Chapter 1: Introduction

引言 — 几何深度学习的统一视角

> ℹ️ **本章概述**
>
> 本章是全书的**起点**，建立了几何深度学习（Geometric Deep Learning, GDL）的宏观视角。我们将回答以下核心问题：
>
> - **为什么**几何深度学习如此重要?
> - **什么是**深度学习中的"几何统一"?
> - **5G 框架**如何统一 CNN、RNN、GNN 和 Transformer?
> - Erlangen Program 的精神如何指导架构设计?
> - **阅读路线图**：本书各章如何串联?
>
> **预计阅读时间**：1.5 小时 | **先修知识**：基础线性代数、微积分、Python

为什么需要几何深度学习?

Why Geometric Deep Learning?

深度学习革命

> 过去十年见证了数据科学和机器学习领域的**实验性革命**，其标志是深度学习方法的崛起。许多此前被认为不可能的高维学习任务——如**计算机视觉**、**围棋博弈**或**蛋白质折叠**——实际上在适当的计算规模下是可行的。
>
> ---
>
> *The last decade has witnessed an experimental revolution in data science and machine learning, epitomised by deep learning methods. Indeed, many high-dimensional learning tasks previously thought to be beyond reach — such as computer vision, playing Go, or protein folding — are in fact feasible with appropriate computational scale.*

让我们用一组数字来感受这场革命的规模：

里程碑	年份	意义
AlexNet (ImageNet)	2012	CNN 在图像识别上首次大幅超越传统方法，开启深度学习时代
AlphaGo	2016	CNN + MCTS 在围棋上击败世界冠军李世乭
Transformer / Attention is All You Need	2017	自注意力机制统一 NLP，开启大模型时代
AlphaFold 2	2020	等变 GNN 解决蛋白质结构预测问题
GPT-3	2020	1750 亿参数语言模型展示 few-shot 学习能力
GNS (粒子仿真)	2020	GNN 实现高精度物理仿真，超越传统数值方法

但这场革命背后有一个**深刻的统一主题**：所有这些成功都源于**对数据底层几何结构的正确利用**。这正是几何深度学习要揭示的核心洞察。

超越欧几里得：非结构化数据的崛起

> 传统深度学习成功主要集中在**欧几里得域**的数据上：图像（2D 网格）、语音（1D 序列）、文本（1D 序列）。但现实世界的大量数据天然具有**非欧几里得**结构：
>
> - **社交网络** — 用户和关系构成图结构
> - **分子** — 原子和化学键构成图；3D 结构在旋转下不变
> - **蛋白质** — 氨基酸链折叠成复杂的 3D 表面
> - **交通网络** — 道路和交叉口构成空间图
> - **粒子系统** — 物理粒子之间的相互作用构成动态图
> - **医疗数据** — 患者关系网络、脑功能连接图
>
> 对这些数据，传统的 CNN 或 RNN 无法直接适用。我们需要一种**更通用的理论框架**。

> *While traditional deep learning succeeded on Euclidean-structured data (images, speech, text), the real world is full of non-Euclidean data: social networks, molecules, proteins, road networks, particle systems, medical data, and more. We need a more general theoretical framework.*

> 🤖 **PhysRobot 项目中的非欧几里得数据**
>
> 我们的医疗机器人物理仿真项目正是这种非欧几里得数据的典型案例：
>
> - **软组织**模型为三角网格（离散流形）
> - **粒子系统**由动态图表示（k-NN 图随仿真演化）
> - **手术器械**在 SE(3) 空间中运动（旋转 + 平移）
> - 物理定律具有**旋转不变性**和**平移不变性**
>
> GDL 提供了正确利用这些对称性的理论基础。

对称性：物理学的指导原则

> 利用大型系统的**已知对称性**是对抗维度灾难的经典而强大的方法，也是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，研究者就调整神经网络来利用物理测量产生的低维几何结构：
>
> - **图像**中的网格结构 → **平移对称性** → CNN
> - **时间序列**中的序列结构 → **时间不变性** → RNN
> - **分子**中的位置和动量 → **旋转对称性** → 等变网络
>
> 本书的核心论点是：这些看似不同的架构实际上是**同一个底层原则——几何正则性**的自然实例。
>
> ---
>
> *Exploiting the known symmetries of a large system is a powerful and classical remedy against the curse of dimensionality, and forms the basis of most physical theories. Deep learning systems are no exception. Throughout our exposition, we will describe these models as natural instances of the same underlying principle of geometric regularity.*

$$\text{对称性} \xrightarrow{\text{约束}} \text{函数空间} \xrightarrow{\text{减少}} \text{需要的样本数} \xrightarrow{\text{提升}} \text{泛化性能}$$

💡 核心逻辑链：对称性约束了可能的函数空间（假设类），使得学习算法不需要从所有可能的函数中搜索，而只需在一个**更小但更正确**的函数类中搜索。这直接导致更好的**样本效率**和**泛化性能**。

让我们用一个直觉性的例子来理解：

```
# 对称性如何减少参数量的直觉
import numpy as np

# 不利用对称性：全连接层
# 输入：32x32 图像 = 1024 维
# 输出：1024 维
fc_params = 1024 * 1024  # = 1,048,576 参数

# 利用平移对称性：卷积层
# 3x3 卷积核，共享权重
conv_params = 3 * 3  # = 9 参数（减少了 ~100,000 倍！）

print(f"全连接层参数：{fc_params:,}")
print(f"卷积层参数：{conv_params}")
print(f"减少倍数：{fc_params / conv_params:.0f}x")

# 但卷积层并没有损失"有用的"表达能力
# 因为图像的统计规律确实具有平移不变性
# 这就是"对称性 → 更好的归纳偏置"的核心思想
```

两个核心算法原则

Two Algorithmic Principles

> 深度学习的本质建立在两个简单的算法原则之上：
>
> ---
>
> *The essence of deep learning is built from two simple algorithmic principles.*

原则一：表征学习 (Representation Learning)

> **表征学习**（或**特征学习**）是指通过自适应的、通常是层级化的特征来捕获每个任务的适当正则性概念。与传统机器学习中需要人工设计特征不同，深度学习**自动学习**数据表征。
>
> *Representation or feature learning, whereby adapted, often hierarchical, features capture the appropriate notion of regularity for each task.*

$$x \xrightarrow{f_1} h^{(1)} \xrightarrow{f_2} h^{(2)} \xrightarrow{\cdots} h^{(L)} \xrightarrow{g} \hat{y}$$

💡 **层级表征**：输入 x 经过多层非线性变换 $f_1, f_2, \ldots$，逐步提取越来越抽象的特征 $h^{(k)}$，最终通过输出层 g 做出预测 $\hat{y}$。例如在图像识别中：$h^{(1)}$ 捕获边缘，$h^{(2)}$ 捕获纹理/形状，$h^{(3)}$ 捕获对象部件，$h^{(L)}$ 捕获完整对象。

```
# 层级特征学习的直觉
import torch
import torch.nn as nn

class HierarchicalFeatureLearner(nn.Module):
    """层级表征学习的简单示例"""
    def __init__(self):
        super().__init__()
        # 每一层学习越来越抽象的特征
        self.layer1 = nn.Conv2d(3, 32, 3, padding=1)    # 低级: 边缘、颜色
        self.layer2 = nn.Conv2d(32, 64, 3, padding=1)   # 中级: 纹理、形状
        self.layer3 = nn.Conv2d(64, 128, 3, padding=1)  # 高级: 对象部件
        self.classifier = nn.Linear(128, 10)             # 分类: 完整对象
        self.relu = nn.ReLU()
        self.pool = nn.AdaptiveAvgPool2d(1)

    def forward(self, x):
        h1 = self.relu(self.layer1(x))   # h^(1): 边缘特征
        h2 = self.relu(self.layer2(h1))  # h^(2): 纹理特征
        h3 = self.relu(self.layer3(h2))  # h^(3): 对象特征
        pooled = self.pool(h3).flatten(1)
        return self.classifier(pooled)
```

```
# 关键洞察：这些特征不是人工设计的，
# 而是通过梯度下降自动学习的！
```

原则二：局部梯度下降 (Backpropagation)

> 第二个原则是通过**局部梯度下降**进行学习，通常以**反向传播**（backpropagation）的形式实现。给定损失函数 $\mathcal{L}(\theta)$，我们通过计算梯度 $\nabla_\theta \mathcal{L}$ 并沿负梯度方向更新参数来最小化损失。
>
> *Learning by local gradient-descent, typically implemented as backpropagation.*

$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla_\theta \mathcal{L}(\theta^{(t)})$$

💡 **梯度下降**：θ 是模型参数，η 是学习率，$\mathcal{L}$ 是损失函数。反向传播通过链式法则高效计算 $\nabla_\theta \mathcal{L}$。这两个原则（表征学习 + 梯度下降）在各种架构中是**通用的**——无论是 CNN、RNN、GNN 还是 Transformer，都使用同样的学习机制。

维度灾难预览

> 虽然在高维空间中学习通用函数是一个**被诅咒的**估计问题（需要指数级的样本），但大多数实际任务并不是通用的。它们来自物理世界底层的**低维性**和**结构性**，带有本质的预定义正则性。
>
> 本书的核心目标就是：通过**统一的几何原则**暴露这些正则性，并将它们应用于广泛的应用领域。
>
> *While learning generic functions in high dimensions is a cursed estimation problem, most tasks of interest are not generic, and come with essential pre-defined regularities arising from the underlying low-dimensionality and structure of the physical world.*

⚠️ **维度灾难的直觉**

在 d 维空间中，如果我们在每个维度上取 m 个样本点，总共需要 m^d 个样本。对于 $d = 100$（一张 10×10 的灰度图），即使每维只取 2 个点，也需要 $2^{100} \approx 10^{30}$ 个样本——远超宇

宙中的原子数（$\sim 10^{80}$...等等，虽然没超过，但这仅仅是 10×10 的图像！对于现实的 224×224 RGB 图像，维度是 $224 \times 224 \times 3 = 150{,}528$）。

这就是为什么我们需要归纳偏置——利用数据的结构来约束搜索空间。详见 Chapter 2。

5G 统一框架

The 5G Framework: Grids, Groups, Graphs, Geodesics, Gauges

本书提出的**"几何统一"**方案——以 Erlangen Program 的精神——有双重目的：一方面，它提供了一个**共同的数学框架**来研究最成功的神经网络架构；另一方面，它提供了一个**建设性程序**来将先验物理知识融入神经架构，并为设计未来的新架构提供原则性方法。

我们将这个框架称为 **5G**：五个以 G 开头的几何域，覆盖了几乎所有的数据结构。

Such a 'geometric unification' endeavour in the spirit of the Erlangen Program serves a dual purpose: providing a common mathematical framework to study the most successful architectures, and giving a constructive procedure to incorporate prior physical knowledge into neural architectures.

Vergleichende Betrachtungen

über

neuere geometrische Forschungen

von

Dr. Felix Klein,
o. ö. Professor der Mathematik an der Universität Erlangen.

Programm

zum Eintritt in die philosophische Facultät und den Senat
der k. Friedrich-Alexanders-Universität
zu Erlangen.

Erlangen.
Verlag von Andreas Deichert.
1872.

5G 框架概览：五种几何域及其对应的对称性和神经网络架构。从左到右，对称性从大到小，结构从通用到特殊。

🔲 Grids — 网格

> **网格**是最简单也是最成功的几何域。图像是 2D 网格上的信号，语音和文本是 1D 网格上的信号。网格的关键对称性是**平移对称性**（translation symmetry）。
>
> ---
>
> *Grids are the simplest and most successful geometric domain. Images are signals on 2D grids, speech/text on 1D grids. The key symmetry is translation.*

$$\Omega = \mathbb{Z}^d, \quad G = (\mathbb{Z}^d, +), \quad \text{对应架构: CNN}$$

💡 **域**：d 维整数网格。**对称群**：整数平移群。**等变操作**：离散卷积。CNN 的成功正是因为它**正确利用**了图像域的平移对称性。卷积核在所有位置共享权重（权重共享 = 平移等变性）。

```
# 网格域：图像卷积
import torch
import torch.nn.functional as F

# 3x3 卷积核（权重共享 = 平移等变性）
kernel = torch.tensor([
    [-1, -1, -1],
    [-1,  8, -1],
    [-1, -1, -1]
], dtype=torch.float).view(1, 1, 3, 3)

# 同一个核应用于图像的每个位置
# 这就是 "平移等变性" — 如果输入平移，输出也平移
image = torch.randn(1, 1, 28, 28)  # MNIST-like
output = F.conv2d(image, kernel, padding=1)
print(f"输入形状：{image.shape}，输出形状：{output.shape}")
print(f"卷积核参数：{kernel.numel()}（vs 全连接：{28*28*28*28:,}）")
```

👥 Groups — 群

> 当域不仅有平移对称性，还有**旋转**、**反射**等更丰富的对称性时，我们需要考虑**群**域上的信号处理。典型例子包括球面上的信号（天文学、气象学）和分子系统。
>
> *When the domain has richer symmetries beyond translation (rotation, reflection, etc.), we need signal processing on group domains.*

$$\Omega = G \text{ (群本身)}, \quad \text{对称群: } G \text{ 自身的左/右作用}$$

$$\text{对应架构: Group Equivariant CNN, Spherical CNN}$$

💡 在群域上，信号定义为 $f : G \to \mathbb{R}^c$。群卷积定义为：$(f \star \psi)(g) = \int_G f(g'^{-1}g)\psi(g')dg'$。这保证了对群作用的等变性。例如，SO(3)-等变的网络在分子性质预测中至关重要。

🔗 Graphs — 图

> **图**可能是最通用的非欧几里得数据结构。社交网络、分子、知识图谱、交通网络——它们都是图。图的关键对称性是**节点置换对称性**：图的性质不应该依赖于节点的编号顺序。
>
> *Graphs are perhaps the most general non-Euclidean data structure. The key symmetry is node permutation: graph properties should not depend on the ordering of nodes.*

$$\Omega = (\mathcal{V}, \mathcal{E}), \quad G = \Sigma_n, \quad \text{对应架构: GNN (消息传递)}$$

💡 **域**：图 $(\mathcal{V}, \mathcal{E})$，节点集 $\mathcal{V}$，边集 $\mathcal{E}$。**对称群**：节点置换群 Σ_n。**等变操作**：消息传递（Message Passing）。GNN 通过邻域聚合（$h_i \leftarrow \mathrm{AGG}(\{h_j : j \in \mathcal{N}(i)\})$）实现置换等变性。

```
# 图域: 消息传递 GNN
import torch

def message_passing(node_features, edge_index, message_fn, update_fn):
```

```
    """
    基本的消息传递操作
    - 置换等变：重新排列节点不改变计算结果
    """
    num_nodes = node_features.size(0)
    messages = torch.zeros_like(node_features)

    for src, dst in edge_index.t():
        # 从邻居 src 向节点 dst 发送消息
        msg = message_fn(node_features[src], node_features[dst])
        messages[dst] += msg  # 聚合（求和是置换不变的!）

    # 更新节点特征
    new_features = update_fn(node_features, messages)
    return new_features

# 消息传递是置换等变的：
# 如果我们打乱节点顺序，结果只是相应地打乱
# 而不会改变每个节点的实际计算结果
```

📏 Geodesics — 测地线

> **测地线**（Geodesics）指的是**流形**（manifold）和**网格**（mesh）上的**内蕴度量**和**等距对称性**。典型例子是蛋白质表面、人体形状等，其中形状的内蕴几何（如曲率、测地距离）比外在嵌入更重要。
>
> ---
>
> *Geodesics refer to intrinsic metrics and isometric symmetries on manifolds and meshes. Shape analysis (e.g., protein surfaces, body shapes) relies on intrinsic geometry rather than extrinsic embedding.*

$$\Omega = \mathcal{M}\ (\text{流形}), \quad G = \mathrm{Iso}(\mathcal{M}), \quad \text{对应架构: Mesh CNN, Geodesic CNN}$$

💡 在流形上，"距离" 由测地线（表面上的最短路径）定义，而不是直线距离。等距对称群 $\mathrm{Iso}(\mathcal{M})$ 保持测地距离不变。例如，当人体弯曲手臂时，皮肤表面上两点的测地距离不变（等距变换），但欧几里得距离变了。

🔬 Gauges — 规范

> **规范**（Gauge）是最精细的几何结构，涉及流形上的**参考框架**（frame）和**纤维丛**（fiber bundle）。在流形上定义卷积需要选择局部参考方向——这个选择就是"规范"。规范等变意味着计算结果不依赖于这个任意选择。
>
> *Gauges involve reference frames on manifolds and fiber bundles. Defining convolution on manifolds requires choosing local reference directions — this choice is a "gauge". Gauge equivariance means computations are independent of this arbitrary choice.*

$$\Omega = (\mathcal{M}, \text{gauge}), \quad G = \text{Structure group of the bundle}$$

$$\text{对应架构: Gauge Equivariant CNN}$$

💡 规范等变 CNN 是最通用的几何 CNN 形式。它在流形的每个点上定义局部滤波器，并确保结果不依赖于局部坐标系的选择。这与物理学中的规范理论（如电磁学中的 U(1) 规范不变性）有深刻的类比。

ℹ️ 5G 之间的关系

域	对称性	典型架构	应用例子
集合 (Sets)	Σ_n (置换)	DeepSets, Transformer	点云、NLP
网格 (Grids)	$\mathbb{Z}^d$ (平移)	CNN, RNN	图像、语音
群 (Groups)	G (群作用)	Group CNN, Spherical CNN	天文学、分子
图 (Graphs)	Σ_n (节点置换)	GCN, GAT, MPNN	社交网络、分子
流形 (Manifolds)	Iso($\mathcal{M}$) / Gauge	Mesh CNN, Gauge CNN	蛋白质表面、3D形状

> 这些域形成一个**层级结构**：集合 ⊃ 图 ⊃ 网格；流形 ⊃ 网格。5G 框架用**对称性**这一统一语言描述所有这些域。

Erlangen Program 的精神

The Spirit of the Erlangen Programme

历史背景

> 1872 年，年仅 23 岁的数学家 **Felix Klein** 在德国 Erlangen 大学的就职演讲中提出了著名的 **Erlangen Program**（Erlangen 纲领）。他的革命性想法是：不同的**几何学**可以通过其**对称群**来分类和统一。
>
> - **欧几里得几何** = 在欧几里得群 $E(d)$ 下不变的性质
> - **仿射几何** = 在仿射群下不变的性质
> - **射影几何** = 在射影群下不变的性质
> - **拓扑学** = 在同胚群下不变的性质
>
> Klein 的核心洞察：**几何就是研究在某个变换群下不变的性质**。
>
> ---
>
> *In 1872, the 23-year-old mathematician Felix Klein proclaimed group theory to be the organising principle of geometry in his famous Erlangen Programme. His key insight: geometry is the study of properties invariant under a transformation group.*

对机器学习的启示

> 本书将 Erlangen Program 的精神应用于深度学习：**不同的深度学习架构可以通过其利用的对称群来分类和统一**。
>
> - **CNN** = 利用平移对称群 $(\mathbb{Z}^d, +)$ 的网络
> - **GNN** = 利用置换对称群 Σ_n 的网络

- **Transformer** = 在完全图上利用置换对称性的网络
- **Spherical CNN** = 利用旋转对称群 $SO(3)$ 的网络

这不仅是事后的"理解"，更是**前瞻性的设计原则**：给定一个新问题，确定其数据的对称性，然后构建等变架构。

We apply the Erlangen Program spirit to deep learning: different architectures can be classified and unified by the symmetry group they exploit. This is not just retrospective understanding, but a prospective design principle.

$$\text{Erlangen for ML:} \quad \boxed{\text{问题}} \xrightarrow{\text{识别}} \boxed{\text{对称群 } G} \xrightarrow{\text{构建}} \boxed{\text{G-等变架构}}$$

💡 这是 GDL 的**建设性程序**：(1) 分析问题的数据域 Ω；(2) 确定相关的对称群 G；(3) 构建 G-等变的神经网络层；(4) 通过局部池化和全局池化得到 G-不变的输出。

架构景观

The Landscape of Architectures

CNN 家族 — 网格上的等变网络

卷积神经网络（CNN）是深度学习最早也是最成功的架构之一。从 GDL 的视角看，CNN 是在d **维网格**上利用**平移对称性**的等变网络。其核心操作——卷积——正是唯一的平移等变线性操作。

- **1D CNN**：用于序列数据（WaveNet 语音合成、时间序列）
- **2D CNN**：用于图像（AlexNet → VGG → ResNet → EfficientNet）
- **3D CNN**：用于视频和体积数据（医学 CT/MRI）

CNNs are equivariant networks on d-dimensional grids exploiting translation symmetry. Convolution is the unique translation-equivariant linear operation.

RNN 家族 — 序列上的处理

> **循环神经网络**（RNN）处理可变长度的序列数据。从 GDL 视角看，RNN 可以视为在 1D 网格上利用了**因果结构**（只看过去，不看未来）的架构。LSTM 和 GRU 通过门控机制解决了梯度消失问题。
>
> ---
>
> *RNNs process variable-length sequences. From GDL perspective, they exploit causal structure on 1D grids. LSTM and GRU solve vanishing gradient via gating.*

GNN 家族 — 图上的消息传递

> **图神经网络**（GNN）是 GDL 的核心成员，处理图结构数据。其基本操作是**消息传递**（message passing）：每个节点聚合其邻居的信息来更新自身表示。这个操作是**置换等变**的。
>
> - **GCN**（Kipf & Welling, 2016）：谱方法的简化
> - **GAT**（Veličković et al., 2018）：注意力加权的消息传递
> - **MPNN**（Gilmer et al., 2017）：统一消息传递框架
> - **GNS**（Sanchez-Gonzalez et al., 2020）：用于物理仿真
>
> ---
>
> *GNNs are the core of GDL, processing graph-structured data via message passing — a permutation-equivariant operation.*

Transformer — 集合上的全局注意力

> **Transformer** 从 GDL 视角可以理解为在**完全图**上的 GNN——每个元素都关注所有其他元素。自注意力机制是一种在**集合**（无固定拓扑结构的域）上的置换等变操作。
>
> Transformer 的位置编码打破了纯置换对称性，引入了序列/空间位置信息。这是一个"减少对称性"的例子——当数据确实有序列结构时，我们**不应该**保持完全的置换不变性。

> *From a GDL perspective, Transformers can be understood as GNNs on a complete graph — each element attends to all others. Self-attention is a permutation-equivariant operation on sets.*

```
# GDL 视角下的自注意力 = 完全图上的消息传递
import torch
import torch.nn.functional as F
import math

def self_attention_as_gnn(Q, K, V):
    """
    自注意力 ≡ 完全图上的注意力加权消息传递
    Q, K, V: [batch, seq_len, d_k]
    """
    d_k = Q.size(-1)

    # 计算注意力权重（完全图的边权重）
    # 每个节点关注所有其他节点
    attention_weights = torch.matmul(Q, K.transpose(-2, -1)) / math.sqrt(d_k)
    attention_weights = F.softmax(attention_weights, dim=-1)

    # 聚合消息（加权求和 = 注意力消息传递）
    output = torch.matmul(attention_weights, V)
    return output

# 关键洞察：
# - Transformer = GNN on complete graph
# - 如果限制注意力只在局部邻居：得到类似 GAT 的东西
# - 如果注意力权重只依赖于距离：得到类似 CNN 的东西
# - 这就是 GDL 统一视角的力量！
```

本书的范围与非范围

Scope and Non-Scope

> 值得注意的是，本书关注的是**表征学习架构**及其中的**数据对称性利用**。以下内容**不是**本书的核心关注点（虽然我们认为几何原则在这些领域同样重要）：

> *Our work concerns representation learning architectures and exploiting the symmetries of data therein. The following exciting pipelines are not our central focus.*

主题	关系	代表性工作
自监督学习	使用 GDL 架构作为编码器	BERT, GPT, SimCLR
生成模型	VAE, GAN, Flow 的编码器/解码器可以是 GDL 架构	VAE, GAN, Normalizing Flows
强化学习	策略/价值网络可以使用 GDL 架构	DQN, PPO, AlphaGo
优化技术	Adam, Dropout, BatchNorm 等是通用训练技巧	Adam, Dropout, BN
互信息最大化	DGI 等将 GNN 与自监督结合	DGI, InfoNCE

阅读路线图

Reading Roadmap

> ℹ️ **本书结构**
>
> 全书由 7 章组成，形成一个**从理论到应用**的完整链条。

Chapter 2: 高维学习 (Learning in High Dimensions)

> **核心问题**：为什么通用高维函数学习是不可能的?
>
> - 维度灾难的数学论证
> - 归纳偏置的概念
> - 函数正则性与复杂度度量
> - 万能逼近定理及其局限性

> **关键收获**：理解为什么我们*必须*利用数据结构——这不是可选的优化，而是*必要的*。

Chapter 3: 几何先验 (Geometric Priors)

> **核心概念**：对称性、不变性、等变性
>
> - 群论基础
> - G-不变性和 G-等变性
> - 形变稳定性
> - 尺度分离
> - **GDL 蓝图**——统一所有架构的抽象框架
>
> **关键收获**：掌握 GDL 的核心数学语言，理解"对称性 → 约束 → 更好的学习"这条逻辑链。

Chapter 4: 几何域 (Geometric Domains)

> **核心内容**：5G 的详细数学描述
>
> - 集合、网格、群、图、流形——每个域的定义和性质
> - 每个域上的信号空间
> - 每个域的对称群
> - Fourier 分析的推广
>
> **关键收获**：具体理解每种数据结构的数学本质。

Chapter 5: GDL 模型 (Geometric Deep Learning Models)

> **核心内容**：具体架构的推导和分析

- CNN 从 GDL 蓝图的推导
- GNN 的三种风格（卷积型、注意力型、消息传递型）
- Transformer 的几何理解
- 等变网络（EGNN, TFN, PaiNN）

关键收获：能够从第一性原理*推导*出各种架构，而不是死记硬背。

Chapter 6: 应用 (Problems and Applications)

核心内容：GDL 在实际领域的应用

- 计算机视觉、药物发现、粒子物理
- 蛋白质结构预测（AlphaFold）
- 推荐系统、社交网络
- 交通预测、游戏 AI
- **医疗机器人**——与我们项目的直接关联

关键收获：看到理论如何落地，理解每个应用中的对称性选择。

Chapter 7: 历史视角 (Historic Perspective)

核心内容：GDL 的思想渊源

- 从 Erlangen Program 到 Noether 定理
- CNN 的发明史
- GNN 的独立起源（AI vs 计算化学）
- 谱方法和调和分析
- WL 图同构测试与 GNN 表达力

> **关键收获**：理解思想的演进脉络，发现不同领域之间的意外联系。

与 PhysRobot 项目的关联

Connection to PhysRobot

> 🤖 **为什么 GDL 对医疗机器人仿真如此重要?**
>
> 我们的 PhysRobot 项目（医疗机器人物理仿真）是 GDL 原则的**完美应用场景**。以下是核心连接点：

PhysRobot 组件	几何域	对称群	GDL 架构
粒子系统仿真	动态图	SE(3) × Σ_n	GNS (Graph Network Simulator)
软组织变形	网格/流形	SE(3)	MeshGraphNet
刚体动力学	SE(3) 群	SE(3)	等变消息传递
器械-组织交互	异构图	SE(3)	异构 GNN
力场预测	粒子图	SE(3) (等变)	EGNN / PaiNN
能量预测	粒子图	SE(3) (不变)	SchNet / DimeNet

```
# PhysRobot 中 GDL 原则的体现
"""
GNS (Graph Network Simulator) 是一个完美的 GDL 实例:

1. 域: 动态图 (粒子 = 节点, 相互作用 = 边)
2. 对称群:
   - SE(3): 物理定律不依赖于坐标系选择
   - Σ_n: 粒子没有固有排序
3. 等变操作: 消息传递 (置换等变)
4. 关键设计:
   - 用相对位置 (r_ij = r_j - r_i) 作为边特征 → 平移不变
```

```
    - 用距离 |r_ij| 作为标量特征 → 旋转不变
    - 预测加速度而非位置 → 保持物理因果性
"""

class GNS_GDL_Blueprint:
    """
    GNS 在 GDL 蓝图中的位置:

    Signal:     x = (particle_type, position, velocity)
    Domain:     Ω = dynamic graph (k-NN in 3D space)
    Symmetry:   G = SE(3) × Σ_n
    Equivariant: message passing layers
    Invariant:  aggregation for global properties
    Coarsening: not used (single-scale)
    Output:     acceleration (SE(3)-equivariant vector)
    """
    pass
```

代码环境准备

Setting Up Your Environment

> 本教程中的代码示例使用 Python 和 PyTorch 生态系统。以下是推荐的环境配置：

```
# 创建虚拟环境
conda create -n gdl python=3.10
conda activate gdl

# 核心依赖
pip install torch torchvision torchaudio
pip install torch-geometric          # PyG: GNN 框架
pip install torch-scatter torch-sparse torch-cluster

# 可视化和数值计算
pip install numpy scipy matplotlib networkx
pip install plotly                   # 3D 可视化

# 可选: 等变网络
pip install e3nn                     # E(3)-等变网络框架
```

```
# 验证安装
python -c "
import torch
import torch_geometric
print(f'PyTorch: {torch.__version__}')
print(f'PyG: {torch_geometric.__version__}')
print(f'CUDA available: {torch.cuda.is_available()}')
print(f'MPS available: {torch.backends.mps.is_available()}')  # Apple Silicon
"
```

```
# 验证 GDL 概念的快速脚本
import torch
import torch_geometric
from torch_geometric.nn import GCNConv
from torch_geometric.data import Data

# 创建一个简单的图：5 个节点，6 条边
edge_index = torch.tensor([
    [0, 1, 1, 2, 3, 4],
    [1, 0, 2, 1, 4, 3]
], dtype=torch.long)

# 节点特征：每个节点 3 维
x = torch.randn(5, 3)

# 创建 PyG Data 对象
data = Data(x=x, edge_index=edge_index)
print(f"图：{data.num_nodes} 节点，{data.num_edges} 边")
print(f"节点特征维度：{data.x.shape}")

# GCN 层（置换等变！）
conv = GCNConv(3, 16)
out = conv(data.x, data.edge_index)
print(f"GCN 输出：{out.shape}")

# 验证置换等变性
perm = torch.randperm(5)
x_perm = x[perm]
edge_index_perm = perm[edge_index]  # 简化，实际需要更细致的处理
# 理论上：conv(x_perm, edge_perm) == conv(x, edge)[perm]
print("✅ GDL 环境配置完成！")
```

✏️ 练习题 Exercises

1. **对称性识别**：对以下数据类型，识别其自然的几何域和对称群： (a) 一组点云中的 3D 物体 (b) 化学分子 (c) 社交媒体上用户之间的关系 (d) 全球天气数据（定义在地球表面上） (e) 手术视频中器械与组织的交互
2. **参数量比较**：计算以下架构对 32×32 灰度图像的参数量： (a) 全连接层（输出 1024 维） (b) 3×3 卷积层（64 个滤波器） (c) 讨论参数量差异的原因（提示：对称性！）
3. **Erlangen 思考**：如果一个机器学习问题的数据具有 SE(3)（旋转+平移）对称性，但你使用了一个不具有 SE(3) 等变性的网络，会发生什么？从样本效率和泛化的角度讨论。
4. **5G 分类**：将以下具体问题映射到 5G 框架中最合适的域： (a) 图像语义分割 (b) 蛋白质-蛋白质对接 (c) 交通流量预测 (d) 球面上的气候模式识别 (e) 粒子物理中的 jet 分类
5. **PhysRobot 思考**：在医疗机器人仿真中，当手术器械刺入软组织时： (a) 描述系统的几何域 (b) 系统有哪些对称性？ (c) 有哪些对称性被*打破*了（例如重力方向）？ (d) 如何在网络设计中处理这种"部分对称性"？
6. **编程练习**：使用 PyTorch Geometric 创建一个简单的消息传递 GNN，应用于 Karate Club 数据集。 (a) 加载数据并可视化图结构 (b) 实现 2 层 GCN (c) 验证输出的维度和可视化节点嵌入（使用 t-SNE）
7. **思考题**：Transformer 中的位置编码（positional encoding）从 GDL 的角度意味着什么？它增加了还是减少了对称性？为什么有时候需要"减少对称性"？

📚 总目录

Chapter 2: 高维学习 →