geometrical object (vector field), whose representation as a function  $\mathbf{x} \in \mathcal{X}(\Omega, \mathbb{R}^s)$  *depends on the choice of gauge*  $\omega$ . If we change the gauge, we also need to change  $\mathbf{x}$  so as to preserve the underlying vector field being represented.

**Tangent bundles and the Structure group** When we change the gauge, we need to apply at each point an invertible matrix that maps the old gauge to the new one. This matrix is unique for every pair of gauges at each point, but possibly different at different points. In other words, a *gauge transformation* is a mapping  $\mathfrak{g} : \Omega \rightarrow \text{GL}(s)$ , where  $\text{GL}(s)$  is the *general linear group* of invertible  $s \times s$  matrices. It acts on the gauge  $\omega_u : \mathbb{R}^s \rightarrow T_u\Omega$  to produce a new gauge  $\omega'_u = \omega_u \circ \mathfrak{g}_u : \mathbb{R}^s \rightarrow T_u\Omega$ . The gauge transformation acts on

a coordinate vector field at each point via  $\mathbf{x}'(u) = \mathbf{g}_u^{-1}\mathbf{x}(u)$  to produce the coordinate representation  $\mathbf{x}'$  of  $X$  relative to the new gauge. The underlying vector field remains unchanged:

$$X(u) = \omega'_u(\mathbf{x}'(u)) = \omega_u(\mathbf{g}_u\mathbf{g}_u^{-1}\mathbf{x}(u)) = \omega_u(\mathbf{x}(u)) = X(u),$$

which is exactly the property we desired. More generally, we may have a field of geometric quantities that transform according to a representation  $\rho$  of  $\mathrm{GL}(s)$ , e.g. a field of 2-tensors (matrices)  $\mathbf{A}(u) \in \mathbb{R}^{s \times s}$  that transform like  $\mathbf{A}'(u) = \rho_2(\mathbf{g}_u^{-1})\mathbf{A}(u) = \rho_1(\mathbf{g}_u)\mathbf{A}(u)\rho_1(\mathbf{g}_u^{-1})$ . In this case, the gauge transformation  $\mathbf{g}_u$  acts via  $\rho(\mathbf{g}_u)$ .

Sometimes we may wish to restrict attention to frames with a certain property, such as orthogonal frames, right-handed frames, etc. Unsurprisingly, we are interested in a set of some property-preserving transformations that form a group. For instance, the group that preserves orthogonality is the orthogonal group  $\mathrm{O}(s)$  (rotations and reflections), and the group that additionally preserves orientation or ‘handedness’ is  $\mathrm{SO}(s)$  (pure rotations). Thus, in general we have a group  $\mathfrak{G}$  called the *structure group* of the bundle, and a gauge transformation is a map  $\mathbf{g} : \Omega \rightarrow \mathfrak{G}$ . A key observation is that in all cases with the given property, for any two frames at a given point there exists exactly one gauge transformation relating them.

As mentioned before, gauge theory extends beyond tangent bundles, and in general, we can consider a bundle of vector spaces whose structure and dimensions are not necessarily related to those of the base space  $\Omega$ . For instance, a color image pixel has a position  $u \in \Omega = \mathbb{Z}^2$  on a 2D grid and a value  $\mathbf{x}(u) \in \mathbb{R}^3$  in the RGB space, so the space of pixels can be viewed as a vector bundle with base space  $\mathbb{Z}^2$  and a fibre  $\mathbb{R}^3$  attached at each point. It is customary to express an RGB image relative to a gauge that has basis vectors for R, G, and B (in that order), so that the coordinate representation of the image looks like  $\mathbf{x}(u) = (r(u), g(u), b(u))^\top$ . But we may equally well permute the basis vectors (color channels) independently at each position,

We use  $s$  to denote the dimension of the base space  $\Omega$  and  $d$  referring to the dimension of the fibre. For tangent bundles,  $d = s$  is the dimension of the underlying manifold. For RGB image,  $s = 2$  and  $d = 3$ .

In this example we have chosen  $\mathfrak{G} = \Sigma_3$  the permutations of the 3 color channels as the structure group of the bundle. Other choices, such as a Hue rotation  $\mathfrak{G} = \text{SO}(2)$  are also possible.

as long as we remember the frame (order of channels) in use at each point. As a computational operation this is rather pointless, but as we will see shortly it is conceptually useful to think about gauge transformations for the space of RGB colors, because it allows us to express a gauge symmetry – in this case, an equivalence between colors – and make functions defined on images respect this symmetry (treating each color equivalently).

As in the case of a vector field on a manifold, an RGB gauge transformation changes the numerical representation of an image (permuting the RGB values independently at each pixel) but not the underlying image. In machine learning applications, we are interested in constructing functions  $f \in \mathcal{F}(\mathcal{X}(\Omega))$  on such images (e.g. to perform image classification or segmentation), implemented as layers of a neural network. It follows that if, for whatever reason, we were to apply a gauge transformation to our image, we would need to also change the function  $f$  (network layers) so as to preserve their meaning. Consider for simplicity a  $1 \times 1$  convolution, i.e. a map that takes an RGB pixel  $\mathbf{x}(u) \in \mathbb{R}^3$  to a feature vector  $\mathbf{y}(u) \in \mathbb{R}^C$ . According to our Geometric Deep Learning blueprint, the output is associated with a group representation  $\rho_{\text{out}}$ , in this case a  $C$ -dimensional representation of the structure group  $\mathfrak{G} = \Sigma_3$  (RGB channel permutations), and similarly the input is associated with a representation  $\rho_{\text{in}}(\mathfrak{g}) = \mathfrak{g}$ . Then, if we apply a gauge transformation to the input, we would need to change the linear map ( $1 \times 1$  convolution)  $f : \mathbb{R}^3 \rightarrow \mathbb{R}^C$  to  $f' = \rho_{\text{out}}^{-1}(\mathfrak{g}) \circ f \circ \rho_{\text{in}}(\mathfrak{g})$  so that the output feature vector  $\mathbf{y}(u) = f(\mathbf{x}(u))$  transforms like  $\mathbf{y}'(u) = \rho_{\text{out}}(\mathfrak{g}_u)\mathbf{y}(u)$  at every point. Indeed we verify:

Here the notation  $\rho^{-1}(\mathfrak{g})$  should be understood as the inverse of the group representation (matrix)  $\rho(\mathfrak{g})$ .

$$\mathbf{y}' = f'(\mathbf{x}') = \rho_{\text{out}}^{-1}(\mathfrak{g})f(\rho_{\text{in}}(\mathfrak{g})\rho_{\text{in}}^{-1}(\mathfrak{g})\mathbf{x}) = \rho_{\text{out}}^{-1}(\mathfrak{g})f(\mathbf{x}).$$

**Gauge Symmetries** To say that we consider gauge transformations to be symmetries is to say that any two gauges related by a gauge transformation are to be considered equivalent. For instance, if we take  $\mathfrak{G} = \text{SO}(d)$ , any

two right-handed orthogonal frames are considered equivalent, because we can map any such frame to any other such frame by a rotation. In other words, there are no distinguished local directions such as “up” or “right”. Similarly, if  $\mathfrak{G} = O(d)$  (the orthogonal group), then any left and right handed orthogonal frame are considered equivalent. In this case, there is no preferred orientation either. In general, we can consider a group  $\mathfrak{G}$  and a collection of frames at every point  $u$  such that for any two of them there is a unique  $\mathfrak{g}(u) \in \mathfrak{G}$  that maps one frame onto the other.

Regarding gauge transformations as symmetries in our Geometric Deep Learning blueprint, we are interested in making the functions  $f$  acting on signals defined on  $\Omega$  and expressed with respect to the gauge should equivariant to such transformation. Concretely, this means that if we apply a gauge transformation to the input, the output should undergo the same transformation (perhaps acting via a different representation of  $\mathfrak{G}$ ). We noted before that when we change the gauge, the function  $f$  should be changed as well, but for a gauge equivariant map this is not the case: changing the gauge leaves the mapping invariant. To see this, consider again the RGB color space example. The map  $f : \mathbb{R}^3 \rightarrow \mathbb{R}^C$  is equivariant if  $f \circ \rho_{\text{in}}(\mathfrak{g}) = \rho_{\text{out}}(\mathfrak{g}) \circ f$ , but in this case the gauge transformation applied to  $f$  has no effect:  $\rho_{\text{out}}^{-1}(\mathfrak{g}) \circ f \circ \rho_{\text{in}}(\mathfrak{g}) = f$ . In other words, the coordinate expression of a gauge equivariant map is independent of the gauge, in the same way that in the case of graph, we applied the same function regardless of how the input nodes were permuted. However, unlike the case of graphs and other examples covered so far, gauge transformations act *not on*  $\Omega$  but separately *on each of the feature vectors*  $\mathbf{x}(u)$  by a transformation  $\mathfrak{g}(u) \in \mathfrak{G}$  for each  $u \in \Omega$ .

Further considerations enter the picture when we look at filters on manifolds with a larger spatial support. Let us first consider an easy example of a mapping  $f : \mathcal{X}(\Omega, \mathbb{R}) \rightarrow \mathcal{X}(\Omega, \mathbb{R})$  from scalar fields to scalar fields on an

$s$ -dimensional manifold  $\Omega$ . Unlike vectors and other geometric quantities, scalars do not have an orientation, so a scalar field  $x \in \mathcal{X}(\Omega, \mathbb{R})$  is *invariant* to gauge transformations (it transforms according to the trivial representation  $\rho(\mathfrak{g}) = 1$ ). Hence, any linear map from scalar fields to scalar fields is *gauge equivariant* (or invariant, which is the same in this case). For example, we could write  $f$  similarly to (19), as a convolution-like operation with a position-dependent filter  $\theta : \Omega \times \Omega \rightarrow \mathbb{R}$ ,

$$(x \star \theta)(u) = \int_{\Omega} \theta(u, v) x(v) dv. \quad (22)$$

This implies that we have a potentially different filter  $\theta_u = \theta(u, \cdot)$  at each point, i.e., no spatial weight sharing — which gauge symmetry alone does not provide.

Consider now a more interesting case of a mapping  $f : \mathcal{X}(\Omega, T\Omega) \rightarrow \mathcal{X}(\Omega, T\Omega)$  from vector fields to vector fields. Relative to a gauge, the input and output vector fields  $X, Y \in \mathcal{X}(\Omega, T\Omega)$  are vector-valued functions  $\mathbf{x}, \mathbf{y} \in \mathcal{X}(\Omega, \mathbb{R}^s)$ . A general linear map between such functions can be written using the same equation we used for scalars (22), only replacing the scalar kernel by a matrix-valued one  $\Theta : \Omega \times \Omega \rightarrow \mathbb{R}^{s \times s}$ . The matrix  $\Theta(u, v)$  should map tangent vectors in  $T_v\Omega$  to tangent vectors in  $T_u\Omega$ , but these points have *different gauges* that we may change *arbitrarily and independently*. That is, the filter would have to satisfy  $\Theta(u, v) = \rho^{-1}(\mathfrak{g}(u))\Theta(u, v)\rho(\mathfrak{g}(v))$  for all  $u, v \in \Omega$ , where  $\rho$  denotes the action of  $\mathfrak{G}$  on vectors, given by an  $s \times s$  rotation matrix. Since  $\mathfrak{g}(u)$  and  $\mathfrak{g}(v)$  can be chosen freely, this is an overly

Indeed  $\Theta$  would have to be zero in this case strong constraint on the filter.

A better approach is to first transport the vectors to a common tangent space by means of the connection, and then impose gauge equivariance w.r.t. a single gauge transformation at one point only. Instead of (22), we can then define the following map between vector fields,

$$(\mathbf{x} \star \Theta)(u) = \int_{\Omega} \Theta(u, v) \rho(\mathfrak{g}_{v \rightarrow u}) \mathbf{x}(v) dv, \quad (23)$$

where  $\mathbf{g}_{v \rightarrow u} \in \mathfrak{G}$  denotes the parallel transport from  $v$  to  $u$  along the geodesic connecting these two points; its representation  $\rho(\mathbf{g}_{v \rightarrow u})$  is an  $s \times s$  rotation matrix rotating the vector as it moves between the points. Note that this geodesic is assumed to be unique, which is true only locally and thus the filter must have a local support. Under a gauge transformation  $\mathbf{g}_u$ , this element transforms as  $\mathbf{g}_{u \rightarrow v} \mapsto \mathbf{g}_u^{-1} \mathbf{g}_{u \rightarrow v} \mathbf{g}_v$ , and the field itself transforms as  $\mathbf{x}(v) \mapsto \rho(\mathbf{g}_v) \mathbf{x}(v)$ . If the filter commutes with the structure group representation  $\Theta(u, v) \rho(\mathbf{g}_u) = \rho(\mathbf{g}_u) \Theta(u, v)$ , equation (23) defines a *gauge-equivariant convolution*, which transforms as

$$(\mathbf{x}' \star \Theta)(u) = \rho^{-1}(\mathbf{g}_u)(\mathbf{x} \star \Theta)(u).$$

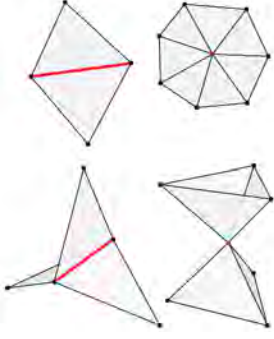
under the aforementioned transformation.

## 4.6 几何图与网格

We will conclude our discussion of different geometric domains with *geometric graphs* (i.e., graphs that can be realised in some geometric space) and *meshes*. In our ‘5G’ of geometric domains, meshes fall somewhere between graphs and manifolds: in many regards, they are similar to graphs, but their additional structure allows to also treat them similarly to continuous objects. For this reason, we do not consider meshes as a standalone object in our scheme, and in fact, will emphasise that many of the constructions we derive in this section for meshes are directly applicable to general graphs as well.

As we already mentioned in Section 4.4, two-dimensional manifolds (surfaces) are a common way of modelling 3D objects (or, better said, the boundary surfaces of such objects). In computer graphics and vision applications, such surfaces are often discretised as *triangular meshes*, which can be roughly thought of as a piece-wise planar approximation of a surface obtained by gluing triangles together along their edges. Meshes are

Triangular meshes are examples of topological structures known as *simplicial complexes*.



Examples of manifold (top) and non-manifold (bottom) edges and nodes. For manifolds with boundary, one further defines *boundary edges* that belong to exactly one triangle.

thus (undirected) *graphs with additional structure*: in addition to nodes and edges, a mesh  $\mathcal{T} = (\mathcal{V}, \mathcal{E}, \mathcal{F})$  also have ordered triplets of nodes forming *triangular faces*  $\mathcal{F} = \{(u, v, q) : u, v, q \in \mathcal{V} \text{ and } (u, v), (u, q), (q, v) \in \mathcal{E}\}$ ; the order of the nodes defines the face *orientation*.

It is further assumed that that each edge is shared by exactly two triangles, and the boundary of all triangles incident on each node forms a single loop of edges. This condition guarantees that 1-hop neighbourhoods around each node are disk-like and the mesh thus constitutes a *discrete manifold* – such meshes are referred to as *manifold meshes*. Similarly to Riemannian manifolds, we can define a *metric* on the mesh. In the simplest instance, it can be induced from the embedding of the mesh nodes  $\mathbf{x}_1, \dots, \mathbf{x}_n$  and expressed through the Euclidean length of the edges,  $\ell_{uv} = \|\mathbf{x}_u - \mathbf{x}_v\|$ . A metric defined in this way automatically satisfies properties such as the *triangle inequality*, i.e., expressions of the form  $\ell_{uv} \leq \ell_{uq} + \ell_{vq}$  for any  $(u, v, q) \in \mathcal{F}$  and any combination of edges. Any property that can be expressed solely in terms of  $\ell$  is *intrinsic*, and any deformation of the mesh preserving  $\ell$  is an *isometry* – these notions are already familiar to the reader from our discussion in Section 4.4.

**拉普拉斯矩阵** By analogy to our treatment of graphs, let us assume a (manifold) mesh with  $n$  nodes, each associated with a  $d$ -dimensional feature vector, which we can arrange (assuming some arbitrary ordering) into an  $n \times d$  matrix  $\mathbf{X}$ . The features can represent the geometric coordinates of the nodes as well as additional properties such as colors, normals, etc, or in specific applications such as chemistry where geometric graphs model molecules, properties such as the atomic number.

Let us first look at the spectral convolution (17) on meshes, which we remind the readers, arises from the Laplacian operator. Considering the mesh as a discretisation of an underlying continuous surface, we can discretise the



Laplacian as

$$(\Delta \mathbf{X})_u = \sum_{v \in \mathcal{N}_u} w_{uv} (\mathbf{x}_u - \mathbf{x}_v), \quad (24)$$

or in matrix-vector notation, as an  $n \times n$  symmetric matrix  $\Delta = \mathbf{D} - \mathbf{W}$ , where  $\mathbf{D} = \text{diag}(d_1, \dots, d_n)$  is called the *degree matrix* and  $d_u = \sum_v w_{uv}$  the *degree* of node  $u$ . It is easy to see that equation (24) performs local permutation-invariant aggregation of neighbour features  $\phi(\mathbf{x}_u, \mathbf{X}_{\mathcal{N}_u}) = d_u \mathbf{x}_u - \sum_{v \in \mathcal{N}_u} w_{uv} \mathbf{x}_v$ , and  $\mathbf{F}(\mathbf{X}) = \Delta \mathbf{X}$  is in fact an instance of our general blueprint (13) for constructing permutation-equivariant functions on graphs.

Note that insofar there is nothing *specific to meshes* in our definition of Laplacian in (24); in fact, this construction is valid for arbitrary graphs as well, with edge weights identified with the adjacency matrix,  $\mathbf{W} = \mathbf{A}$ , i.e.,  $w_{uv} = 1$  if  $(u, v) \in \mathcal{E}$  and zero otherwise. Laplacians constructed in this way are often called *combinatorial*, to reflect the fact that they merely capture the connectivity structure of the graph. For geometric graphs (which do not necessarily have the additional structure of meshes, but whose nodes do have spatial coordinates that induces a metric in the form of edge lengths), it is common to use weights inversely related to the metric, e.g.  $w_{uv} \propto e^{-\ell_{uv}}$ .

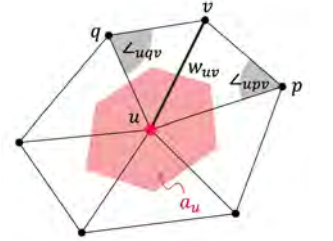
On meshes, we can exploit the additional structure afforded by the faces, and define the edge weights in equation (24) using the *cotangent formula* (Pinkall and Polthier, 1993; Meyer et al., 2003)

$$w_{uv} = \frac{\cot \angle_{uqv} + \cot \angle_{upv}}{2a_u} \quad (25)$$

where  $\angle_{uqv}$  and  $\angle_{upv}$  are the two angles in the triangles  $(u, q, v)$  and  $(u, p, v)$  opposite the shared edge  $(u, v)$ , and  $a_u$  is the local area element, typically computed as the area of the polygon constructed upon the barycenters of the triangles  $(u, p, q)$  sharing the node  $u$  and given by  $a_u = \frac{1}{3} \sum_{v, q: (u, v, q) \in \mathcal{F}} a_{uvq}$ .

The cotangent Laplacian can be shown to have multiple convenient properties (see e.g. Wardetzky et al. (2007)): it is a *positive-semidefinite* matrix,

The degree in this case equals the number of neighbours. If the graph is directed, the corresponding Laplacian is non-symmetric.



The earliest use of this formula dates back to the PhD thesis of MacNeal (1949), who developed it to solve PDEs on the Caltech Electric Analog Computer.

$\Delta \succcurlyeq 0$  and thus has non-negative eigenvalues  $\lambda_1 \leq \dots \leq \lambda_n$  that can be regarded as an analogy of frequency, it is symmetric and thus has orthogonal eigenvectors, and it is *local* (i.e., the value of  $(\Delta \mathbf{X})_u$  depends only on 1-hop neighbours,  $\mathcal{N}_u$ ). Perhaps the most important property is the convergence of the cotangent mesh Laplacian matrix  $\Delta$  to the continuous operator  $\Delta$  when the mesh is infinitely refined (Wardetzky, 2008). Equation (25) constitutes thus an appropriate *discretisation* of the Laplacian operator defined on Riemannian manifolds in Section 4.4.

Some technical conditions must be imposed on the refinement, to avoid e.g.

triangles becoming pathological. One such example is a bizarre triangulation of the cylinder

known in German as the *Schwarzscher Stiefel* (Schwarz’s boot) or in

English literature as the ‘Schwarz lantern’, proposed in 1880 by Hermann Schwarz, a German

mathematician known from the Cauchy-Schwarz inequality fame.

While one expects the Laplacian to be intrinsic, this is not very obvious from equation (25), and it takes some effort to express the cotangent weights entirely in terms of the discrete metric  $\ell$  as

$$w_{uv} = \frac{-\ell_{uv}^2 + \ell_{vq}^2 + \ell_{uq}^2}{8a_{uvq}} + \frac{-\ell_{uv}^2 + \ell_{vp}^2 + \ell_{up}^2}{8a_{uvp}}$$

where the area of the triangles  $a_{ijk}$  is given as

$$a_{uvq} = \sqrt{s_{uvq}(s_{uvq} - \ell_{uv})(s_{uvq} - \ell_{vq})(s_{uvq} - \ell_{uq})}$$

using *Heron’s semiperimeter formula* with  $s_{uvq} = \frac{1}{2}(\ell_{uv} + \ell_{uq} + \ell_{vq})$ . This endows the Laplacian (and any quantities associated with it, such as its eigenvectors and eigenvalues) with *isometry invariance*, a property for which it is so loved in geometry processing and computer graphics (see an excellent review by Wang and Solomon (2019)): any deformation of the mesh that does not affect the metric  $\ell$  (does not ‘stretch’ or ‘squeeze’ the edges of the mesh) does not change the Laplacian.



Laplacian-based filters are isotropic. In the plane, such filters have radial symmetry.

Finally, as we already noticed, the definition of the Laplacian (25) is invariant to the permutation of nodes in  $\mathcal{N}_u$ , as it involves aggregation in the form of summation. While on general graphs this is a necessary evil due to the lack of canonical ordering of neighbours, on meshes we can order the 1-hop neighbours according to some orientation (e.g., clock-wise), and the only ambiguity is the selection of the first node. Thus, instead of any

possible permutation we need to account for *cyclic shifts* (rotations), which intuitively corresponds to the ambiguity arising from  $\text{SO}(2)$  gauge transformations discussed in Section 4.5. For a fixed gauge, it is possible to define an *anisotropic Laplacian* that is sensitive to local directions and amounts to changing the metric or the weights  $w_{uv}$ . Constructions of this kind were used to design shape descriptors by Andreux et al. (2014); Boscaini et al. (2016b) and in early Geometric Deep Learning architectures on meshes by Boscaini et al. (2016a).

**谱分析与网格** The orthogonal eigenvectors  $\Phi = (\varphi_1, \dots, \varphi_n)$  diagonalising the Laplacian matrix ( $\Delta = \Phi \Lambda \Phi^\top$ , where  $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_n)$  is the diagonal matrix of Laplacian eigenvalues), are used as the non-Euclidean analogy of the Fourier basis, allowing to perform spectral convolution on the mesh as the product of the respective Fourier transforms,

$$\mathbf{X} \star \boldsymbol{\theta} = \Phi \text{diag}(\Phi^\top \boldsymbol{\theta})(\Phi^\top \mathbf{X}) = \Phi \text{diag}(\hat{\boldsymbol{\theta}}) \hat{\mathbf{X}},$$

where the filter  $\hat{\boldsymbol{\theta}}$  is designed directly in the Fourier domain. Again, nothing in this formula is specific to meshes, and one can use the Laplacian matrix of a generic (undirected) graph. It is tempting to exploit this spectral definition of convolution to generalise CNNs to graphs, which in fact was done by one of the authors of this text, Bruna et al. (2013). However, it appears that the non-Euclidean Fourier transform is extremely sensitive to even minor perturbations of the underlying mesh or graph (see Figure 12 in Section 4.4) and thus can only be used when one has to deal with different signals on a *fixed* domain, but not when one wishes to generalise across *different domains*. Unluckily, many computer graphics and vision problems fall into the latter category, where one trains a neural network on one set of 3D shapes (meshes) and test on a different set, making the Fourier transform-based approach inappropriate.

The fact that the graph is assumed to be undirected is important: in this case the Laplacian is symmetric and has orthogonal eigenvectors.

As noted in Section 4.4, it is preferable to use spectral filters of the form (18)

applying some transfer function  $\hat{p}(\lambda)$  to the Laplacian matrix,

$$\hat{p}(\Delta)\mathbf{X} = \Phi \hat{p}(\Lambda) \Phi^\top \mathbf{X} = \Phi \operatorname{diag}(\hat{p}(\lambda_1), \dots, \hat{p}(\lambda_n)) \hat{\mathbf{X}}.$$

When  $\hat{p}$  can be expressed in terms of matrix-vector products, the eigende-

In the general case, the complexity of eigendecomposition is  $\mathcal{O}(n^3)$ . [Defferrard et al. \(2016\)](#) used *polynomials* of degree  $r$  as filter functions,

$$\hat{p}(\Delta)\mathbf{X} = \sum_{k=0}^r \alpha_k \Delta^k \mathbf{X} = \alpha_0 \mathbf{X} + \alpha_1 \Delta \mathbf{X} + \dots + \alpha_r \Delta^r \mathbf{X},$$

amounting to the multiplication of the  $n \times d$  feature matrix  $\mathbf{X}$  by the  $n \times n$  Laplacian matrix  $r$  times. Since the Laplacian is typically sparse (with  $\mathcal{O}(|\mathcal{E}|)$  non-zero elements) this operation has low complexity of  $\mathcal{O}(|\mathcal{E}|dr) \sim \mathcal{O}(|\mathcal{E}|)$ . Furthermore, since the Laplacian is local, a polynomial filter of degree  $r$  is localised in  $r$ -hop neighbourhood.

Meshes are nearly-regular graphs, with each node having  $\mathcal{O}(1)$  neighbours, resulting in  $\mathcal{O}(n)$  non-zeros

in  $\Delta$ .

However, this exact property comes at a disadvantage when dealing with meshes, since the actual support of the filter (i.e., the radius it covers) depends on the *resolution* of the mesh. One has to bear in mind that meshes arise from the discretisation of some underlying continuous surface, and one may have two different meshes  $\mathcal{T}$  and  $\mathcal{T}'$  representing *the same object*. In a finer mesh, one might have to use larger neighbourhoods (thus, larger degree  $r$  of the filter) than in a coarser one.



Two-hop neighbourhoods on meshes of different resolution.

For this reason, in computer graphics applications it is more common to use *rational filters*, since they are resolution-independent. There are many ways to define such filters (see, e.g. [Patanè \(2020\)](#)), the most common being as a polynomial of some rational function, e.g.,  $\frac{\lambda-1}{\lambda+1}$ . More generally, one can use a complex function, such as the *Cayley transform*  $\frac{\lambda-i}{\lambda+i}$  that maps the real line into the unit circle in the complex plane. [Levie et al. \(2018\)](#) used spectral filters expressed as *Cayley polynomials*, real rational functions with complex coefficients  $\alpha_l \in \mathbb{C}$ ,

$$\hat{p}(\lambda) = \operatorname{Re} \left( \sum_{l=0}^r \alpha_l \left( \frac{\lambda-i}{\lambda+i} \right)^l \right).$$

Cayley transform is a particular case of a *Möbius transformation*. When applied to the Laplacian (a positive-semidefinite matrix), it maps its non-negative eigenvalues to the complex half-circle.

When applied to matrices, the computation of the Cayley polynomial requires matrix inversion,

$$\hat{p}(\Delta) = \operatorname{Re} \left( \sum_{l=0}^r \alpha_l (\Delta - \mathbf{i}\mathbf{I})^l (\Delta + \mathbf{i}\mathbf{I})^{-l} \right),$$

In signal processing, polynomial filters are termed *finite impulse response* (FIR), whereas rational filters are *infinite impulse response* (IIR).

which can be carried out approximately with linear complexity. Unlike polynomial filters, rational filters do not have a local support, but have exponential decay (Levie et al., 2018). A crucial difference compared to the direct computation of the Fourier transform is that polynomial and rational filters are stable under approximate isometric deformations of the underlying graph or mesh – various results of this kind were shown e.g. by Levie et al. (2018, 2019); Gama et al. (2020); Kenlay et al. (2021).

**Mesher as operators and Functional maps** The paradigm of functional maps suggests thinking of meshes as *operators*. As we will show, this allows obtaining more interesting types of invariance exploiting the additional structure of meshes. For the purpose of our discussion, assume the mesh  $\mathcal{T}$  is constructed upon embedded nodes with coordinates  $\mathbf{X}$ . If we construct an intrinsic operator like the Laplacian, it can be shown that it encodes completely the structure of the mesh, and one can recover the mesh (up to its isometric embedding, as shown by Zeng et al. (2012)). This is also true for some other operators (see e.g. Boscaini et al. (2015); Corman et al. (2017); Chern et al. (2018)), so we will assume a general operator, or  $n \times n$  matrix  $\mathbf{Q}(\mathcal{T}, \mathbf{X})$ , as a representation of our mesh.

In this view, the discussion of Section 4.1 of learning functions of the form  $f(\mathbf{X}, \mathcal{T})$  can be rephrased as learning functions of the form  $f(\mathbf{Q})$ . Similar to graphs and sets, the nodes of meshes also have no canonical ordering, i.e., functions on meshes must satisfy the permutation invariance or equivariance

conditions,

$$\begin{aligned} f(\mathbf{Q}) &= f(\mathbf{PQP}^\top) \\ \mathbf{PF}(\mathbf{Q}) &= \mathbf{F}(\mathbf{PQP}^\top) \end{aligned}$$

for any permutation matrix  $\mathbf{P}$ . However, compared to general graphs we now have more structure: we can assume that our mesh arises from the discretisation of some underlying continuous surface  $\Omega$ . It is thus possible to have a different mesh  $\mathcal{T}' = (\mathcal{V}', \mathcal{E}', \mathcal{F}')$  with  $n'$  nodes and coordinates  $\mathbf{X}'$  representing the same object  $\Omega$  as  $\mathcal{T}$ . Importantly, the meshes  $\mathcal{T}$  and  $\mathcal{T}'$  can have a different connectivity structure and even different number of nodes ( $n' \neq n$ ). Therefore, we cannot think of these meshes as isomorphic graphs with mere reordering of nodes and consider the permutation matrix  $\mathbf{P}$  as correspondence between them.

Functional maps were introduced by [Ovsjanikov et al. \(2012\)](#) as a generalisation of the notion of correspondence to such settings, replacing the correspondence between *points* on two domains (a map  $\eta : \Omega \rightarrow \Omega'$ ) with correspondence between *functions* (a map  $\mathbf{C} : \mathcal{X}(\Omega) \rightarrow \mathcal{X}(\Omega')$ , see Figure 13). A *functional map* is a linear operator  $\mathbf{C}$ , represented as a matrix  $n' \times n$ , establishing correspondence between signals  $\mathbf{x}'$  and  $\mathbf{x}$  on the respective domains as

$$\mathbf{x}' = \mathbf{Cx}.$$

In most cases the functional map is implemented in the spectral domain, as a  $k \times k$  map  $\hat{\mathbf{C}}$  between the Fourier coefficients,  $\mathbf{x}' = \hat{\mathbf{C}}\mathbf{x}$ , where  $\Phi$  and  $\Phi'$  are the respective  $n \times k$  and  $n' \times k$  matrices of the (truncated) Laplacian eigenbases, with  $k \ll n, n'$ .

[Rustamov et al. \(2013\)](#) showed that in order to guarantee *area-preserving* mapping, the functional map must be orthogonal,  $\mathbf{C}^\top \mathbf{C} = \mathbf{I}$ , i.e., be an element of the orthogonal group  $\mathbf{C} \in \text{O}(n)$ . In this case, we can invert the map using  $\mathbf{C}^{-1} = \mathbf{C}^\top$ .

The functional map also establishes a relation between the operator representation of meshes,

$$\mathbf{Q}' = \mathbf{CQC}^\top, \quad \mathbf{Q} = \mathbf{C}^\top \mathbf{Q}' \mathbf{C},$$

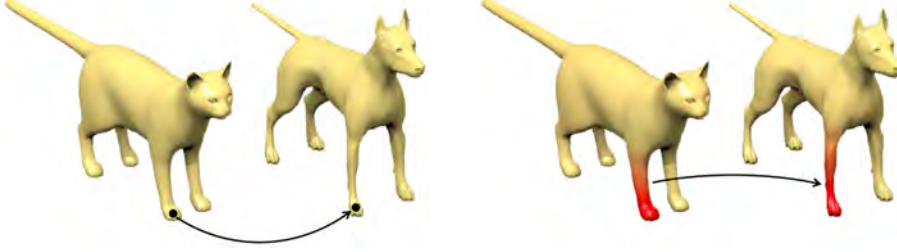


图 13: Pointwise map (left) vs functional map (right).

which we can interpret as follows: given an operator representation  $\mathbf{Q}$  of  $\mathcal{T}$  and a functional map  $\mathbf{C}$ , we can construct its representation  $\mathbf{Q}'$  of  $\mathcal{T}'$  by first mapping the signal from  $\mathcal{T}'$  to  $\mathcal{T}$  (using  $\mathbf{C}^\top$ ), applying the operator  $\mathbf{Q}$ , and then mapping back to  $\mathcal{T}'$  (using  $\mathbf{C}$ ). This leads us to a more general class of *remeshing invariant* (or equivariant) functions on meshes, satisfying

$$\begin{aligned} f(\mathbf{Q}) &= f(\mathbf{C}\mathbf{Q}\mathbf{C}^\top) = f(\mathbf{Q}') \\ \mathbf{C}\mathbf{F}(\mathbf{Q}) &= \mathbf{F}(\mathbf{C}\mathbf{Q}\mathbf{C}^\top) = \mathbf{F}(\mathbf{Q}') \end{aligned}$$

for any  $\mathbf{C} \in \mathbf{O}(n)$ . It is easy to see that the previous setting of permutation invariance and equivariance is a particular case, which can be thought of as a trivial remeshing in which only the order of nodes is changed.

Note that we read these operations *right-to-left*.

Wang et al. (2019a) showed that given an eigendecomposition of the operator  $\mathbf{Q} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top$ , any remeshing invariant (or equivariant) function can be expressed as  $f(\mathbf{Q}) = f(\mathbf{\Lambda})$  and  $\mathbf{F}(\mathbf{Q}) = \mathbf{V}\mathbf{F}(\mathbf{\Lambda})$ , or in other words, remeshing-invariant functions *involve only the spectrum of  $\mathbf{Q}$* . Indeed, functions of Laplacian eigenvalues have been proven in practice to be robust to surface discretisation and perturbation, explaining the popularity of spectral constructions based on Laplacians in computer graphics, as well as in deep learning on graph (Defferrard et al., 2016; Levie et al., 2018). Since this result refers to a generic operator  $\mathbf{Q}$ , multiple choices are available besides the ubiquitous Laplacian – notable examples include the Dirac (Liu et al., 2017; Kostrikov et al., 2018) or Steklov (Wang et al., 2018) operators, as

This follows from the orthogonality of permutation matrices,  $\mathbf{P}^\top \mathbf{P} = \mathbf{I}$ .

well as learnable parametric operators ([Wang et al., 2019a](#)).



## 5 几何深度学习模型

已经彻底研究了我们的几何深度学习蓝图的各种实例 (对于域、对称群和局部性概念的不同选择), 我们准备讨论如何执行这些规定可以生成一些最流行的深度学习体系结构.

我们的论述, 再一次, 将不会严格按照普遍性的顺序. 我们最初涵盖了三种体系结构, 其实现几乎直接遵循我们之前的讨论: 卷积神经网络 (CNNs)、群等变 CNNs 和图神经网络 (GNNs).

然后, 我们将仔细研究图形结构未知的情况下的 GNNs 变体 (即无序集), 通过我们的讨论, 我们将把流行的深度集和转换器架构描述为 GNNs 的实例.

根据我们对几何图形和格网的讨论, 我们首先描述等变信息传递网络, 它将显式几何对称性引入 GNN 计算. 然后, 我们展示了我们的测地线和量规对称理论可以在深度学习中实现的方法, 恢复了一族内在的格网神经网络 (包括测地线神经网络、MoNet 和量规等变格网神经网络).

最后, 我们从时序的 (*temporal*) 角度回顾格网领域. 这个讨论将引导我们到递归神经网络 (RNNs). 我们将展示 RNNs 在时间格网上平移等变的方式, 还将研究它们对时间扭曲变换的稳定性. 这个属性对于正确处理远程依赖关系是非常理想的, 并且对这样的转换执行类不变性产生了门控神经网络的类 (包括流行的 RNN 模型, 如 LSTM 或 GRU).

虽然我们希望以上讨论了在写作时使用的大多数关键的深度学习架构, 但我们很清楚每天都有新的神经网络实例被提出. 因此, 我们不打算覆盖每一个可能的架构, 我们希望下面的部分足够说明问题, 以至于读者能够使用不变性 (invariances) 和对称性 (symmetries) 的镜头轻松地对任何未来的几何深度学习发展进行分类.

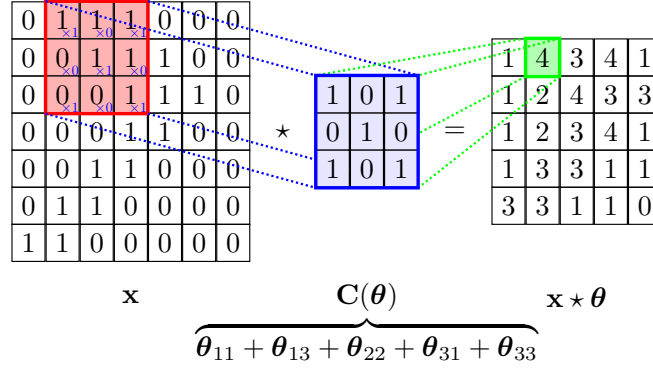


图 14: 用滤波器  $\mathbf{C}(\boldsymbol{\theta})$  为图像  $\mathbf{x}$  做卷积的过程. 滤波器参数  $\boldsymbol{\theta}$  可以表示为发生器  $\boldsymbol{\theta}_{vw}$  的线性组合.

### 5.1 卷积神经网络

卷积神经网络可能是最早和最广为人知的深度学习体系结构的例子, 它遵循 3.5 节中概述的几何深度学习的蓝图. 在第 4.2 节中, 我们已经完全刻画了线性性和局部平移等变算子的类, 由带局部滤波器  $\boldsymbol{\theta}$  的卷积  $\mathbf{C}(\boldsymbol{\theta})\mathbf{x} = \mathbf{x} \star \boldsymbol{\theta}$  给出. 让我们首先关注标量值 ( “单通道” 或 “灰度” ) 离散化图像, 其中域是格网  $\Omega = [H] \times [W]$ ,  $\mathbf{u} = (u_1, u_2)$  和  $\mathbf{x} \in \mathcal{X}(\Omega, \mathbb{R})$ .

与尺寸为  $H^f \times W^f$  的紧支撑滤波器的任何卷积可以写成发生器  $\boldsymbol{\theta}_{1,1}, \dots, \boldsymbol{\theta}_{H^f, W^f}$ , 例如由单位峰值  $\boldsymbol{\theta}_{vw}(u_1, u_2) = \delta(u_1 - v, u_2 - w)$  给出. 因此, 任何局部线性等变映射都可以表示为

请注意, 我们通常想象  $\mathbf{x}$  和  $\boldsymbol{\theta}_{vw}$  是 2D 矩阵, 但是在这个等式中,  $\mathbf{x}$  和  $\boldsymbol{\theta}_{vw}$  都将它们的两个坐标维展平为一维, 使得  $\mathbf{x}$  是一个向量,  $\mathbf{C}(\boldsymbol{\theta}_{vw})$  是一个矩阵. 其在坐标上对应于熟悉的 2D 卷积 (参见图 14 的概览):

$$\mathbf{F}(\mathbf{x}) = \sum_{v=1}^{H^f} \sum_{w=1}^{W^f} \alpha_{vw} \mathbf{C}(\boldsymbol{\theta}_{vw}) \mathbf{x}, \quad (26)$$

$$\mathbf{F}(\mathbf{x})_{uv} = \sum_{a=1}^{H^f} \sum_{b=1}^{W^f} \alpha_{ab} x_{u+a, v+b}. \quad (27)$$

基础  $\boldsymbol{\theta}_{vw}$  的其他选择也是可能的, 并且将产生等效操作 (对于  $\alpha_{vw}$  的潜在不同选择). 一个常见的例子是方向导数 (*directional derivatives*):  $\boldsymbol{\theta}_{vw}(u_1, u_2) =$

$\delta(u_1, u_2) - \delta(u_1 - v, u_2 - w), (v, w) \neq (0, 0)$  加上局部平均值  $\theta_0(u_1, u_2) = \frac{1}{H_f W_f}$ . 事实上, 方向导数可以被认为是图形上扩散过程的格网特定模拟, 如果我们假设每个像素是连接到格网中紧邻像素的节点, 则可以恢复方向导数.

当标量输入通道被多个通道 (例如, RGB 颜色, 或者更一般地, 任意数量的特征映射) 代替时, 卷积滤波器变成卷积张量, 其将输入特征的任意线性组合表示成输出特征映射. 在坐标中, 这可以表示为:

$$\mathbf{F}(\mathbf{x})_{uvj} = \sum_{a=1}^{H^f} \sum_{b=1}^{W^f} \sum_{c=1}^M \alpha_{jabc} x_{u+a, v+b, c}, \quad j \in [N], \quad (28)$$

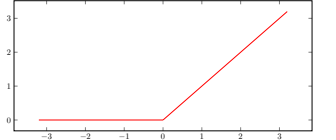
其中  $M$  和  $N$  分别是输入和输出通道的数量. 这一基本操作包含了一大类神经网络架构, 正如我们将在下一节中展示的那样, 这些架构对计算机视觉、信号处理等许多领域产生了深远的影响. 在这里, 与其剖析 CNNs 的无数可能的架构变体, 我们更喜欢关注一些使其得以广泛使用的重要创新.

**高效多尺度计算** 正如在 GDL 通用对称模板中所讨论的, 从卷积算子中提取平移不变特征需要一个非线性步骤. 卷积特征通过非线性激活函数  $\sigma$  进行处理, 以元素方式作用于输入, 即  $\sigma : \mathcal{X}(\Omega) \rightarrow \mathcal{X}(\Omega)$ , 如  $\sigma(\mathbf{x})(u) = \sigma(\mathbf{x}(u))$ . 也许在撰写本文时最流行的例子是修正线性单位 (ReLU):  $\sigma(x) = \max(x, 0)$ . 这种非线性有效地校正 (*rectifies*) 了信号, 将它们的能量推向更低的频率, 并通过迭代结构来计算跨尺度的高阶相互作用.

早在 Fukushima and Miyake (1982) 和 LeCun et al. (1998) 的早期工作中, CNNs 和类似架构就具有多尺度结构, 其中在每个卷积层 (28) 之后, 执行格网粗化  $\mathbf{P} : \mathcal{X}(\Omega) \rightarrow \mathcal{X}(\Omega')$ , 其中格网  $\Omega'$  具有比  $\Omega$  更粗的分辨率. 这使得多尺度滤波器能够有效地增加感受野, 但每个尺度保持恒定数量的参数. 可以使用几种信号粗化策略 (称为池化), 最常见的是应用低通抗混叠滤波器 (例如局部平均), 然后是格网下采样或非线性最大池化.

总之, “普通” CNN 层可以表示为几何深度学习蓝图中已经介绍的基本对象的组成:

$$\mathbf{h} = \mathbf{P}(\sigma(\mathbf{F}(\mathbf{x}))), \quad (29)$$



ReLU, 通常被认为是一种“现代”的架构选择, 已经被用于 Neocognitron (Fukushima and Miyake, 1982). 整流相当于解调原理, 在电气工程中是基础, 作为很多传输协议的基础, 比如 FM 收音机; 并且在神经元活动的模型中也具有突出的作用.

只由本段提到的运算组成的神经网络通常被称为“全卷积”. 相比之下, 一旦应用了足够的等变层和粗化层, 许多中枢神经系统就会在空间轴上展平图像, 并将它们传递给 MLP 分类器. 这就失去了平移不变性.

即等变线性层  $\mathbf{F}$ 、粗化操作  $\mathbf{P}$  和非线性  $\sigma$ . 也可以在 CNNs 内执行平移不变的全局池化操作. 直观地说, 这涉及到每个像素——经过几次卷积后, 总结出一个以它为中心的面片——提出图像的最终表示, 最终的选择由这些建议的聚合形式来指导. 这里一个流行的选择是平均函数, 因为它的输出将保持相似的大小, 而与图像大小无关 (Springenberg et al., 2014).

遵循 CNN 蓝图的突出例子 (其中一些我们将在下面讨论) 显示在图15中.

**深度与残差网络** 因此, 最简单形式的 CNN 架构由超参数  $(H_k^f, W_k^f, N_k, p_k)_{k \leq K}$  指定, 其中  $M_{k+1} = N_k, p_k = 0, 1$  表示是否进行格网粗化. 虽然所有这些超参数在实践中都很重要, 但一个特别重要的问题是理解深度  $K$  在 CNN 架构中的作用, 以及在选择这样一个关键超参数时涉及到哪些基本的权衡, 尤其是与滤波器大小  $(H_k^f, W_k^f)$  的关系.

从历史上看, ResNet 模型早于高速网络 (Srivastava et al., 2015), 高速网络允许更通用的门控机制来控制残差信息流. 虽然这个问题的严格答案仍然难以捉摸, 但近年来收集的越来越多的经验证据表明, 更深 (大  $K$ ) 但更薄 (小  $((H_k^f, W_k^f))$ ) 的模型是一个有利的折衷. 在这种情况下, He et al. (2016) 的一个关键见解是重新参数化每个卷积层, 以模拟先前特征的扰动, 而不是一般的非线性变换:

$$\mathbf{h} = \mathbf{P}(\mathbf{x} + \sigma(\mathbf{F}(\mathbf{x}))) . \quad (30)$$

在这种情况下, ResNet 正在对一个常微分方程进行前向欧拉离散化:  $\dot{\mathbf{x}} = \sigma(\mathbf{F}(\mathbf{x}))$  所得的残差网络提供了优于先前公式的几个关键优点. 本质上, 残差参数化与深层网络是底层连续动力系统的离散化的观点一致, 建模为常微分方程 (ODE). 至关重要的是, 通过模拟其速度来学习动力系统比直接学习其位置要容易得多. 在我们的学习环境中, 这转化为具有更有利几何形状的优化景观, 从而能够训练比以前更深入的架构. 正如将在未来的工作中讨论的, 使用深度学习的学习定义了一个非凸优化问题, 它可以在某些简化的情况下使用梯度下降方法有效地解决. ResNet 参数化的主要优势已经在简单场景中进行了严格分析 (Hardt and Ma, 2016), 并且仍然是理论研究的活跃领域. 最后, 神经微分方程组 (Chen et al., 2018) 是一种最近流行的体系结构, 它通过直接学习  $\text{ODE} \dot{\mathbf{x}} = \sigma(\mathbf{F}(\mathbf{x}))$  的参数并依靠标准数值积分, 将与微分方程组的类比推得更远.

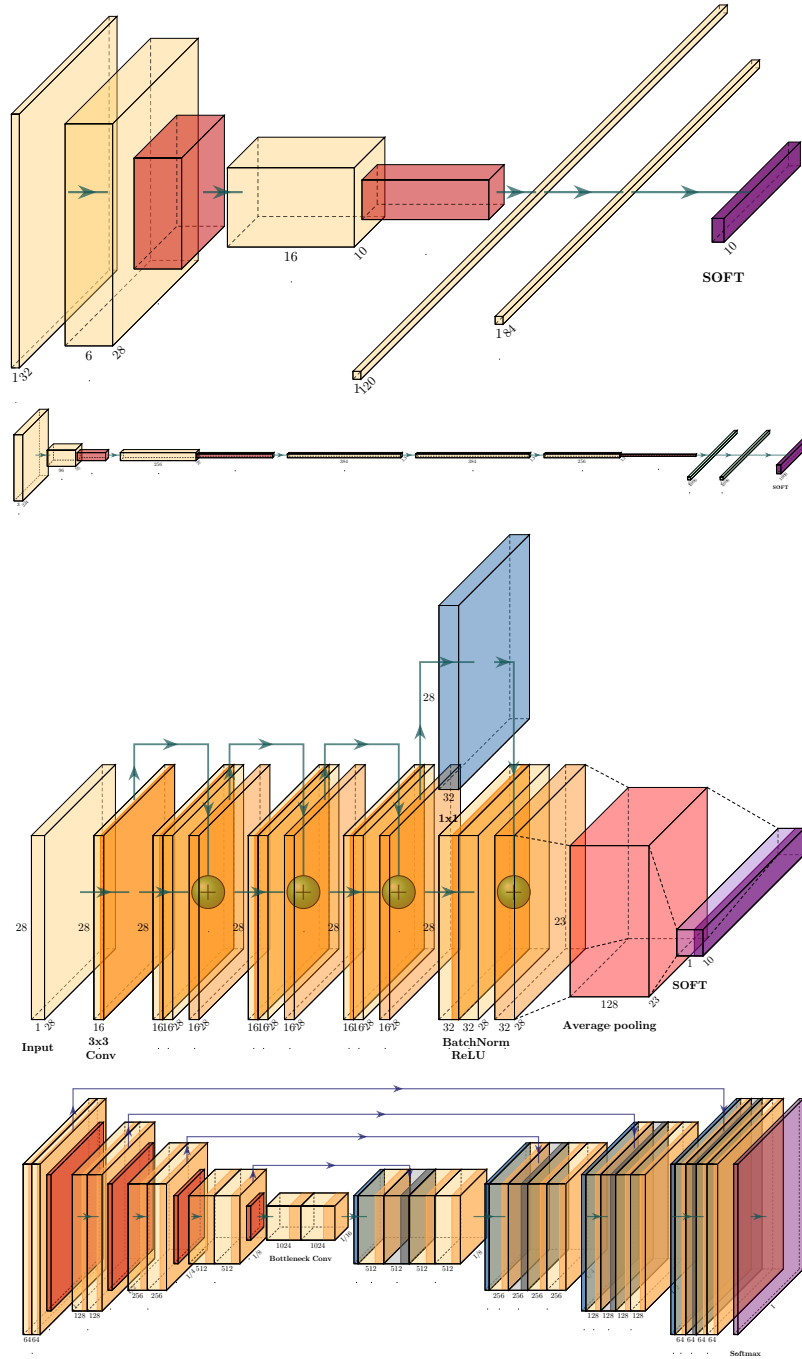


图 15: CNN 架构的突出例子. 自上而下: LeNet (LeCun et al., 1998)、AlexNet (Krizhevsky et al., 2012)、ResNet (He et al., 2016) 和 U-Net (Ronneberger et al., 2015). 使用绘图神经网络包绘制 (Iqbal, 2018).

**归一化** 另一个显著提高中枢神经系统经验性能的重要算法创新是标准化的概念. 在早期的神经活动模型中, 假设神经元执行某种形式的局部“增益控制”, 其中层系数  $\mathbf{x}_k$  由  $\mathbf{x}_k = \sigma_k^{-1} \odot (\mathbf{x}_k - \mu_k)$  代替. 这里,  $\mu_k$  和  $\sigma_k$  分别表示  $\mathbf{x}_k$  的一阶和二阶矩信息. 此外, 它们可以是全局计算的, 也可以是局部计算的.

在深度学习的背景下, 这一原则通过批归一化 (*batch normalisation*) 层被广泛采用 (Ioffe and Szegedy, 2015), 随后还有几个变体 (Ba et al., 2016; Salimans and Kingma, 2016; Ulyanov et al., 2016; Cooijmans et al., 2016; Wu and He, 2018). 尽管有人试图从更好的条件优化景观的角度来严格解释归一化的好处 (Santurkar et al., 2018), 但在撰写本文时, 仍然缺乏一个可以提供指导原则的一般理论.

我们注意到, 甚至在批量标准化出现之前, 神经网络的标准  
化激活就已经受到关注. 例如,  
见Lyu and Simoncelli  
(2008).

**数据增强** 虽然中枢神经系统编码与平移不变性和尺度分离相关的几何先验, 但它们没有明确说明保留语义信息的其他已知变换, 例如闪电或颜色变化, 或小的旋转和膨胀. 用最小的体系结构变化来合并这些先验的实用方法是执行数据扩充, 其中人工地对输入图像执行所述转换, 并将它们添加到训练集中.

数据增强已经在实践中取得了成功, 并得到了广泛的应用——不仅用于训练最先进的视觉架构, 还用于支持自我监督和因果表示学习的若干发展 (Chen et al., 2020; Grill et al., 2020; Mitrovic et al., 2020). 然而, 就样本复杂度而言, 它是可证明次优的 (Mei et al., 2021); 更有效的策略是考虑具有更丰富不变性群的架构——正如我们接下来讨论的.

## 5.2 群等变卷积神经网络

如第4.3节所讨论的, 我们可以将卷积运算从欧几里得空间上的信号推广到由群  $\mathfrak{G}$  作用的任何齐次空间 (*homogeneous space*)  $\Omega$  上的信号. 类似于欧几里得卷积, 其中平移的滤波器与信号匹配, 群卷积的思想是使用群作用在域中移动滤波器, 例如通过旋转和平移. 借助群作用的传递性 (*transitivity*), 我们回想一下, 齐次空间是配备有传递群作用的集合  $\Omega$ , 这意味着对于任何  $u, v \in \Omega$ , 存在  $g \in \mathfrak{G}$ , 使得  $g.u = v$ .

可以将滤波器移动到  $\Omega$  上的任何位置. 在本节中, 我们将讨论群卷积一般思想的几个具体例子, 包括实现方面和架构选择.

**离散群卷积** 我们首先考虑域  $\Omega$  和群  $\mathfrak{G}$  是离散的情况. 作为我们的第一个例子, 我们将医学体积图像表示为具有离散平移和旋转对称性的 3D 格网上的信号. 该域是 3D 立方体格网  $\Omega = \mathbb{Z}^3$ , 并且图像 (例如 MRI 或 CT 3D 扫描) 被建模为函数  $x: \mathbb{Z}^3 \rightarrow \mathbb{R}$ , 即  $x \in \mathcal{X}(\Omega)$ . 虽然实际上这样的图像在有限长方体  $[W] \times [H] \times [D] \subset \mathbb{Z}^3$  上有支撑, 但是我们更愿意将它们看作  $\mathbb{Z}^3$  上具有适当零填充的函数. 作为我们的对称性, 我们考虑  $\mathbb{Z}^3$  上的群  $\mathfrak{G} = \mathbb{Z}^3 \rtimes O_h$  的距离和方向保留变换. 这个群包括平移 ( $\mathbb{Z}^3$ ) 和绕三个轴旋转 90 度产生的离散旋转  $O_h$  (见图 16).

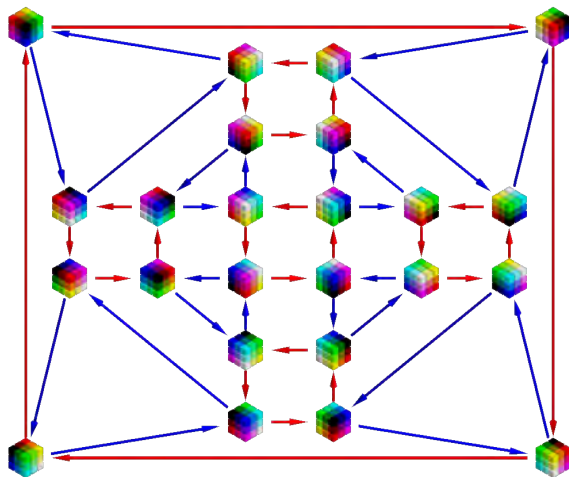
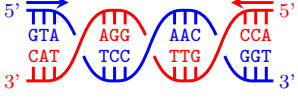


图 16:  $3 \times 3$  滤波器, 由离散旋转群  $O_h$  的所有 24 个元素旋转, 由围绕垂直轴旋转 90 度 (红色箭头) 和围绕对角轴旋转 120 度 (蓝色箭头) 生成.

作为我们的第二个例子, 我们考虑由四个字母组成的 DNA 序列: C、G、A 和 T. 这些序列可以在 1D 格网  $\Omega = \mathbb{Z}$  上表示为信号  $x: \mathbb{Z} \rightarrow \mathbb{R}^4$ , 其中每个字母在  $\mathbb{R}^4$  被单热编码. 自然地, 我们在格网上有一个离散的 1D 平移对称性, 但是 DNA 序列有一个额外的有趣的对称性. 这种对称性源于 DNA 物理上体现为双螺旋的方式, 以及细胞的分子机制读取它的方式. 双螺旋的每一条

DNA 是一种生物聚合物分子, 由四个称为核苷酸 (胞嘧啶、鸟嘌呤、腺嘌呤和胸腺嘧啶) 的重复单元组成, 排列成两条以双螺旋形式相互缠绕的链, 其中每个核苷酸与互补的核苷酸 (碱基对 A/T 和 C/G) 相反.





脱氧核糖核酸双螺旋结构的示意图, 两条链被染成蓝色和红色. 注意螺旋中的序列是如何互补和反向阅读的 (从 5' 到 3').

链都从所谓的 50 端开始, 以 30 端结束, 一条链上的 50 端由另一条链上的 30 端补充. 换句话说, 两股具有相反的方向. 由于 DNA 分子总是从 50 端开始被读出, 但我们不知道是哪一个, 所以像 ACCCTGG 这样的序列相当于每个字母都被它的互补序列 CCAGGGT 取代的反向序列. 这就是所谓的字母序列的反补对称 (*reverse-complement symmetry*). 因此, 我们有对应于恒等式 0 和逆补变换 1 (以及组合  $1 + 1 = 0 \pmod{2}$ ) 的二元群  $\mathbb{Z}_2 = \{0, 1\}$ . 整个群包含了平移 (translations) 和逆补 (reverse-complement) 变换.

在我们的例子中, 我们在第 4.3 节中定义的群卷积 (14) 给出如下:

$$(x \star \theta)(\mathbf{g}) = \sum_{u \in \Omega} x_u \rho(\mathbf{g}) \theta_u, \quad (31)$$

(单通道) 输入信号  $x$  与  $\mathbf{g} \in \mathfrak{G}$  通过  $\rho(\mathbf{g})\theta_u = \theta_{\mathbf{g}^{-1}u}$  转换的滤波器  $\theta$  和输出的内积  $x \star \theta$  是  $\mathfrak{G}$  上的一个函数. 注意, 由于  $\Omega$  是离散的, 我们用一个和代替了等式 (14) 中的积分.

**变换 + 卷积方法** 我们将展示群卷积可以分两步实现: 一个滤波器变换步骤, 和一个平移卷积步骤. 滤波器变换步骤包括创建基本滤波器的旋转 (或逆补变换) 副本, 而平移卷积与标准 CNNs 相同, 因此可以在 GPU 等硬件上高效计算. 要看到这一点, 请注意, 在我们的两个例子中, 我们可以将一般变换  $\mathbf{g} \in \mathfrak{G}$  写成变换  $\mathbf{h} \in \mathfrak{H}$  (例如旋转或逆补变换), 然后是平移  $\mathbf{t} \in \mathbb{Z}^d$ , 即  $\mathbf{g} = \mathbf{t}\mathbf{h}$  (并置表示群元素  $\mathbf{k}$  和  $\mathbf{H}$  的组成). 通过群表示的性质, 我们得到了  $\rho(\mathbf{g}) = \rho(\mathbf{t}\mathbf{h}) = \rho(\mathbf{t})\rho(\mathbf{h})$ . 因此,

$$\begin{aligned} (x \star \theta)(\mathbf{t}\mathbf{h}) &= \sum_{u \in \Omega} x_u \rho(\mathbf{t})\rho(\mathbf{h})\theta_u \\ &= \sum_{u \in \Omega} x_u (\rho(\mathbf{h})\theta)_{u-\mathbf{t}} \end{aligned} \quad (32)$$

我们认为最后一个等式是信号  $x$  和变换滤波器  $\rho(\mathbf{h})\theta$  的标准 (平面欧几里德) 卷积. 因此, 为了实现这些群的群卷积, 我们采用正则滤波器  $\theta$ , 为每个  $\mathbf{h} \in \mathfrak{H}$  (例如, 每个旋转  $\mathbf{h} \in O_h$  或反向互补 DNA 对称性  $\mathbf{h} \in \mathbb{Z}_2$ ) 创建变换副本  $\theta_{\mathbf{h}} = \rho(\mathbf{h})\theta$ , 然后将  $x$  与这些滤波器卷积:  $(x \star \theta)(\mathbf{t}\mathbf{h}) = (x \star \theta_{\mathbf{h}})(\mathbf{t})$ . 对



于我们的两个例子, 对称性通过简单地置换滤波器系数来作用于滤波器, 如图16所示的离散旋转. 因此, 这些操作可以使用具有预先计算的索引的索引操作来有效地实现.

当我们定义由群卷积输出  $x \star \theta$  的特征映射作为  $\mathfrak{G}$  上的函数时, 我们可以将  $\mathfrak{g}$  分为  $\mathfrak{h}$  和  $\mathfrak{k}$  的事实意味着我们也可以将它们视为欧几里德特征映射 (有时称为方向通道) 的堆栈, 每个滤波器变换/方向  $\mathfrak{k}$  有一个特征映射. 例如, 在我们的第一个示例中, 我们将每个滤波器旋转 (图16中的每个节点) 与一个特征映射相关联, 该特征映射是通过卷积 (在传统的平移意义下) 旋转的滤波器获得的. 因此, 这些特征映射仍然可以存储为  $W \times H \times C$  数组, 其中通道  $C$  的数量等于独立滤波器的数量乘以变换  $\mathfrak{h} \in \mathfrak{H}$  的数量 (例如旋转).

如4.3节所示, 群卷积是等变的:  $(\rho(\mathfrak{g})x) \star \theta = \rho(\mathfrak{g})(x \star \theta)$ . 这在定向通道方面的意思是, 在  $\mathfrak{h}$  的作用下, 每个定向通道都被变换, 定向通道本身被置换. 例如, 如果我们在图16中为每个变换关联一个方向通道, 并围绕  $z$  轴旋转 90 度 (对应于红色箭头), 则要素地图将被置换, 如红色箭头所示. 这种描述清楚地表明, 群卷积神经网络与传统的 CNN 有很大的相似性. 因此, 第5.1节中讨论的许多网络设计模式, 如残差网络, 也可以用于群卷积.

**傅立叶域球面卷积神经网络** 对于我们在第4.3节中看到的球体的连续对称群, 可以使用适当的傅里叶变换在谱域中实现卷积 (我们提醒读者  $S^2$  上的卷积是  $SO(3)$  上的函数, 因此我们需要在这两个域上定义傅里叶变换, 以便实现多层球形 CNNs). 球谐函数 (*Spherical harmonics*) 是 2D 球面上的正交基, 类似于复指数的经典傅里叶基. 在特殊的正交群上, 傅里叶基被称为维格纳函数 (*Wigner D-functions*). 在这两种情况下, 傅里叶变换 (系数) 都是作为基函数的内积来计算的, 卷积定理的一个类比成立: 可以将傅里叶域中的卷积作为傅里叶变换的元素积来计算. 此外, 还存在类似快速傅立叶变换的算法来有效地计算  $S^2$  和  $S^3$  上的傅立叶变换. 进一步细节可以参考 [Cohen et al. \(2018\)](#).

### 5.3 图神经网络

图神经网络是利用置换群的性质在图形上实现我们的几何深度学习蓝图。GNNs 是目前存在的最普通的深度学习体系结构, 正如我们将在本文中看到的, 大多数其他深度学习体系结构可以被理解为具有附加几何结构的 GNN 的特例。

根据我们在第4.1节中的讨论, 我们考虑用邻接矩阵  $\mathbf{A}$  和节点特征  $\mathbf{X}$  来指定图。我们将研究 GNN 体系结构, 它们是通过在局部邻域上应用共享置换不变函数  $\mathbf{F}(\mathbf{X}, \mathbf{A})$  而构造的置换等变函数  $\phi(\mathbf{x}_u, \mathbf{X}_{\mathcal{N}_u})$ 。在各种伪装下, 这个局部函数 可以被称为“扩散”、“传播”或“消息传递”, 并且这种  $\mathbf{F}$  的整体计算被称为“GNN 层”。

GNN 图层的设计和研究是写作时深度学习最活跃的领域之一, 使其成为一个具有挑战性的景观。幸运的是, 我们发现绝大多数文献可能仅来自三种“调料”(flavours) 的 GNN 层面 (图17), 我们将在此介绍。这些调料决定了  $\phi$  变换邻域特征的程度, 从而允许在整个图形的交互进行建模时有不同程度的复杂性。

在所有三种调料中, 置换不变性是通过用一些置换不变函数  $\psi$  聚集来自  $\mathbf{X}_{\mathcal{N}_u}$  的特征 (潜在地, 通过一些函数  $\psi$  变换), 然后通过一些函数  $\oplus$  更新节点  $u$  的特征来保证的。一般来说,  $\psi$  和  $\phi$  是可学习的, 其中  $\oplus$  实现为非参数运算, 如和、均值或最大值, 尽管它也可以使用递归神经网络来构建 (Murphy et al., 2018)。

在卷积层调料中 (Kipf and Welling, 2016a; Defferrard et al., 2016; Wu et al., 2019), 邻域节点的特征用固定权重直接聚合,

$$\mathbf{h}_u = \phi \left( \mathbf{x}_u, \bigoplus_{v \in \mathcal{N}_u} c_{uv} \psi(\mathbf{x}_v) \right). \quad (33)$$

这里,  $c_{uv}$  是指节点  $v$  对节点  $u$  的重要性。它是一个常数, 经常直接依赖于  $\mathbf{A}$  中代表图形结构的元素。请注意, 当聚合算子  $\oplus$  被选为求和时, 它可以被视为线性扩散或位置相关的线性滤波, 卷积的泛化。特别是, 我们在第 4.4节和

最常见的是,  $\psi$  和  $\phi$  是具有激活函数的可学习仿射变换; 例如,  $\psi(\mathbf{x}) = \mathbf{W}\mathbf{x} + \mathbf{b}$ ;  $\phi(\mathbf{x}, \mathbf{z}) = \sigma(\mathbf{W}\mathbf{x} + \mathbf{U}\mathbf{z} + \mathbf{b})$ , 其中,  $\mathbf{W}, \mathbf{U}, \mathbf{b}$  是可学习参数,  $\sigma$  是激活函数, 如修正线性单位。  $\mathbf{x}_u$  到  $\phi$  的额外输入代表可选的残差链接, 这通常非常有用。

值得注意的是, 这种风格并不表示卷积的每个 GNN 层 (在与图结构交换的意义上), 而是涵盖了实践中提出的大多数这样的方法。我们将在未来的工作中提供详细的讨论和扩展

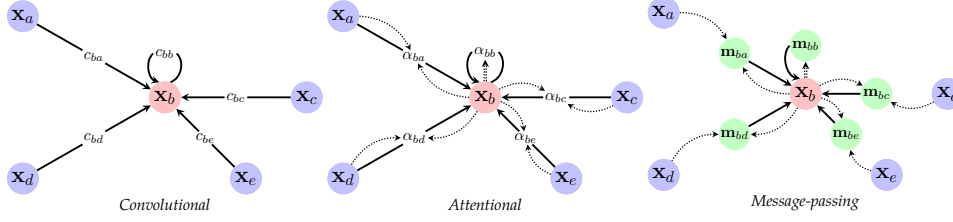


图 17: GNN 层三种调料的数据流可视化, 例如, 我们使用图10中节点  $b$  的邻域来说明这一点. 从左到右: 卷积, 其中发送者节点特征乘以常数  $c_{uv}$  注意, 其中该乘数是通过接收者对发送者的注意机制隐式计算的:  $\alpha_{uv} = a(\mathbf{x}_u, \mathbf{x}_v)$ ; 和消息传递, 其中基于向量的消息是基于发送方和接收方计算的:  $\mathbf{m}_{uv} = \psi(\mathbf{x}_u, \mathbf{x}_v)$ .

第4.6节中看到的频谱滤波器属于这一类, 因为它们相当于将固定的局部算子 (例如拉普拉斯矩阵) 应用于逐节点信号.

在注意力层调料中 (Veličković et al., 2018; Monti et al., 2017; Zhang et al., 2018), 这些相互作用是隐式的

$$\mathbf{h}_u = \phi \left( \mathbf{x}_u, \bigoplus_{v \in \mathcal{N}_u} a(\mathbf{x}_u, \mathbf{x}_v) \psi(\mathbf{x}_v) \right). \quad (34)$$

这里,  $a$  表示可学习的自注意力机制, 它隐式计算重要性系数  $\alpha_{uv} = a(\mathbf{x}_u, \mathbf{x}_v)$ . 它们通常在所有邻居中被 softmax 归一化. 当  $\bigoplus$  表示求和时, 聚合仍然是邻域节点特征的线性组合, 但是现在权重依赖于特征.

最后, 信息传递的调料 (Gilmer et al., 2017; Battaglia et al., 2018) 相当于计算边的任意向量 ( “消息” ),

$$\mathbf{h}_u = \phi \left( \mathbf{x}_u, \bigoplus_{v \in \mathcal{N}_u} \psi(\mathbf{x}_u, \mathbf{x}_v) \right). \quad (35)$$

这里,  $\psi$  是一个可学习的消息函数, 计算发送给  $u$  的  $v$  的向量, 聚合可以被认为是在图上传递消息的一种形式.

需要注意的一件重要的事情是这些方法之间的表示包含: 卷积  $\subseteq$  注意  $\subseteq$  消息传递. 事实上, 注意力 GNNs 可以通过一种被实现为查找表  $a(\mathbf{x}_u, \mathbf{x}_v) = c_{uv}$

的注意力机制来表示卷积 GNNs, 并且卷积 GNNs 和注意 GNNs 都是消息传递的特殊情况, 其中消息仅仅是发送者节点的特征: 对于卷积 GNNs,  $\psi(\mathbf{x}_u, \mathbf{x}_v) = c_{uv}\psi(\mathbf{x}_v)$ , 对于注意力 GNNs,  $\psi(\mathbf{x}_u, \mathbf{x}_v) = a(\mathbf{x}_u, \mathbf{x}_v)\psi(\mathbf{x}_v)$ .

这并不意味着消息传递 GNNs 总是最有用的变体; 由于它们必须计算边上的向量值消息, 它们通常更难训练, 并且需要庞大的内存. 此外, 在大量自然生成的图中, 图的边编码为类相似性 (即边  $(u, v)$  意味着  $u$  和  $v$  可能具有相同的输出). 对于这样的图 (通常被称为同构图), 从正则性 (regularisation) 和可扩展性 (scalability) 两个方面来看, 邻域之间的卷积聚合通常是更好的选择. 注意力 GNNs 提供了一个“中间地带”: 它们允许模拟邻里之间的复杂互动, 同时只计算边上的标量值, 使它们比消息传递更具可扩展性.

这里提出的“三种调料”分别考虑了简洁性, 不可避免地忽略了 GNN 模型的大量细微差别、洞察、泛化性和历史背景. 重要的是, 它排除了基于 Weisfeiler-Lehman 层次的高维 GNN 和依赖于显式计算图傅里叶变换的谱 GNNs.

## 5.4 Deep Sets, Transformers, 潜在图推理

我们通过评论用于学习无序集 (*unordered sets*) 表示的置换-等变神经网络结构来结束对神经网络的讨论. 虽然集合在我们在本文中讨论的领域中具有最少的结构, 但是它们的重要性最近被高度流行的架构所强调, 例如 Transformers (Vaswani et al., 2017) 和 Deep Sets (Zaheer et al., 2017).. 在第4.1节的语言中, 我们假设给定了一个节点特征矩阵, 但是节点之间没有任何指定的邻接或排序信息. 具体的架构将通过决定在多大程度上对节点之间的交互进行建模来产生.

**空边集** 无序集合是指没有任何额外结构或几何信息的——因此, 可以认为处理它们的最自然的方法是完全独立地处理每个集合元素. 这转化为在这些输入作用上置换等变函数, 这已经在第4.1节中介绍过了: 一个应用于每个孤立节点的共享变换. 假设采用与 GNNs 相同的符号 (第5.3节), 这样的模型可

以表示为

$$\mathbf{h}_u = \psi(\mathbf{x}_u),$$

其中  $\psi$  是一个可学习的变换. 可以观察到, 这是卷积 GNN 的一种特殊情况, 其  $\mathcal{N}_u = \{u\}$ —或等价地,  $\mathbf{A} = \mathbf{I}$ . 这种体系结构通常被称为 Deep Sets, [Zaheer et al. \(2017\)](#) 的工作从理论上证明了这种体系结构的几个通用近似性质. 需要注意的是, 在处理点云时, 处理无序集合需要计算机视觉和图形学模型; 其中, 有类模型被称为 PointNets ([Qi et al., 2017](#)).

**完全集** 虽然假设空边集是在无序集上构建函数的一种非常有效的构造, 但我们通常会期望该集的元素表现出某种形式的关系结构, 即节点之间存在潜在的图. 假定  $\mathbf{A} = \mathbf{I}$  会丢弃任何此类结构, 并可能产生次优性能. 相反, 我们可以假设, 在没有任何其他先验知识的情况下, 我们不能预先排除节点之间的任何可能的链接. 在这种方法中, 我们假设完全图,  $\mathbf{A} = \mathbf{1}\mathbf{1}^\top$ ; 等价地, 由于我们不假设访问任何交互作用系数, 在这样的图上运行卷积型神经网络将等于:

$$\mathbf{h}_u = \phi\left(\mathbf{x}_u, \bigoplus_{v \in \mathcal{V}} \psi(\mathbf{x}_v)\right),$$

其中, 第二个输入  $\bigoplus_{v \in \mathcal{V}} \psi(\mathbf{x}_v)$  对于所有节点  $u$  都是相同的, 因此使得模型的表达等同于完全忽略该输入; 即上述情况  $\mathbf{A} = \mathbf{I}$ .

这是  $\bigoplus$  置换不变性的直接结果.

这激发了人们使用更富表现力的 GNN “调料”, 注意力

$$\mathbf{h}_u = \phi\left(\mathbf{x}_u, \bigoplus_{v \in \mathcal{V}} a(\mathbf{x}_u, \mathbf{x}_v) \psi(\mathbf{x}_v)\right) \quad (36)$$

这就产生了“自注意力”的算子, Transformer 架构的核心 ([Vaswani et al., 2017](#)). 假设对注意力系数 (例如 softmax) 进行某种归一化, 我们可以将所有标量  $a(\mathbf{x}_u, \mathbf{x}_v)$  限制在  $[0, 1]$  范围内; 因此, 我们可以把自注意力看作是推断一个软邻接矩阵  $a_{uv} = a(\mathbf{x}_u, \mathbf{x}_v)$ , 作为一些下游任务梯度优化的副产品.

上述观点意味着我们可以在一个完全图上把 Transformers 精确地看作注意力 GNNs ([Joshi, 2020](#)). 然而, 这显然与 Transformers 最初被提议用于建模

采用信息传递“调料”也是合适的. 虽然流行的物理模拟和关系推理 (e.g. [Battaglia et al. \(2016\)](#); [Santoro et al. \(2017\)](#)) 还没有像 Transformers 那样被广泛使用. 这可能是由于在一个完全图中计算向量消息的内存问题, 或者基于向量的消息比基于“自注意力”的“软邻接”

序列相冲突—— $\mathbf{h}_u$  的表示应该注意节点  $u$  在序列中的位置, 而完全图聚合会忽略这一点. Transformers 通过引入位置编码来解决这个问题: 节点特征  $\mathbf{x}_u$  被扩充以对序列中节点  $u$  的位置进行编码, 通常作为正弦波的样本, 其频率取决于  $u$ .

在图中, 不存在节点的自然排序, 对这种位置编码提出了多种选择. 虽然我们将这些替代方案放到以后讨论, 但我们注意到一个有希望的方向是实现 Transformers 中使用的位置编码可以直接与离散傅立叶变换相关, 从而与“环形格网”的图拉普拉斯特征向量相关. 因此, Transformers 的位置编码隐含地代表了我们的假设, 即输入节点连接在一个格网中. 对于更一般的图形结构, 人们可以简单地使用 (假设的) 图拉普拉斯特征向量——这是 Dwivedi and Bresson (2020) 在其经验强大的图 Transformer 模型中采用的.

**推断边集合** 最后, 你可以试着学习潜在的关系结构, 得到一些既不是  $\mathbf{I}$  也不是  $\mathbf{1}\mathbf{1}^\top$  的一般  $\mathbf{A}$ . 推导出 GNN 使用的潜在邻接矩阵  $\mathbf{A}$  (通常称为潜图推理 (*latent graph inference*)) 是图表示学习的一个重要问题. 这是因为假设  $\mathbf{A} = \mathbf{I}$  可能表现较差, 并且由于内存要求和要聚合的大邻居,  $\mathbf{A} = \mathbf{1}\mathbf{1}^\top$  可能难以实现. 此外, 它最接近“真实”问题: 推断邻接矩阵意味着检测  $\mathbf{X}$  行之间的有用结构, 这可能有助于构造假设, 如变量之间的因果关系.

不幸的是, 这样的框架必然导致建模复杂性的增加. 具体来说, 它需要适当平衡结构学习目标 (这是离散的, 因此对基于梯度的优化具有挑战性) 和基于图下游任务. 这使得潜图推理成为一个极具挑战性的复杂问题.

## 5.5 等变消息传递网络

在图神经网络的许多应用中, 节点特征 (或其部分) 不是任意向量, 而是几何实体的坐标. 例如, 在处理分子图时就是这种情况: 表示原子的节点可能包含关于原子类型及其 3D 空间坐标的信息. 以与分子在空间中变换相同的方式来处理特征的后一部分是我们希望的, 换句话说, 除了之前讨论的标准置换等变性, 与描述刚体运动 (旋转, 平移, 镜像) 的欧几里得群  $E(3)$  等价.



为了给我们的 (稍微简化的) 分析奠定基础, 我们将区分节点特征  $\mathbf{f}_u \in \mathbb{R}^d$  和节点空间坐标  $\mathbf{x}_u \in \mathbb{R}^3$ ; 后者被赋予欧几里得对称结构. 在这种设置下, 等变图层分别显式转换这两个输入, 产生修改的节点特征  $\mathbf{f}'_u$  和坐标  $\mathbf{x}'_u$ .

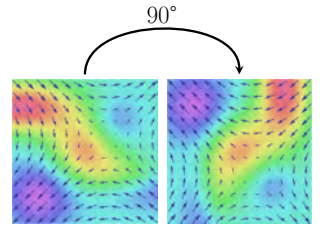
我们现在可以按照几何深度学习蓝图, 陈述我们期望的等变性质. 如果输入的空间分量由  $\mathbf{g} \in \text{E}(3)$  变换 (表示为  $\rho(\mathbf{g})\mathbf{x} = \mathbf{R}\mathbf{x} + \mathbf{b}$ , 其中  $\mathbf{R}$  是模拟旋转和反射的正交矩阵,  $\mathbf{b}$  是平移向量), 输出的空间分量以相同的方式变换 (如  $\mathbf{x}'_u \mapsto \mathbf{R}\mathbf{x}'_u + \mathbf{b}$ ), 而  $\mathbf{f}_u$  保持不变.

就像我们之前在一般图的上下文中讨论的置换等变函数的空间一样, 存在大量满足上述约束的  $\text{E}(3)$ -等变层, 但是并非所有这些层都是几何稳定的, 或者易于实现. 事实上, 实际上有用的等变层的空间可以很容易地通过简单的分类来描述, 这与我们的空间 GNN 层的“三种调料”并无不同. Satorras et al. (2021) 以等变消息传递的形式提出了一个优雅的方案. 他们的模式如下:

$$\begin{aligned}\mathbf{f}'_u &= \phi \left( \mathbf{f}_u, \bigoplus_{v \in \mathcal{N}_u} \psi_f(\mathbf{f}_u, \mathbf{f}_v, \|\mathbf{x}_u - \mathbf{x}_v\|^2) \right), \\ \mathbf{x}'_u &= \mathbf{x}_u + \sum_{v \neq u} (\mathbf{x}_u - \mathbf{x}_v) \psi_c(\mathbf{f}_u, \mathbf{f}_v, \|\mathbf{x}_u - \mathbf{x}_v\|^2)\end{aligned}$$

其中  $\psi_f$  和  $\psi_c$  表示两个不同的 (可学) 函数. 可以表明, 在空间坐标的欧几里得变换下, 这种聚集是等变的. 这是因为  $\mathbf{f}'_u$  on  $\mathbf{x}_u$  的唯一依赖性距离  $\|\mathbf{x}_u - \mathbf{x}_v\|^2$ , 而  $\text{E}(3)$  的作用必然使节点之间的距离保持不变. 此外, 这种层的计算可以被视为“消息传递”GNN 调料的特定实例, 因此它们实现起来是有效的.

总之, 与普通的神经网络相比, Satorras et al. (2021) 能够正确处理图形中每个点的“坐标”. 它们现在被视为  $\text{E}(3)$  群的成员, 这意味着网络输出在输入的旋转、反射和平移下表达正确. 然而, 特征  $\mathbf{f}_u$  是以通道方式处理的, 并且仍然被认为是在这些转换下不变的标量. 这限制了可以在这样的框架内捕获的空间信息的类型. 例如, 可能希望将某些特征编码为矢量 (例如点速度), 在这种变换下, 这些矢量会改变方向. Satorras et al. (2021) 通过在他们架构的一个变体中引入速度的概念, 部分缓解了这个问题. 速度是每个适当旋转的点的 3D 矢量属性. 然而, 这只是一般表示的一个小的子空间, 可以用  $\text{E}(3)$  等



虽然标量要素 (热图) 在旋转时不会改变, 但矢量要素 (箭头) 可能会改变方向. 之前给出的简单的  $\text{E}(3)$  等变 GNN 没有考虑到这一点.

变网络来学习. 一般来说, 节点特征可以编码任意维数的张量, 这些张量仍然会以明确定义的方式根据  $E(3)$  进行变换.

因此, 虽然上面讨论的体系架构已经为许多实际的输入表示提供了一个优雅的等变解决方案, 但在某些情况下, 可能需要探索满足等变属性的更广泛的函数集合. 处理这种场景的现有方法可以分为两类: 不可约表示 (其中前面提到的层是简化的实例) 和正则表示. 我们在这里简单介绍一下, 把详细的讨论留给以后的工作.

**不可约表示** 不可约表示建立在旋转平移群的所有元素都可以变成不可约形式的发现之上: 由块对角矩阵旋转的向量. 至关重要的是, 这些块中的每一个都是维格纳矩阵 (*Wigner D-matrix*)(前面提到的球面神经网络的傅里叶基). 使用等变核从一组不可约表示映射到另一组不可约表示的方法. 为了找到等变映射的全部集合, 可以直接求解这些核上的等变约束. 其解构成了由 *Clebsch-Gordan* 矩阵和球谐函数导出的等变基矩阵的线性组合.

不可约表示方法的早期例子包括张量场网络 (Tensor Field Networks ([Thomas et al., 2018](#))) 和 3D 可控卷积神经网络 (3D Steerable CNNs ([Weiler et al., 2018](#))), 这两种卷积模型都在点云上运行.[Fuchs et al. \(2020\)](#) 的  $SE(3)$ -Transformer 将该框架扩展到图形域, 使用了注意力层而不是卷积层. 此外, 虽然我们的讨论集中在[Satorras et al. \(2021\)](#) 的特例解上, 但我们注意到, 对图的旋转或平移等变预测的动机在历史上已经在其他领域得到了探索, 包括诸如用于点云的动态图 CNN(Dynamic Graph CNN ([Wang et al., 2019b](#))) 和用于量子化学的高效消息传递模型的架构, 例如 SchNet ([Schütt et al., 2018](#)) 和 DimeNet ([Klicpera et al., 2020](#)).

**正则表示** 虽然不可约表示的方法很有吸引力, 但它需要直接推理底层的群表示, 这可能很繁琐, 并且只适用于完备群. 正则表示方法更为普遍, 但会带来额外的计算负担——为了精确的等变, 它们需要存储所有群元素的潜在特征嵌入的副本.

事实上, 这种方法是由我们在前面几节中介绍的群卷积神经网络开创的.



这个领域一个有前途的方法是通过指数和对数映射的定义来观察李群 (*Lie groups*) 的等变, 并有望在各种对称群之间快速原型化. 虽然李群不在本节讨论范围内, 但我们请读者参考这一方向的两个最近成功的例子: LieConv [Finzi et al. \(2020\)](#) 和 LieTransformer [Hutchinson et al. \(2020\)](#).

本节中涉及的方法代表了在几何图形上处理数据的流行方法, 这种方法明显等同于基础几何图形. 如4.6节所述, 网格 (*meshes*) 是几何图形的一个特殊例子, 可以理解为连续曲面的离散. 接下来我们将研究特定网格的等变神经网络.

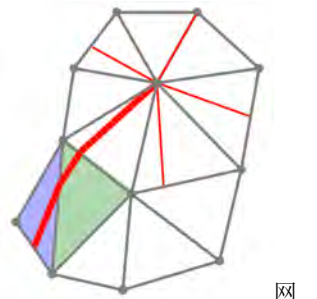
## 5.6 内在网格卷积神经网络

网格, 尤其是三角形网格, 是计算机图形学的“面包和黄油”, 也可能是建模 3D 对象的最常见方式. 深度学习的显著成功, 特别是计算机视觉中的中枢神经系统, 导致了 2010 年代中期图形和几何处理社区对构建网格数据的类似架构的浓厚兴趣.

测地线面片的例子. 为了使得到的网格成为拓扑圆盘, 其半径  $R$  必须小于内射半径 (injectivity radius).

**测地线面片** 大多数用于网格深度学习的体系结构通过离散化或近似指数映射并在三角平面的坐标系中表示滤波器来实现公式 (21) 的卷积滤波器. 跟踪测地线  $\gamma : [0, T] \rightarrow \Omega$ , 从  $u = \gamma(0)$  到领域点  $v = \gamma(T)$ , 定义测地线极坐标 (*geodesic polar coordinates*)  $(r(u, v), \vartheta(u, v))$  的局部系统, 其中  $r$  是  $u$  和  $v$  之间的测地线距离 (测地线  $\gamma$  的长度),  $\vartheta$  是  $\gamma'(0)$  和某个局部参考方向之间的角度. 这允许定义测地线面片  $x(u, r, \vartheta) = x(\exp_u \tilde{\omega}(r, \vartheta))$ , 其中  $\tilde{\omega}_u : [0, R] \times [0, 2\pi) \rightarrow T_u\Omega$  是局部极坐标系.

在离散为网格的曲面上, 测地线是穿过三角形面的折线. 传统上, 测地线是用快速行进 (Fast Marching) 算法 [Kimmel and Sethian \(1998\)](#) 计算的, 它是在介质中波传播的物理模型中遇到的非线性偏微分方程 (*eikonal equation*) 的有效数值近似. 该方案由 [Kokkinos et al. \(2012\)](#) 改编用于计算局部测地线面片, 后来由 [Masci et al. \(2015\)](#) 重新用于构建测地线 CNNs (*Geodesic CNNs*), 这是网格上第一个内在的 (intrinsic) 类似 CNN 的架构.



网格上离散测地线的构造.

**各向同性滤波器** 重要的是, 在定义中, 在参考方向和面片方向的选择上, 面片具有模糊性. 这正是量具选择的模糊性, 我们的局部坐标系被定义为任意旋转 (或角度坐标的移动,  $x(u, r, \vartheta + \vartheta_0)$ ), 这在每个节点上都可能不同. 也许最直接的解决方案是使用  $\theta(r)$  形式的各向同性滤波器, 该滤波器对相邻特征进行方向无关的聚集,

$$(x \star \theta)(u) = \int_0^R \int_0^{2\pi} x(u, r, \vartheta) \theta(r) dr d\vartheta.$$

第4.4-4.6节中讨论的谱滤波器属于这一类: 它们基于各向同性的拉普拉斯算子. 然而, 这种方法丢弃了重要的方向信息, 并且可能无法提取类似边的特征.

**固定量规** 我们已经在第4.4节中提到了另一种方法, 即固定一些量规. Monti et al. (2017) 使用了主曲率方向: 虽然这种选择不是内在的, 可能在平坦点 (曲率消失的地方) 或均匀曲率 (如在完美的球体上) 模糊不清, 但作者表明, 这对于处理近似分段刚性的可变形人体形状是合理的. 后来的工作, 如 Melzi et al. (2019), 显示了网格上的规的可靠的内在结构, 计算为内在函数的 (内在) 梯度. 虽然这样的切场可能具有奇异性 (即, 在某些点消失), 但是整个过程对于噪声和重网格化是非常鲁棒的.

**角度池化** Masci et al. (2015) 使用了另一种方法, 称为角度最大池化 (*angular max pooling*). 在这种情况下, 滤波器  $\theta(r, \vartheta)$  是各向异性的, 但它与函数的匹配是在所有可能的旋转上执行的, 然后这些旋转被聚合:

$$(x \star \theta)(u) = \max_{\vartheta_0 \in [0, 2\pi)} \int_0^R \int_0^{2\pi} x(u, r, \vartheta) \theta(r, \vartheta + \vartheta_0) dr d\vartheta.$$

从概念上来说, 这可以被可视化为将测地线面片与旋转滤波器相关联, 并收集最强的响应.

在网格上, 连续积分可以使用称为面片算子 (*patch operators*) 的结构来离散化 (Masci et al., 2015). 在节点  $u$  周围的测地线面片中, 在局部极坐标中表

示为  $(r_{uv}, \vartheta_{uv})$  的相邻节点  $\mathcal{N}_u$  由一组加权函数  $w_1(r, \vartheta), \dots, w_K(r, \vartheta)$  加权聚合 (如图 18 所示, 作为 “软像素” ).

$$(x \star \theta)_u = \frac{\sum_{k=1}^K w_k \sum_{v \in \mathcal{N}_u} (r_{uv}, \vartheta_{uv}) x_v \theta_k}{\sum_{k=1}^K w_k \sum_{v \in \mathcal{N}_u} (r_{uv}, \vartheta_{uv}) \theta_k}$$

(这里  $\theta_1, \dots, \theta_K$  是滤波器的可学习系数). 多通道功能是按通道处理的, 有一系列合适的滤波器. Masci et al. (2015); Boscaini et al. (2016a) 使用预定义的加权函数  $w$ , 而 Monti et al. (2017) 进一步允许它们是可学习的.

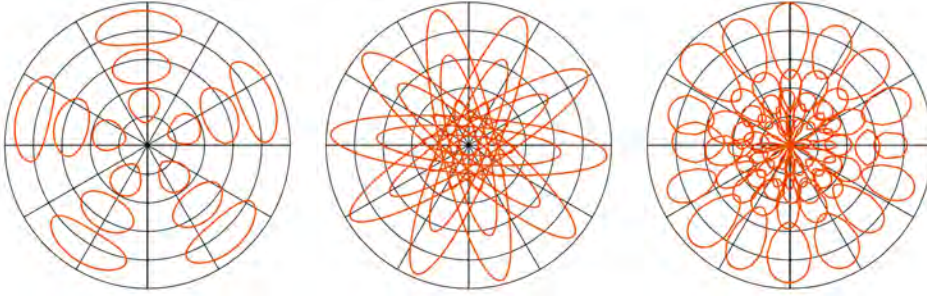


图 18: 从左到右: 在测地线 CNN (Geodesic CNN (Masci et al., 2015))、各向异性 CNN (Anisotropic CNN (Boscaini et al., 2016b)) 和 MoNet ((Monti et al., 2017)) 中使用的面片算子示例, 权重函数  $w_k(r, \vartheta)$  显示为红色.

**量规等变滤波器** 各向同性滤波器和角度最大化池化都导出对量规变换不变 (*invariant*) 的特征; 它们根据平凡表示  $\rho(\mathbf{g}) = 1$  (其中  $\mathbf{g} \in \text{SO}(2)$  是局部坐标框架的旋转) 进行变换. 这一观点启发了 Cohen et al. (2019); de Haan et al. (2020) 提出的另一种方法, 在第 4.5 节中讨论过, 其中由网络计算的特征与结构群  $\mathfrak{G}$  的任意表示  $\rho$  相关联 (例如,  $\text{SO}(2)$  和  $\text{O}(2)$  分别是坐标框架的旋转和旋转 + 镜像). 切向量根据标准表示  $\rho(\mathbf{g}) = \mathbf{g}$  进行变换. 作为另一个示例, 通过匹配同一滤波器的  $n$  个旋转副本获得的特征向量, 在量规旋转下通过循环移位进行变换; 这就是循环群  $C_n$  的正则表示.

正如 4.5 节所讨论的, 当处理这样的几何特征 (与非平凡表示相关联) 时, 我们必须在应用滤波器之前首先将它们平行传输到相同的向量空间. 在网格上,

这可以通过de Haan et al. (2020) 描述的消息传递机制来实现. 设  $\mathbf{x}_u \in \mathbb{R}^d$  为网格节点  $u$  处的  $d$  维输入特征, 该特征可用相应于  $u$  处的量规 (任意) 来表示, 并假设根据  $\mathfrak{G} = \text{SO}(2)$  在量规旋转下的表示  $\rho_{\text{in}}$  来变换. 类似地, 网格卷积的输出特征是  $d'$  维的, 并且应该根据  $\rho_{\text{out}}$  进行变换 ( $\rho_{\text{out}}$  可以由网络设计者随意选择).

与图神经网络类似, 我们可以通过从  $u$  的邻居  $\mathcal{N}_u$  (以及从  $u$  本身) 向  $u$  发送消息来实现网格上的量规等变卷积 (23);

$$\mathbf{h}_u = \Theta_{\text{self}} \mathbf{x}_u + \sum_{v \in \mathcal{N}_u} \Theta_{\text{neigh}}(\vartheta_{uv}) \rho(\mathbf{g}_{v \rightarrow u}) \mathbf{x}_v, \quad (37)$$

请注意,  $d$  是特征尺寸, 不一定等于 2, 网格的维数.

其中  $\Theta_{\text{self}}, \Theta_{\text{neigh}}(\vartheta_{uv}) \in \mathbb{R}^{d' \times d}$  学习好的滤波器矩阵. 结构群元素  $\mathbf{g}_{v \rightarrow u} \in \text{SO}(2)$  表示从  $v$  到  $u$  的平行传输效果, 相对于  $u$  和  $v$  处的量规表示, 可以以为每个网格预先计算. 它的作用由变换矩阵  $\rho(\mathbf{g}_{v \rightarrow u}) \in \mathbb{R}^{d \times d}$  编码. 矩阵  $\Theta_{\text{neigh}}(\vartheta_{uv})$  取决于邻居  $v$  相对于参考方向 (例如, 帧的第一轴) 的角度  $\vartheta_{uv}$ , 所以这个核是各向异性的: 不同的邻居被不同地对待.

这里我们滥用符号, 用角度  $\vartheta$  识别 2D 旋转.

如第4.5节所述, 为了使  $\mathbf{h}(u)$  成为一个定义明确的计量量, 在量规变换下, 它应该变换为  $\mathbf{h}(u) \mapsto \rho_{\text{out}}(\mathbf{g}^{-1}(u)) \mathbf{h}(u)$ . 对于所有  $\vartheta \in \text{SO}(2)$ , 当  $\Theta_{\text{self}} \rho_{\text{in}}(\vartheta) = \rho_{\text{out}}(\vartheta) \Theta_{\text{self}}$ , 代表所有  $\vartheta \in \text{SO}(2)$  和  $\Theta_{\text{neigh}}(\vartheta_{uv} - \vartheta) \rho_{\text{in}}(\vartheta) = \rho_{\text{out}}(\vartheta) \Theta_{\text{neigh}}(\vartheta_{uv})$  时, 情况就满足. 由于这些约束是线性的, 满足这些约束的矩阵  $\Theta_{\text{self}}$  和矩阵值函数  $\Theta_{\text{neigh}}$  的空间是线性子空间, 因此我们可以将它们参数化为具有可学习系数的基核的线性组合:  $\Theta_{\text{self}} = \sum_i \alpha_i \Theta_{\text{self}}^i$  和  $\Theta_{\text{neigh}} = \sum_i \beta_i \Theta_{\text{neigh}}^i$ .

## 5.7 递归神经网络

到目前为止, 我们的讨论一直假设输入在给定的域中是唯一空间 (*spatial*) 的输入. 然而, 在许多常见的用例中, 输入也可以被认为是顺序的 (例如, 视频、文本或语音). 在这种情况下, 我们假设输入由任意多的步骤 (*steps*) 组成, 其中在每个步骤  $t$ , 我们得到一个输入信号, 我们将其表示为  $\mathbf{X}^{(t)} \in \mathcal{X}(\Omega^{(t)})$ .

该域是静态还是动态取决于时间尺度 (*time scales*): 例如,

道路网络确实随着时间而变化 (随着新道路的修建和旧道路的拆除), 但与交通流量相比明显较慢. 类似地, 在社交网络中, 参与度的变化 (例如推特用户转发推文) 发生的频率比下图中的变化高得多.

虽然一般来说, 该域可以随着其上的信号一起随时间演变, 但通常假设该域

在所有  $t$  上保持固定, 即  $\Omega^{(t)} = \Omega$ . 在这里, 我们将专门关注这种情况, 但请注意, 例外是常见的. 社交网络是一个例子, 人们经常不得不考虑随着时间的推移而变化的领域, 因为新的链接经常被创建和删除. 该设置中的域通常被称为动态图 (*dynamic graph*) (Xu et al., 2020a; Rossi et al., 2020).

通常, 单个的  $\mathbf{X}^{(t)}$  输入会表现出有用的对称性, 因此可能会被我们之前讨论的任何体系结构所忽略. 常见的例子有: 视频 ( $\Omega$  是固定网格, 信号是帧序列); 功能磁共振成像扫描 ( $\Omega$  是一个代表大脑皮层几何形状 of 固定网格, 不同的区域在不同的时间被激活, 作为对呈现的刺激的反应); 和交通流网络 ( $\Omega$  是表示道路网络的固定图形, 例如在其上记录了各个节点的平均交通速度).

让我们假设一个编码器函数  $f(\mathbf{X}^{(t)})$  在适合问题的粒度级别上提供潜在表示, 并尊重输入域 of 对称性. 例如, 考虑处理视频帧: 也就是说, 在每个时间步长, 我们得到一个格网结构的输入 (*grid-structured input*), 表示为  $n \times d$  矩阵  $\mathbf{X}^{(t)}$ , 其中  $n$  是像素数 (时间固定),  $d$  是输入通道数 (例如, 对于 RGB 帧,  $d = 3$ ). 此外, 我们对整个帧级别的分析感兴趣, 在这种情况下, 将  $f$  实现为平移不变 CNN 是合适的, 在时间步长  $t$  输出帧的  $k$  维表示  $\mathbf{z}^{(t)} = f(\mathbf{X}^{(t)})$ .

我们现在剩下的任务是在所有步骤中适当地汇总 (*summarising*) 一系列向量  $\mathbf{z}^{(t)}$ . 一种规范的方法是使用递归神经网络 (RNN), 以相对输入的时序方式动态地 (*dynamically*) 聚合该信息, 并且还容易接收在线生成新的数据点. 我们将在这里展示的是, RNNs 本身是一种有趣的几何架构, 因为它们在输入  $\mathbf{z}^{(t)}$  上实现了一种相当不寻常的对称.

**简单递归神经网络** 在每一步, 递归神经网络计算所有输入步骤的  $m$  维汇总向量  $\mathbf{h}^{(t)}$ , 直到包括  $t$ . 这个 (部分) 汇总是根据当前步骤的特征和前一步的汇总, 通过一个共享的更新函数  $R: \mathbb{R}^k \times \mathbb{R}^m \rightarrow \mathbb{R}^m$ , 如下所示 (见图 19):

$$\mathbf{h}^{(t)} = R(\mathbf{z}^{(t)}, \mathbf{h}^{(t-1)}) \quad (38)$$

因为  $\mathbf{z}^{(t)}$  和  $\mathbf{h}^{(t-1)}$  都是平面 (*flat*) 向量表示, 所以  $R$  最容易表示为单个完全连接的神经网络层 (通常称为简单递归神经网络 (SimpleRNNs); 见 see

在我们的例子中, 我们失去一般性; 可以对例如时空图上的节点级输出进行等价分析; 唯一的区别在于编码器  $f$  的选择 (它将是置换等变 GNN).

请注意,  $\mathbf{z}^{(t)}$  向量可以被视为时序格网 (*temporal grid*) 上的点: 因此, 用 CNN 处理它们在某些情况下也是可行的. Transformers 也是越来越受欢迎的处理一般顺序输入 of 模型.

尽管名字很简单, 但它们非常有表现力. 例如, Siegelmann and Sontag (1995) 表明, 这种模型是图灵完备的 (*Turing-complete*), 这意味着它们可能代表我们可能在计算机上执行的任何计算.



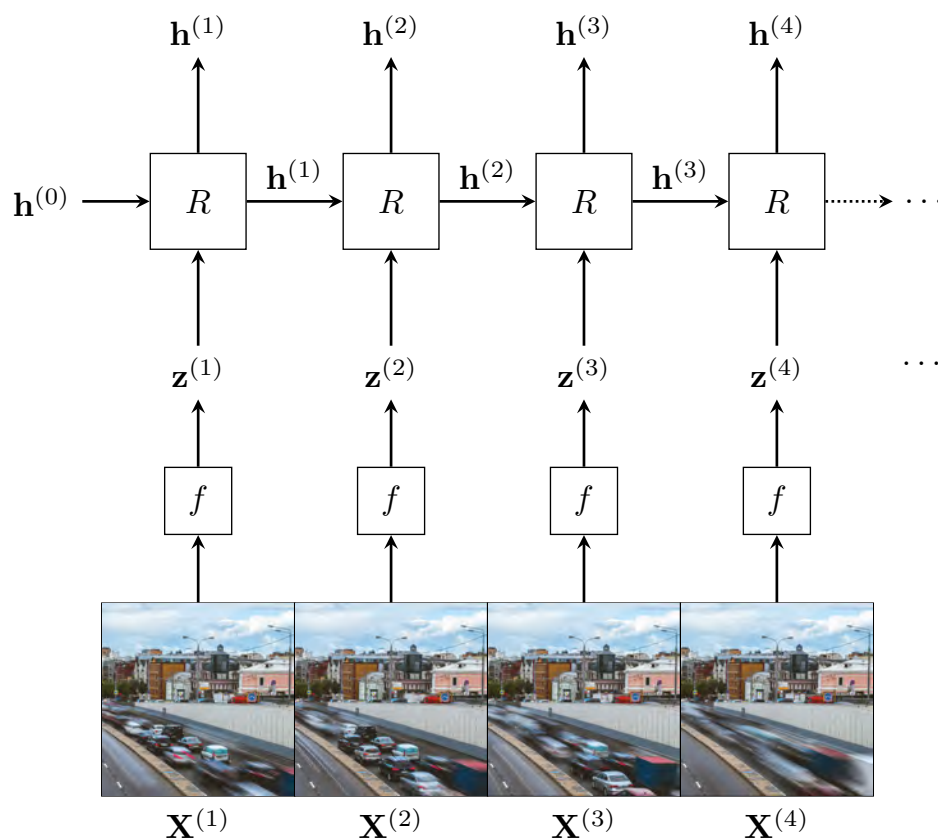


图 19: 使用 RNNs 处理视频输入. 每个输入视频帧  $\mathbf{X}^{(t)}$  使用共享函数  $f$  (例如, 平移不变 CNN) 处理成平面 (flat) 表示  $\mathbf{z}^{(t)}$ . 然后, RNN 更新函数  $R$  在这些向量上迭代, 迭代地更新汇总 (*summary*) 向量  $\mathbf{h}^{(t)}$ , 该向量汇总了直到并包括  $\mathbf{z}^{(t)}$  的所有输入. 该计算以初始汇总向量  $\mathbf{h}^{(0)}$  作为种子, 该向量可以是预先确定的或可学习的.

Elman (1990); Jordan (1997)):

$$\mathbf{h}^{(t)} = \sigma(\mathbf{W}\mathbf{z}^{(t)} + \mathbf{U}\mathbf{h}^{(t-1)} + \mathbf{b}) \quad (39)$$

其中  $\mathbf{W} \in \mathbb{R}^{k \times m}$ ,  $\mathbf{U} \in \mathbb{R}^{m \times m}$  和  $\mathbf{b} \in \mathbb{R}^m$  可学习参数,  $\sigma$  为激活函数. 虽然这在网络的计算图中引入了循环, 但在实践中, 网络被展开适当的步数, 允许通过时间反向传播 (Robinson and Fallside, 1987; Werbos, 1988; Mozer, 1989) 将被应用.

然后下游任务可以适当地利用汇总向量——如果在序列的每一步都需要预测, 则可以将共享预测器单独应用于每个  $\mathbf{h}^{(t)}$ . 为了对整个序列进行分类, 通常将最后的汇总  $\mathbf{h}^{(T)}$  传递给分类器. 这里,  $T$  是序列的长度.

特别地, 初始汇总向量通常要么被设置为零向量, 即  $\mathbf{h}^{(0)} = \mathbf{0}$ , 要么被设置为可学习的. 分析初始汇总向量的设置方式也允许我们推导出由 RNNs 具有的一种有趣的平移等变 (*translation equivariance*) 形式.

**递归神经网络的平移等变** 因为我们把单个的时间步长  $t$  解释为离散的时间步长 (*discrete time-steps*), 所以输入向量  $\mathbf{z}^{(t)}$  可以被看作是时间步长上的一维格网. 虽然在这里尝试从 CNNs 扩展我们的平移等变分析可能是有吸引力的, 但它不能用平凡的方式得到.

为了了解原因, 让我们假设我们已经通过将序列左移一步产生了一个新的序列  $\mathbf{z}'^{(t)} = \mathbf{z}^{(t+1)}$ . 尝试揭露  $\mathbf{h}'^{(t)} = \mathbf{h}^{(t+1)}$  可能很有诱惑力, 正如人们对平移等变的期望; 然而, 这通常是不成立的. 考虑  $t = 1$ ; 直接应用和扩展更新函数, 我们可以得到以下内容:

$$\mathbf{h}'^{(1)} = R(\mathbf{z}'^{(1)}, \mathbf{h}^{(0)}) = R(\mathbf{z}^{(2)}, \mathbf{h}^{(0)}) \quad (40)$$

$$\mathbf{h}^{(2)} = R(\mathbf{z}^{(2)}, \mathbf{h}^{(1)}) = R(\mathbf{z}^{(2)}, R(\mathbf{z}^{(1)}, \mathbf{h}^{(0)})) \quad (41)$$

因此, 除非我们能够保证  $\mathbf{h}^{(0)} = R(\mathbf{z}^{(1)}, \mathbf{h}^{(0)})$ , 否则我们将无法恢复平移等变, 然后可以对步骤  $t > 1$  进行类似的分析.

幸运的是, 通过对我们如何表示  $\mathbf{z}$  的稍微重构, 选择一个合适的  $R$ , 有可能满足上面的等式, 这表示一种移位 (*shifts*) 等变的 RNNs. 我们的问题主要是

Note that this construction is extendable to grids in higher dimensions, allowing us to, e.g., process signals living on images in a *scanline* fashion. Such a construction powered a popular series of models, such as the PixelRNN from van den Oord et al. (2016b).

边界条件 *boundary conditions* 之一: 上面的等式包括  $\mathbf{z}^{(1)}$ , 我们的左移位操作破坏了. 为了抽象出这个问题, 我们将关注 RNN 如何处理一个适当的左填充 (*left-padded*) 序列  $\bar{\mathbf{z}}^{(t)}$ , 定义如下:

$$\bar{\mathbf{z}}^{(t)} = \begin{cases} \mathbf{0} & t \leq t' \\ \mathbf{z}^{(t-t')} & t > t' \end{cases}$$

请注意, 如果我们使用不同于  $\mathbf{0}$  的填充向量, 将需要等价分析. 这样的序列现在允许向左移位多达  $t'$  步, 而不破坏任何原始输入元素. 此外, 请注意, 我们不需要单独处理右移位; 事实上, 从 RNN 等式中自然可以得出右移位的等变形式.

我们现在可以再次分析 RNN 在  $\bar{\mathbf{z}}^{(t)}$  的左移位版本上的运算, 我们用  $\bar{\mathbf{z}}^{(t)} = \bar{\mathbf{z}}^{(t+1)}$  表示, 正如我们在等式 40-41 中所做的:

$$\begin{aligned} \mathbf{h}'^{(1)} &= R(\bar{\mathbf{z}}^{(1)}, \mathbf{h}^{(0)}) = R(\bar{\mathbf{z}}^{(2)}, \mathbf{h}^{(0)}) \\ \mathbf{h}^{(2)} &= R(\bar{\mathbf{z}}^{(2)}, \mathbf{h}^{(1)}) = R(\bar{\mathbf{z}}^{(2)}, R(\bar{\mathbf{z}}^{(1)}, \mathbf{h}^{(0)})) = R(\bar{\mathbf{z}}^{(2)}, R(\mathbf{0}, \mathbf{h}^{(0)})) \end{aligned}$$

其中只要  $t' \geq 1$ , 即只要应用了任何填充, 替换  $\bar{\mathbf{z}}^{(1)} = \mathbf{0}$  就成立. 现在, 只要  $\mathbf{h}^{(0)} = R(\mathbf{0}, \mathbf{h}^{(0)})$ , 我们可以通过一步 ( $\mathbf{h}'^{(t)} = \mathbf{h}^{(t+1)}$ ) 保证左移位等变.

换句话说,  $\mathbf{h}^{(0)}$  必须选择为函数  $\gamma(\mathbf{h}) = R(\mathbf{0}, \mathbf{h})$  的不动点 (*fixed point*). 如果更新函数  $R$  选择方便, 那么不仅可以保证这类不动点的存在, 甚至可以通过迭代  $R$  直到收敛; 举例如下:

$$\mathbf{h}_0 = \mathbf{0} \quad \mathbf{h}_{k+1} = \gamma(\mathbf{h}_k), \quad (42)$$

其中指数  $k$  指的是在我们的计算中  $R$  的迭代, 与表示 RNN 时间步长的指数 ( $t$ ) 相反. 如果我们选择  $R$  使得  $\gamma$  是压缩映射 (*contraction mapping*), 这样的迭代确实会收敛到唯一的不动点. 因此, 我们可以迭代方程 (42), 直到  $\mathbf{h}_{k+1} = \mathbf{h}_k$ , 我们可以设置  $\mathbf{h}^{(0)} = \mathbf{h}_k$ . 请注意, 这种计算相当于用“足够多的”零向量对序列进行左填充.

压缩是函数  $\gamma: \mathcal{X} \rightarrow \mathcal{X}$ , 使得在  $\mathcal{X}$  上的某个范数  $\|\cdot\|$ , 应用  $\gamma$  压缩点之间的距离: 对于所有的  $\mathbf{x}, \mathbf{y} \in \mathcal{X}$ , 以及某些  $q \in [0, 1)$ , 它认为  $\|\gamma(\mathbf{x}) - \gamma(\mathbf{y})\| \leq q\|\mathbf{x} - \mathbf{y}\|$ . 迭代这样的函数必然会收敛到一个唯一的不动点, 这是巴拿赫不动点定理的直接结果 (*Banach's Fixed Point Theorem* (Banach, 1922)).

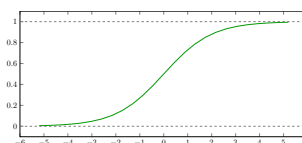
**递归神经网络的深度** 堆叠多个神经网络也很容易——只需将  $\mathbf{h}^{(t)}$  向量用作第二个 RNN 的输入序列. 这种架构通常被称为“深 RNN”, 这可能会产



生误导. 实际上, 由于循环操作的重复应用, 甚至单个 RNN “层” 的深度也等于输入步骤的数量.

在优化神经网络时, 这通常会引入独特的挑战性学习动态, 因为每个训练示例都会对更新网络的共享 (*shared*) 参数进行多次梯度更新. 在这里, 我们将重点关注最突出的问题——消失 (*vanishing*) 和爆炸 (*exploding*) 梯度 (Bengio et al., 1994), 这在 RNNs 中尤其成问题, 因为它们的深度和参数共享. 此外, 它还刺激了一些对神经网络最有影响力的研究. 为了获得更详细的概述, 我们请读者参考 Pascanu et al. (2013), 他们非常详细地研究了 RNNs 的训练动力学, 并从多种角度揭露了这些挑战: 分析、几何和动力系统的视角.

为了说明消失梯度, 考虑一个带有  $\sigma$  激活函数的 SimpleRNNs, 它的导数幅度  $|\sigma'|$  总是在 0 和 1 之间. 将许多这样的值相乘会导致梯度迅速趋于零, 这意味着输入序列中的早期变化可能根本不能影响网络参数的更新.



例如, 考虑下一个单词预测任务 (在预测键盘中常见), 并且输入文本 “*Petar is Serbian. He was born on ...[long paragraph] ...Petar currently lives in \_\_\_\_\_*”. 在这里, 预测下一个单词是 “Serbia” 可能只能通过考虑段落的开头来合理地得出结论——但是当梯度到达这个输入步骤时, 它们可能已经消失了, 使得从这样的例子中学习变得非常困难.

这种激活的例子包括 logistic 函数  $\sigma(x) = \frac{1}{1+\exp(-x)}$ , 以及双曲正切 (*hyperbolic tangent*),  $\sigma(x) = \tanh x$ . 他们被称为 sigmoidal, 因为他们的图形是截然不同的 S 形.

深度前馈神经网络也受到消失梯度问题的困扰, 直到 ReLU 激活 (其梯度恰好等于零或一——从而解决了消失梯度问题) 的发明. 然而, 在 RNNs 中, 使用 ReLUs 可能很容易导致梯度爆炸 (*exploding*), 因为更新函数的输出空间现在是无界的 (*unbounded*), 梯度下降将为每个输入步骤更新一次单元格, 快速建立更新的规模. 历史上, 消失的梯度现象很早就被认为是使用递归网络的一个重大障碍. 解决这个问题推动了更复杂的 RNN 层的发展, 我们将在下面描述.

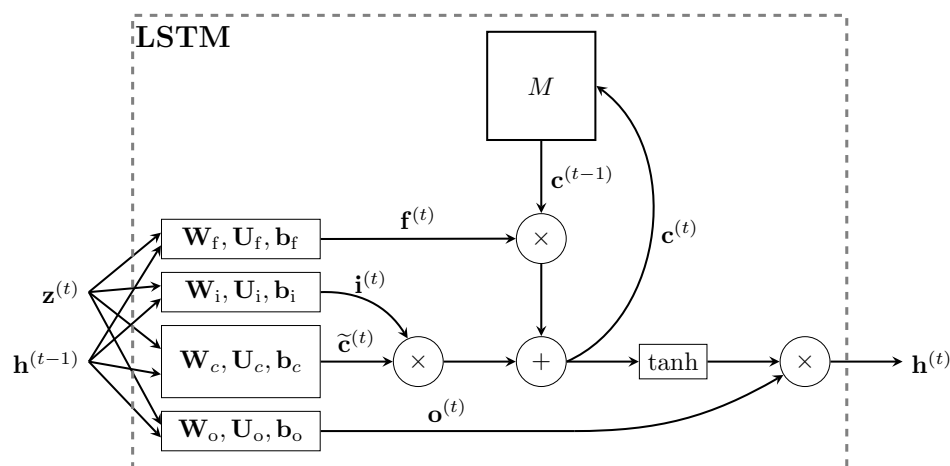


图 20: 长短期记忆的数据流 (LSTM), 其组成部分和记忆单元 ( $M$ ) 清楚地突出. 基于当前输入  $\mathbf{z}^{(t)}$ 、先前汇总  $\mathbf{h}^{(t-1)}$  和先前单元状态  $\mathbf{c}^{(t-1)}$ , LSTM 预测更新的单元状态  $\mathbf{c}^{(t)}$  和汇总  $\mathbf{h}^{(t)}$ .

## 5.8 长短期记忆网络

显著降低 RNNs 中消失梯度影响的一项关键发明是门控机制 (*gating mechanisms*), 它允许网络以数据驱动的方式选择性地覆盖信息. 这些门控神经网络的突出例子包括长短期记忆 (LSTM; Hochreiter and Schmidhuber (1997)) 和门控递归单位 (GRU; Cho et al. (2014)). 在这里, 我们将主要讨论 LSTM——具体来说, Graves (2013) 提出的变体——以说明这种模型的操作. LSTMs 的概念很容易迁移到其他门控 RNNs.

在本节中, 参考图 20 可能会有所帮助, 它说明了我们将在文本中讨论的所有 LSTM 操作.

LSTM 通过引入记忆单元 (*memory cell*) 来增加递归计算, 该记忆单元存储在计算步骤之间保留的单元状态 (*cell state*) 向量  $\mathbf{c}^{(t)} \in \mathbb{R}^m$ . LSTM 直接基于  $\mathbf{c}^{(t)}$  计算汇总向量  $\mathbf{h}^{(t)}$ , 而  $\mathbf{c}^{(t)}$  又通过  $\mathbf{z}^{(t)}$ ,  $\mathbf{h}^{(t-1)}$  和  $\mathbf{c}^{(t-1)}$  来计算. 重要的是, 基于  $\mathbf{z}^{(t)}$  和  $\mathbf{h}^{(t-1)}$ , 单元没有被完全覆盖, 这将使网络面临与 SimpleRNNs 相同的问题. 取而代之的是, 可以保留一定数量的先前单元状态——并且这

种情况发生的比例是从数据中明确获知的。

就像在 SimpleRNN 中一样, 我们通过在当前输入步骤和之前的汇总向量上使用单个完全连接的神经网络层来计算特征:

$$\tilde{\mathbf{c}}^{(t)} = \tanh(\mathbf{W}_c \mathbf{z}^{(t)} + \mathbf{U}_c \mathbf{h}^{(t-1)} + \mathbf{b}_c) \quad (43)$$

但是, 如上所述, 我们不允许所有的向量进入“单元”——因此我们称之为候选 (*candidate*) 特征向量, 并将其表示为  $\tilde{\mathbf{c}}^{(t)}$ . 相反, LSTM 直接学习选通向量 (*gating vectors*)(范围为  $[0, 1]$  的实值向量), 并决定应该允许多少信号进入、退出和覆盖记忆单元。

计算三个这样的门, 全部基于  $\mathbf{z}^{(t)}$  和  $\mathbf{h}^{(t-1)}$ : 输入门  $\mathbf{i}^{(t)}$ , 它计算允许进入单元的候选向量的比例; 忘记门  $\mathbf{f}^{(t)}$  计算要保留的先前单元状态的比例, 输出门  $\mathbf{o}^{(t)}$  计算要用于最终汇总向量的新单元状态的比例. 一般来说, 所有这些门都是使用单个完全连接的层导出的, 采用 logistic sigmoid 激活函数  $\text{logistic}(x) = \frac{1}{1+\exp(-x)}$ , 以保证输出在  $[0, 1]$  范围内:

$$\mathbf{i}^{(t)} = \text{logistic}(\mathbf{W}_i \mathbf{z}^{(t)} + \mathbf{U}_i \mathbf{h}^{(t-1)} + \mathbf{b}_i) \quad (44)$$

$$\mathbf{f}^{(t)} = \text{logistic}(\mathbf{W}_f \mathbf{z}^{(t)} + \mathbf{U}_f \mathbf{h}^{(t-1)} + \mathbf{b}_f) \quad (45)$$

$$\mathbf{o}^{(t)} = \text{logistic}(\mathbf{W}_o \mathbf{z}^{(t)} + \mathbf{U}_o \mathbf{h}^{(t-1)} + \mathbf{b}_o) \quad (46)$$

最后, 这些门被适当地应用于解码新的单元状态  $\mathbf{c}^{(t)}$ , 然后由输出门对其进行调制以产生汇总向量  $\mathbf{h}^{(t)}$ , 如下所示:

$$\mathbf{c}^{(t)} = \mathbf{i}^{(t)} \odot \tilde{\mathbf{c}}^{(t)} + \mathbf{f}^{(t)} \odot \mathbf{c}^{(t-1)} \quad (47)$$

$$\mathbf{h}^{(t)} = \mathbf{o}^{(t)} \odot \tanh(\mathbf{c}^{(t)}) \quad (48)$$

其中  $\odot$  是逐元素向量乘法. 一起应用, 等式 (43)–(48) 完全指定了 LSTM 的更新规则, 现在也考虑了单元向量  $\mathbf{c}^{(t)}$ :

$$(\mathbf{h}^{(t)}, \mathbf{c}^{(t)}) = R(\mathbf{z}^{(t)}, (\mathbf{h}^{(t-1)}, \mathbf{c}^{(t-1)}))$$

请注意, 由于  $\mathbf{f}^{(t)}$  的值是从  $\mathbf{z}^{(t)}$  和  $\mathbf{h}^{(t-1)}$  导出的, 因此可以直接从数据中学习, LSTM 有效地学会了如何适当地忘记过去的经历. 事实上,  $\mathbf{f}^{(t)}$  的值直接

注意我们这里已经把激活函数设置为  $\tanh$ , 由于 LSTMs 的设计是为了改善消失的梯度问题, 现在适合使用 sigmoidal 激活函数.

请注意, 三个门本身就是向量, 即  $\mathbf{i}^{(t)}, \mathbf{f}^{(t)}, \mathbf{o}^{(t)} \in [0, 1]^m$ . 这允许它们控制  $m$  个维度中的每一个维度允许通过门的量.

这仍然与等式 (38) 中的 RNN 更新蓝图兼容; 简单地认为汇总向量是  $\mathbf{h}^{(t)}$  和  $\mathbf{c}^{(t)}$  的连接; 有时用  $\mathbf{h}^{(t)} \parallel \mathbf{c}^{(t)}$  表示.

出现在所有 LSTM 参数 ( $\mathbf{W}_*$ ,  $\mathbf{U}_*$ ,  $\mathbf{b}_*$ ) 的反向传播更新中, 允许网络以数据驱动的方式明确控制梯度在时间步长上的消失程度.

除了正面解决消失的梯度问题, 门控 RNNs 还释放了一种非常有用的对时间扭曲 (*time-warping*) 的不变性, 这是 SimpleRNNs 无法实现的.

我们专注于连续的场景, 因为在那里更容易推理时间的操控.

**门控递归神经的时间扭曲不变性** 我们将首先说明, 在连续时间场景中, 扭曲时间 (*warp time*) 意味着什么, 以及为了实现对这种变换的不变性, 需要一个递归模型. 我们的阐述将在很大程度上遵循 [Tallec and Ollivier \(2018\)](#) 的工作, 他们最初描述了这一现象——事实上, 他们是第一批从不变性的角度实际研究 RNNs 的人.

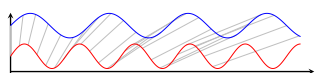
我们将使用  $h(t)$  表示时间  $t$  的连续信号,  $\mathbf{h}^{(t)}$  表示时间步长  $t$  的离散信号.

让我们假设一个连续的时域信号  $z(t)$ , 我们希望在其上应用 RNN. 对齐 RNN 的汇总向量  $\mathbf{h}^{(t)}$  的离散时间计算. 用连续域中的一个类似物,  $h(t)$ , 我们将观察它的线性泰勒展开:

$$h(t + \delta) \approx h(t) + \delta \frac{dh(t)}{dt} \quad (49)$$

同时, 假定  $\delta = 1$ , 我们恢复  $h(t)$  和  $h(t + 1)$  之间的关系, 这正是 RNN 更新函数 (等式38) 所计算的. 即, RNN 更新函数满足以下微分方程:

$$\frac{dh(t)}{dt} = h(t + 1) - h(t) = R(z(t + 1), h(t)) - h(t) \quad (50)$$



这种扭曲操作可以很简单, 例如时间重新缩放; 例如,  $\tau(t) = 0.7t$  (如上所示), 在离散设置中, 这相当于每隔  $\sim 1.43$  步接收新的输入. 然而, 它也允许宽范围的可变 (*variably-changing*) 采样速率, 例如, 采样可以在整个时域中自由加速或减速.

我们希望 RNN 能够适应信号采样的方式 (例如, 通过改变测量的时间单位), 以解决其中的任何缺陷或不规则性. 在形式上, 我们将时间扭曲操作  $\tau : \mathbb{R}^+ \rightarrow \mathbb{R}^+$  表示为时间之间任何单调递增的可微映射. 选择符号  $\tau$  是因为时间扭曲表示时间的自同构 (*automorphism*).

此外, 我们声明一类模型对于时间扭曲是不变的 (*invariant*), 如果对于该类的任何模型和任何这样的  $\tau$ , 存在来自该类的另一个 (可能相同的) 模型, 该模型以与原始模型在非扭曲情况下相同的方式处理扭曲的数据.

这是一个潜在的非常有用的属性. 如果我们有一个能够很好地模拟短期相关性的 RNN 类, 并且我们还可以证明这个类对时间扭曲是不变的, 那么我

们知道有可能以一种同样有效地捕捉长期相关性的方式来训练这样一个模型 (因为它们对应于具有短期相关性的信号的时间膨胀扭曲 (time dilation warping)). 正如我们将很快看到的那样, 像 LSTM 这样的门控 RNN 模型被提出来模拟长程相关性并不是巧合. 实现时间扭曲不变性与门控机制的存在密切相关, 例如 LSTMs 的输入/忘记/输出门.

当时间被  $\tau$  扭曲时, RNN 在时间  $t$  观察到的信号是  $z(\tau(t))$ , 并且为了保持对这种扭曲的不变性, 它应该预测等变扭曲的汇总函数  $h(\tau(t))$ . 再次使用泰勒展开参数, 我们导出了扭曲时间的公式50, RNN 更新  $R$  应满足:

$$\frac{dh(\tau(t))}{d\tau(t)} = R(z(\tau(t+1)), h(\tau(t))) - h(\tau(t)) \quad (51)$$

然而, 上述导数是相对于扭曲时间  $\tau(t)$  计算的, 因此没有考虑原始信号. 为了使我们的模型明确地考虑扭曲变换, 我们需要区分关于  $t$  的扭曲汇总函数. 应用链式规则, 这产生以下微分方程:

$$\frac{dh(\tau(t))}{dt} = \frac{dh(\tau(t))}{d\tau(t)} \frac{d\tau(t)}{dt} = \frac{d\tau(t)}{dt} R(z(\tau(t+1)), h(\tau(t))) - \frac{d\tau(t)}{dt} h(\tau(t)) \quad (52)$$

同时, 为了使我们的 (连续时间)RNN 对于任何时间扭曲 ( $t$ ) 保持不变, 它需要能够显式地表示导数  $\frac{d\tau(t)}{dt}$ , 这是预先不知道的假设! 我们需要引入一个逼近这个导数的可学习函数  $\Gamma$ . 例如,  $\Gamma$  可以是考虑  $z(t+1)$  和  $h(t)$  并预测标量 (scalar) 输出的神经网络.

现在, 请注意, 从时间扭曲下的离散 RNN 模型的角度来看, 其输入  $\mathbf{z}^{(t)}$  将对应于  $z(\tau(t))$ , 其汇总  $\mathbf{h}^{(t)}$  将对应于  $h(\tau(t))$ . 为了获得  $\mathbf{h}^{(t)}$  与  $\mathbf{h}^{(t+1)}$  的所需关系, 以便保持对时间扭曲的不变性, 我们将使用  $h(\tau(t))$  的一步泰勒展开:

$$h(\tau(t+\delta)) \approx h(\tau(t)) + \delta \frac{dh(\tau(t))}{dt}$$

同时, 再次假定  $\delta = 1$ , 代入等式52, 然后离散化

$$\begin{aligned} \mathbf{h}^{(t+1)} &= \mathbf{h}^{(t)} + \frac{d\tau(t)}{dt} R(\mathbf{z}^{(t+1)}, \mathbf{h}^{(t)}) - \frac{d\tau(t)}{dt} \mathbf{h}^{(t)} \\ &= \frac{d\tau(t)}{dt} R(\mathbf{z}^{(t+1)}, \mathbf{h}^{(t)}) + \left(1 - \frac{d\tau(t)}{dt}\right) \mathbf{h}^{(t)} \end{aligned}$$

最后, 我们用前面提到的可学习函数  $\Gamma$  交换  $\frac{d\tau(t)}{dt}$ . 这给了我们时间扭曲不变 RNN 所需的形式:

$$\mathbf{h}^{(t+1)} = \Gamma(\mathbf{z}^{(t+1)}, \mathbf{h}^{(t)})R(\mathbf{z}^{(t+1)}, \mathbf{h}^{(t)}) + (1 - \Gamma(\mathbf{z}^{(t+1)}, \mathbf{h}^{(t)}))\mathbf{h}^{(t)} \quad (53)$$

我们可以很快推导出 SimpleRNNs(等式39) 不是时间扭曲不变的, 因为它们没有等式53中的第二项. 相反, 它们用  $R(\mathbf{z}^{(t+1)}, \mathbf{h}^{(t)})$  完全覆盖  $\mathbf{h}^{(t)}$ , 这对应于假设完全没有时间扭曲;  $\frac{d\tau(t)}{dt} = 1$ , 即  $\tau(t) = t$ .

此外, 我们在连续时间 RNNs 和基于  $R$  的离散 RNN 之间的联系建立在泰勒近似的准确性上, 只有当时间扭曲导数不太大, 即  $\frac{d\tau(t)}{dt} \lesssim 1$  时, 泰勒近似才成立. 对此的直观解释是: 如果我们的时间扭曲操作 (time warping operation) 以使时间增量 ( $t \rightarrow t + 1$ ) 足够大以至于中间数据变化没有被采样的方式压缩时间 (*contracts time*), 模型就永远不能希望以与原始输入相同的方式处理时间扭曲的输入——它根本就不能访问相同的信息. 相反, 任何形式的时间膨胀 (*dilations*)(用离散术语来说, 相当于在输入时间序列中散布零) 在我们的框架内都是完全允许的.

结合我们的单调递增  $\tau$  ( $\frac{d\tau(t)}{dt} > 0$ ) 的要求, 我们可以将  $\Gamma$  的输出空间限制为  $0 < \Gamma(\mathbf{z}^{(t+1)}, \mathbf{h}^{(t)}) < 1$ , 这激励我们为  $\Gamma$  使用逻辑 sigmoid 激活函数, 例如:

$$\Gamma(\mathbf{z}^{(t+1)}, \mathbf{h}^{(t)}) = \text{logistic}(\mathbf{W}_\Gamma \mathbf{z}^{(t+1)} + \mathbf{U}_\Gamma \mathbf{h}^{(t)} + \mathbf{b}_\Gamma)$$

精确 (*exactly*) 匹配 LSTM 门控方程 (例如等式44). 主要区别在于, LSTMs 计算门控向量, 而等式53暗示  $\Gamma$  应该输出标量. 向量化门 (Hochreiter, 1991) 允许在  $\mathbf{h}(t)$  的每个维度上拟合不同的扭曲 (warping) 导数, 允许在多个时阈上同时推理.

这里值得停下来总结一下我们所做的工作. 通过要求我们的 RNN 类对于 (非破坏性的) 时间扭曲是不变的, 我们导出了它必须具有的必要形式 (等式53), 并且表明它精确地对应于门控类 RNNs. 在这种观点下, 门的主要作用是精确拟合扭曲变换的导数  $\frac{d\tau(t)}{dt}$ .

类不变性 (*class invariance*) 的概念与我们先前研究的不变性有些不同. 也就是说, 一旦我们在  $\tau_1(t)$  的时间扭曲输入上训练门控 RNN, 我们通常不能将



其零拍转移 (zero-shot transfer) 为被不同  $\tau_2(t)$  扭曲的信号. 相反, 类不变性只能保证门控神经网络足够强大, 能够以相同的方式拟合这两种信号, 但可能具有非常不同的模型参数. 也就是说, 认识到有效的门控机制与拟合翘曲导数密切相关, 可以为门控 RNN 优化提供有用的处方, 正如我们现在简要展示的那样.

例如, 我们可以经常假设, 希望我们跟踪的信号中依存关系的范围将在  $[T_l, T_h]$  时间步长内.

通过分析方程52的解析解, 可以看出我们的选通 RNN 对  $\mathbf{h}^{(t)}$  的特征遗忘时间与  $\frac{1}{\Gamma(\mathbf{z}^{(t+1)}, \mathbf{h}^{(t)})}$  成正比. 因此, 我们希望我们的选通值在  $\left[\frac{1}{T_h}, \frac{1}{T_m}\right]$  之间, 以便有效地记住假设范围内的信息.

此外, 如果我们假设  $\mathbf{z}^{(t)}$  和  $\mathbf{h}^{(t)}$  大致以零为中心——这是应用层归一化等变换的常见副产品 (Ba et al., 2016)——我们可以假设  $\mathbb{E}[\Gamma(\mathbf{z}^{(t+1)}, \mathbf{h}^{(t)})] \approx \text{logistic}(\mathbf{b}_\Gamma)$ . 因此, 控制选通机制的偏置向量是控制有效选通值的非常有效的方法.

结合这两个观察结果, 我们得出结论, 通过初始化  $\mathbf{b}_\Gamma \sim -\log(\mathcal{U}(T_l, T_h) - 1)$  可以获得适当范围的选通值, 其中  $\mathcal{U}$  是均匀实分布. 这样的建议被Tallec and Ollivier (2018) 称为计时初始化 (*chrono initialisation*), 并已被经验证明可以改善门控 RNNs 的长期依赖性建模.

**利用递归神经进行序列到序列学习** 使用 RNN 支持的计算的一个突出的历史例子是序列到序列 (*sequence-to-sequence*) 的翻译任务, 例如自然语言的机器翻译.Sutskever et al. (2014) 的开创性工作通过传递汇总向量实现了这一点,  $\mathbf{h}^{(T)}$  作为解码器 RNN 的初始输入, RNN 块的输出作为下一步的输入给出.

这给汇总向量  $\mathbf{h}^{(T)}$  带来了相当大的表征压力. 在深度学习的背景下,  $\mathbf{h}^{(T)}$  有时被称为瓶颈. 它的固定容量必须足以表示整个输入序列的内容, 以有助于生成相应序列的方式, 同时还支持长度基本不同的输入序列 (图21).

有可能进行零拍转移的一种情况是, 假设第二次时间扭曲是第一次的时间重新缩放 ( $\tau_2(t) = \alpha \tau_1(t)$ ). 将预先在  $\tau_1$  上训练的选通 RNN 转换为由  $\tau_2$  扭曲的信号需要重新调整选通:  $\Gamma_2(\mathbf{z}^{(t+1)}, \mathbf{h}^{(t)}) = \alpha \Gamma_1(\mathbf{z}^{(t+1)}, \mathbf{h}^{(t)})$ .  $R$  可以保留其参数 ( $R_1 = R_2$ ).

Gers and Schmidhuber (2000); Jozefowicz et al. (2015) 已经发现了这一观点, 他根据经验建议将 LSTMs 的遗忘门偏置初始化为一个恒定的正向量, 比如 1.



瓶颈效应最近在图形表示学习社区 (Alon and Yahav, 2020) 以及神经算法推理 (Cappart et al., 2021) 中受到了极大的关注.

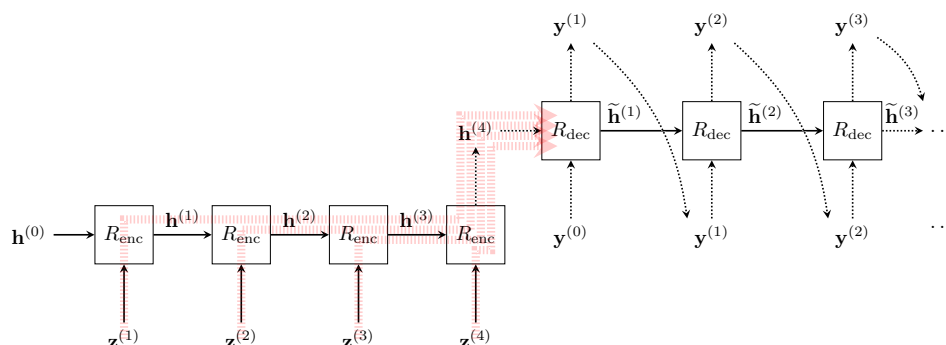


图 21: seq2seq 架构的一个典型例子是 RNN 编码器  $R_{\text{enc}}$  和 RNN 解码器  $R_{\text{dec}}$ . 解码器以来自编码器的最终汇总向量  $\mathbf{h}^{(T)}$  作为种子, 然后以自回归 (autoregressive) 方式进行: 在每一步, 来自前一步的预测输出被反馈作为  $R_{\text{dec}}$  的输入. 瓶颈问题也用红线表示: 汇总向量  $\mathbf{h}^{(T)}$  被迫存储翻译输入序列的所有相关信息, 随着输入长度的增长, 这变得越来越具有挑战性.

实际上, 输出的不同步骤可能希望关注输入的不同部分, 并且所有这样的选择很难通过瓶颈向量来表示. 根据这一观察, Bahdanau et al. (2014) 提出了流行的递归注意力 (recurrent attention) 模型. 在处理的每一步, 查询向量由 RNN 生成; 然后, 这个查询向量与每个时间步长  $\mathbf{h}^{(t)}$  的表示相互作用, 主要是通过计算它们的加权和. 这一模型开创了基于神经内容的注意力, 并先于 Transformer 模型获得成功.

最后, 虽然试图提出一种动态关注部分输入内容的软方法, 但实质性的工作也让我们学会了用明确的 (explicit) 方法将注意力引向输入. 一种强大的基于算法的方法是 Vinyals et al. (2015) 的指针网络 (pointer network), 该网络提出了一种简单的递归注意力修改模块, 以允许指向可变大小的输入元素. 这些发现随后被推广到 set2set 架构 (Vinyals et al., 2016), 该架构将 seq2seq 模型推广到无序集, 由指针网络支持的 LSTMs 支持.



## 6 问题与应用

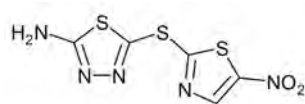
不变性和对称性在源自现实世界的数据中普遍存在。因此，毫不奇怪，21 世纪机器学习的一些最受欢迎的应用是作为几何深度学习的直接副产品出现的，也许有时没有完全意识到这一事实。我们想为读者提供一个概述——绝不是全面的——几何深度学习中有影响力的作品以及令人兴奋和有前途的新应用。我们的动机是双重的：展示五个几何领域共同出现的科学和工业问题的具体实例，并为进一步研究几何深度学习原则和体系结构提供额外的动机。

### 6.1 化学与药物设计

图形表示学习最有前途的应用之一是在计算化学和药物开发。传统药物是小分子，被设计成与某些目标分子（通常是蛋白质）化学连接（结合），以激活或破坏某些疾病相关的化学过程。不幸的是，药物开发是一个极其漫长和昂贵的过程：在撰写本文时，将一种新药推向市场通常需要十多年的时间，费用超过十亿美元。原因之一是许多药物在不同阶段失败的测试成本——不到 5% 的候选药物进入了最后阶段（例如，见 [Gaudelet et al. \(2020\)](#)）。

由于化学合成分子的空间非常大（估计在  $10^{60}$  左右），寻找具有正确结合特性的候选分子，如靶结合亲和力、低毒性、溶解性等。不能通过实验完成，而是采用虚拟或电子筛选（即使用计算技术来识别有希望的分子）。机器学习技术在这项任务中发挥着越来越重要的作用。[Stokes et al. \(2020\)](#) 最近展示了一个使用几何深度学习（Geometric Deep Learning）进行虚拟药物筛选的突出例子，该方法使用经过训练的图形神经网络来预测候选分子是否抑制模型细菌大肠杆菌的生长，他们能够有效地发现 *Halicin*（一种最初用于治疗糖尿病分子）是一种高效抗生素，即使是针对已知抗生素耐药性的细菌菌株。这一发现被科学和大众媒体广泛报道。

许多药物不是设计出来的，而是偶然发现的。植物王国的许多药物的历史来源反映在它们的名字上：例如，乙酰水杨酸，通常被称为阿司匹林 (*aspirin*)，包含在柳树皮（柳属）中，柳属植物的药用特性自古以来就为人所知。



Halicin 的分子图。

## 6.2 药物重新定位

虽然产生全新的候选药物是一种潜在的可行方法，但开发新疗法的更快、更便宜的途径是药物重新定位 (*drug repositioning*)，即寻求评估已经批准的药物 (单独或联合) 的新用途。这通常会显著减少将药物投放市场所需的临床评估量。在某种抽象层次上，药物对身体生物化学的作用以及它们彼此之间以及与其他生物分子之间的相互作用可以被建模为图形，从而产生了由著名网络科学家阿尔伯特-拉斯洛·巴拉斯 (Albert-László Barabási) 创造的“网络医学”概念，并倡导使用生物网络 (如蛋白质-蛋白质相互作用和代谢途径) 来开发新的疗法 (Barabási et al., 2011)。

几何深度学习为这类方法提供了一种现代的方式。一个突出的早期例子是 Zitnik et al. (2018) 的工作，他使用图形神经网络来预测药物重新定位形式的副作用，称为组合疗法或多药疗法，在药物-药物相互作用图中表示为边缘预测。在撰写本文时，新型冠状病毒大流行仍在进行中，这引发了人们对尝试将此类方法应用于新冠肺炎的特别兴趣 (Gysi et al., 2020)。最后，我们应该注意到，药物重新定位不一定局限于合成分子: Veselkov et al. (2019) 对食物中包含的药物样分子应用了类似的方法 (因为，正如我们提到的，许多植物基食物包含用于肿瘤治疗的化合物的生物类似物)。本文的一位作者参与了一项合作，通过与一位分子厨师合作，为这项研究增添了创造性，这位分子厨师根据富含这种药物样分子的“超级食物”成分设计了令人兴奋的食谱。

## 6.3 蛋白质生物学

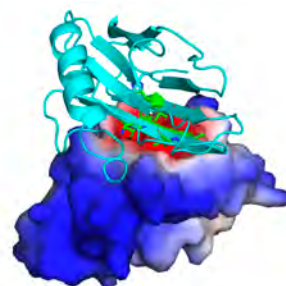
既然我们已经提到了蛋白质作为药物靶点，让我们在这个话题上多花一些时间。蛋白质可以说是我们身体中具有多种功能的最重要的生物分子之一，包括抵御病原体 (抗体)、赋予皮肤结构 (胶原蛋白)、向细胞输送氧气 (血红蛋白)、催化化学反应 (酶) 和发出信号 (许多激素是蛋白质)。从化学角度来说，蛋白质是一种生物聚合物，或者是一系列被称为氨基酸的小构件，在静电力的影响下折叠成复杂的 3D 结构。正是这种结构赋予了蛋白质功能，因此，

一个常见的隐喻，可以追溯到诺贝尔化学奖得主埃米尔·费舍尔 (Emil Fischer)，是施吕塞尔-施洛斯-普林齐普 (Schlus-Schloss-Prinzip) (“钥匙锁原理”，1894 年): 两种蛋白质通常只有在几何和化学结构互补的情况下才会相互作用。

理解蛋白质如何工作以及它们做什么至为重要。由于蛋白质是药物治疗的常见目标，制药工业对这一领域有着浓厚的兴趣。

蛋白质生物信息学中典型的问题层次是从蛋白质序列 (20 个不同氨基酸字母表上的 1D 字符串) 到 3D 结构 (一个被称为“蛋白质折叠”的问题) 到功能 (“蛋白质功能预测”)。Senior et al. (2020) 最近的方法，如 DeepMind 的 AlphaFold，使用接触图来表示蛋白质结构。Gligorijevic et al. (2020) 表明，将图形神经网络应用于此类图形，可以实现比使用纯粹基于序列的方法更好的功能预测。

Gainza et al. (2020) 开发了一个名为 MaSIF 的几何深度学习管道，从蛋白质的 3D 结构预测蛋白质之间的相互作用。MaSIF 将蛋白质建模为离散为网格的分子表面，认为这种表示在处理相互作用时是有利的，因为它允许抽象内部折叠结构。该体系结构基于网格卷积神经网络，该网络在小的局部测地线片中对预先计算的化学和几何特征进行操作。该网络使用蛋白质数据库中的几千个共晶体蛋白质 3D 结构进行培训，以解决多项任务，包括界面预测、配体分类和对接，并允许重新设计蛋白质 (从零开始)，这些蛋白质原则上可用作抗癌的生物免疫治疗药物——这些蛋白质旨在抑制程序性细胞死亡蛋白质复合物 (PD-1/PD-L1) 各部分之间的蛋白质-蛋白质相互作用 (PPI)，并赋予免疫系统攻击肿瘤细胞的能力。



肿瘤靶向 PD-L1 蛋白表面 (热图显示预测的结合位点) 和设计的结合剂 (如带状图所示)。

## 6.4 推荐系统与社交网络

图表示学习的第一个普及的大规模应用出现在社交网络中 *social networks*，主要是在推荐系统的环境中。推荐者的任务是决定向用户提供哪些内容，这可能取决于他们以前在服务上的交互历史。这通常是通过链接预测目标来实现的：监督各种节点 (内容片段) 的嵌入，使得如果它们被认为是相关的 (例如，通常一起观看)，则它们被保持在一起。然后，两个嵌入 (例如，它们的内积) 的接近度 (*proximity*) 可以被解释为它们通过内容图中的边链接的概率，因此对于用户查询的任何内容，一种方法可以服务于嵌入空间中的  $k$  个最近邻居。

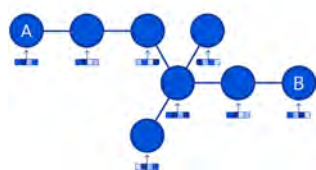


Pinterest 还有后续工作 PinnerSage(Pal et al., 2020), 该工作有效地将用户特定的上下文信息集成到推荐器中。

这种方法的先驱之一是美国图像共享和社交媒体公司 Pinterest: 除了展示 GNNs 在生产中的第一次成功部署之外, 他们的方法 PinSage成功地使图形表示学习可扩展到数百万个节点和数十亿条边的图形 (Ying et al., 2018)。相关的应用, 特别是在产品推荐领域, 很快接踵而至。目前投入生产的热门 GNNbacked 推荐产品包括阿里巴巴的 Aligraph (Zhu et al., 2019) 和亚马逊的 P-Companion(Hao et al., 2020)。这样, 图形深度学习每天都在影响着数百万人。

在社交网络内容分析方面, 另一个值得注意的努力是 Fabula AI, 它是第一批被收购的 GNN 初创企业之一 (2019 年, 被 Twitter 收购)。这家初创公司由该文本的一位作者及其团队创建, 开发了一种新技术来检测社交网络上的错误信息 (Monti et al., 2019)。Fabula 的解决方案包括通过分享某个特定新闻项目的用户网络对其传播进行建模。如果其中一个用户重新分享了另一个用户的信息, 并且他们在社交网络上相互关注, 那么这些用户就是有联系的。然后, 这个图形被输入到一个图形神经网络中, 该网络将整个图形分为“真”或“假”内容——标签基于事实核查机构之间的一致。除了表现出快速稳定的强大预测能力 (通常在新闻传播的几个小时内), 分析单个用户节点的嵌入揭示了倾向于共享不正确信息的用户的清晰聚类, 例证了众所周知的“回声室” (‘echo chamber’) 效应。

## 6.5 交通预测



道路网络 (顶部) 及其相应的图形表示 (底部)。

交通网络是几何深度学习技术已经对全球数十亿用户产生切实影响的另一个领域。例如, 在道路网络上, 我们可以将交叉点视为节点, 将路段视为连接它们的边, 然后可以通过路段的道路长度、当前或历史速度等来表征这些边。

这个空间中的一个标准预测问题是预测估计到达时间 (ETA): 对于给定的候选路线, 提供穿越它所需的预期行驶时间。在这个领域, 这样一个问题至关重要, 不仅对于面向用户的流量推荐应用程序, 而且对于在自己的运营中利用这些预测的企业 (如送餐或拼车服务) 也是如此。

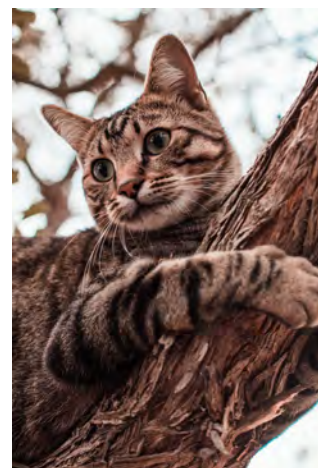
图神经网络在这一领域也显示出巨大的潜力: 例如, 它们可以用于直接预测道路网络相关子图的预计到达时间 (实际上是图形回归任务)。DeepMind 成功地利用了这种方法, 产生了一个基于 GNNbased 的预计到达时间预测器, 该预测器现已在谷歌地图 (Derrow-Pinion et al., 2021) 上投入生产, 为全球几个主要大都市地区的预计到达时间查询提供服务。百度地图团队也观察到了类似的回报, 其中旅行时间预测目前由 ConSTGAT 模型提供, 该模型本身基于图形注意力网络模型的时空变体 (Fang et al., 2020)。



在一些大都市地区, 全球导航卫星系统在谷歌地图上提供查询服务, 预测质量有所提高 (悉尼等城市提高了 40%)。

## 6.6 物体识别

计算机视觉中机器学习技术的一个主要基准是对所提供图像中的中心对象进行分类的能力。ImageNet 大规模视觉识别挑战 (Russakovsky et al., 2015, ILSVRC) 是一项年度对象分类挑战, 推动了几何深度学习的早期发展。ImageNet 要求模型将从网络上抓取的真实图像分为 1000 个类别之一: 这些类别同时是多样的 (包括有生命的和无生命的物体), 以及特定的 (许多类别侧重于区分各种猫和狗的品种)。因此, 在 ImageNet 上的良好性能通常意味着从普通照片中提取特征的坚实水平, 这为来自预先训练的 ImageNet 模型的各种迁移学习设置奠定了基础。



一个输入图像的例子, 类似的可以在 ImageNet 中找到, 代表“斑猫” (“tabby cat”) 类。有趣的是, VGG-16 体系结构有 16 个卷积层, 被作者称为“非常深”。随后的发展迅速将这种模型扩大到数百甚至数千层。

卷积神经网络在 ImageNet 上的成功——特别是 Krizhevsky et al. (2012) 的 AlexNet 模型, 该模型在很大程度上横扫了 ILSVRC 2012 在很大程度上引领了学术界和工业界对深度学习的整体采用。自那以后, CNNs 一直在 ILSVRC 中名列前茅, 产生了许多流行的架构, 如 VGG-16 (Simonyan and Zisserman, 2014)、Inception (Szegedy et al., 2015) 和 ResNets (He et al., 2016), 这些架构在这项任务中的性能已成功超过了人类水平。这些架构所采用的设计决策和正则化技术 (如校正线性激活 (Nair and Hinton, 2010)、dropout (Srivastava et al., 2014)、跳跃连接 (He et al., 2016) 和批量归一化 (Ioffe and Szegedy, 2015)) 构成了当今使用的许多有效 CNN 模型的主干。

在物体分类的同时, 物体检测也取得了重大进展; 也就是说, 隔离图像中所有感兴趣的对象, 并用特定的类标记它们。这一任务与各种下游问题相关,



从图像字幕一直到自动车辆。它需要一种更精细的方法，因为预测需要本地化；因此，通常情况下，平移等变模型已经证明了它们在这个领域的价值。这一领域一个有影响力的例子包括 R-CNN 模型家族 (Girshick et al., 2014; Girshick, 2015; Ren et al., 2015; He et al., 2017)，而在语义分割的相关领域，Badrinarayanan et al. (2017) 的 SegNet 模型被证明具有影响力，其编码器-解码器架构依赖于 VGG-16 主干。

## 6.7 博弈对抗

无论何时观察到的状态可以在网格域中表示，卷积神经网络在强化学习 (RL) 环境中作为平移不变特征提取器也发挥着突出的作用；例如，当从像素学习玩视频游戏时就是这种情况。在这种情况下，CNN 负责将输入减少到平面向量表示，然后用于导出驱动强化学习智能体行为的策略或值函数。虽然强化学习的细节不是本节的重点，但我们确实注意到，过去十年中深度学习的一些最有影响力的结果是通过 CNN 支持的强化学习产生的。

这里值得一提的一个特别的例子是 DeepMind 的 *AlphaGo* (Silver et al., 2016)。它通过将 CNN 应用于代表放置的石头的位置的  $19 \times 19$  网格来编码围棋游戏中的当前状态。然后，通过从以前的专家动作中学习、蒙特卡罗树搜索和自我游戏的结合，它成功地达到了围棋大师的水平，足以在一场在世界范围内广泛宣传的五轮挑战比赛中击败有史以来最强的围棋选手之一李世石 (Lee Sedol)。



围棋的游戏是在  $19 \times 19$  的棋盘上进行的，两个玩家在空旷的场地上放置白色和黑色的石头。据估计，合法态的数量约为  $\approx 2 \times 10^{170}$  (Tromp and Farnebäck, 2006)，远远超过宇宙中的原子数量。

虽然这已经代表了更广泛的人工智能的一个重要里程碑——围棋的状态空间比象棋复杂得多——但 AlphaGo 的发展并没有就此停止。作者逐渐从体系结构中移除了越来越多的围棋特定偏好，*AlphaGo Zero* 移除了人类偏好，纯粹通过自我游戏进行优化 (Silver et al., 2017)，*AlphaZero* 将该算法扩展到相关的双人游戏，如象棋和将棋；最后，*MuZero* (Schrittwieser et al., 2020) 采用了一种能够动态学习游戏规则模型，这种模型允许在 Atari 2600 控制台以及围棋、象棋和将棋中达到强大的性能，而无需任何规则的前期知识。在所有这些发展过程中，中枢神经系统仍然是这些模型表示输入的基础。

虽然多年来为 Atari2600 平台提出了几种高性能的强化学习智能体 (Mnih et al., 2015, 2016; Schulman et al., 2017), 在很长一段时间里, 他们无法在其中提供的 57 款游戏中达到人类水平的表现。这一障碍最终被 Agent57 (Badia et al., 2020) 打破, 该智能体使用了一系列参数化的策略, 从强烈的探索性到纯粹的剥削性, 并在不同的培训阶段以不同的方式对其进行优先排序。它 also 通过一个卷积神经网络为视频游戏的帧缓冲区提供大部分计算能力。

## 6.8 文本与语音合成

除了图像 (自然地映射到二维网格), 一些 (几何) 深度学习的最大成功发生在一维网格上。自然的例子是文本和语音, 在自然语言处理和数字信号处理等不同领域折叠几何深度学习蓝图。

这个领域中一些最广泛应用和宣传的作品侧重于合成: 能够无条件地或以特定提示为条件合成 (*synthesis*) 语音或文本。这种设置可以支持大量有用的任务, 如文本到语音 (TTS)、预测文本补全和机器翻译。在过去十年中, 已经提出了用于文本和语音生成的各种神经架构, 最初主要基于递归神经网络 (例如, 前述 seq2seq 模型 (Sutskever et al., 2014) 或递归注意力 (Bahdanau et al., 2014))。然而, 近年来, 它们已经逐渐被卷积神经网络和基于变压器的体系结构所取代。

在这种情况下, 简单 1D 卷积的一个特殊限制是它们线性增长的感受野 (*receptive field*), 需要许多层来覆盖到目前为止产生的序列。相反, 扩张卷积提供了一个指数增长的感受野, 具有相同数量的参数。因此, 它们被证明是一个非常强大的替代方案, 最终在机器翻译方面与 RNNs 竞争 (Kalchbrenner et al., 2016), 同时由于它们在所有输入位置的并行性, 大大降低了计算复杂性。最广为人知的扩张卷积应用是 van den Oord et al. (2016a) 的 WaveNet 模型。WaveNets 证明, 使用扩展有可能在原始波形 (通常每秒 16,000 个样本或更多) 的水平上合成语音, 产生比以前最好的文本到语音 (TTS) 系统更“像人”的语音样本。随后, 进一步证明了 WaveNets 的计算可以在一个

扩张卷积也被称为 *à trous* 卷积, 法语中字面意思是“有孔” (“*holed*”).

这种技术在蛋白质-蛋白质相互作用等各种问题上也优于 RNNs (Deac et al., 2019).

除此之外, WaveNet 模型被证明能够生成钢琴曲。

更简单的模型——*WaveRNN* (Kalchbrenner et al., 2018) 中进行提炼——并且该模型能够在工业规模上有效地部署该技术。这不仅允许其部署大规模的语音生成服务，如谷歌助手，还允许有效的设备上的计算；例如使用端到端加密的 Google Duo。

Transformers (Vaswani et al., 2017) 已经设法超越了递归和卷积架构的限制，表明自注意力 (*self-attention*) 足以实现机器翻译的最新性能。随后，它们彻底改变了自然语言处理。通过 BERT (Devlin et al., 2018) 等模型提供的预先训练的嵌入，Transformer 计算已经能够用于自然语言处理的大量下游应用——例如，谷歌使用 BERT 嵌入来为其搜索引擎提供动力。

可以说，在过去几年里，Transformer 最广泛的应用是文本生成，主要是由创成式预先训练的 Transformer (GPT, Radford et al. (2018, 2019); Brown et al. (2020)) 来自 OpenAI 的模型家族。特别是，GPT-3 (Brown et al., 2020) 成功地将语言模型学习缩放到 1750 亿个可学习参数，在刮擦的 (scraped) 文本语料库的网络规模的数量上训练下一个单词预测。这使得它不仅能够成为一个在各种基于语言的任务中非常有效的少样本学习器，而且能够生成连贯的、听起来像人的文本。这种能力不仅意味着大量的下游应用，还引发了广泛的媒体报道。

## 6.9 医疗保健

医学领域的应用是几何深度学习的另一个有前途的领域。这些方法有多种使用方式。首先，更多的传统架构 (如中枢神经系统) 已应用于网格结构的数据，例如，预测重症监护病房的住院时间 (Rocheteau et al., 2020)，或通过视网膜扫描诊断威胁视力的疾病 (De Fauw et al., 2018)。Winkels and Cohen (2019) 表明，与传统的中枢神经系统相比，使用 3D 旋转平移群卷积网络可以提高肺结节检测的准确性。

第二，将器官建模为几何表面，网格卷积神经网络被证明能够解决各种各样的任务，从遗传学相关信息重建面部结构 (Mahdi et al., 2020) 到大脑皮层



分组 (Cucurull et al., 2018) 到从皮层表面结构回归人口统计属性 (Besson et al., 2020)。后一个例子代表了神经科学中的一种增长趋势, 即认为大脑是一个具有复杂褶皱的表面, 从而产生了高度非欧几里得结构。

大脑皮层的这种结构在解剖学文献中被称为脑沟 (*sulci*) 和脑回 (*gyri*)。

与此同时, 神经科学家经常试图构建和分析大脑的功能网络 (*functional networks*), 这些功能网络代表大脑中在执行某些认知功能时被一起激活的各个区域; 这些网络通常使用功能性磁共振成像 (fMRI) 来构建, 该成像实时显示大脑的哪些区域消耗更多的血液。这些功能网络可以揭示患者的人口统计数据 (例如, 区分男性和女性, Arslan et al. (2018)), 并用于神经病理学诊断, 这是几何深度学习在医学中的第三个应用领域, 我们希望在此强调。在这种情况下, Ktena et al. (2017) 率先使用图形神经网络来预测神经疾病, 如自闭症谱系障碍。大脑的几何结构和功能结构似乎密切相关, 最近 Itani and Thanou (2021) 指出了在神经疾病分析中联合利用它们的好处。

典型地, 使用血氧水平依赖 (BOD) 对比成像。

第四, 患者网络 (*patient networks*) 在基于最大似然估计的医学诊断中变得越来越突出。这些方法背后的基本原理是, 患者人口统计学、基因型和表型相似性的信息可以改善对他们疾病的预测。Parisot et al. (2018) 将图形神经网络应用于根据人口统计学特征创建的患者网络, 用于神经疾病诊断, 表明图形的使用改善了预测结果。Cosmo et al. (2020) 展示了在这种情况下潜在图形学习 (网络通过它学习未知的患者图形) 的好处。后一项工作使用了英国生物银行的数据, 这是一个包括脑成像在内的大规模医学数据集合 (Miller et al., 2016)。

关于医院病人的大量数据可以在电子健康记录 (*electronic health records* (EHRs)) 中找到。除了给出患者病情进展的全面视图, EHR 分析还允许将相似的患者联系在一起。这与诊断中常用的模式识别方法 (*pattern recognition method*) 相一致。其中, 临床医生使用经验来识别临床特征的模式, 并且当临床医生的经验可以使他们快速诊断病情时, 这可能是使用的主要方法。沿着这些思路, 一些工作试图基于数据构建患者图, 或者通过分析他们的医生笔记的嵌入 (Malone et al., 2018), 入院时的诊断相似性 (Rocheteau et al., 2021), 或者甚至假设完全连通的图 (Zhu and Razavian, 2019)。在所有情况下, 有希望的结果已经显示出支持使用图形表示学习来处理 EHRs。

公开的匿名 EHR 重症监护数据集包括 MIMIC-III (Johnson et al., 2016) 和 eICU (Pollard et al., 2018)。

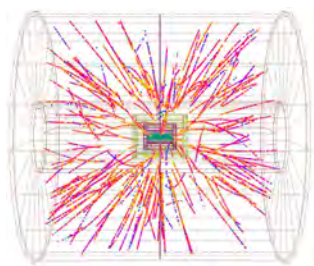
## 6.10 粒子物理和天体物理



大型强子对撞机探测器的一部分.

高能物理学家可能是自然科学领域第一批拥抱新的闪亮工具——图神经网络的领域专家之一。在最近的一篇综述论文中, [Shlomi et al. \(2020\)](#)指出, 机器学习在粒子物理实验中一直被大量使用, 要么学习复杂的反函数, 允许从探测器中测量的信息推断底层物理过程, 要么执行分类和回归任务。对于后者, 为了能够使用标准的深度学习架构, 如 CNN, 通常需要将数据强制转换为不自然的表示, 如格网 (grid)。然而, 物理学中的许多问题涉及到具有丰富关系和相互作用的无序集合形式的数据, 这些数据可以自然地表示为图形。

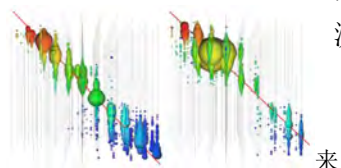
高能物理的一个重要应用是粒子射流 (*particle jets*) 的重建和分类——由多次连续相互作用产生的稳定粒子喷雾和由单个初始事件产生的粒子衰变。在欧洲粒子物理研究所建造的最大和最著名的粒子加速器大型哈顿对撞机中, 这种射流是质子以接近光速的速度碰撞的结果。这些碰撞产生了大量的粒子, 比如长粒子——希格斯玻色子 (Higgs boson) 或顶夸克 (top quark)。碰撞事件的识别和分类至关重要, 因为它可能为新粒子的存在提供实验证据。



粒子射流示例.

最近, 多种几何深度学习方法已经被提出用于粒子射流分类任务, 例如, 分别由 [Komiske et al. \(2019\)](#) 和 [Qu and Gouskos \(2019\)](#) 基于 DeepSet 和 Dynamic Graph CNN 架构。最近, 人们还对开发基于物理考虑的特殊架构感兴趣, 并结合符合哈密顿量或拉格朗日力学的归纳偏差 (例如, [Sanchez-Gonzalez et al. \(2019\)](#); [Cranmer et al. \(2020\)](#)), 洛伦兹群的等变 (物理学中空间和时间的基本对称性)([Bogatskiy et al., 2020](#)), 甚至结合符号推理 ([Cranmer et al., 2019](#)), 能够从数据中学习物理规律。这种方法更容易解释 (因此被领域专家认为更“可信”), 也提供了更好的泛化性。

除了粒子加速器之外, 粒子探测器现在正被天体物理学家用于多信使天文学 (*multi-messenger astronomy*)——一种协调观察来自同一来源的不同信号 (如电磁辐射、引力波和中微子) 的新方法。中微子天文学尤其令人感兴趣, 因为中微子很少与物质相互作用, 因此可以不受影响地传播很远的距离。探测中微子可以观察光学望远镜无法观察到的物体, 但需要非常大的探测器



来自背景事件 (muon 子束, 左) 和天体物理中微子 (高能单 muon 子, 右) 的 IceCube 探测器中光沉积 (light deposition) 的特征模式。

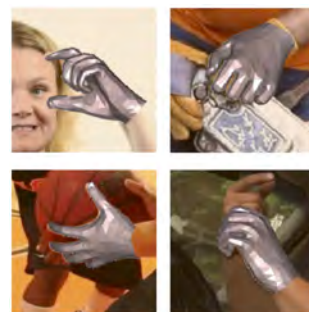
——冰立方中微子天文台使用南极一立方公里的南极冰架作为探测器。探测高能中微子可能会照亮宇宙中一些最神秘的物体，如黑洞和黑洞。Choma et al. (2018) 使用几何神经网络对冰立方中微子探测器的不规则几何形状进行建模，在探测来自天体物理源的中微子并将它们与背景事件分离方面显示出显著更好的性能。

虽然中微子天文学为宇宙研究提供了巨大的希望，但传统的光学和射电望远镜仍然是天文学家的“战马”。有了这些传统的工具，几何深度学习仍然可以为数据分析提供新的方法。例如，Scaife and Porter (2021) 使用旋转等变氯化萘对射电星系进行分类，McEwen et al. (2021) 使用球形氯化萘对宇宙微波背景辐射进行分析，这是大爆炸的遗迹，可能会揭示原始宇宙的形成。正如我们已经提到的，这种信号自然地在球体上表示，等变神经网络是研究它们的合适工具。

### 6.11 虚拟与增强现实

另一个应用领域是计算机视觉和图形学，特别是处理虚拟和增强现实的三维人体模型，它是开发一大类几何深度学习方法的动力。在《阿凡达》等电影中用于产生特殊效果的运动捕捉技术通常分两个阶段运行：首先，捕捉演员身体或面部运动的 3D 扫描仪的输入与一些典型形状（通常建模为离散流形或网格）对应（这个问题通常称为“分析”）。其次，生成一个新的形状来重复输入的运动（“合成”）。计算机图形学与视觉中的几何深度学习 (Masci et al., 2015; Boscaini et al., 2016a; Monti et al., 2017) 开发了网状卷积神经网络来解决分析问题，或者更具体地说，可变形形状对应。

Litany et al. (2018) 和 Ranjan et al. (2018) 独立提出了第一个用于 3D 形状合成的几何自动编码器架构。在这些体系结构中，假设（身体、面部或手的）标准网格是已知的，并且合成任务包括回归节点的 3D 坐标（表面的嵌入，使用微分几何的行话）。Kulon et al. (2020) 展示了一种用于 3D 手姿态估计的混合管道，该管道具有基于 CNN 的图像编码器和几何解码器。该系统的演示是与英国初创公司阿里埃勒人工智能 (Ariel AI) 合作开发的，并在

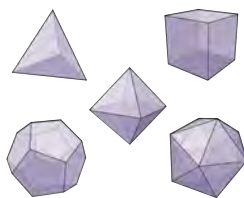


从野外 2D 图像重建的复杂 3D 手部姿势的示例 (Kulon et al., 2020).

CVPR 2020 上展示，它允许通过手机上的视频输入，以比实时更快的速度，用完全铰接的手创建逼真的身体化身。Ariel AI 于 2020 年被 Snap 收购，在撰写本文时，其技术用于 Snap 的增强现实产品。

## 7 历史视角

“对称, 无论你怎么定义它的含义, 它都是一种观念, 自古以来人类就试图通过它来理解和创造秩序、美和完美。”伟大数学家 Hermann 的同名著作中给出了这种有点诗意的对称定义 Weyl (2015), 他在普林斯顿高等研究院退休前夕去世。Weyl 将对称在科学和艺术中的特殊地位追溯到古代, 从苏美尔 (Sumerian) 的对称设计到毕达哥拉斯学派 (Pythagoreans), 他们相信圆是完美的, 因为它是旋转对称的。柏拉图认为今天以他的名字命名的五个正多面体是如此的基本, 以至于它们必须是塑造物质世界的基本构件。然而, 尽管柏拉图被认为创造了  $\sigma\upsilon\mu\mu\epsilon\tau\rho\acute{\iota}\alpha$  这个词, 字面意思是“同样的度量”, 但他只是模糊地用它来传达艺术中的比例美和音乐中的和谐美。天文学家和数学家约翰尼斯·开普勒首次尝试对水晶体的对称形状进行严格的分析。天文学家和数学家约翰尼斯·开普勒 (Johannes Kepler) 首次尝试对水晶体的对称形状进行严格的分析。在他的论文“六角雪花”中, 他将雪花的六重二面角结构归因于颗粒的六边形堆积——这一想法虽然先于对物质如何形成的清晰理解, 但今天仍然是结晶学的基础 (Ball, 2011)。



四面体、立方体、八面体、十二面体和二十面体被称为柏拉图多面体。

书名为“新年礼物”, 或“六角雪花”, 是开普勒在 1611 年作为圣诞礼物送给他的赞助人和朋友 Johannes Matthäus Wackher von Wackenfels 的小册子。

### 7.1 数学与物理中的对称

在现代数学中, 对称性几乎是用群论的语言来表达的。这一理论的起源通常归功于埃瓦里斯特·伽罗瓦 (Évariste Galois), 他在 19 世纪 30 年代创造了这个术语, 并用它来研究多项式方程的可解性。另外两个与群论有关的名字是索福思·李 (Sophus Lie) 和费利克斯·克莱因 (Felix Klein), 他们相遇并一起工作了一段时间, 并取得了丰硕的成果 (Tobies, 2019)。前者将发展至今以他的名字命名的连续对称理论; 后者在他的埃尔兰根纲领中宣称群论是几何学的组织原则, 我们在本文开头提到过。黎曼几何被明确排除在克莱因的统一几何图景之外, 又过了五十年才被整合, 这很大程度上要归功于 20 世纪 20 年代埃利·卡坦 (Élie Cartan) 的工作。

艾米·诺特 (Emmy Noether), 克莱因在哥廷根的同事, 证明了一个物理系统



的每一个可微的对称性都有相应的守恒定律 (Noether, 1918). 在物理学中, 这是一个惊人的结果: 在此之前, 需要细致的实验观察来发现能量守恒等基本定律, 即使在那时, 这也是一个来自任何地方的经验结果. 诺瑟定理——用诺贝尔奖得主弗兰克·维尔泽克 (Frank Wilczek) 的话说, 是“20 世纪和 21 世纪物理学的一颗指路之星”——表明能量守恒源于时间的平移对称性, 这是一个相当直观的想法, 即实验的结果不应取决于它是在今天还是明天进行.

1919 年, Weyl 第一次推测 (错误地) 尺度或“量规”变化下的不变性是电磁学的局部对称性. 量规一词, 或德语中的 *Eich*, 是通过类比铁路的各种量规来选择的. 在量子力学发展之后, Weyl (1929) 通过用波相的变化代替标度因子来修改量规选择. 见 Straumann (1996).

与电荷守恒相关的对称性 是电磁场的整体量规不变性, 首次出现在麦克斯韦的电动力学公式中 (Maxwell, 1865); 然而, 它的重要性最初并未引起注意. 同样是赫尔曼·韦尔 (Hermann Weyl) 如此犹豫不决地写下了对称性, 他是 20 世纪初第一个在物理学中引入量规不变性概念的人, 强调它作为一种可以导出电磁学的原理的作用. 直到几十年后, 这一基本原理——由杨和米尔斯 Yang and Mills (1954) 发展的广义形式——才被证明成功地提供了一个统一的框架来描述电磁学的量子力学行为以及强弱力, 最终形成了标准模型, 该模型捕捉了除重力之外的所有基本自然力. 因此, 我们可以与另一位诺贝尔奖获得者物理学家 Philip Anderson (1972) 一起得出结论, “说物理是对对称性的研究只是稍微夸大其词.”

## 7.2 机器学习中早期使用对称

在机器学习及其在模式识别和计算机视觉中的应用中, 对称性的重要性早已得到认可. 设计用于模式识别的等变特征检测器的早期工作是由 Amari (1978), Kanatani (2012), 和 Lenz (1990) 做的. 在神经网络文献中, Minsky and Papert (2017) 的著名感知器群不变性定理对 (单层) 感知器学习不变量的能力提出了基本限制. 这是研究多层架构的主要动机之一 (Sejnowski et al., 1986; Shawe-Taylor, 1989, 1993), 最终导致深度学习.

在神经网络社区中, Neocognitron (Fukushima and Miyake, 1982) 被认为是神经网络中“不受位置变化影响的模式识别”的移动不变性的第一个实现. 他的解决方案是以具有局部连通性的分层神经网络的形式出现的, 灵感来自二十年前神经科学家大卫·胡贝尔 (David Hubel) 和托尔斯滕·威塞尔

(Torsten Wiesel) 在视觉皮层中发现的感受野 (Hubel and Wiesel, 1959). 这些想法在 Yann LeCun 和合著者 (LeCun et al., 1998) 的开创性工作中的卷积神经网络中达到顶峰. Wood and Shawe-Taylor (1996) 完成了对不变和等变神经网络的表示理论观点的第一项工作, 不幸的是很少被引用. 这些思想的最新体现包括 Makadia et al. (2007); Esteves et al. (2020) 和本文作者之一 (Cohen and Welling, 2016).

这项经典的工作得到了 1981 年诺贝尔医学奖的认可, 由 Hubel, Wiesel 与 Roger Sperry 分享.

### 7.3 图神经网络

图神经网络的概念是如何开始出现的很难解释——部分是因为大多数早期的工作没有将图形作为一等公民, 部分是因为 GNNs 直到 2010 年代后期才变得实用, 部分是因为这个领域是从几个研究领域的融合中出现的. 也就是说, 图神经网络的早期形式至少可以追溯到 20 世纪 90 年代, 例子包括 Alessandro Sperduti's Labeling RAAM (Sperduti, 1994), Goller and Kuchler (1996) 的“通过结构的反向传播”, 和数据结构的自适应处理 (Sperduti and Starita, 1997; Frasconi et al., 1998). 虽然这些工作主要关注于对“结构”(通常是树或有向无环图)的操作, 但它们的架构中保留的许多不变性让人想起今天更常用的 GNNs.

第一次正确处理一般图形结构的过程 (以及“图神经网络”一词的诞生) 发生在 21 世纪之交. 在锡耶纳大学 (Università degli Studi di Siena) 的人工智能实验室里, 由 Marco Gori 和 Franco Scarselli 领导的论文提出了第一个“GNN” (Gori et al., 2005; Scarselli et al., 2008). 他们依赖循环机制, 需要神经网络参数来指定压缩映射, 从而通过搜索固定点来计算节点表示——这本身就需要一种特殊形式的反向传播 (Almeida, 1990; Pineda, 1988) 并且根本不依赖于节点特征. Li et al. (2015) 的门控模型纠正了上述所有问题. GNNs 带来了现代神经网络的许多好处, 如 (Cho et al., 2014) 和通过时间的反向传播, 到 GNN 模型, 并保持流行至今.

与此同时, 阿莱西奥·米歇尔提出了图神经网络 (*neural network for graphs*, NN4G) 模型, 该模型侧重于前馈 (*feedforward*) 而不是递归范式 (Micheli, 2009).

## 7.4 计算化学

同样非常重要的是要注意到 GNNs 的一条独立且并行的发展路线: 一条完全由计算化学需求驱动的路线, 其中分子最自然地表达为由化学键 (边) 连接的原子 (节点) 的图形. 这邀请了直接在这样的图形结构上操作的分子性质预测的计算技术, 它在 20 世纪 90 年代已经出现在机器学习中: 这包括 Kireev (1995) 的 ChemNet 模型 (1995) 和 Baskin et al. (1997) 的工作. 引人注目的是, Merkwirth and Lengauer (2005) 的“分子图网络”早在 2005 年就明确提出了许多在当代 GNNs 中常见的元素——例如边缘类型条件权重或全局汇集. 化学研究动机继续推动 GNN 发展到 2010 年代, GNN 的两项重大进展集中在改进分子指纹 (Duvenaud et al., 2015) 和预测小分子的量子化学性质 (Gilmer et al., 2017). 在撰写本文时, 分子性质预测是 GNNs 最成功的应用之一, 在新抗生素药物的虚拟筛选中产生了有影响力的结果 (Stokes et al., 2020).

## 7.5 节点嵌入

一些关于图的深度学习的早期成功故事涉及到基于图结构以无监督的方式学习节点的表示. 鉴于他们的结构灵感, 这个方向也提供了图形表示学习和网络科学社区之间最直接的联系之一. 这一领域的关键早期方法依赖于基于随机行走的嵌入: 学习节点表示, 如果节点在短时间内随机行走, 则使它们更接近. 这个空间中有代表性的方法有 DeepWalk (Perozzi et al., 2014)、node2vec (Grover and Leskovec, 2016) 和 LINE (Tang et al., 2015), 都是纯自监督的. Planetoid (Yang et al., 2016) 是第一个在这个空间纳入监管标签信息 (如果有).

最近, Srinivasan and Ribeiro (2019) 开发了一个理论框架, 其中证明了结构和位置表示的等效性. 此外, Qiu et al. (2018) 已经证明, 所有基于随机行走的嵌入技术相当于适当设定的矩阵分解任务. 曾多次尝试用 GNN 编码器统一随机行走目标, 有代表性的方法包括变分图自动编码器 (VGAE, Kipf and Welling (2016b))、嵌入传播 (García-Durán and Niepert, 2017) 和 GraphSAGE 的无监督变体 (Hamilton et al., 2017). 然而, 结果喜忧参半, 人们很快发现, 将相邻的节点表示推到一起已经是 GNNs 感应偏置的一个关键部分. 事实证明, 在节点功能可用的情况下, 未经训练的



GNN 已经表现出与 DeepWalk 相当的性能 (Veličković et al., 2019; Wu et al., 2019). 这开启了一个方向, 从将随机行走目标与 GNNs 相结合, 转向受互信息最大化启发的对比方法, 并与图像领域的成功方法相一致. 这方面的突出例子包括 Deep Graph Informax(DGI, Veličković et al. (2019))、GRACE(Zhu et al., 2020)、BERT 样目标 (Hu et al., 2020) 和 BGRL (Thakoor et al., 2021).

## 7.6 概率图模型

同时, 图神经网络也通过嵌入概率图模型的计算而复兴. 概念图模型 (PGMs, Wainwright and Jordan (2008)) 是处理图形数据的强大工具, 它们的效用来自于对图形边的概率: 也就是说, 节点被视为随机变量, 而图形结构编码条件独立假设, 允许显著简化联合分布的计算和采样. 事实上, 许多 (精确地或近似地) 支持 PGMs 上的学习和推理的算法依赖于通过它们的边传递消息的形式 (Pearl, 2014), 例子包括变分平均场推理和循环信念传播 (Yedidia et al., 2001; Murphy et al., 2013).

PGMs 和消息传递之间的这种联系随后被发展到体系结构中, 由 structure2vec(Dai et al., 2016) 的作者建立了早期的理论联系. 也就是说, 通过将图形表示学习设置提出为马尔可夫随机场 (对应于输入特征和潜在表示的节点), 作者直接将平均场推理和循环信念传播的计算与当今常用的 GNNs 模型进行了比较.

关键的“技巧”是使用分布的希尔伯特空间嵌入 (*Hilbert-space embeddings*), 它允许将 GNN 的潜在表示与概率模型保持的概率分布联系起来 (Smola et al., 2007). 给定  $\phi$ , 一个为特征  $\mathbf{x}$  适当选择的嵌入函数, 有可能嵌入它们的概率分布  $p(\mathbf{x})$  作为期望嵌入  $\mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})} \phi(\mathbf{x})$ . 这种对应允许我们执行类似 GNN 的计算, 知道由 GNN 计算的表示将总是对应于在节点特征上的某种概率分布的嵌入.

最终, structure2vec 模型本身是一个 GNN 架构, 它很容易被放在我们的框架中, 但是它的设置启发了一系列 GNN 架构, 这些架构更直接地结合了 PGMs 中的计算. 新兴的例子已经成功地将 GNNs 与条件随机场相结合 (Gao et al.,

2019; Spalević et al., 2020), 关系马尔科夫网络 (Qu et al., 2019) 和马尔科夫逻辑网 (Zhang et al., 2020).

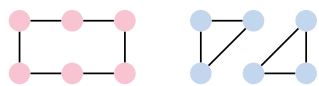
## 7.7 Weisfeiler-Lehman 形式化

图形神经网络的复兴紧随其后的是理解其基本局限性的努力, 特别是在表达能力方面. 虽然 GNNs 是一个强大的图形结构数据建模工具变得越来越明显, 但很明显, 它们不能完美地解决图形上指定的任何任务. 一个典型的例子是决定图的同构 (*graph isomorphism*): 我们的 GNN 能够给两个给定的非同构图附加不同的表示吗? 这是一个有用的框架, 原因有二. 如果 GNN 不能做到这一点, 那么它在任何需要区分这两个图形的任务上都是没有希望的. 此外, 目前还不知道决定图同构的是在 P 中, 这是所有 GNN 通常的计算复杂性.

由于它们的排列不变性, GNNs 将把相同的表示附加到两个同构的图上, 所以这种情况被简单地解决了.

目前最著名的决定图同构的算法是由 Babai and Luks (1983) 提出的, 尽管 Babai (2016) 最近的一个提议 (未被完全审查) 暗示了一个准多项式时间的解决方案.

将 GNNs 绑定到图同构的关键框架是 Weisfeiler-Lehman (WL) 图同构测试 (Weisfeiler and Leman, 1968). 该测试通过沿着图的边缘反复传递节点特征, 然后随机散列它们在邻近区域的总和, 来生成图表示. 与随机初始化的卷积神经网络的联系是显而易见的, 并且已经在早期观察到了: 例如, 在 Kipf and Welling (2016a) 的 GCN 模型中. 除了这种联系之外, WL 迭代以前是由 Shervashidze et al. (2011) 引入图核领域的, 并且它仍然为全图表示的无监督学习提供了强基线.



一个简单的例子: WL 测试无法区分 6 个环和 2 个三角形.

虽然 WL 测试在概念上很简单, 并且有许多它无法区分的非同构图的简单例子, 但它的表达能力最终与 GNNs 紧密相连. Morris et al. (2019) 和 Xu et al. (2018) 的分析都得出了一个惊人的结论: 任何符合我们在第 5.3 节中概述的三种口味之一的都不可能比 WL 测试更有效!

为了准确地达到这一表示水平, GNN 更新规则必须存在某些限制. Xu et al. (2018) 表明, 在离散特征域中, 使用的聚集函数必须是单射的 (*injective*), 而 *summation* 是一个关键表示. 基于他们的分析结果, Xu et al. (2018) 提出了图同构网络 (GIN), 它是在此框架下最大表达的一个简单而有力的例子.

最大化和平均化等流行的聚合器在这方面有所欠缺, 因为它们无法区分, 比如领域多集合  $\{\{a, b\}\}$  and  $\{\{a, a, b, b\}\}$ .

它也可以在我们提出的卷积 GNN 风味下表达.

最后, 值得注意的是, 这些发现不能推广到连续的节点特征空间. 事实上, 使用 Borsuk-Ulam 定理 (Borsuk, 1933), Corso et al. (2020) 已经证明, 假设实值节点特征, 获得单射聚合函数需要多个聚合器 (具体地, 等于接收节点的度). 他们的发现推动了主要邻域聚合 (PNA) 架构, 该架构提出了一个经验上强大且稳定的多聚合器 GNN.

这种聚合器的一个例子是多集邻居的时刻.

## 7.8 高阶方法

前面几段的发现并不与神经网络的实际效用相矛盾. 事实上, 在许多现实世界的应用中, 输入特征足够丰富, 以支持对图形结构的有用的区分计算, 尽管有上述限制.

相比之下, 几乎总是认为无特征或分类特征的图形.

然而, 一个关键的推论是, 神经网络在检测图形中的一些基本结构方面相对较弱. 在 WL 试验的具体限制或失败案例的指导下, 一些工作已经提供了比 WL 试验更强有力的 GNNs 变体, 因此可能对需要这种结构检测的任务有用.

一个突出的例子是计算化学, 其中分子的化学功能会受到分子图中芳香环的强烈影响.

或许寻找更具表现力的神经网络最直接的地方是 WL 测试本身. 事实上, 原始 WL 测试的强度可以通过考虑 WL 测试的层次结构来增强, 使得  $k$ -WL 测试将表示附加到节点的  $k$ -元组 (Morris et al., 2017). Morris et al. (2019) 已经将  $k$ -WL 测试直接转化为更高阶的  $k$ -GNN 架构, 这种架构比我们之前考虑的 GNN 风格更强大. 然而, 它维护元组表示的要求意味着, 实际上很难扩展到  $k = 3$  以上.

已经有一些努力, 例如  $\delta$ - $k$ -LGNN (Morris et al., 2020), 来稀疏化  $k$ -GNN 的计算.

同时, Maron et al. (2018, 2019) 研究了节点  $k$  元组上不变和等变图网络的特征. 除了证明任何不变的或等变的图形网络都可以表示为有限数量的生成器的线性组合的令人惊讶的结果——其数量仅取决于  $k$ ——作者还证明了这种层的表达能力相当于  $k$ -WL 测试, 并提出了一个经验上可扩展的变体, 该变体可证明是 3-WL 强大的.

除了对计算表示的领域进行概括之外, 还投入了大量的精力来分析 1-WL 的具体故障情况, 并增加 GNN 的输入, 以帮助他们区分这种情况. 一个常见的例子是将识别特征附加到节点, 这可以帮助检测结构. 这样做的建议包括独热 (*one-hot*) 表示 (Murphy et al., 2019), 以及纯随机特征 (Sato et al., 2020). 例如, 如果一个节点看到它自己的标识符  $k$  跳了, 它是一个直接的指示, 表明它在一个  $k$  环内. 更广泛地说, 已经有许多努力将结构信息结合到消息传递过程中, 或者通过调制消息函数或者通过传递计算的图形. 这里有几个有趣的工作包括对锚节点集 (*anchor node sets*) 进行采样 (You et al., 2019), 基于拉普拉斯特征向量 (*Laplacian eigenvectors*) 进行聚合 (Stachenfeld et al., 2020; Beaini et al., 2020; Dwivedi and Bresson, 2020), 或进行拓扑数据分析, 或用于位置嵌入 (Bouritsas et al., 2020) 或驱动信息传递 (Bodnar et al., 2021).

在计算化学领域, 通常认为分子功能是由子结构 (功能团) 驱动的, 这直接启发了分子在模体 (*motif*) 水平上的建模. 参考文献 Jin et al. (2018, 2020); Fey et al. (2020).

## 7.9 信号处理与调和分析

自从卷积神经网络的早期成功以来, 研究人员已经求助于谐波分析、图像处理和计算神经科学的工具, 试图提供一个理论框架来解释它们的效率. *M-theory* 是一个受视觉皮层启发的框架, 由托马索·波吉欧 (Tomaso Poggio) 及其合作者首创 (Riesenhuber and Poggio, 1999; Serre et al., 2007), 基于模板的概念, 可以在某些对称群下操作. 由计算神经科学产生的另一个值得注意的模型是可操纵金字塔, 这是一种多尺度小波分解的形式, 对某些输入变换具有有利的性质, 由 Simoncelli and Freeman (1995) 开发. 它们是早期纹理生成模型的核心元素 (Portilla and Simoncelli, 2000), 后来通过用深层 CNN 特征替换可控小波特征进行了改进 Gatys et al. (2015). 最后, 由 Stéphane Mallat (2012) 引入并由 Bruna and Mallat (2013) 开发的散射变换, 通过用多尺度小波分解代替可训练滤波器, 提供了一个理解中枢神经系统的框架, 也展示了变形稳定性和深度在体系结构中的作用.

### 7.10 图与网格上信号处理

图神经网络的另一个重要类别, 通常被称为谱 (*spectral*), 已经出现在本文作者之一的工作中 (Bruna et al., 2013), 使用了图傅立叶变换 (*Graph Fourier transform*) 的概念. 这种结构的根源在于信号处理和计算谐波分析领域, 在 2000 年代末和 2010 年代初, 处理非欧几里德信号变得非常突出. 来自 Pierre Vandergheynst (Shuman et al., 2013) 和 José Moura (Sandryhaila and Moura, 2013) 小组的有影响力的论文普及了“图信号处理”(GSP) 的概念和基于图形邻接和拉普拉斯矩阵的特征向量的傅立叶变换的推广. Defferrard et al. (2016) 和 Kipf and Welling (2016a) 的依赖于谱滤波器的图卷积神经网络是该领域引用最多的网络之一, 并且可能被认为是近年来重新点燃机器学习对图的兴趣的网络.

值得注意的是, 在计算机图形学和几何处理领域, 非欧调和与分析比图形信号处理至少早了十年. 我们可以追溯流形和网格上的谱滤波器到 Taubin et al. (1996) 的工作. 在 Karni and Gotsman (2000) 关于谱几何压缩和 Lévy (2006) 关于使用拉普拉斯特征向量作为非欧几里德傅立叶基础的有影响力的论文之后, 这些方法在 2000 年代成为主流. 谱方法已被广泛应用 其中最突出的是构造形状描述符 (Sun et al., 2009) 和函数映射 (functional maps) (Ovsjanikov et al., 2012); 在撰写本文时, 这些方法仍广泛用于计算机图形学.

Roe Litman 和 Alex Bronstein (2013) 提出了类似于谱图 CNNs 的可学习形状描述符, 后者是本文作者的孪生兄弟.

### 7.11 计算机图形学与几何处理

基于内在度量不变量的形状分析模型由计算机图形和几何处理领域的不同作者引入 (Elad and Kimmel, 2003; Mémoli and Sapiro, 2005; Bronstein et al., 2006), 并在其中一个作者的早期著作 (Bronstein et al., 2008) 中得到深入讨论. Raviv et al. (2007); Ovsjanikov et al. (2008) 在同一领域也探讨了内在对称性的概念. 网格深度学习的第一个体系结构是测地线 CNNs, 是由 (Masci et al., 2015) 该文的一位作者的团队开发的. 该模型使用具有共享权重的局部滤波器, 应用于测地线径向面片. 这是后来由该文本的另一位作者 (Cohen et al., 2019) 开发的量规等变 CNNs 的一种特例. Federico Monti

et al. (2017) 在同一团队中提出了一种具有可学习聚合操作的测地线 CNNs 的一般化方法 MoNet, 该方法在网格的局部结构特征上使用了一种类似注意力的机制, 这种机制在一般的图形上也同样有效. 图形注意网络 (GAT) 从技术上来说可以被认为是 MoNet 的一个特定实例, 它是由本文的另一位作者引入的 (Veličković et al., 2018). GATs 囊括了 MoNet 的注意力机制, 也纳入了节点特征信息, 脱离了先前工作的纯粹结构衍生的相关性. 它是目前使用的最受欢迎的 GNN 架构之一.

在计算机图形学的背景下, 还值得一提的是, 在集合上学习的想法 (Zaheer et al., 2017) 是由斯坦福大学的 Leo Guibas 团队以 PointNet (Qi et al., 2017) 的名义同时开发的, 用于分析 3D 点云. 这种架构导致了多个后续作品, 包括本文作者之一的动态图形 CNN (DGCNN, Wang et al. (2019b)). DGCNN 使用最近邻图来捕捉点云的局部结构, 以便在节点之间交换信息; 这种体系结构的关键特征是图形是动态构建的, 并在与下游任务相关的神经网络层之间进行更新. 后一个特性使 DGCNN 成为“潜在图形学习”的第一个体现, 这反过来又产生了重大的后续影响. 扩展到 DGCNN 的  $k$ -近邻图建议包括通过双层优化 (Franceschi et al., 2019)、强化学习 (Kazi et al., 2020) 或直接监督 (Veličković et al., 2020) 对这些图的边进行更明确的控制. 独立地, 通过 NRI 模型出现了一个变化的方向 (从计算的后验分布中概率性地采样边)(Kipf et al., 2018). 虽然它仍然依赖于节点数量的二次计算, 但它允许对所选边的不确定性进行显式编码.

另一个非常流行的方向是在没有提供图形的情况下学习图形, 它依赖于在一个完全图形上执行 GNN 风格的计算, 让网络推断自己的方式来利用连通性. 这种需求尤其出现在自然语言处理中, 在自然语言处理中, 句子中的各种单词以非常不寻常和不连续的方式相互作用. 对一个完整的单词图进行操作带来了 Transformer 模型的第一个化身 (Vaswani et al., 2017), 它超越了递归和卷积模型, 成为神经机器翻译的最新技术, 并引发了相关工作的雪崩, 超越了自然语言处理和其他领域的界限. 全连通 GNN 计算也同时出现在模拟 (Battaglia et al., 2016)、推理 (Santoro et al., 2017) 和多智能体 (Hoshen, 2017) 应用中, 并且当节点数量相当少时仍然是一种流行的选择.



## 7.12 算法推理

在本节提出的大部分讨论中, 我们给出了空间诱导几何的例子, 这些例子反过来又塑造了底层的定义域, 以及它的不变性和对称性. 然而, 大量不变性和对称性的例子也出现在计算环境中. 许多常见场景几何深度学习的一个关键区别是, 链接不再需要为任何类型的相似性、邻近性或关系类型进行编码——它们只是为它们连接的数据点之间的数据流指定“配方”.

相反, 神经网络的计算模拟算法的推理过程 (Cormen et al., 2009), 由算法的控制流和中间结果引起的附加不变性. 在算法空间中, 假设的输入不变量通常被称为前置条件, 而算法保留的不变量被称为后置条件.

例如, Bellman-Ford 寻路算法 (Bellman, 1958) 的一个不变量是, 在  $k$  步之后, 它将总是计算到使用不超过  $k$  条边的源节点的最短路径.

同名, 算法推理 (*algorithmic reasoning*) 的研究方向 (Cappart et al., 2021, 第 3.3 节) 是寻求产生适当保存算法不变量的神经网络体系结构. 该领域研究了通用神经计算机的构造, 例如神经测试机 (*neural Turing machine*) (Graves et al., 2014) 和可微分神经计算机 (*differentiable neural computer*) (Graves et al., 2016). 虽然这种架构具有一般计算的所有特征, 但它们同时引入了几个组件, 这使得它们经常难以优化, 并且在实践中, 它们几乎总是被简单的关系推理器超越, 例如 Santoro et al. (2017, 2018) 提出的推理器.

由于模拟复杂的后条件具有挑战性, 关于学习执行的归纳偏置的大量工作 (Zaremba and Sutskever, 2014) 集中在原始算法上 (例如简单的算术). 这一领域的突出例子包括神经 GPU (*neural GPU*) (Kaiser and Sutskever, 2015)、神经 RAM (*neural RAM*) (Kurach et al., 2015)、神经程序解释器 (*neural programmer-interpreters*) (Reed and De Freitas, 2015)、神经算术-逻辑单元 (*neural arithmetic-logic units*) (Trask et al., 2018; Madsen and Johansen, 2020) 和神经执行引擎 (*neural execution engines*) (Yan et al., 2020).

随着 GNN 体系结构的迅速发展, 模拟超线性 (*superlinear*) 复杂度的组合算法成为可能. Xu et al. (2019) 首创的算法对齐框架从理论上证明了 GNNs 与动态规划 (Bellman, 1966) 对齐, 动态规划是一种可以表达大多数算法的语言. 本文的一位作者同时从经验上表明, 设计和训练符合实践中算法不变量

的 GNNs 是可能的 (Veličković et al., 2019). 此后, 通过迭代算法 (*iterative algorithms* (Tang et al., 2020))、线性算法 (*linearithmic algorithms* (Freivalds et al., 2019))、数据结构 (*data structures* (Veličković et al., 2020)) 和持久存储器 (*persistent memory* (Strathmann et al., 2021)) 实现了对齐. 这种模型在隐式规划器 (*implicit planners*) 中也有实际应用 (Deac et al., 2020), 突破了强化学习算法的空间.

同时, 在将神经网络用于物理模拟 (*physics simulations*) 方面取得了重大进展 (Sanchez-Gonzalez et al., 2020; Pfaff et al., 2020). 这一方向为广义神经网络的设计提供了许多相同的建议. 这种对应是意料之中的: 给定算法可以被表述为离散时间模拟, 并且模拟通常被实现为逐步算法, 两个方向都需要保持相似种类的不变量.

与算法推理研究紧密相连的是外推法 (*extrapolation*). 对于神经网络来说, 这是一个臭名昭著的痛点, 因为它们的大部分成功故事都是在推广内分布时获得的; 即当在训练数据中发现的模式正确地预测了在测试数据中发现的模式时. 然而, 算法不变量必须被保留, 而与例如输入的大小或生成分布无关, 这意味着训练集可能不会覆盖实践中遇到的任何可能的场景. Xu et al. (2020b) 提出了一个几何论证, 证明了整流器激活对外推的要求: 需要对其组件和特征进行设计, 以使其组成模块 (如消息函数) 仅学习线性目标函数. Bevilacqua et al. (2021) 提出在因果推理的透镜下观察外推, 产生环境不变的图形表示.

### 7.13 几何深度学习

我们最后的历史评论是关于这篇文章的名字. “几何深度学习”一词最早是由本文的作者之一在 2015 年的 ERC 授权中提出的, 并在同名的《电气和电子工程师协会信号处理杂志》论文中得到推广 (Bronstein et al., 2017). 这篇论文宣告了“一个新领域正在诞生”的迹象, 尽管“有些谨慎”鉴于最近图形神经网络的流行, 不变性和等方差思想在广泛的机器学习应用中的日益使用, 以及我们写这篇文章的事实, 认为这个预言至少部分实现可能是正确的. “4G: 格网、图形、群和规划”这个名字是由马克斯·韦林 (Max Welling) 为



ELLIS 几何深度学习项目创造的, 该项目由本文的两位作者共同领衔. 诚然, 最后一个“G”有点牵强, 因为底层结构是流形 (manifolds) 和丛 (bundles), 而不是规划 (gauges). 对于这篇文章, 我们添加了另一个“G”, 测地线, 参考度量不变量和流形的内在对称性.

## 致谢

本文通过不变性和对称性的几何透镜, 尝试总结和融合深度学习架构中几十年现有知识. 希望我们的观点将使新人和从业者更容易统揽这个领域, 使研究人员更容易融合新的架构, 作为我们蓝图的实例. 在某种程度上, 我们希望已经展示了“构建您所需要的架构所需的一切”——一部受启发的文字剧 [Vaswani et al. \(2017\)](#).

正文的大部分是在 2020 年末和 2021 年初写的. 正如经常发生的那样, 我们对整本书是否有意义有成千上万的怀疑, 并利用同事提供的机会帮助我们打破“怯场”, 展示我们工作的早期版本, 这在 Petar 在剑桥的演讲 (由 Pietro Liò 提供) 和迈克尔在牛津 (由 Xiaowen Dong 提供) 和帝国理工学院 (由 Michael Huth 和 Daniel Rueckert 主持) 的演讲中看到了曙光. Petar 还在“埃尔兰根纲领”的发源地——弗里德里希-亚历山大-埃尔兰根-纽伦堡大学 (Friedrich-Alexander-Universität Erlangen-Nürnberg) 展示了我们的作品!——感谢 Andreas Maier 的盛情邀请. 我们从这些会谈中得到的反馈对保持我们的高昂情绪以及进一步完善工作非常宝贵. 最后, 但同样重要的是, 我们感谢 ICLR 2021 的组委会, 我们的工作将在一个由迈克尔发表的主题演讲中被突出.

我们应该注意到, 协调如此大量的研究很少是由仅仅四个人的专业知识实现的. 因此, 我们要对所有认真研究了我们的案文的各个方面的研究人员给予应有的赞扬, 他们为我们提供了仔细的评论和参考资料: Yoshua Bengio, Charles Blundell, Andreea Deac, Fabian Fuchs, Francesco di Giovanni, Marco Gori, Raia Hadsell, Will Hamilton, Maksym Korablyov, Christian Merkwirth, Razvan Pascanu, Bruno Ribeiro, Anna Scaife, Jürgen Schmidhuber, Marwin Segler, Corentin Tallec, Ngan Vũ, Peter Wirnsberger and David Wong. 他们的专家式反馈对巩固我们的统一努力非常宝贵. 当然, 本文中的任何违规行为都是我们的责任. 这是一项正在进行的工作, 我们很高兴在任何阶段收到评论. 如果您发现任何错误或遗漏, 请联系我们.

## 参考文献

- Yonathan Aflalo and Ron Kimmel. Spectral multidimensional scaling. *PNAS*, 110(45):18052–18057, 2013.
- Yonathan Aflalo, Haim Brezis, and Ron Kimmel. On the optimality of shape and data representation in the spectral domain. *SIAM J. Imaging Sciences*, 8(2):1141–1160, 2015.
- Luis B Almeida. A learning rule for asynchronous perceptrons with feedback in a combinatorial environment. In *Artificial neural networks: concept learning*, pages 102–111. 1990.
- Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. *arXiv:2006.05205*, 2020.
- Sl Amari. Feature spaces which admit and detect invariant signal transformations. In *Joint Conference on Pattern Recognition*, 1978.
- Philip W Anderson. More is different. *Science*, 177(4047):393–396, 1972.
- Mathieu Andreux, Emanuele Rodola, Mathieu Aubry, and Daniel Cremers. Anisotropic Laplace-Beltrami operators for shape analysis. In *ECCV*, 2014.
- Salim Arslan, Sofia Ira Ktena, Ben Glocker, and Daniel Rueckert. Graph saliency maps through spectral convolutional networks: Application to

sex classification with brain connectivity. In *Graphs in Biomedical Image Analysis and Integrating Medical Imaging and Non-Imaging Modalities*, pages 3–13. 2018.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv:1607.06450*, 2016.

László Babai. Graph isomorphism in quasipolynomial time. In *ACM Symposium on Theory of Computing*, 2016.

László Babai and Eugene M Luks. Canonical labeling of graphs. In *ACM Symposium on Theory of computing*, 1983.

Francis Bach. Breaking the curse of dimensionality with convex neural networks. *JMLR*, 18(1):629–681, 2017.

Adrià Puigdomènech Badia, Bilal Piot, Steven Kapturowski, Pablo Sprechmann, Alex Vitvitskyi, Zhaohan Daniel Guo, and Charles Blundell. Agent57: Outperforming the atari human benchmark. In *ICML*, 2020.

Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *Trans. PAMI*, 39(12):2481–2495, 2017.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv:1409.0473*, 2014.

Philip Ball. In retrospect: On the six-cornered snowflake. *Nature*, 480(7378):455–455, 2011.

Bassam Bamieh. Discovering transforms: A tutorial on circulant matrices, circular convolution, and the discrete fourier transform. *arXiv:1805.05533*, 2018.

- Stefan Banach. Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales. *Fundamenta Mathematicae*, 3(1):133–181, 1922.
- Albert-László Barabási, Natali Gulbahce, and Joseph Loscalzo. Network medicine: a network-based approach to human disease. *Nature Reviews Genetics*, 12(1):56–68, 2011.
- Andrew R Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Trans. Information Theory*, 39(3):930–945, 1993.
- Igor I Baskin, Vladimir A Palyulin, and Nikolai S Zefirov. A neural device for searching direct correlations between structures and properties of chemical compounds. *J. Chemical Information and Computer Sciences*, 37(4):715–721, 1997.
- Peter W Battaglia, Razvan Pascanu, Matthew Lai, Danilo Rezende, and Koray Kavukcuoglu. Interaction networks for learning about objects, relations and physics. *arXiv:1612.00222*, 2016.
- Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv:1806.01261*, 2018.
- Dominique Beaini, Saro Passaro, Vincent Létourneau, William L Hamilton, Gabriele Corso, and Pietro Liò. Directional graph networks. *arXiv:2010.02863*, 2020.
- Richard Bellman. On a routing problem. *Quarterly of Applied Mathematics*, 16(1):87–90, 1958.
- Richard Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.

- Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. Neural Networks*, 5(2):157–166, 1994.
- Marcel Berger. *A panoramic view of Riemannian geometry*. Springer, 2012.
- Pierre Besson, Todd Parrish, Aggelos K Katsaggelos, and S Kathleen Bandt. Geometric deep learning on brain shape predicts sex and age. *BioRxiv:177543*, 2020.
- Beatrice Bevilacqua, Yangze Zhou, and Bruno Ribeiro. Size-invariant graph representations for graph classification extrapolations. *arXiv:2103.05045*, 2021.
- Guy Blanc, Neha Gupta, Gregory Valiant, and Paul Valiant. Implicit regularization for deep neural networks driven by an ornstein-uhlenbeck like process. In *COLT*, 2020.
- Cristian Bodnar, Fabrizio Frasca, Yu Guang Wang, Nina Otter, Guido Montúfar, Pietro Liò, and Michael Bronstein. Weisfeiler and lehman go topological: Message passing simplicial networks. *arXiv:2103.03212*, 2021.
- Alexander Bogatskiy, Brandon Anderson, Jan Offermann, Marwah Roussi, David Miller, and Risi Kondor. Lorentz group equivariant neural network for particle physics. In *ICML*, 2020.
- Karol Borsuk. Drei sätze über die n-dimensionale euklidische sphäre. *Fundamenta Mathematicae*, 20(1):177–190, 1933.
- Davide Boscaini, Davide Eynard, Drosos Kourounis, and Michael M Bronstein. Shape-from-operator: Recovering shapes from intrinsic operators. *Computer Graphics Forum*, 34(2):265–274, 2015.
- Davide Boscaini, Jonathan Masci, Emanuele Rodoià, and Michael Bronstein. Learning shape correspondence with anisotropic convolutional neural networks. In *NIPS*, 2016a.

- Davide Boscaini, Jonathan Masci, Emanuele Rodolà, Michael M Bronstein, and Daniel Cremers. Anisotropic diffusion descriptors. *Computer Graphics Forum*, 35(2):431–441, 2016b.
- Sébastien Bogleux, Luc Brun, Vincenzo Carletti, Pasquale Foggia, Benoit Gaüzere, and Mario Vento. A quadratic assignment formulation of the graph edit distance. *arXiv:1512.07494*, 2015.
- Giorgos Bouritsas, Fabrizio Frasca, Stefanos Zafeiriou, and Michael M Bronstein. Improving graph neural network expressivity via subgraph isomorphism counting. *arXiv:2006.09252*, 2020.
- Alexander M Bronstein, Michael M Bronstein, and Ron Kimmel. Generalized multidimensional scaling: a framework for isometry-invariant partial surface matching. *PNAS*, 103(5):1168–1172, 2006.
- Alexander M Bronstein, Michael M Bronstein, and Ron Kimmel. *Numerical geometry of non-rigid shapes*. Springer, 2008.
- Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond Euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.
- Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv:2005.14165*, 2020.
- Joan Bruna and Stéphane Mallat. Invariant scattering convolution networks. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1872–1886, 2013.
- Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. In *ICLR*, 2013.

- Quentin Cappart, Didier Chételat, Elias Khalil, Andrea Lodi, Christopher Morris, and Petar Veličković. Combinatorial optimization and reasoning with graph neural networks. *arXiv:2102.09544*, 2021.
- Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. *arXiv:1806.07366*, 2018.
- Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020.
- Albert Chern, Felix Knöppel, Ulrich Pinkall, and Peter Schröder. Shape from metric. *ACM Trans. Graphics*, 37(4):1–17, 2018.
- Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv:1406.1078*, 2014.
- Nicholas Choma, Federico Monti, Lisa Gerhardt, Tomasz Palczewski, Zahra Ronaghi, Prabhat Prabhat, Wahid Bhimji, Michael M Bronstein, Spencer R Klein, and Joan Bruna. Graph neural networks for icecube signal classification. In *ICMLA*, 2018.
- Taco Cohen and Max Welling. Group equivariant convolutional networks. In *ICML*, 2016.
- Taco Cohen, Maurice Weiler, Berkay Kicanaoglu, and Max Welling. Gauge equivariant convolutional networks and the icosahedral CNN. In *ICML*, 2019.
- Taco S Cohen, Mario Geiger, Jonas Köhler, and Max Welling. Spherical cnns. *arXiv:1801.10130*, 2018.
- Tim Cooijmans, Nicolas Ballas, César Laurent, Çağlar Gülçehre, and Aaron Courville. Recurrent batch normalization. *arXiv:1603.09025*, 2016.



- Etienne Corman, Justin Solomon, Mirela Ben-Chen, Leonidas Guibas, and Maks Ovsjanikov. Functional characterization of intrinsic and extrinsic geometry. *ACM Trans. Graphics*, 36(2):1–17, 2017.
- Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2009.
- Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal neighbourhood aggregation for graph nets. *arXiv:2004.05718*, 2020.
- Luca Cosmo, Anees Kazi, Seyed-Ahmad Ahmadi, Nassir Navab, and Michael Bronstein. Latent-graph learning for disease prediction. In *MICCAI*, 2020.
- Miles Cranmer, Sam Greydanus, Stephan Hoyer, Peter Battaglia, David Spergel, and Shirley Ho. Lagrangian neural networks. *arXiv:2003.04630*, 2020.
- Miles D Cranmer, Rui Xu, Peter Battaglia, and Shirley Ho. Learning symbolic physics with graph networks. *arXiv:1909.05862*, 2019.
- Guillem Cucurull, Konrad Wagstyl, Arantxa Casanova, Petar Veličković, Estrid Jakobsen, Michal Drozdal, Adriana Romero, Alan Evans, and Yoshua Bengio. Convolutional neural networks for mesh-based parcellation of the cerebral cortex. 2018.
- George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, 1989.
- Hanjun Dai, Bo Dai, and Le Song. Discriminative embeddings of latent variable models for structured data. In *ICML*, 2016.
- Jeffrey De Fauw, Joseph R Ledsam, Bernardino Romera-Paredes, Stanislav Nikolov, Nenad Tomasev, Sam Blackwell, Harry Askham, Xavier Glorot, Brendan O’ Donoghue, Daniel Visentin, et al. Clinically applicable deep

learning for diagnosis and referral in retinal disease. *Nature Medicine*, 24(9):1342–1350, 2018.

Pim de Haan, Maurice Weiler, Taco Cohen, and Max Welling. Gauge equivariant mesh CNNs: Anisotropic convolutions on geometric graphs. In *NeurIPS*, 2020.

Andreea Deac, Petar Veličković, and Pietro Sormanni. Attentive cross-modal paratope prediction. *Journal of Computational Biology*, 26(6):536–545, 2019.

Andreea Deac, Petar Veličković, Ognjen Milinković, Pierre-Luc Bacon, Jian Tang, and Mladen Nikolić. Xlvin: executed latent value iteration nets. *arXiv:2010.13146*, 2020.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *NIPS*, 2016.

Austin Derrow-Pinion, Jennifer She, David Wong, Oliver Lange, Todd Hester, Luis Perez, Marc Nunkesser, Seongjae Lee, Xueying Guo, Peter W Battaglia, Vishal Gupta, Ang Li, Zhongwen Xu, Alvaro Sanchez-Gonzalez, Yujia Li, and Petar Veličković. Traffic Prediction with Graph Neural Networks in Google Maps. 2021.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv:1810.04805*, 2018.

David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. *NIPS*, 2015.

- Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. *arXiv:2012.09699*, 2020.
- Asi Elad and Ron Kimmel. On bending invariant signatures for surfaces. *Trans. PAMI*, 25(10):1285–1295, 2003.
- Jeffrey L Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.
- Carlos Esteves, Ameesh Makadia, and Kostas Daniilidis. Spin-weighted spherical CNNs. *arXiv:2006.10731*, 2020.
- Xiaomin Fang, Jizhou Huang, Fan Wang, Lingke Zeng, Haijin Liang, and Haifeng Wang. ConSTGAT: Contextual spatial-temporal graph attention network for travel time estimation at baidu maps. In *KDD*, 2020.
- Matthias Fey, Jan-Gin Yuen, and Frank Weichert. Hierarchical inter-message passing for learning on molecular graphs. *arXiv:2006.12179*, 2020.
- Marc Finzi, Samuel Stanton, Pavel Izmailov, and Andrew Gordon Wilson. Generalizing convolutional neural networks for equivariance to lie groups on arbitrary continuous data. In *ICML*, 2020.
- Jon Folkman. Regular line-symmetric graphs. *Journal of Combinatorial Theory*, 3(3):215–232, 1967.
- Luca Franceschi, Mathias Niepert, Massimiliano Pontil, and Xiao He. Learning discrete structures for graph neural networks. In *ICML*, 2019.
- Paolo Frasconi, Marco Gori, and Alessandro Sperduti. A general framework for adaptive processing of data structures. *IEEE Trans. Neural Networks*, 9(5):768–786, 1998.
- Kārlis Freivalds, Emīls Ozoliņš, and Agris Šostaks. Neural shuffle-exchange networks—sequence processing in  $o(n \log n)$  time. *arXiv:1907.07897*, 2019.

- Fabian B Fuchs, Daniel E Worrall, Volker Fischer, and Max Welling. SE(3)-transformers: 3D roto-translation equivariant attention networks. *arXiv:2006.10503*, 2020.
- Kunihiko Fukushima and Sei Miyake. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and Cooperation in Neural Nets*, pages 267–285. Springer, 1982.
- Pablo Gainza, Freyr Sverrisson, Federico Monti, Emanuele Rodola, D Boscaini, MM Bronstein, and BE Correia. Deciphering interaction fingerprints from protein molecular surfaces using geometric deep learning. *Nature Methods*, 17(2):184–192, 2020.
- Fernando Gama, Alejandro Ribeiro, and Joan Bruna. Diffusion scattering transforms on graphs. In *ICLR*, 2019.
- Fernando Gama, Joan Bruna, and Alejandro Ribeiro. Stability properties of graph neural networks. *IEEE Trans. Signal Processing*, 68:5680–5695, 2020.
- Hongchang Gao, Jian Pei, and Heng Huang. Conditional random field enhanced graph convolutional neural networks. In *KDD*, 2019.
- Alberto García-Durán and Mathias Niepert. Learning graph representations with embedding propagation. *arXiv:1710.03059*, 2017.
- Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Texture synthesis using convolutional neural networks. *arXiv preprint arXiv:1505.07376*, 2015.
- Thomas Gaudelot, Ben Day, Arian R Jamasb, Jyothish Soman, Cristian Regep, Gertrude Liu, Jeremy BR Hayter, Richard Vickers, Charles Roberts, Jian Tang, et al. Utilising graph machine learning within drug discovery and development. *arXiv:2012.05716*, 2020.

- Felix A Gers and Jürgen Schmidhuber. Recurrent nets that time and count. In *IJCNN*, 2000.
- Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. *arXiv:1704.01212*, 2017.
- Ross Girshick. Fast R-CNN. In *CVPR*, 2015.
- Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.
- Vladimir Gligoričević, P Douglas Renfrew, Tomasz Kosciółek, Julia Koehler Leman, Daniel Berenberg, Tommi Vatanen, Chris Chandler, Bryn C Taylor, Ian M Fisk, Hera Vlamakis, et al. Structure-based function prediction using graph convolutional networks. *bioRxiv:786236*, 2020.
- Christoph Goller and Andreas Kuchler. Learning task-dependent distributed representations by backpropagation through structure. In *ICNN*, 1996.
- Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *arXiv:1406.2661*, 2014.
- Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *IJCNN*, 2005.
- Alex Graves. Generating sequences with recurrent neural networks. *arXiv:1308.0850*, 2013.
- Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv:1410.5401*, 2014.
- Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Ed-

ward Grefenstette, Tiago Ramalho, John Agapiou, et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471–476, 2016.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv:2006.07733*, 2020.

Mikhael Gromov. *Structures métriques pour les variétés riemanniennes*. Cedric, 1981.

Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *KDD*, 2016.

Suriya Gunasekar, Blake E Woodworth, Srinadh Bhojanapalli, Behnam Neyshabur, and Nati Srebro. Implicit regularization in matrix factorization. In *NIPS*, 2017.

Deisy Morselli Gysi, Ítalo Do Valle, Marinka Zitnik, Asher Ameli, Xiao Gan, Onur Varol, Helia Sanchez, Rebecca Marlene Baron, Dina Ghiassian, Joseph Loscalzo, et al. Network medicine framework for identifying drug repurposing opportunities for COVID-19. *arXiv:2004.07229*, 2020.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NIPS*, 2017.

Junheng Hao, Tong Zhao, Jin Li, Xin Luna Dong, Christos Faloutsos, Yizhou Sun, and Wei Wang. P-companion: A principled framework for diversified complementary product recommendation. In *Information & Knowledge Management*, 2020.

Moritz Hardt and Tengyu Ma. Identity matters in deep learning. *arXiv:1611.04231*, 2016.

- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.
- Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *CVPR*, 2017.
- Claude Adrien Helvétius. *De l'esprit*. Durand, 1759.
- R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In *ICLR*, 2019.
- Sepp Hochreiter. *Untersuchungen zu dynamischen neuronalen Netzen*. PhD thesis, Technische Universität München, 1991.
- Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.
- Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991.
- Yedid Hoshen. Vain: Attentional multi-agent predictive modeling. *arXiv:1706.06122*, 2017.
- Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. In *ICLR*, 2020.
- David H Hubel and Torsten N Wiesel. Receptive fields of single neurones in the cat's striate cortex. *J. Physiology*, 148(3):574–591, 1959.
- Michael Hutchinson, Charline Le Lan, Sheheryar Zaidi, Emilien Dupont, Yee Whye Teh, and Hyunjik Kim. LieTransformer: Equivariant self-attention for Lie groups. *arXiv:2012.10885*, 2020.

- Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.
- Haris Iqbal. Harisqbal88/plotneuralnet v1.0.0, December 2018. URL <https://doi.org/10.5281/zenodo.2526396>.
- Sarah Itani and Dorina Thanou. Combining anatomical and functional networks for neuropathology identification: A case study on autism spectrum disorder. *Medical Image Analysis*, 69:101986, 2021.
- Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. In *ICML*, 2018.
- Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Hierarchical generation of molecular graphs using structural motifs. In *ICML*, 2020.
- Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-Wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. *Scientific Data*, 3(1):1–9, 2016.
- Michael I Jordan. Serial order: A parallel distributed processing approach. In *Advances in Psychology*, volume 121, pages 471–495. 1997.
- Chaitanya Joshi. Transformers are graph neural networks. *The Gradient*, 2020.
- Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. An empirical exploration of recurrent network architectures. In *ICML*, 2015.
- Łukasz Kaiser and Ilya Sutskever. Neural GPUs learn algorithms. *arXiv:1511.08228*, 2015.
- Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. Neural machine translation in linear time. *arXiv:1610.10099*, 2016.



- Nal Kalchbrenner, Erich Elsen, Karen Simonyan, Seb Noury, Norman Casagrande, Edward Lockhart, Florian Stimberg, Aaron van den Oord, Sander Dieleman, and Koray Kavukcuoglu. Efficient neural audio synthesis. In *ICML*, 2018.
- Ken-Ichi Kanatani. *Group-theoretical methods in image understanding*. Springer, 2012.
- Zachi Karni and Craig Gotsman. Spectral compression of mesh geometry. In *Proc. Computer Graphics and Interactive Techniques*, 2000.
- Anees Kazi, Luca Cosmo, Nassir Navab, and Michael Bronstein. Differentiable graph module (DGM) graph convolutional networks. *arXiv:2002.04999*, 2020.
- Henry Kenlay, Dorina Thanou, and Xiaowen Dong. Interpretable stability bounds for spectral graph filters. *arXiv:2102.09587*, 2021.
- Ron Kimmel and James A Sethian. Computing geodesic paths on manifolds. *PNAS*, 95(15):8431–8435, 1998.
- Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.
- Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv:1312.6114*, 2013.
- Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. Neural relational inference for interacting systems. In *ICML*, 2018.
- Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv:1609.02907*, 2016a.
- Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv:1611.07308*, 2016b.

- Dmitry B Kireev. Chemnet: a novel neural network based method for graph/property mapping. *J. Chemical Information and Computer Sciences*, 35(2):175–180, 1995.
- Johannes Klicpera, Janek Groß, and Stephan Günnemann. Directional message passing for molecular graphs. *arXiv:2003.03123*, 2020.
- Iasonas Kokkinos, Michael M Bronstein, Rooe Litman, and Alex M Bronstein. Intrinsic shape context descriptors for deformable shapes. In *CVPR*, 2012.
- Patrick T Komiske, Eric M Metodiev, and Jesse Thaler. Energy flow networks: deep sets for particle jets. *Journal of High Energy Physics*, 2019 (1):121, 2019.
- Ilya Kostrikov, Zhongshi Jiang, Daniele Panozzo, Denis Zorin, and Joan Bruna. Surface networks. In *CVPR*, 2018.
- Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.
- Sofia Ira Ktena, Sarah Parisot, Enzo Ferrante, Martin Rajchl, Matthew Lee, Ben Glocker, and Daniel Rueckert. Distance metric learning using graph convolutional networks: Application to functional brain networks. In *MICCAI*, 2017.
- Dominik Kulon, Riza Alp Guler, Iasonas Kokkinos, Michael M Bronstein, and Stefanos Zafeiriou. Weakly-supervised mesh-convolutional hand reconstruction in the wild. In *CVPR*, 2020.
- Karol Kurach, Marcin Andrychowicz, and Ilya Sutskever. Neural random-access machines. *arXiv:1511.06392*, 2015.
- Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86(11):2278–2324, 1998.

- Reiner Lenz. *Group theoretical methods in image processing*. Springer, 1990.
- Moshe Leshno, Vladimir Ya Lin, Allan Pinkus, and Shimon Schocken. Multi-layer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 6(6):861–867, 1993.
- Ron Levie, Federico Monti, Xavier Bresson, and Michael M Bronstein. Cayleynets: Graph convolutional neural networks with complex rational spectral filters. *IEEE Trans. Signal Processing*, 67(1):97–109, 2018.
- Ron Levie, Elvin Isufi, and Gitta Kutyniok. On the transferability of spectral graph filters. In *Sampling Theory and Applications*, 2019.
- Bruno Lévy. Laplace-Beltrami eigenfunctions towards an algorithm that “understands” geometry. In *Proc. Shape Modeling and Applications*, 2006.
- Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *arXiv:1511.05493*, 2015.
- Or Litany, Alex Bronstein, Michael Bronstein, and Ameesh Makadia. Deformable shape completion with graph convolutional autoencoders. In *CVPR*, 2018.
- Roe Litman and Alexander M Bronstein. Learning spectral descriptors for deformable shape correspondence. *Trans. PAMI*, 36(1):171–180, 2013.
- Hsueh-Ti Derek Liu, Alec Jacobson, and Keenan Crane. A Dirac operator for extrinsic shape analysis. *Computer Graphics Forum*, 36(5):139–149, 2017.
- Siwei Lyu and Eero P Simoncelli. Nonlinear image representation using divisive normalization. In *CVPR*, 2008.
- Richard H MacNeal. *The solution of partial differential equations by means of electrical networks*. PhD thesis, California Institute of Technology, 1949.

- Andreas Madsen and Alexander Rosenberg Johansen. Neural arithmetic units. *arXiv:2001.05016*, 2020.
- Soha Sadat Mahdi, Nele Nauwelaers, Philip Joris, Giorgos Bouritsas, Shunwang Gong, Sergiy Bokhnyak, Susan Walsh, Mark Shriver, Michael Bronstein, and Peter Claes. 3d facial matching by spiral convolutional metric learning and a biometric fusion-net of demographic properties. *arXiv:2009.04746*, 2020.
- VE Maiorov. On best approximation by ridge functions. *Journal of Approximation Theory*, 99(1):68–94, 1999.
- Ameesh Makadia, Christopher Geyer, and Kostas Daniilidis. Correspondence-free structure from motion. *IJCV*, 75(3):311–327, 2007.
- Stéphane Mallat. *A wavelet tour of signal processing*. Elsevier, 1999.
- Stéphane Mallat. Group invariant scattering. *Communications on Pure and Applied Mathematics*, 65(10):1331–1398, 2012.
- Brandon Malone, Alberto Garcia-Duran, and Mathias Niepert. Learning representations of missing data for predicting patient outcomes. *arXiv:1811.04752*, 2018.
- Haggai Maron, Heli Ben-Hamu, Nadav Shamir, and Yaron Lipman. Invariant and equivariant graph networks. *arXiv:1812.09902*, 2018.
- Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably powerful graph networks. *arXiv:1905.11136*, 2019.
- Jean-Pierre Marquis. Category theory and klein’ s erlangen program. In *From a Geometrical Point of View*, pages 9–40. Springer, 2009.
- Jonathan Masci, Davide Boscaini, Michael Bronstein, and Pierre Vandergheynst. Geodesic convolutional neural networks on Riemannian manifolds. In *CVPR Workshops*, 2015.

- James Clerk Maxwell. A dynamical theory of the electromagnetic field. *Philosophical Transactions of the Royal Society of London*, (155):459–512, 1865.
- Jason D McEwen, Christopher GR Wallis, and Augustine N Mavor-Parker. Scattering networks on the sphere for scalable and rotationally equivariant spherical cnns. *arXiv:2102.02828*, 2021.
- Song Mei, Theodor Misiakiewicz, and Andrea Montanari. Learning with invariances in random features and kernel models. *arXiv:2102.13219*, 2021.
- Simone Melzi, Riccardo Spezialetti, Federico Tombari, Michael M Bronstein, Luigi Di Stefano, and Emanuele Rodolà. Gframes: Gradient-based local reference frame for 3d shape matching. In *CVPR*, 2019.
- Facundo Mémoli and Guillermo Sapiro. A theoretical and computational framework for isometry invariant recognition of point cloud data. *Foundations of Computational Mathematics*, 5(3):313–347, 2005.
- Christian Merkwirth and Thomas Lengauer. Automatic generation of complementary descriptors with molecular graph networks. *J. Chemical Information and Modeling*, 45(5):1159–1168, 2005.
- Mark Meyer, Mathieu Desbrun, Peter Schröder, and Alan H Barr. Discrete differential-geometry operators for triangulated 2-manifolds. In *Visualization and Mathematics III*, pages 35–57. 2003.
- Alessio Micheli. Neural network for graphs: A contextual constructive approach. *IEEE Trans. Neural Networks*, 20(3):498–511, 2009.
- Karla L Miller, Fidel Alfaro-Almagro, Neal K Bangerter, David L Thomas, Essa Yacoub, Junqian Xu, Andreas J Bartsch, Saad Jbabdi, Stamatios N Sotiropoulos, Jesper LR Andersson, et al. Multimodal population brain imaging in the uk biobank prospective epidemiological study. *Nature Neuroscience*, 19(11):1523–1536, 2016.

- Marvin Minsky and Seymour A Papert. *Perceptrons: An introduction to computational geometry*. MIT Press, 2017.
- Jovana Mitrovic, Brian McWilliams, Jacob Walker, Lars Buesing, and Charles Blundell. Representation learning via invariant causal mechanisms. *arXiv:2010.07922*, 2020.
- Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.
- Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *ICML*, 2016.
- Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *CVPR*, 2017.
- Federico Monti, Fabrizio Frasca, Davide Eynard, Damon Mannion, and Michael M Bronstein. Fake news detection on social media using geometric deep learning. *arXiv:1902.06673*, 2019.
- Christopher Morris, Kristian Kersting, and Petra Mutzel. Glocalized Weisfeiler-Lehman graph kernels: Global-local feature maps of graphs. In *ICDM*, 2017.
- Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *AAAI*, 2019.
- Christopher Morris, Gaurav Rattan, and Petra Mutzel. Weisfeiler and Leman go sparse: Towards scalable higher-order graph embeddings. In *NeurIPS*, 2020.

- Michael C Mozer. A focused back-propagation algorithm for temporal pattern recognition. *Complex Systems*, 3(4):349–381, 1989.
- Kevin Murphy, Yair Weiss, and Michael I Jordan. Loopy belief propagation for approximate inference: An empirical study. *arXiv:1301.6725*, 2013.
- Ryan Murphy, Balasubramaniam Srinivasan, Vinayak Rao, and Bruno Ribeiro. Relational pooling for graph representations. In *ICML*, 2019.
- Ryan L Murphy, Balasubramaniam Srinivasan, Vinayak Rao, and Bruno Ribeiro. Janossy pooling: Learning deep permutation-invariant functions for variable-size inputs. *arXiv:1811.01900*, 2018.
- Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.
- John Nash. The imbedding problem for Riemannian manifolds. *Annals of Mathematics*, 63(1):20–63, 1956.
- Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. Norm-based capacity control in neural networks. In *COLT*, 2015.
- Emmy Noether. Invariante variationsprobleme. In *König Gesellsch. d. Wiss. zu Göttingen, Math-Phys. Klassc*, pages 235–257. 1918.
- Maks Ovsjanikov, Jian Sun, and Leonidas Guibas. Global intrinsic symmetries of shapes. *Computer Graphics Forum*, 27(5):1341–1348, 2008.
- Maks Ovsjanikov, Mirela Ben-Chen, Justin Solomon, Adrian Butscher, and Leonidas Guibas. Functional maps: a flexible representation of maps between shapes. *ACM Trans. Graphics*, 31(4):1–11, 2012.
- Aditya Pal, Chantat Eksombatchai, Yitong Zhou, Bo Zhao, Charles Rosenberg, and Jure Leskovec. Pinnersage: Multi-modal user embedding framework for recommendations at pinterest. In *KDD*, 2020.

Sarah Parisot, Sofia Ira Ktena, Enzo Ferrante, Matthew Lee, Ricardo Guerrero, Ben Glocker, and Daniel Rueckert. Disease prediction using graph convolutional networks: application to autism spectrum disorder and alzheimer’ s disease. *Medical Image Analysis*, 48:117–130, 2018.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *ICML*, 2013.

Giuseppe Patanè. Fourier-based and rational graph filters for spectral processing. *arXiv:2011.04055*, 2020.

Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Elsevier, 2014.

Roger Penrose. *The road to reality: A complete guide to the laws of the universe*. Random House, 2005.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *KDD*, 2014.

Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W Battaglia. Learning mesh-based simulation with graph networks. *arXiv:2010.03409*, 2020.

Fernando J Pineda. Generalization of back propagation to recurrent and higher order neural networks. In *NIPS*, 1988.

Ulrich Pinkall and Konrad Polthier. Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics*, 2(1):15–36, 1993.

Allan Pinkus. Approximation theory of the mlp model in neural networks. *Acta Numerica*, 8:143–195, 1999.

Tom J Pollard, Alistair EW Johnson, Jesse D Raffa, Leo A Celi, Roger G Mark, and Omar Badawi. The eicu collaborative research database, a



- freely available multi-center database for critical care research. *Scientific Data*, 5(1):1–13, 2018.
- Javier Portilla and Eero P Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *International journal of computer vision*, 40(1):49–70, 2000.
- Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017.
- Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In *WSDM*, 2018.
- H Qu and L Gouskos. Particlenet: jet tagging via particle clouds. *arXiv:1902.08570*, 2019.
- Meng Qu, Yoshua Bengio, and Jian Tang. GMNN: Graph Markov neural networks. In *ICML*, 2019.
- Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.
- Anurag Ranjan, Timo Bolkart, Soubhik Sanyal, and Michael J Black. Generating 3D faces using convolutional mesh autoencoders. In *ECCV*, 2018.
- Dan Raviv, Alexander M Bronstein, Michael M Bronstein, and Ron Kimmel. Symmetries of non-rigid shapes. In *ICCV*, 2007.
- Noam Razin and Nadav Cohen. Implicit regularization in deep learning may not be explainable by norms. *arXiv:2005.06398*, 2020.

- Scott Reed and Nando De Freitas. Neural programmer-interpreters. *arXiv:1511.06279*, 2015.
- Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv:1506.01497*, 2015.
- Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *ICML*, 2015.
- Maximilian Riesenhuber and Tomaso Poggio. Hierarchical models of object recognition in cortex. *Nature neuroscience*, 2(11):1019–1025, 1999.
- AJ Robinson and Frank Fallside. *The utility driven dynamic error propagation network*. University of Cambridge, 1987.
- Emma Rocheteau, Pietro Liò, and Stephanie Hyland. Temporal pointwise convolutional networks for length of stay prediction in the intensive care unit. *arXiv:2007.09483*, 2020.
- Emma Rocheteau, Catherine Tong, Petar Veličković, Nicholas Lane, and Pietro Liò. Predicting patient outcomes with graph representation learning. *arXiv:2101.03940*, 2021.
- Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015.
- Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386, 1958.
- Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. Temporal graph networks for deep learning on dynamic graphs. *arXiv:2006.10637*, 2020.

- Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015.
- Raif M Rustamov, Maks Ovsjanikov, Omri Azencot, Mirela Ben-Chen, Frédéric Chazal, and Leonidas Guibas. Map-based exploration of intrinsic shape differences and variability. *ACM Trans. Graphics*, 32(4):1–12, 2013.
- Tim Salimans and Diederik P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *arXiv:1602.07868*, 2016.
- Alvaro Sanchez-Gonzalez, Victor Bapst, Kyle Cranmer, and Peter Battaglia. Hamiltonian graph networks with ODE integrators. *arXiv:1909.12790*, 2019.
- Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph networks. In *ICML*, 2020.
- Aliaksei Sandryhaila and José MF Moura. Discrete signal processing on graphs. *IEEE Trans. Signal Processing*, 61(7):1644–1656, 2013.
- Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. A simple neural network module for relational reasoning. In *NIPS*, 2017.
- Adam Santoro, Ryan Faulkner, David Raposo, Jack Rae, Mike Chrzanowski, Theophane Weber, Daan Wierstra, Oriol Vinyals, Razvan Pascanu, and Timothy Lillicrap. Relational recurrent neural networks. *arXiv:1806.01822*, 2018.
- Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry.

How does batch normalization help optimization? *arXiv:1805.11604*, 2018.

Ryoma Sato, Makoto Yamada, and Hisashi Kashima. Random features strengthen graph neural networks. *arXiv:2002.03155*, 2020.

Victor Garcia Satorras, Emiel Hoogetboom, and Max Welling. E(n) equivariant graph neural networks. *arXiv:2102.09844*, 2021.

Anna MM Scaife and Fiona Porter. Fanaroff-Riley classification of radio galaxies using group-equivariant convolutional neural networks. *Monthly Notices of the Royal Astronomical Society*, 2021.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Trans. Neural Networks*, 20(1):61–80, 2008.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv:1707.06347*, 2017.

Kristof T Schütt, Huziel E Saucedo, P-J Kindermans, Alexandre Tkatchenko, and K-R Müller. Schnet—a deep learning architecture for molecules and materials. *The Journal of Chemical Physics*, 148(24):241722, 2018.

Terrence J Sejnowski, Paul K Kienker, and Geoffrey E Hinton. Learning symmetry groups with hidden units: Beyond the perceptron. *Physica D: Nonlinear Phenomena*, 22(1-3):260–275, 1986.

- Andrew W Senior, Richard Evans, John Jumper, James Kirkpatrick, Laurent Sifre, Tim Green, Chongli Qin, Augustin Židek, Alexander WR Nelson, Alex Bridgland, et al. Improved protein structure prediction using potentials from deep learning. *Nature*, 577(7792):706–710, 2020.
- Thomas Serre, Aude Oliva, and Tomaso Poggio. A feedforward architecture accounts for rapid categorization. *Proceedings of the national academy of sciences*, 104(15):6424–6429, 2007.
- Ohad Shamir and Gal Vardi. Implicit regularization in relu networks with the square loss. *arXiv:2012.05156*, 2020.
- John Shawe-Taylor. Building symmetries into feedforward networks. In *ICANN*, 1989.
- John Shawe-Taylor. Symmetries and discriminability in feedforward network architectures. *IEEE Trans. Neural Networks*, 4(5):816–826, 1993.
- Nino Shervashidze, Pascal Schweitzer, Erik Jan Van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-lehman graph kernels. *JMLR*, 12(9), 2011.
- Jonathan Shlomi, Peter Battaglia, and Jean-Roch Vlimant. Graph neural networks in particle physics. *Machine Learning: Science and Technology*, 2(2):021001, 2020.
- David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine*, 30(3):83–98, 2013.
- Hava T Siegelmann and Eduardo D Sontag. On the computational power of neural nets. *Journal of Computer and System Sciences*, 50(1):132–150, 1995.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

Eero P Simoncelli and William T Freeman. The steerable pyramid: A flexible architecture for multi-scale derivative computation. In *Proceedings, International Conference on Image Processing*, volume 3, pages 444–447. IEEE, 1995.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556*, 2014.

Alex Smola, Arthur Gretton, Le Song, and Bernhard Schölkopf. A Hilbert space embedding for distributions. In *ALT*, 2007.

Stefan Spalević, Petar Veličković, Jovana Kovačević, and Mladen Nikolić. Hierachial protein function prediction with tail-GNNs. *arXiv:2007.12804*, 2020.

Alessandro Sperduti. Encoding labeled graphs by labeling RAAM. In *NIPS*, 1994.

Alessandro Sperduti and Antonina Starita. Supervised neural networks for the classification of structures. *IEEE Trans. Neural Networks*, 8(3):714–735, 1997.

Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Mar-

- tin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv:1412.6806*, 2014.
- Balasubramaniam Srinivasan and Bruno Ribeiro. On the equivalence between positional node embeddings and structural graph representations. *arXiv:1910.00452*, 2019.
- Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 15(1):1929–1958, 2014.
- Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv:1505.00387*, 2015.
- Kimberly Stachenfeld, Jonathan Godwin, and Peter Battaglia. Graph networks with spectral message passing. *arXiv:2101.00079*, 2020.
- Jonathan M Stokes, Kevin Yang, Kyle Swanson, Wengong Jin, Andres Cubillos-Ruiz, Nina M Donghia, Craig R MacNair, Shawn French, Lindsey A Carfrae, Zohar Bloom-Ackerman, et al. A deep learning approach to antibiotic discovery. *Cell*, 180(4):688–702, 2020.
- Heiko Strathmann, Mohammadamin Barekatain, Charles Blundell, and Petar Veličković. Persistent message passing. *arXiv:2103.01043*, 2021.
- Norbert Straumann. Early history of gauge theories and weak interactions. *hep-ph/9609230*, 1996.
- Jian Sun, Maks Ovsjanikov, and Leonidas Guibas. A concise and provably informative multi-scale signature based on heat diffusion. *Computer Graphics Forum*, 28(5):1383–1392, 2009.
- Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *arXiv:1409.3215*, 2014.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

Corentin Tallec and Yann Ollivier. Can recurrent neural networks warp time? *arXiv:1804.11188*, 2018.

Hao Tang, Zhiao Huang, Jiayuan Gu, Bao-Liang Lu, and Hao Su. Towards scale-invariant graph-related problem solving by iterative homogeneous gnns. In *NeurIPS*, 2020.

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *WWW*, 2015.

Gabriel Taubin, Tong Zhang, and Gene Golub. Optimal surface smoothing as filter design. In *ECCV*, 1996.

Shantanu Thakoor, Corentin Tallec, Mohammad Gheshlaghi Azar, Rémi Munos, Petar Veličković, and Michal Valko. Bootstrapped representation learning on graphs. *arXiv:2102.06514*, 2021.

Nathaniel Thomas, Tess Smidt, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation- and translation-equivariant neural networks for 3D point clouds. *arXiv:1802.08219*, 2018.

Renate Tobies. Felix Klein—mathematician, academic organizer, educational reformer. In *The Legacy of Felix Klein*, pages 5–21. Springer, 2019.

Andrew Trask, Felix Hill, Scott Reed, Jack Rae, Chris Dyer, and Phil Blunsom. Neural arithmetic logic units. *arXiv:1808.00508*, 2018.

John Tromp and Gunnar Farneback. Combinatorics of go. In *International Conference on Computers and Games*, 2006.



- Alexandre B Tsybakov. *Introduction to nonparametric estimation*. Springer, 2008.
- Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv:1607.08022*, 2016.
- Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv:1609.03499*, 2016a.
- Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *ICML*, 2016b.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017.
- Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. *ICLR*, 2018.
- Petar Veličković, Rex Ying, Matilde Padovano, Raia Hadsell, and Charles Blundell. Neural execution of graph algorithms. *arXiv:1910.10593*, 2019.
- Petar Veličković, Lars Buesing, Matthew C Overlan, Razvan Pascanu, Oriol Vinyals, and Charles Blundell. Pointer graph networks. *arXiv:2006.06380*, 2020.
- Petar Veličković, Wiliam Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep Graph Infomax. In *ICLR*, 2019.
- Kirill Veselkov, Guadalupe Gonzalez, Shahad Aljifri, Dieter Galea, Reza Mirnezami, Jozef Youssef, Michael Bronstein, and Ivan Laponogov. Hyperfoods: Machine intelligent mapping of cancer-beating molecules in foods. *Scientific Reports*, 9(1):1–12, 2019.

- Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. *arXiv:1506.03134*, 2015.
- Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. Order matters: Sequence to sequence for sets. In *ICLR*, 2016.
- Ulrike von Luxburg and Olivier Bousquet. Distance-based classification with lipschitz functions. *JMLR*, 5:669–695, 2004.
- Martin J Wainwright and Michael Irwin Jordan. *Graphical models, exponential families, and variational inference*. Now Publishers Inc, 2008.
- Yu Wang and Justin Solomon. Intrinsic and extrinsic operators for shape analysis. In *Handbook of Numerical Analysis*, volume 20, pages 41–115. Elsevier, 2019.
- Yu Wang, Mirela Ben-Chen, Iosif Polterovich, and Justin Solomon. Steklov spectral geometry for extrinsic shape analysis. *ACM Trans. Graphics*, 38(1):1–21, 2018.
- Yu Wang, Vladimir Kim, Michael Bronstein, and Justin Solomon. Learning geometric operators on meshes. In *ICLR Workshops*, 2019a.
- Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph CNN for learning on point clouds. *ACM Trans. Graphics*, 38(5):1–12, 2019b.
- Max Wardetzky. Convergence of the cotangent formula: An overview. *Discrete Differential Geometry*, pages 275–286, 2008.
- Max Wardetzky, Saurabh Mathur, Felix Kälberer, and Eitan Grinspun. Discrete Laplace operators: no free lunch. In *Symposium on Geometry Processing*, 2007.
- Maurice Weiler, Mario Geiger, Max Welling, Wouter Boomsma, and Taco Cohen. 3d steerable cnns: Learning rotationally equivariant features in volumetric data. *arXiv:1807.02547*, 2018.

- Boris Weisfeiler and Andrei Leman. The reduction of a graph to canonical form and the algebra which appears therein. *NTI Series*, 2(9):12–16, 1968.
- Paul J Werbos. Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks*, 1(4):339–356, 1988.
- Hermann Weyl. Elektron und gravitation. i. *Zeitschrift für Physik*, 56(5-6): 330–352, 1929.
- Hermann Weyl. *Symmetry*. Princeton University Press, 2015.
- Marysia Winkels and Taco S Cohen. Pulmonary nodule detection in ct scans with equivariant cnns. *Medical Image Analysis*, 55:15–26, 2019.
- Jeffrey Wood and John Shawe-Taylor. Representation theory and invariant neural networks. *Discrete Applied Mathematics*, 69(1-2):33–60, 1996.
- Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *ICML*, 2019.
- Yuxin Wu and Kaiming He. Group normalization. In *ECCV*, 2018.
- Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. Inductive representation learning on temporal graphs. *arXiv:2002.07962*, 2020a.
- Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv:1810.00826*, 2018.
- Keyulu Xu, Jingling Li, Mozhi Zhang, Simon S Du, Ken-ichi Kawarabayashi, and Stefanie Jegelka. What can neural networks reason about? *arXiv:1905.13211*, 2019.
- Keyulu Xu, Jingling Li, Mozhi Zhang, Simon S Du, Ken-ichi Kawarabayashi, and Stefanie Jegelka. How neural networks extrapolate: From feedforward to graph neural networks. *arXiv:2009.11848*, 2020b.

- Yujun Yan, Kevin Swersky, Danai Koutra, Parthasarathy Ranganathan, and Milad Heshemi. Neural execution engines: Learning to execute sub-routines. *arXiv:2006.08084*, 2020.
- Chen-Ning Yang and Robert L Mills. Conservation of isotopic spin and isotopic gauge invariance. *Physical Review*, 96(1):191, 1954.
- Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *ICML*, 2016.
- Jonathan S Yedidia, William T Freeman, and Yair Weiss. Bethe free energy, kikuchi approximations, and belief propagation algorithms. *NIPS*, 2001.
- Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *KDD*, 2018.
- Jiaxuan You, Rex Ying, and Jure Leskovec. Position-aware graph neural networks. In *ICML*, 2019.
- Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. In *NIPS*, 2017.
- Wojciech Zaremba and Ilya Sutskever. Learning to execute. *arXiv:1410.4615*, 2014.
- Wei Zeng, Ren Guo, Feng Luo, and Xianfeng Gu. Discrete heat kernel determines discrete riemannian metric. *Graphical Models*, 74(4):121–129, 2012.
- Jiani Zhang, Xingjian Shi, Junyuan Xie, Hao Ma, Irwin King, and Dit-Yan Yeung. Gaan: Gated attention networks for learning on large and spatiotemporal graphs. *arXiv:1803.07294*, 2018.

- Yuyu Zhang, Xinshi Chen, Yuan Yang, Arun Ramamurthy, Bo Li, Yuan Qi, and Le Song. Efficient probabilistic logic reasoning with graph neural networks. *arXiv:2001.11850*, 2020.
- Rong Zhu, Kun Zhao, Hongxia Yang, Wei Lin, Chang Zhou, Baole Ai, Yong Li, and Jingren Zhou. Aligraph: A comprehensive graph neural network platform. *arXiv:1902.08730*, 2019.
- Weicheng Zhu and Narges Razavian. Variationally regularized graph-based representation learning for electronic health records. *arXiv:1912.03761*, 2019.
- Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Deep graph contrastive representation learning. *arXiv:2006.04131*, 2020.
- Marinka Zitnik, Monica Agrawal, and Jure Leskovec. Modeling polypharmacy side effects with graph convolutional networks. *Bioinformatics*, 34(13):i457–i466, 2018.