Geometric Deep Learning

Slides from GDL Course 2021
(M. Bronstein, J. Bruna, T. Cohen, P. Velickovic)

Overview

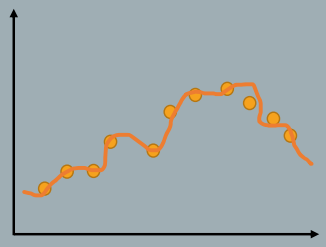
- High-Dimensional Learning
- Geometric Deep Learning
- The 5 Gs
  - Graphs
  - Grids
  - Groups
  - Gauges

HIGH-DIMENSIONAL LEARNING

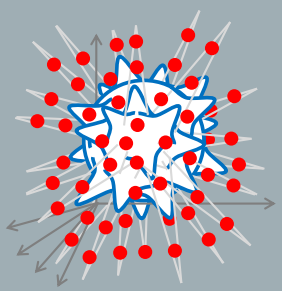
The Curse Of Dimensionality

Outline

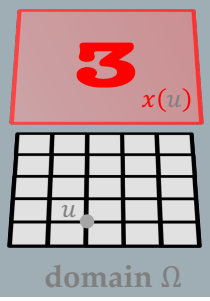
Basics of Statistical Learning: decomposition of error



The Curse of Dimensionality

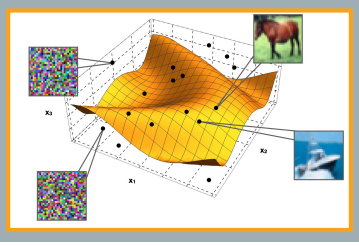


Addressing the curse

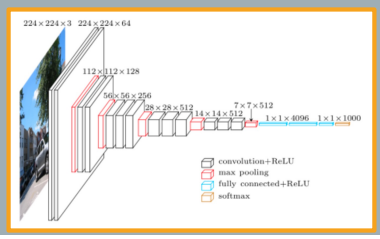


Statistical Learning in a nutshell

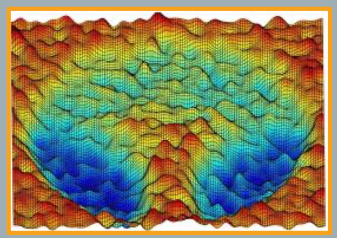
Data Distribution



Approximation Model



Error Metric



Estimation Algorithm

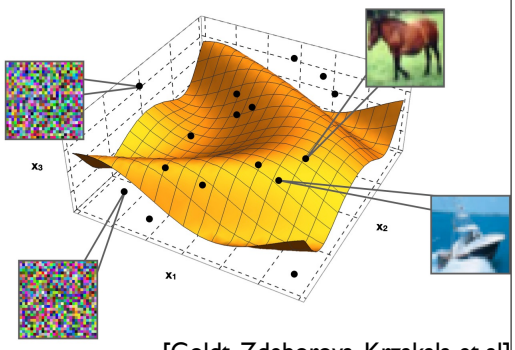
Data

(supervised learning)

$\{(x_i, y_i)\}_i$, *with* $x \in \Omega$: *data domain*, $y \in \mathbb{R}$

$x_i \sim \nu$: data distribution defined over $\mathcal{X}$.

$y_i = f^*(x_i)$ for some unknown $f^*: \mathcal{X} \to \mathbb{R}$



[Goldt, Zdeborova, Krzakala et al]

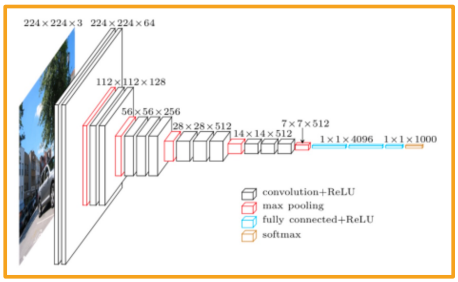
Examples:

- f*(x) : excitation energy of molecule x.
- f*(x) = P(y=1 | x) : logit in a binary classification task.

In order to analyse ML algorithms and provide guarantees, we need assumptions on both ν and f^*.

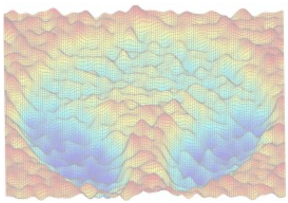
Model

- The model or *hypothesis class* is a subset $\mathcal{F} \subset \{f: \mathcal{X} \to \mathbb{R}\}$.
- Examples:
  - $\mathcal{F} = \{\, polynomials\ of\ degree\ up\ to\ k\}$
  - $\mathcal{F} = \{\, Neural\ networks\ of\ given\ architecture\ \}$
- The hypothesis class is *organized* in terms of a (non-negative) *complexity measure* $\gamma: \mathcal{F} \to \mathbb{R}$, *e.g.* a norm.
- Examples:
  - $\gamma(f)$: number of neurons in a NN.



Error Metric

- Given point-wise convex error measure $\ell(y, y')$, e.g. squared error $\ell(y,y') = |y - y'|^2$, we consider:
  - *Population Loss:* $\mathcal{R}(f) = \mathbb{E}_v[\ell\ (\ f(x), f^*(x))\ ]$
  - *Empirical Loss:* $\hat{\mathcal{R}}(f) = \frac{1}{n}\sum_i \ell\,(f(x_i), f^*(x_i))$



- **Fact**: for each $f \in \mathcal{F}$, $\hat{\mathcal{R}}(f)$ is an unbiased estimator of $\mathcal{R}(f)$, with variance $\sigma^2(f) = \frac{1}{n}\mathbb{E}\left[\ell\left(f(x), f^*(x)\right) - \mathcal{R}(f)\right]^2$

ML Algorithm

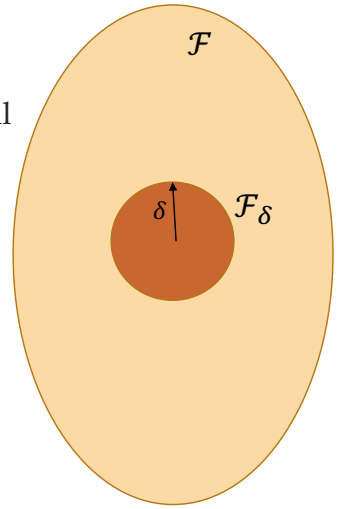
- Underlying Goal: Minimise $\mathcal{R}(f)$ having only access to $\hat{\mathcal{R}}(f)$.
- In order to control fluctuations uniformly, we consider the ball

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

(limited complexity hypothesis space)

- **Empirical Risk Minimisation** (ERM), constrained form:

$$\hat{f}_\delta = arg \min_{f \in \mathcal{F}_\delta} \hat{\mathcal{R}}(f)$$



Basic Decomposition of Error

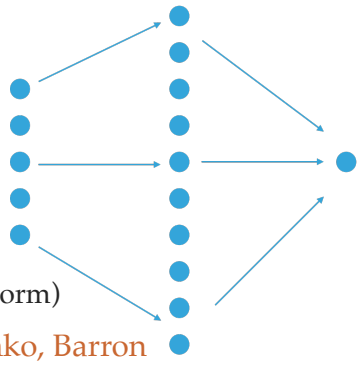
[Bottou, Bousquet]

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

- $inf_{f \in \mathcal{F}} \mathcal{R}(f) = 0$ if $\mathcal{F}$ is dense, e.g. neural networks with non-polynomial activation (Universal Approximation Theorems).
- Approximation error $\varepsilon_{appr} = inf_{f \in \mathcal{F}_\delta} \mathcal{R}(f)$: Small if hypothesis space is such that target f^* has small complexity.
- Statistical error $\varepsilon_{stat} = 2sup_{f \in \mathcal{F}_\delta} |\mathcal{R}(f) - \hat{\mathcal{R}}(f)|$: small when $\mathcal{F}_\delta$ can be covered with 'few' small balls.
- Optimization error $\varepsilon_{opt} = \hat{\mathcal{R}}(\hat{f}) - inf_{f \in \mathcal{F}_\delta} \hat{\mathcal{R}}(f)$ small when ERM can be efficiently solved.

How to simultaneously control all sources of error in the high-dimensional regime?

Curse in Approximation



- Shallow Neural Networks Hypothesis Class:

$$\mathcal{F} = \left\{ f(x) = \sum_{j \le m} a_j \rho(w_j^T x + b_j) \right\}$$

  - $\gamma(f) = m$ (number of neurons), $\gamma(f) = \sum |a_j| (\|w_j\| + |b_j|)$ (path norm)
- **Universal Approximation Theorem** [Hornik, Pinkus, Cybenko, Barron etc.]: If ρ is not a polynomial, then $\mathcal{F}$ is dense in the class of continuous functions, under uniform compact topology. Quantitative Rates?
- Rates are cursed in the *Sobolev* Class: if $f \in \mathcal{H}^s$, then $inf_{g \in \mathcal{F}} \, sup_{x \in K} |f(x) - g(x)| = \Theta(m^{-s/d})$ [DeVore, Maiorov] (*i.e., grows with dimension*)

s: number of finite derivatives of f

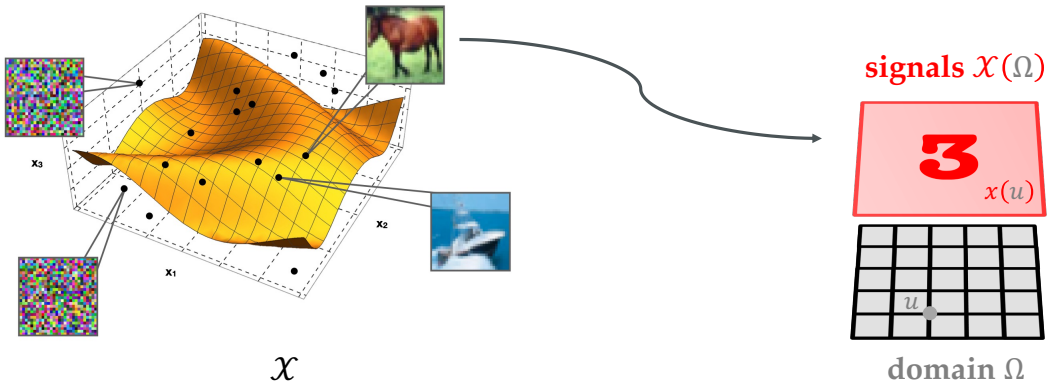
- We need to define more adapted function spaces. How?

Takeaways

- High-dimensional Learning:
  impossible without assumptions; plagued with curse of dimensionality.
- "Classic" regularity assumptions from low-dimensional mathematics:
  not appropriate.
- Input data: signals over low-dimensional geometric domains.
- Study novel notions of regularity.

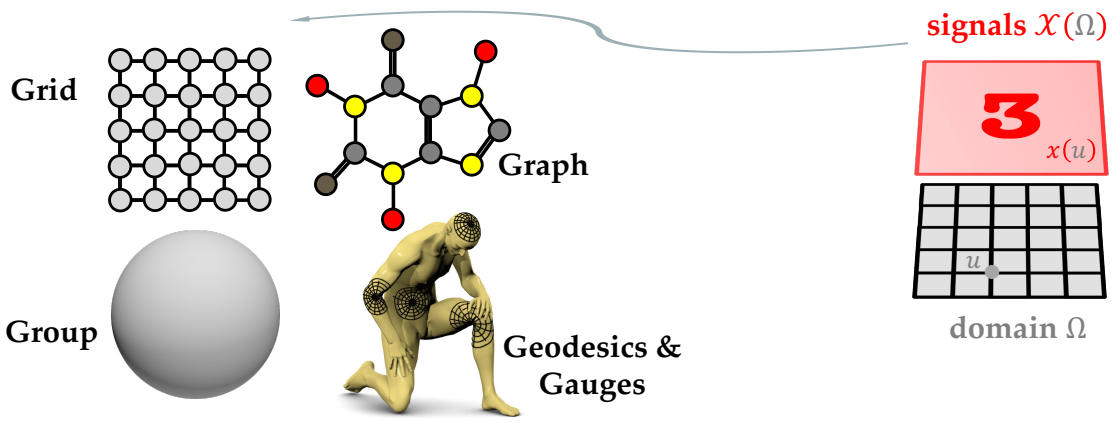
Geometric Function Spaces

- Exploit the underlying low-dimensional structure of the input high-dimensional space $\mathcal{X}$:



Towards Geometric Function Spaces

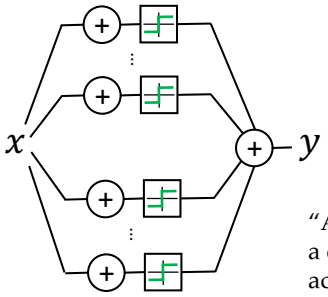
- Geometric domain Ω provides new notions of regularity that can be exploited for more efficient learning.

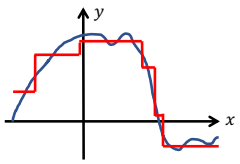


GEOMETRIC DEEP LEARNING

Fundamental Principles underlying
Deep Representation Learning

Universal Approximation

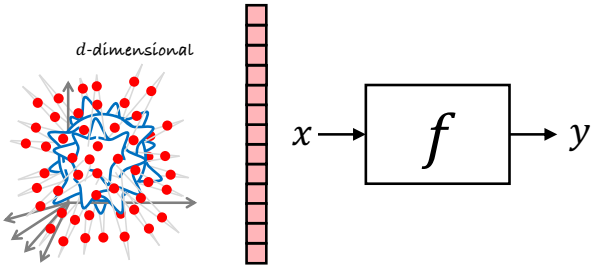




"A 2-layer perceptron can approximate a continuous function to any desired accuracy"

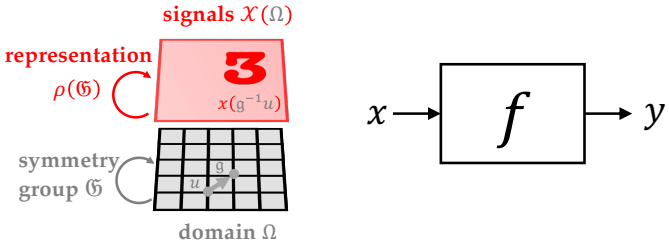
Cybenko 1989; Hornik 1991; Barron 1993; Leshno et al 1993; Maiorov 1999; Pinkus 1999

The Curse of Dimensionality



$O(\varepsilon^{-d})$ samples

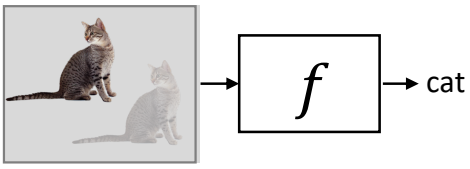
Symmetry Prior



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

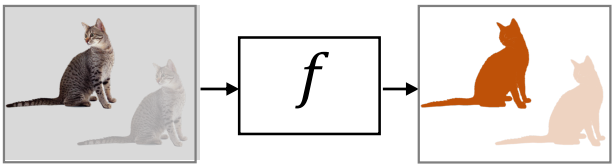
Invariant functions:
ex. Image Classification

$\mathfrak{G}$-invariance $\quad f(\rho(\mathfrak{g})x) = f(x)$

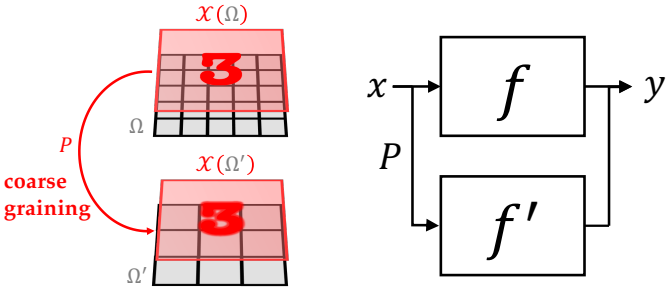


Equivariant functions:
Image Segmentation

$\mathfrak{G}$-equivariance $f(\rho(\mathfrak{g})x) = \rho(\mathfrak{g})f(x)$

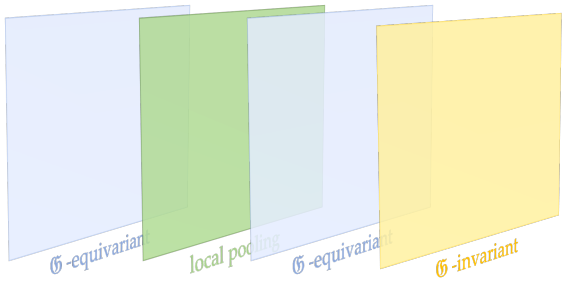


Scale Separation Prior

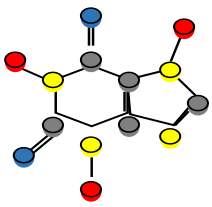


$f \approx f' \circ P$

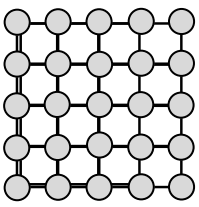
Geometric Deep Learning Blueprint



The "5G" of Geometric Deep Learning



Graphs



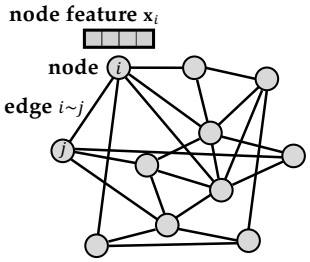
Grids

Groups

Geodesics & Gauges

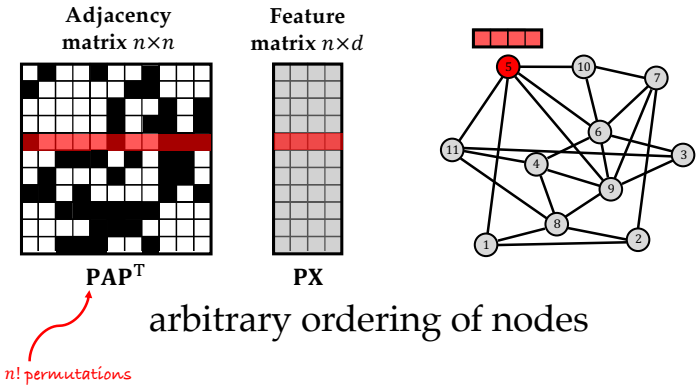
GRAPHS

Graphs



graph

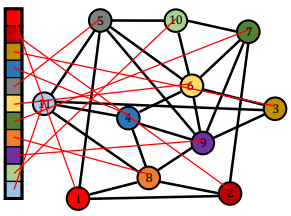
Key Structural Properties of Graphs



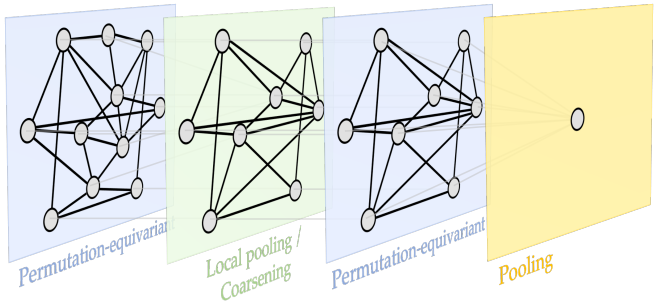
Equivariant Graph Functions

permutation-equivariant

$$\mathbf{F}(\mathbf{PX}, \mathbf{PAP}^{\top}) = \mathbf{PF}(\mathbf{X}, \mathbf{A})$$



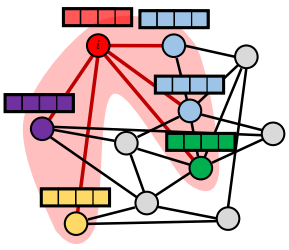
Graph Neural Networks



A General Blueprint for Constructing Graph Functions

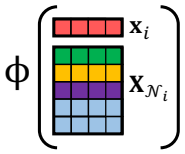
multiset of neighbour features

$$\mathbf{X}_{\mathcal{N}_i} = \{\!\{\mathbf{x}_{j \in \mathcal{N}_i}\}\!\}$$

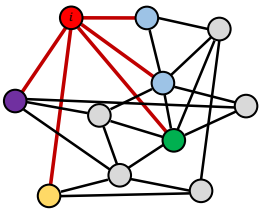


A General Blueprint for Constructing Graph Functions

local function



permutation invariant



Message Passing

permutation-invariant aggregation operator, e.g. sum

$$f(\mathbf{x}_i) = \phi\left(\mathbf{x}_i, \square_{j \in \mathcal{N}_i} \psi(\mathbf{x}_i, \mathbf{x}_j)\right)$$

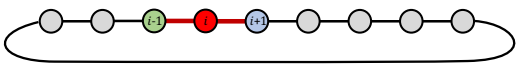
new feature of node i

learnable functions

Gilmer et al. 2017 (MPNN); Battaglia et al 2018; Wang et al. 2018

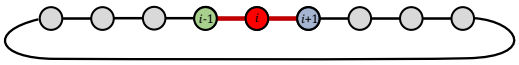
GRIDS

Grids vs Graphs



fixed neighbourhood structure

Grids vs Graphs



linear local aggregation function

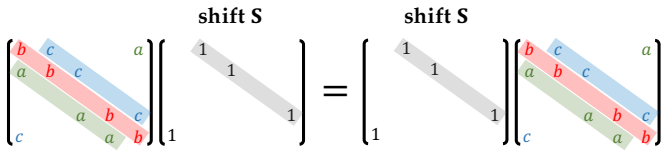
$$f(\mathbf{x}_i) = a\mathbf{x}_{i-1} + b\mathbf{x}_i + c\mathbf{x}_{i+1}$$

Convolution

$$f(\mathbf{X}) = \begin{bmatrix} b & c & & & a \\ a & b & c & & \\ & a & b & c & \\ & & a & b & c \\ c & & & a & b \end{bmatrix} \begin{bmatrix} \\ \mathbf{X} \\ \\ \\ \\ \end{bmatrix}$$

circulant matrix = convolution

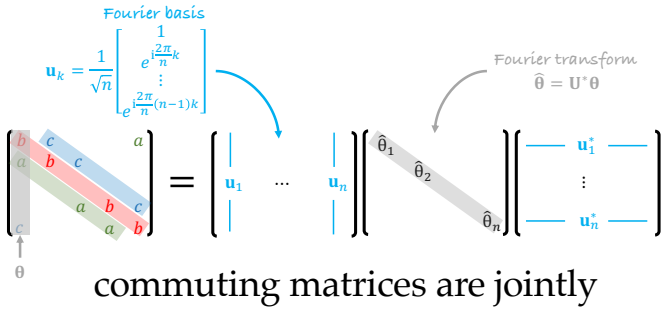
Deriving Convolution from Symmetry



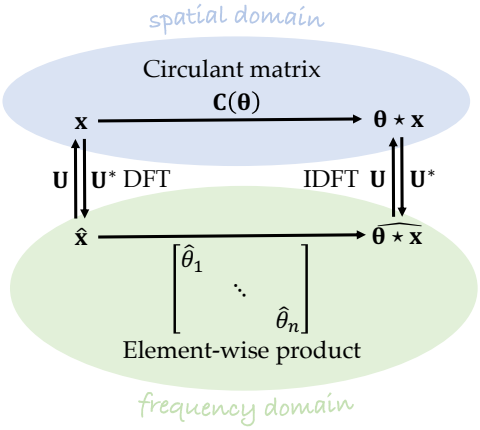
convolution ⟺ shift-equivariant

convolution **emerges** from translation symmetry

Deriving Fourier Transform from Symmetry



commuting matrices are jointly diagonalisable by Fourier Transform


spatial domain
Circulant matrix
$\mathbf{C}(\boldsymbol{\theta})$
$\mathbf{x}$
$\boldsymbol{\theta} \star \mathbf{x}$
$\mathbf{U}$ $\mathbf{U}^*$ DFT
IDFT $\mathbf{U}$ $\mathbf{U}^*$
$\hat{\mathbf{x}}$
$\widehat{\boldsymbol{\theta} \star \mathbf{x}}$
$\begin{bmatrix} \hat{\theta}_1 & & \\ & \ddots & \\ & & \hat{\theta}_n \end{bmatrix}$
Element-wise product
frequency domain

GROUPS

Convolution, revisited

convolution = matching shifted filter

$$(x \star \psi)(u) = \langle x, T_u \psi \rangle = \int_{-\infty}^{+\infty} x(v)\psi(u - v)\mathrm{d}v$$

shift vector — shift operator

domain Ω = symmetry group $\mathfrak{G}$

Group Convolution

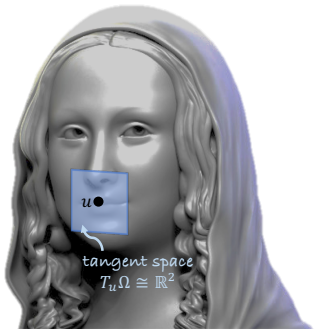
convolution = matching transformed filter

$$(x \star \psi)(\mathfrak{g}) = \langle x, \rho(\mathfrak{g})\psi \rangle = \int_{\Omega} x(v)\psi(\mathfrak{g}^{-1}v)\mathrm{d}v$$

group element — group representation

GEODESICS & GAUGES

Manifolds



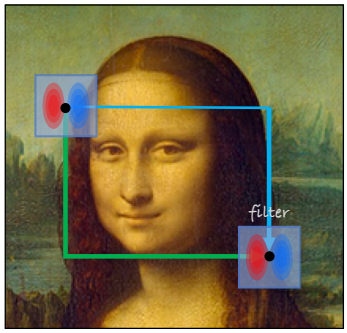
manifold Ω

manifold = locally Euclidean space

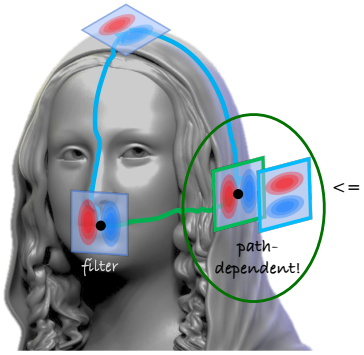
Riemannian metric = local length/direction

Isometry = metric-preserving deformation

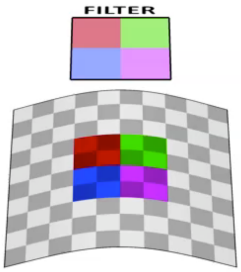
Parallel Transport



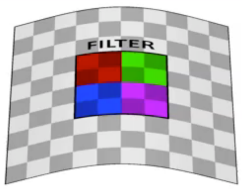
Euclidean



Non-Euclidean

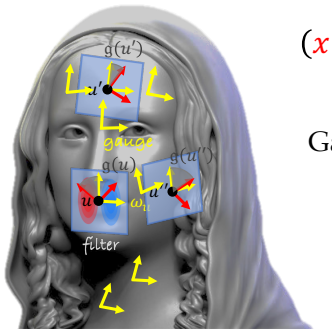


Euclidean (extrinsic) convolution



Geometric (intrinsic) convolution

Gauge-equivariant CNNs



manifold Ω
structure group $\mathfrak{G}$

Cohen et al. 2019; Weiler et al. 2021

$$(x \star \psi)(u) = \int_{\mathbb{R}^2} \psi(\mathbf{v}) x(\exp_u \omega_u \mathbf{v}) \mathrm{d}\mathbf{v}$$

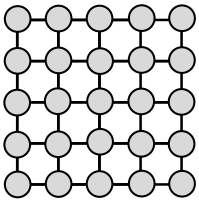
Gauge defined up to gauge transformation

$$\mathfrak{g}: \Omega \to \mathfrak{G}$$

- "Naked" manifold: $\mathrm{GL}(2)$
- Manifold+orientation: $\mathrm{GL}^+(2)$
- Manifold+volume: $\mathrm{SL}(2)$
- Manifold+metric: $\mathrm{O}(2)$
- Manifold+metric+orientation: $\mathrm{SO}(2)$
- Manifold+frame field: $\{\mathrm{id}\}$

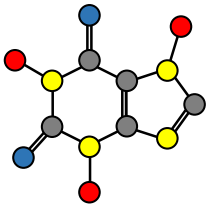
Recap

The "5G" of Geometric Deep Learning



Grids

Groups

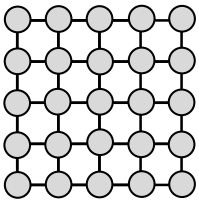


Graphs

Geodesics & Gauges

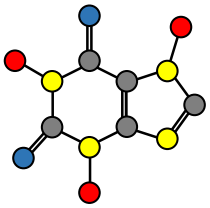
Data Networks

Explicit Representation for Signals



Images & Sequences

Homogeneous spaces

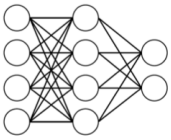


Graphs & Sets

Manifolds, Meshes & Geometric graphs

Coordinate Networks

Implicit Representation for Signals

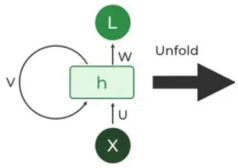


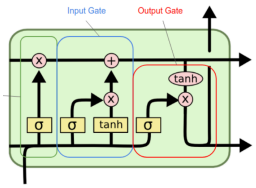
General Media Objects

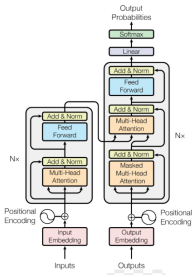
Other Networks

Sequences, Time Dependent

Language Models





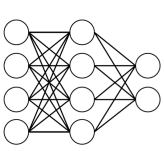


RNN

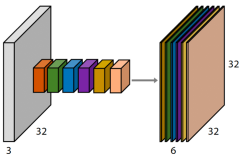
LSTM

Transformer

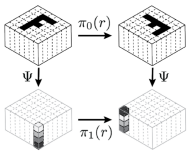
NN Architectures & Symmetries



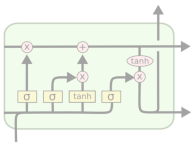
MLP / Perceptrons
Function regularity



CNNs
Translation

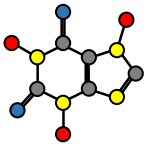


Group-CNNs
Translation+Rotation,
Global groups



LSTMs
Time warping

DeepSets / Transformers
Permutation



GNNs
Permutation

Intrinsic CNNs
Isometry / Gauge choice

Aftertought

Models Ahoy !

- Discriminative Models
  - Analysis
- Descriptive Models
  - Representation
- Generative Models
  - Synthesis
- Simulation Models
  - Action

World Models