MLSS 2021 Summer School

GEOMETRIC DEEP LEARNING

Michael Bronstein

Imperial College London / IDSIA / Twitter

Based on joint work with J. Bruna, T. Cohen, P. Veličković
Slides adapted from AMMI Course

Fundamental Principles underlying Deep Representation Learning architectures

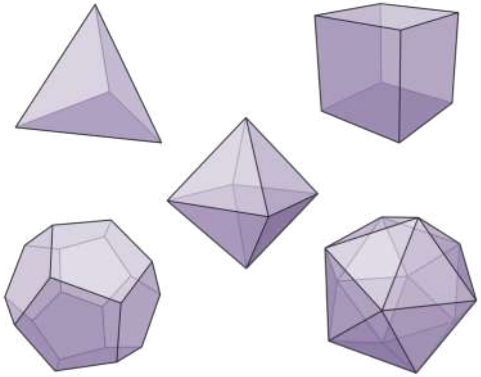
Symmetry

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

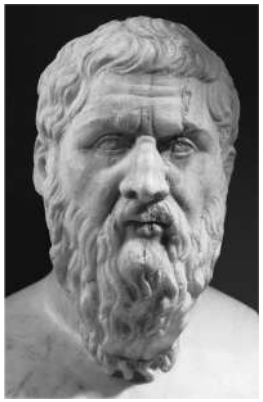
H. Weyl

H. Weyl 1952

συμμετρία

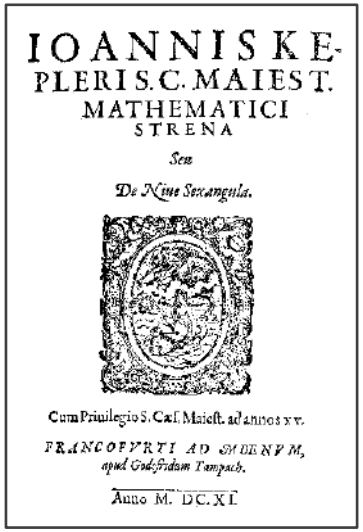


Platonic solids



Plato

~370 BC



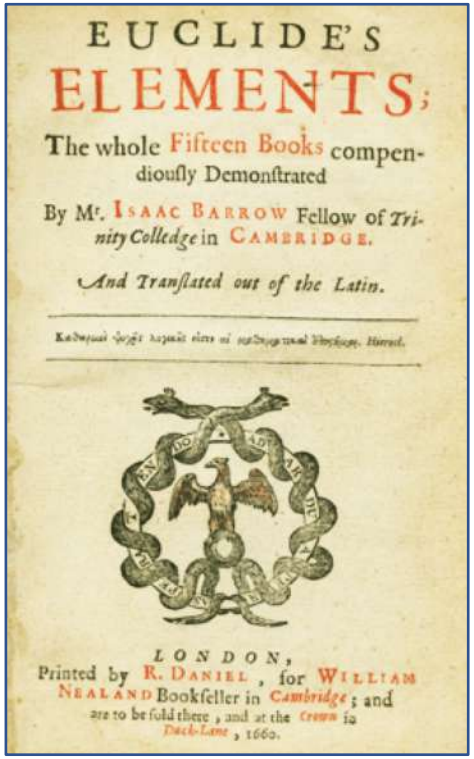
"On the six-cornered snowflake"

J. Kepler

1611

Euclidean geometry

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


EUCLIDE'S
ELEMENTS;
The whole Fifteen Books compendiously Demonstrated
By Mr. ISAAC BARROW Fellow of Trinity Colledge in CAMBRIDGE.
And Translated out of the Latin.
LONDON,
Printed by R. DANIEL, for WILLIAM NEALAND Bookseller in Cambridge; and are to be sold there, and at the Crown in Duck-Lane, 1660.

Euclid

~300 BC

End of Euclid's Monopoly

J. V. Poncelet

1822

N. Lobachevsky

1826

C. F. Gauss

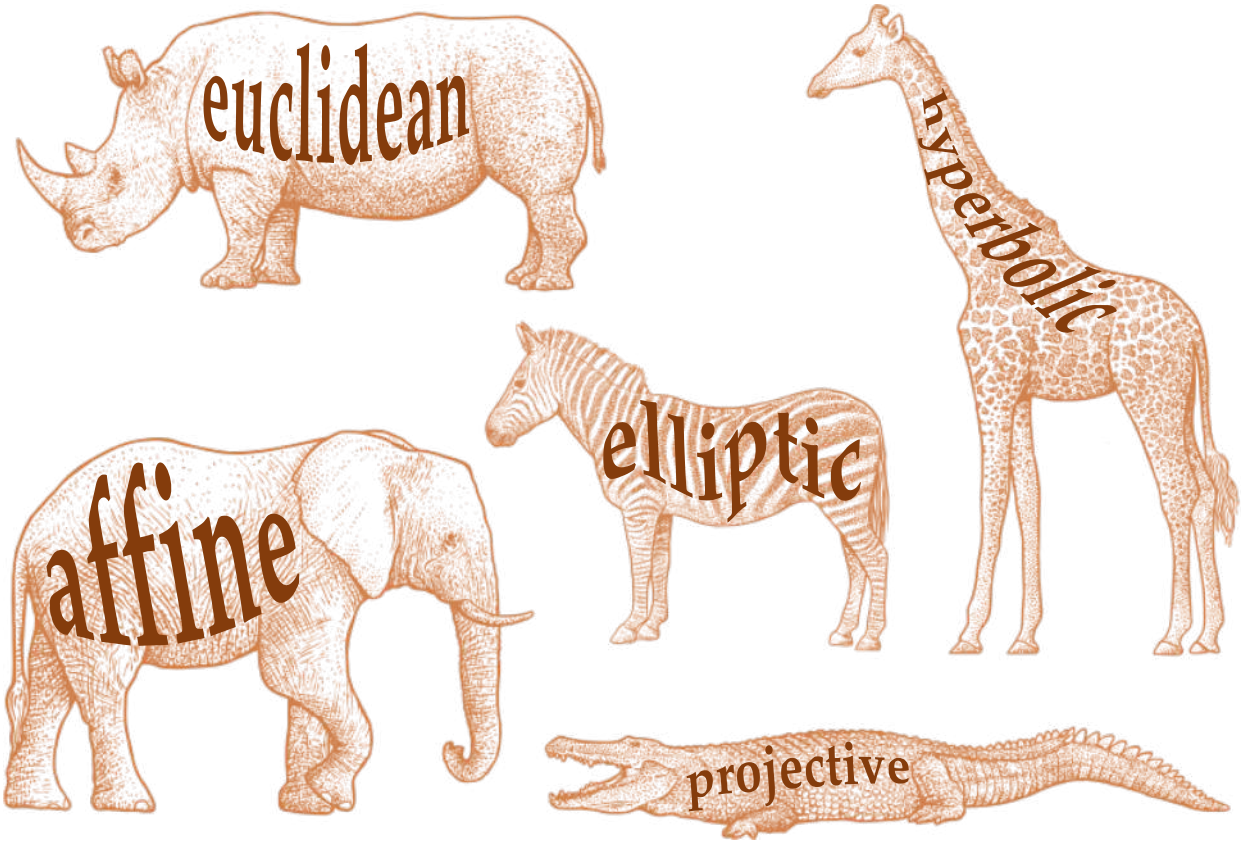
J. Bolyai

1832

B. Riemann

1856

19th Century Zoo of Geometries



The Erlangen Programme

"Given a [homogeneous] manifold and a transformation group acting [transitively] on it, to investigate those properties of figures on that manifold which are invariant under transformations of that group"

Vergleichende Betrachtungen

über

neuere geometrische Forschungen

von

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

Programm

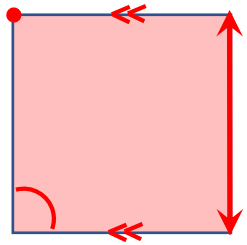
zum Eintritt in die philosophische Facultät und den Senat
der k. Friedrich-Alexanders-Universität
zu Erlangen.

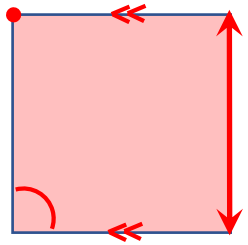
Erlangen.
Verlag von Andreas Deichert.
1872.

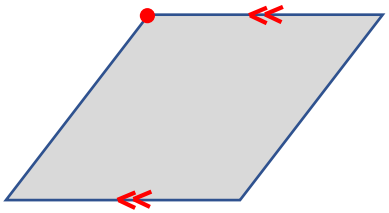
F. Klein

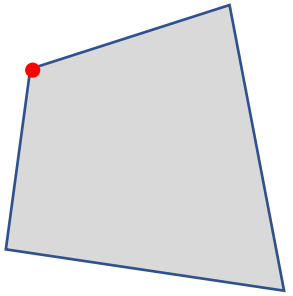
1872

The Erlangen Programme



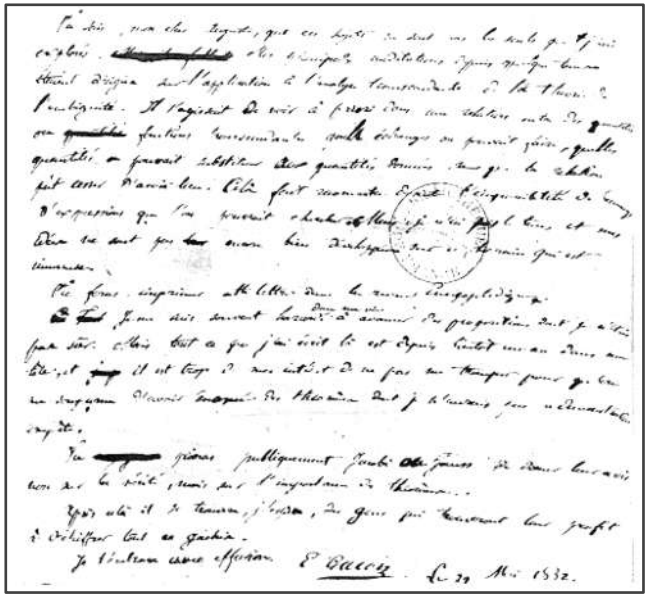






	Euclidean	**Affine**	**Projective**
angle	+	—	—
distance	+	—	—
area	+	—	—
parallelism	+	+	—
intersection	+	+	+

Group Theory



Mon cher ami — Galois' last letter written on the night before his duel

E. Galois

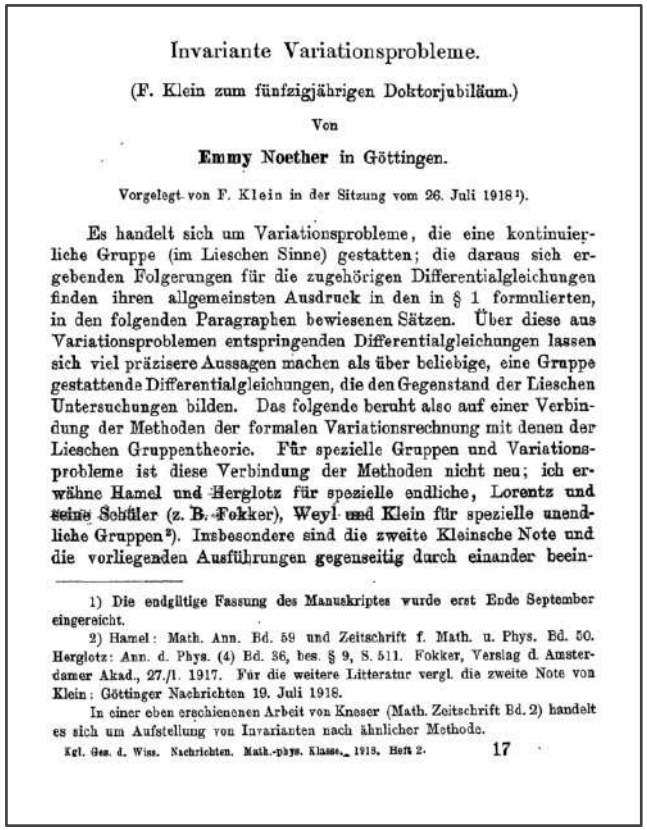
F. Klein

S. Lie

1832

Noether's Theorem

"Every [differentiable] symmetry of the action of a physical system [with conservative forces] has a corresponding conservation law"



Invariante Variationsprobleme.

(F. Klein zum fünfzigjährigen Doktorjubiläum.)

Von

Emmy Noether in Göttingen.

Vorgelegt von F. Klein in der Sitzung vom 26. Juli 1918[1]).

Es handelt sich um Variationsprobleme, die eine kontinuierliche Gruppe (im Lieschen Sinne) gestatten; die daraus sich ergebenden Folgerungen für die zugehörigen Differentialgleichungen finden ihren allgemeinsten Ausdruck in den in § 1 formulierten, in den folgenden Paragraphen bewiesenen Sätzen. Über diese aus Variationsproblemen entspringenden Differentialgleichungen lassen sich viel präzisere Aussagen machen als über beliebige, eine Gruppe gestattende Differentialgleichungen, die den Gegenstand der Lieschen Untersuchungen bilden. Das folgende beruht also auf einer Verbindung der Methoden der formalen Variationsrechnung mit denen der Lieschen Gruppentheorie. Für spezielle Gruppen und Variationsprobleme ist diese Verbindung der Methoden nicht neu; ich erwähne Hamel und Herglotz für spezielle endliche, Lorentz und seine Schüler (z. B. Fokker), Weyl und Klein für spezielle unendliche Gruppen[2]). Insbesondere sind die zweite Kleinsche Note und die vorliegenden Ausführungen gegenseitig durch einander beein-

1) Die endgiltige Fassung des Manuskriptes wurde erst Ende September eingereicht.

2) Hamel: Math. Ann. Bd. 59 und Zeitschrift f. Math. u. Phys. Bd. 50. Herglotz: Ann. d. Phys. (4) Bd. 36, bes. § 9, S. 511. Fokker, Verslag d. Amsterdamer Akad., 27./1. 1917. Für die weitere Litteratur vergl. die zweite Note von Klein: Göttinger Nachrichten 19. Juli 1918.

In einer eben erschienenen Arbeit von Kneser (Math. Zeitschrift Bd. 2) handelt es sich um Aufstellung von Invarianten nach ähnlicher Methode.

Kgl. Ges. d. Wiss. Nachrichten. Math.-phys. Klasse. 1918. Heft 2. 17

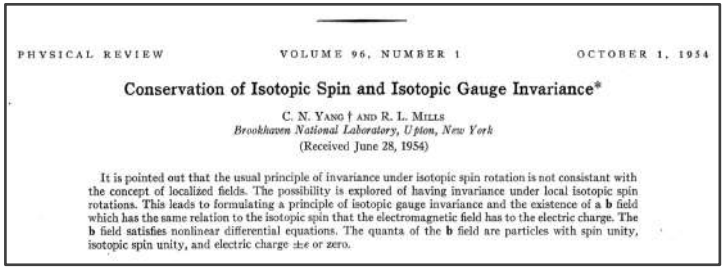
E. Noether

1918

Gauge Invariance

H. Weyl

1929



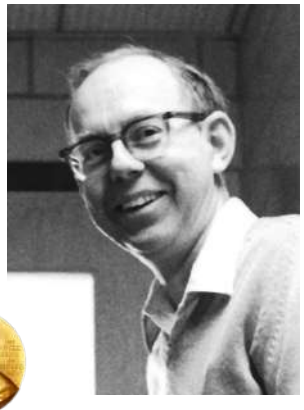
PHYSICAL REVIEW VOLUME 96, NUMBER 1 OCTOBER 1, 1954

Conservation of Isotopic Spin and Isotopic Gauge Invariance*

C. N. YANG † AND R. L. MILLS
Brookhaven National Laboratory, Upton, New York
(Received June 28, 1954)

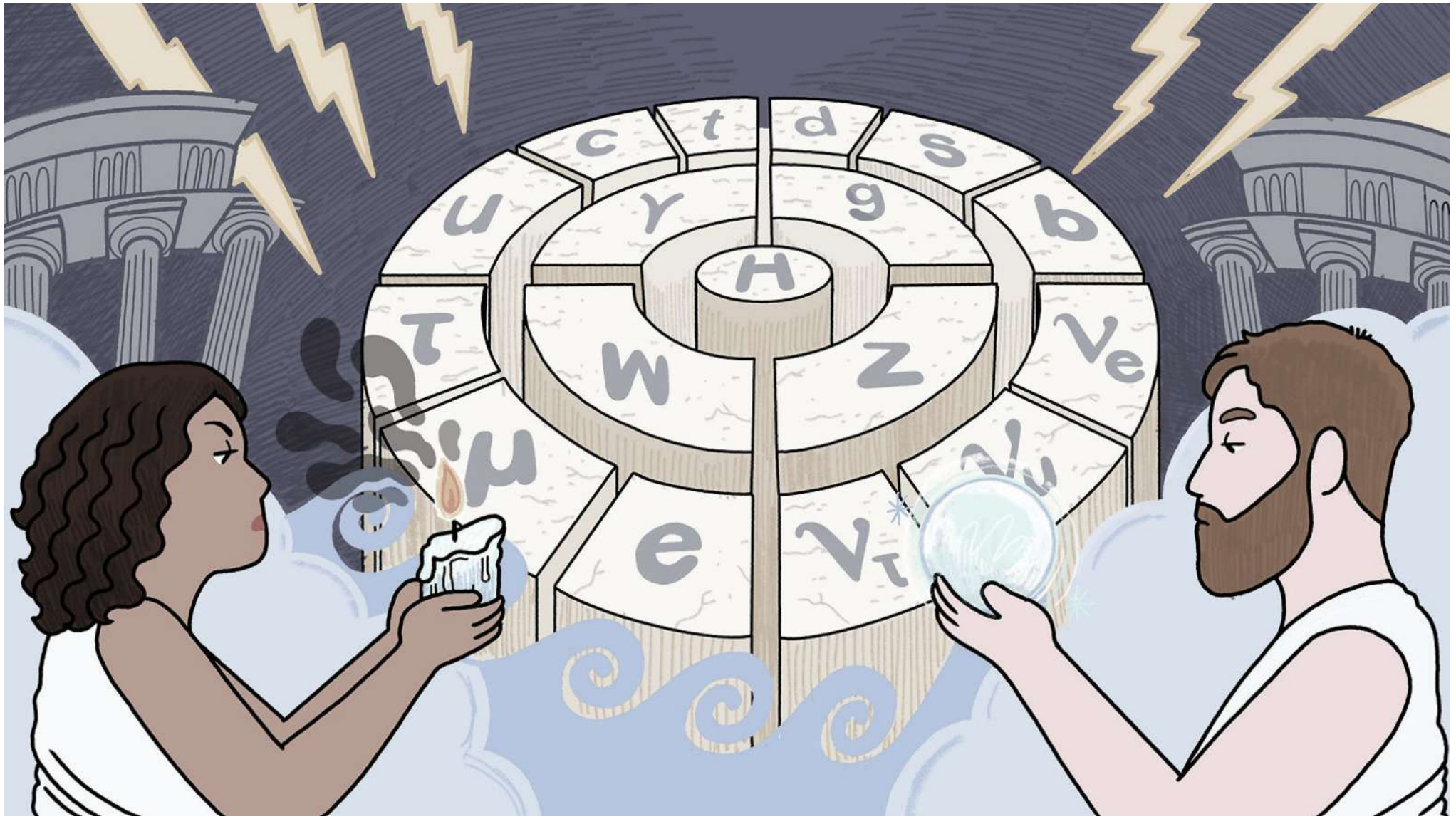
It is pointed out that the usual principle of invariance under isotopic spin rotation is not consistant with the concept of localized fields. The possibility is explored of having invariance under local isotopic spin rotations. This leads to formulating a principle of isotopic gauge invariance and the existence of a **b** field which has the same relation to the isotopic spin that the electromagnetic field has to the electric charge. The **b** field satisfies nonlinear differential equations. The quanta of the **b** field are particles with spin unity, isotopic spin unity, and electric charge $\pm e$ or zero.

C. N. Yang



R. L. Mills

1954


c
t
d
s
u
γ
g
b
H
τ
W
Z
νe
μ
νμ
e
νt

“It is only slightly overstating the case to say that Physics is the study of symmetry”

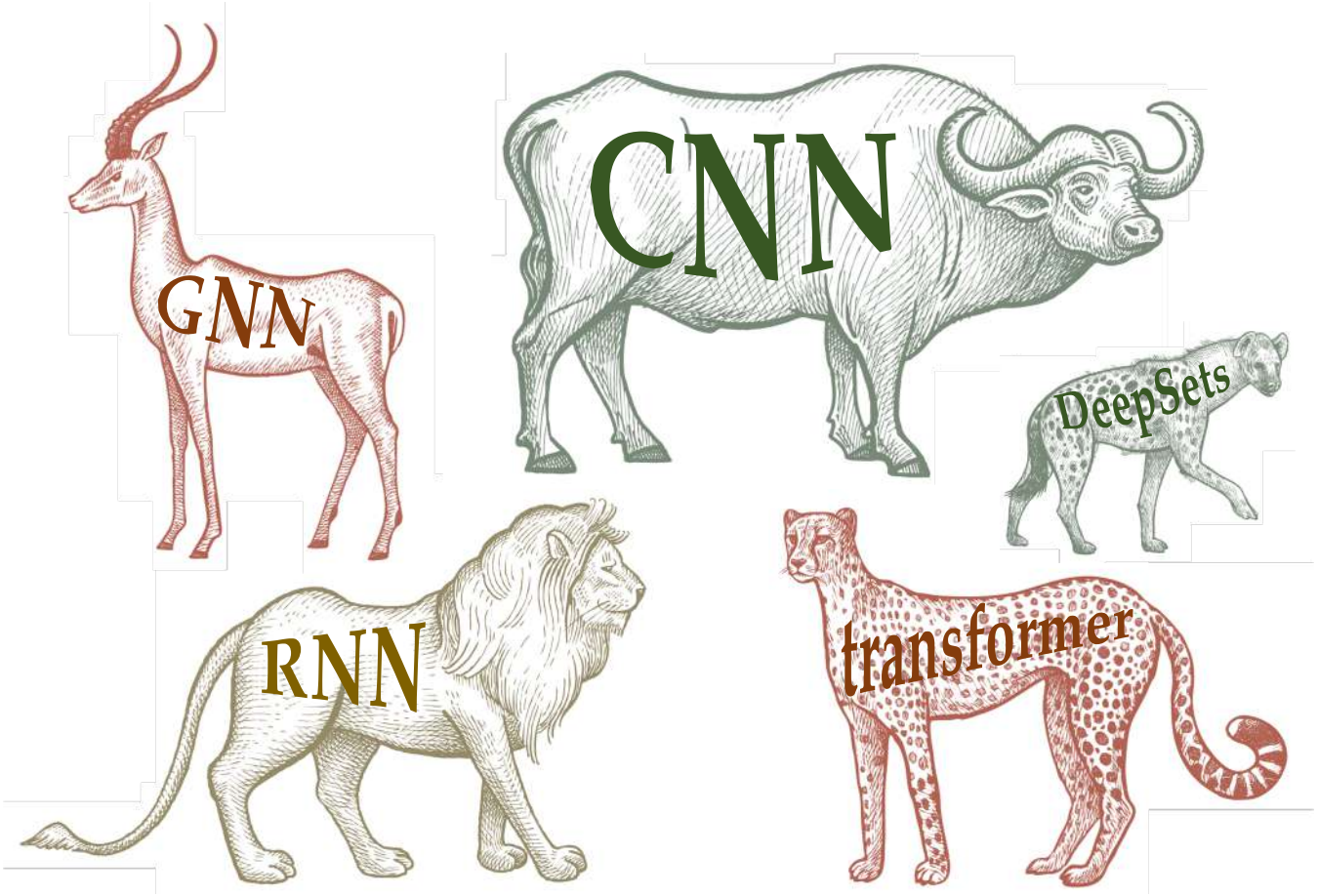
P. Anderson

Anderson 1972

?

REVOLUTION IN DATA SCIENCE

20th Century Zoo of Neural Network Architectures



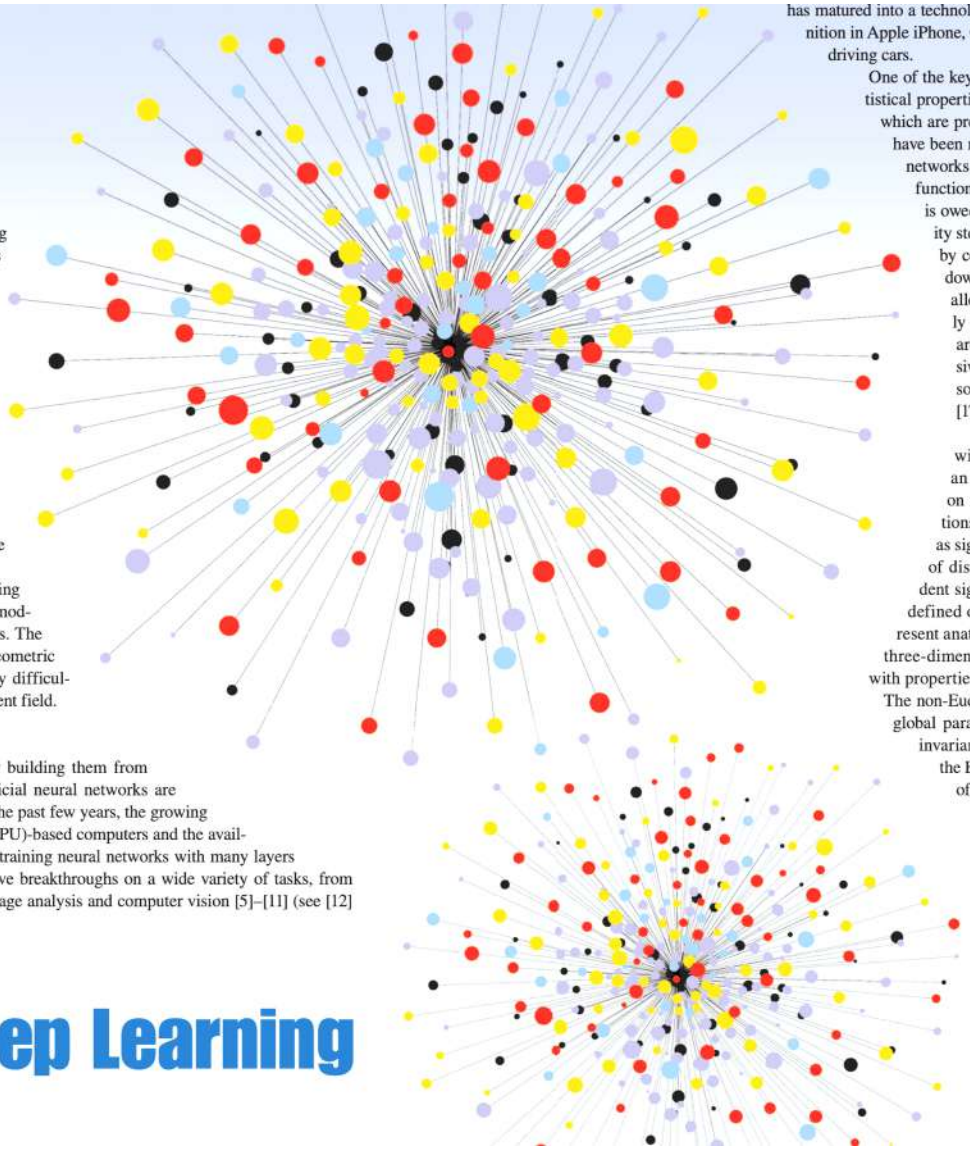
The Erlangen Programme of ML

Geometric Deep Learning

Michael M. Bronstein, Joan Bruna, Yann LeCun,
Arthur Szlam, and Pierre Vandergheynst

Geometric Deep Learning

Going beyond Euclidean data



Many scientific fields study data with an underlying structure that is non-Euclidean. Some examples include social networks in computational social sciences, sensor networks in communications, functional networks in brain imaging, regulatory networks in genetics, and meshed surfaces in computer graphics. In many applications, such geometric data are large and complex (in the case of social networks, on the scale of billions) and are natural targets for machine-learning techniques. In particular, we would like to use deep neural networks, which have recently proven to be powerful tools for a broad range of problems from computer vision, natural-language processing, and audio analysis. However, these tools have been most successful on data with an underlying Euclidean or grid-like structure and in cases where the invariances of these structures are built into networks used to model them.

Geometric deep learning is an umbrella term for emerging techniques attempting to generalize (structured) deep neural models to non-Euclidean domains, such as graphs and manifolds. The purpose of this article is to overview different examples of geometric deep-learning problems and present available solutions, key difficulties, applications, and future research directions in this nascent field.

Overview of deep learning

Deep learning refers to learning complicated concepts by building them from simpler ones in a hierarchical or multilayer manner. Artificial neural networks are popular realizations of such deep multilayer hierarchies. In the past few years, the growing computational power of modern graphics processing unit (GPU)-based computers and the availability of large training data sets have allowed successfully training neural networks with many layers and degrees of freedom (DoF) [1]. This has led to qualitative breakthroughs on a wide variety of tasks, from speech recognition [2], [3] and machine translation [4] to image analysis and computer vision [5]–[11] (see [12] has matured into a technology that is widely used in commercial applications, including Siri speech recognition in Apple iPhone, Google text translation, and Mobileye vision-based technology for autonomously driving cars.

One of the key reasons for the success of deep neural networks is their ability to leverage statistical properties of the data, such as stationarity and compositionality through local statistics, which are present in natural images, video, and speech [14], [15]. These statistical properties have been related to physics [16] and formalized in specific classes of convolutional neural networks (CNNs) [17]–[19]. In image analysis applications, one can consider images as functions on the Euclidean space (plane), sampled on a grid. In this setting, stationarity is owed to shift invariance, locality is due to the local connectivity, and compositionality stems from the multiresolution structure of the grid. These properties are exploited by convolutional architectures [20], which are built of alternating convolutional and downsampling (pooling) layers. The use of convolutions has a twofold effect. First, it allows extracting local features that are shared across the image domain and greatly reduces the number of parameters in the network with respect to generic deep architectures (and thus also the risk of overfitting), without sacrificing the expressive capacity of the network. Second, the convolutional architecture itself imposes some priors about the data, which appear very suitable especially for natural images [17]–[19], [21].

While deep-learning models have been particularly successful when dealing with speech, image, and video signals, in which there are an underlying Euclidean structure, recently there has been a growing interest in trying to apply learning on non-Euclidean geometric data. Such kinds of data arise in numerous applications. For instance, in social networks, the characteristics of users can be modeled as signals on the vertices of the social graph [22]. Sensor networks are graph models of distributed interconnected sensors, whose readings are modeled as time-dependent signals on the vertices. In genetics, gene expression data are modeled as signals defined on the regulatory network [23]. In neuroscience, graph models are used to represent anatomical and functional structures of the brain. In computer graphics and vision, three-dimensional (3-D) objects are modeled as Riemannian manifolds (surfaces) endowed with properties such as color texture.

The non-Euclidean nature of such data implies that there are no such familiar properties as global parameterization, common system of coordinates, vector space structure, or shift invariance. Consequently, basic operations like convolution that are taken for granted in the Euclidean case are even not well defined on non-Euclidean domains. The purpose of this article is to show different methods of translating the key ingredients of successful deep-learning methods, such as CNNs, to non-Euclidean data.

Geometric learning problems

Broadly speaking, we can distinguish between two classes of geometric learning problems. In the first class of problems, the goal is to characterize the structure of the data. The second class of problems deals with analyzing functions defined on a given non-Euclidean domain. These two classes are related, because understanding the properties of functions defined on a domain conveys certain information about the domain, and vice versa, the structure of the domain imposes certain properties on the functions on it.

Structure of the domain

As an example of the first class of problems, assume to be given a set of data points with some underlying low-dimensional structure embedded into a high-dimensional Euclidean space. Recovering that low-dimensional structure

Geometric Deep Learning
Grids, Groups, Graphs, Geodesics, and Gauges

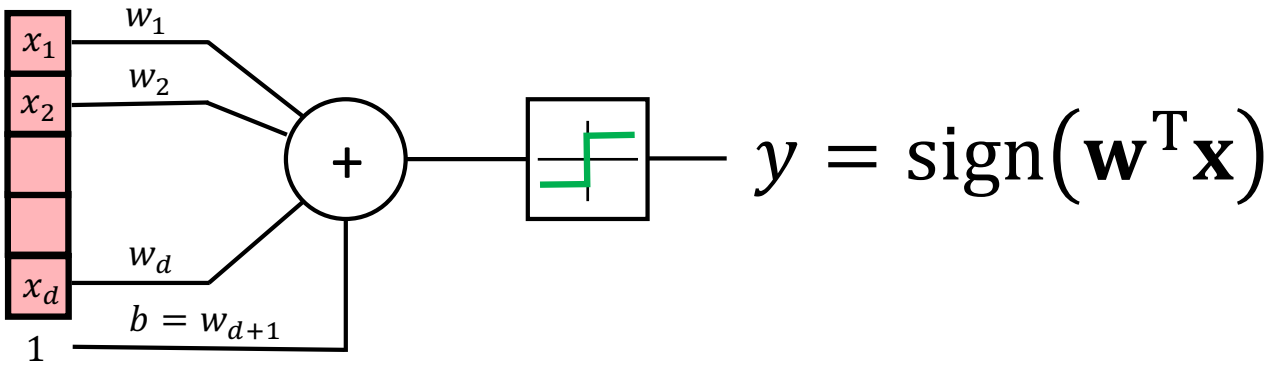
Michael M. Bronstein[1], Joan Bruna[2], Taco Cohen[3], Petar Veličković[4]

May 4, 2021

www.geometricdeeplearning.com

Supervised ML = Function Approximation

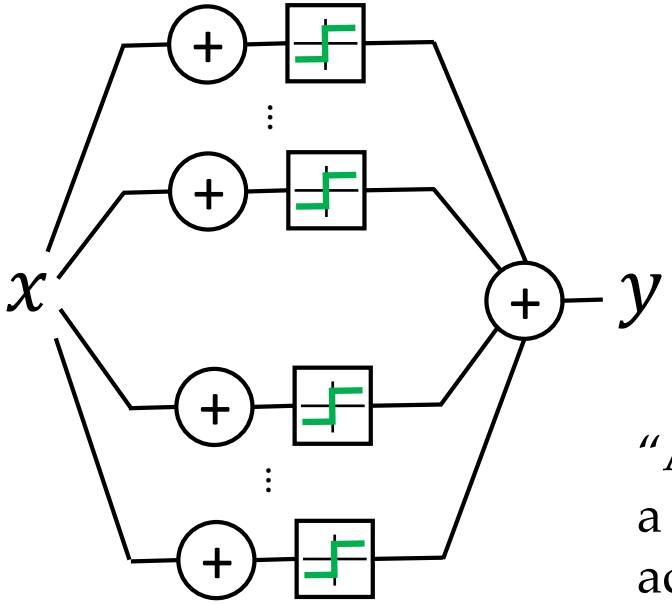
Perceptrons

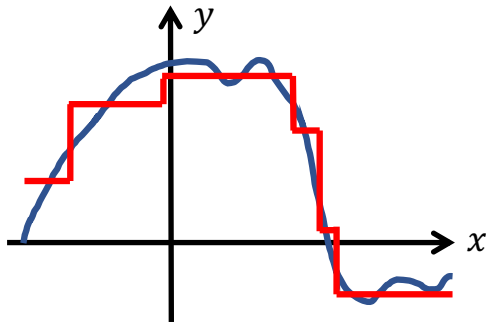


Perceptron: the simplest neural network

Rosenblatt 1957

Universal Approximation

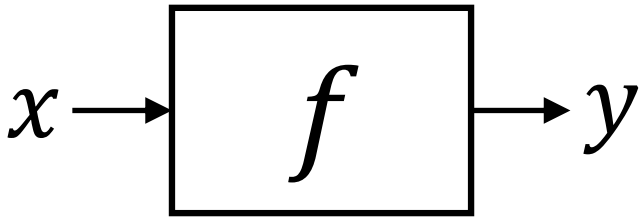




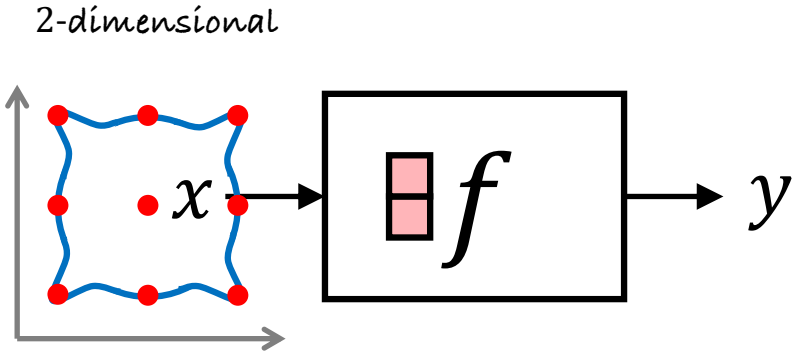
"A 2-layer perceptron can approximate a continuous function to any desired accuracy"

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

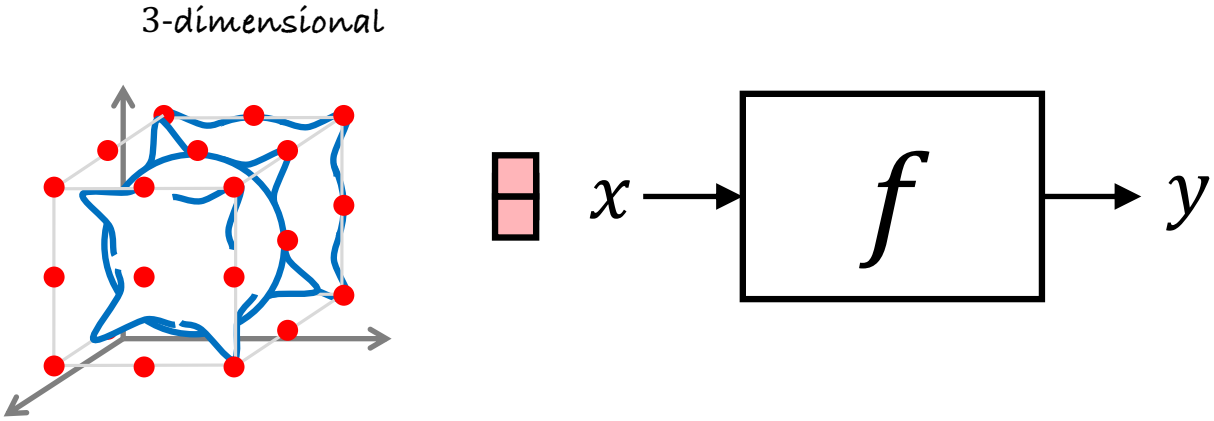
The Curse of Dimensionality



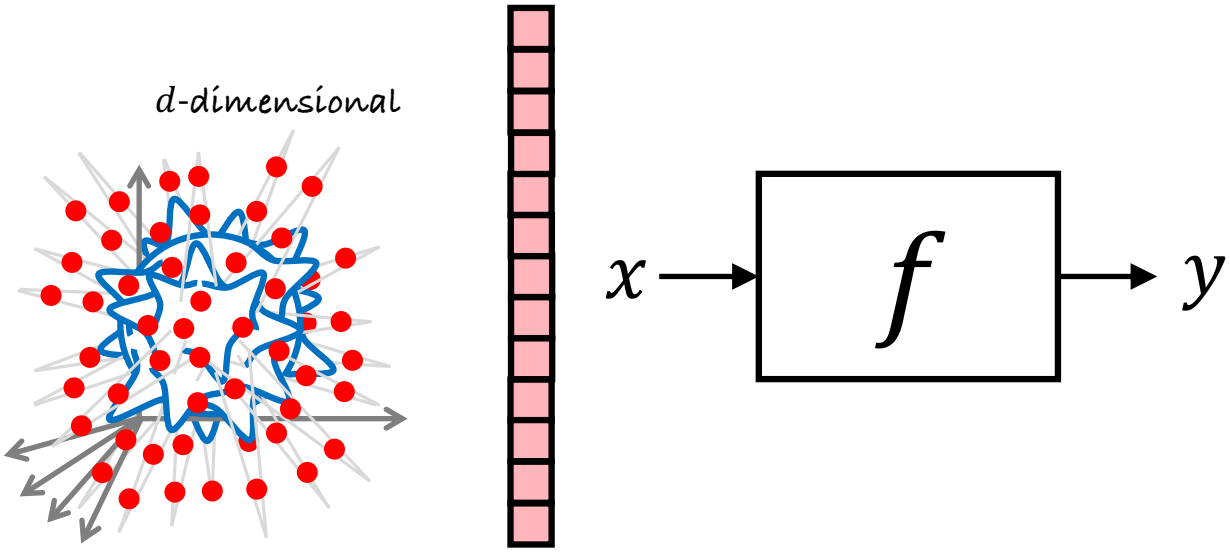
The Curse of Dimensionality



The Curse of Dimensionality

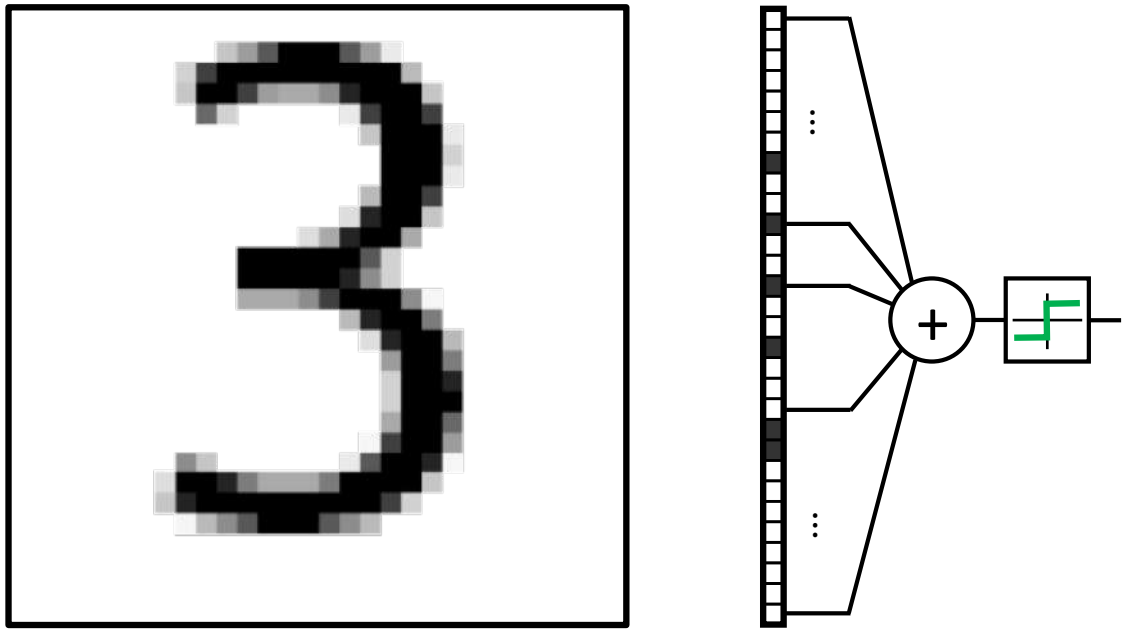


The Curse of Dimensionality



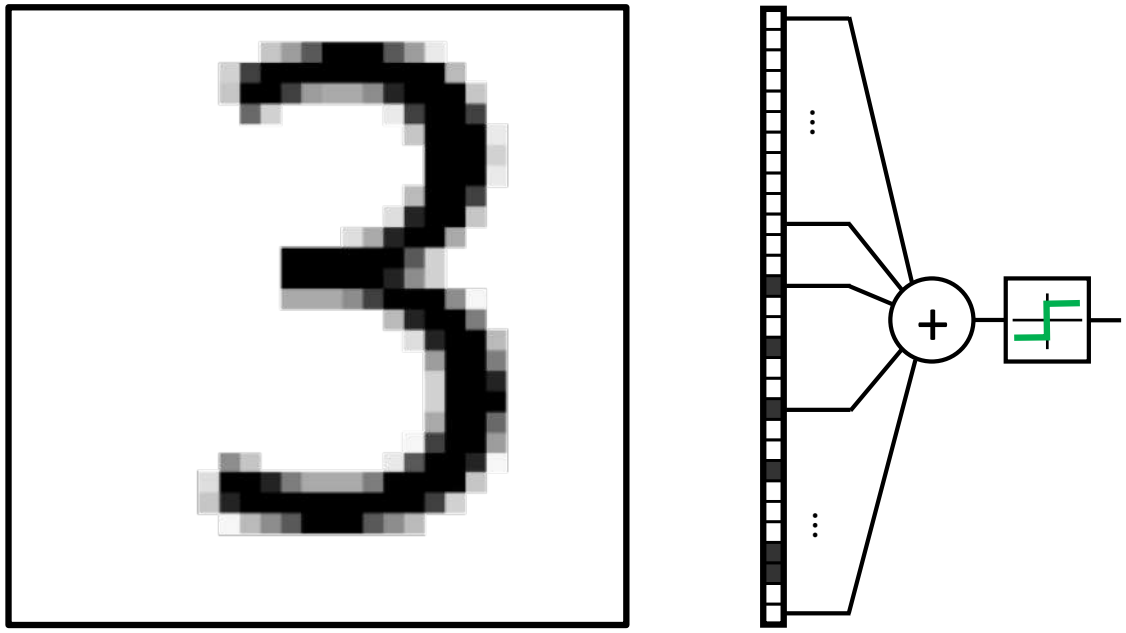
$O(\varepsilon^{-d})$ samples

Computer Vision



input image **input vector**

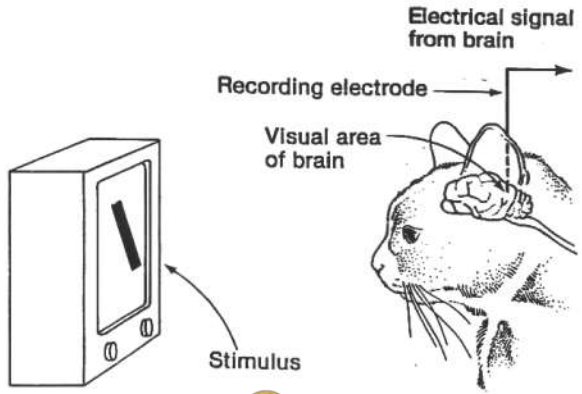
Computer Vision



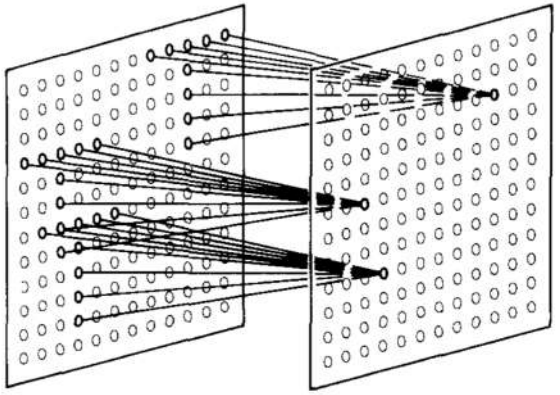
input image input vector

must learn shift invariance from data!

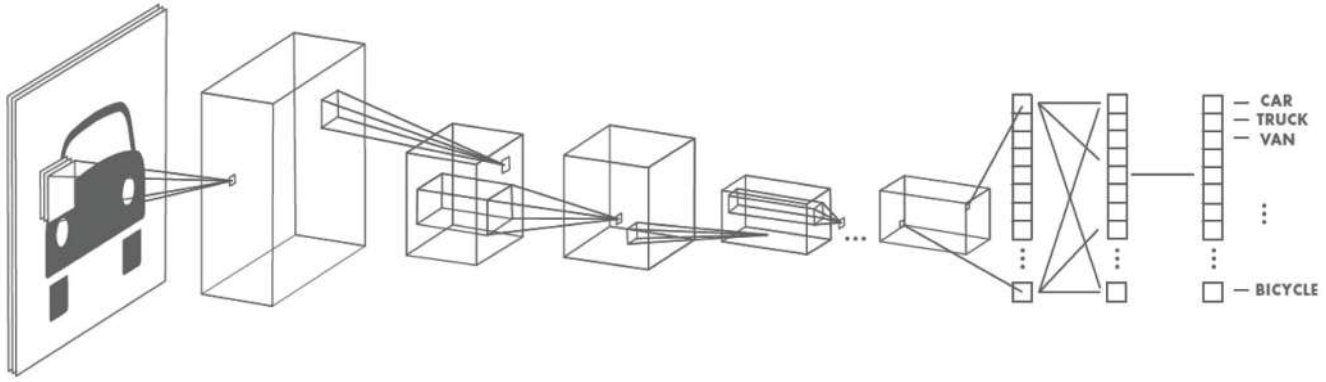
Computer Vision



Hubel, Wiesel 1962;  1981

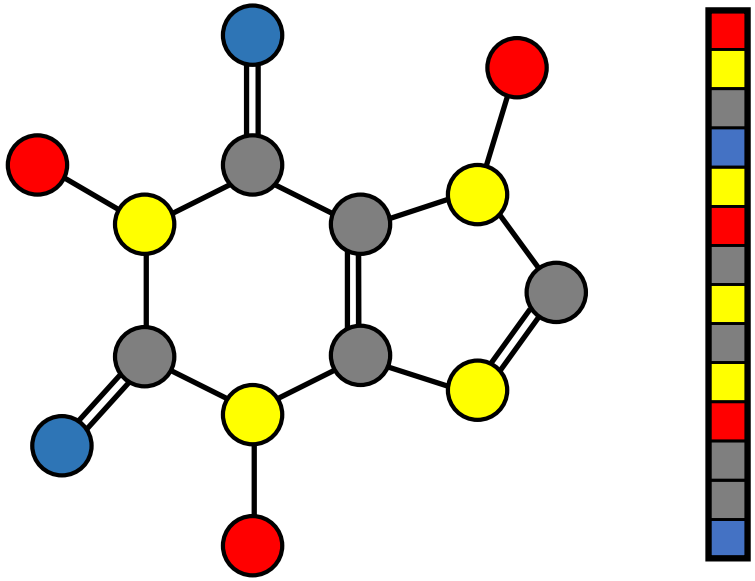


Fukushima 1980



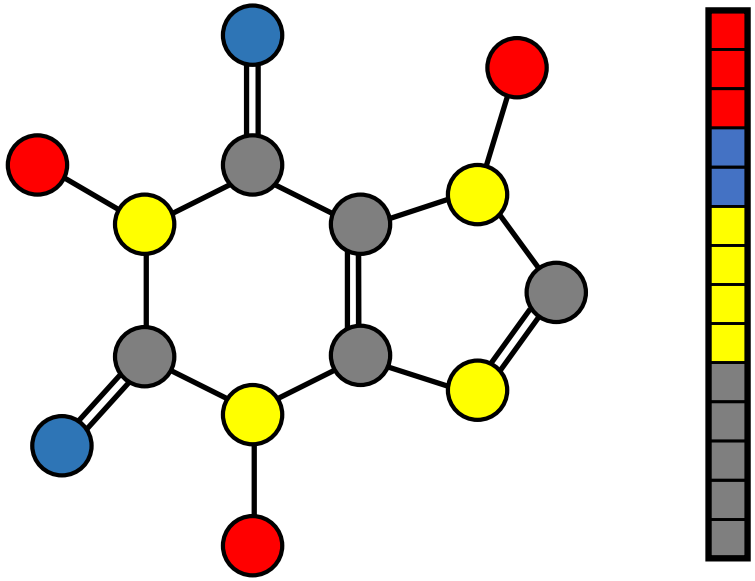
LeCun et al. 1989

Beyond Grids



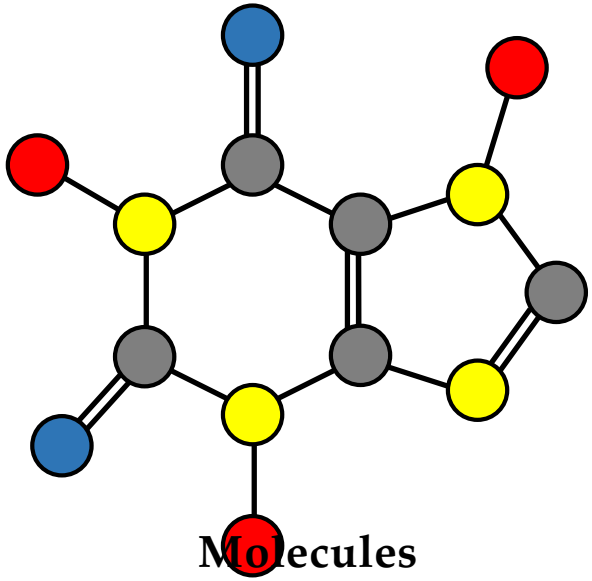
energy U = ?

Beyond Grids

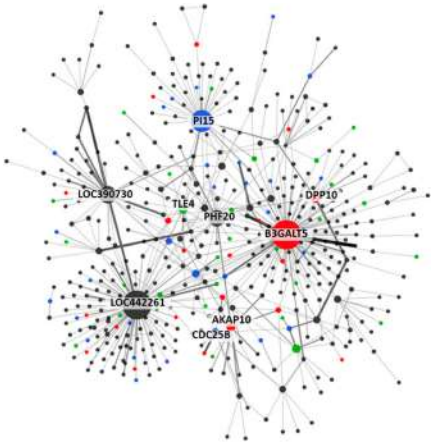


energy U = ?

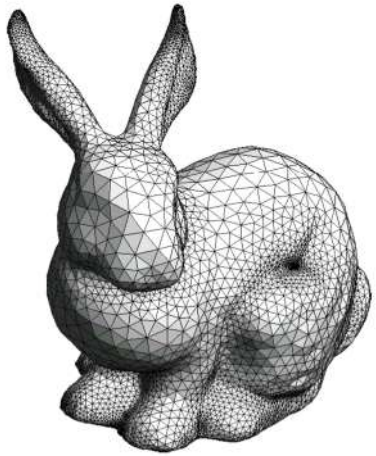
Social networks



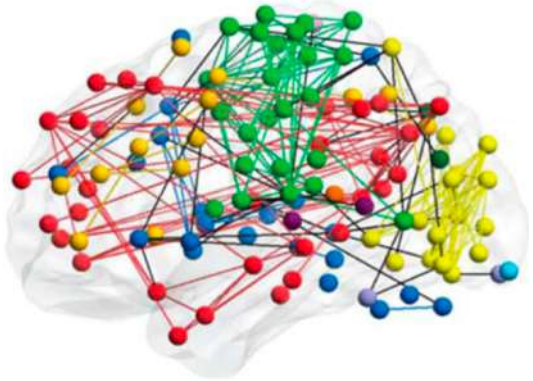
Molecules



Interaction networks

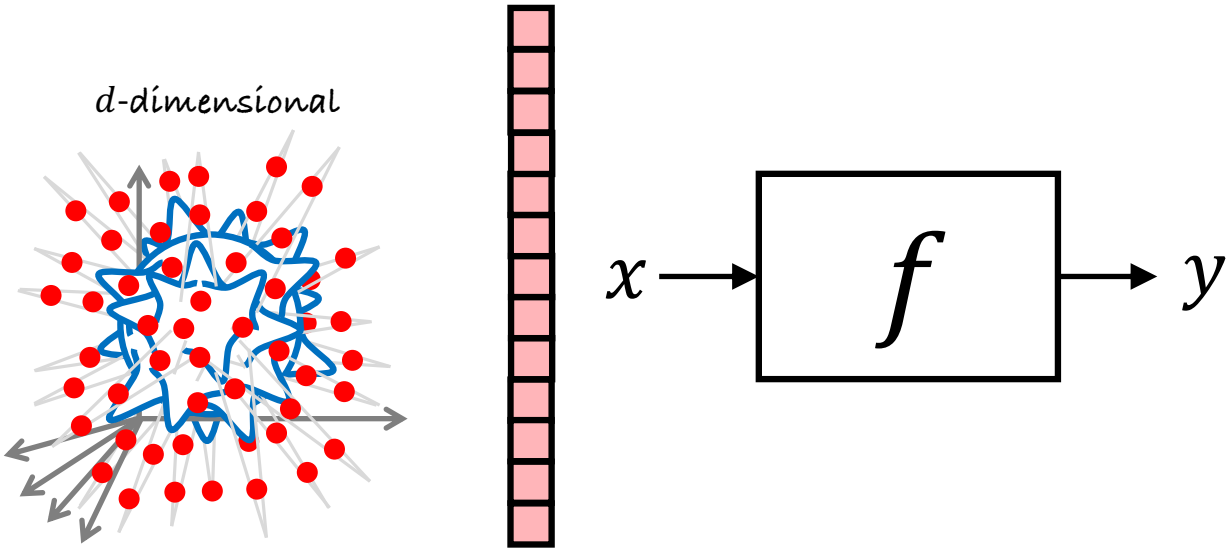


Meshes



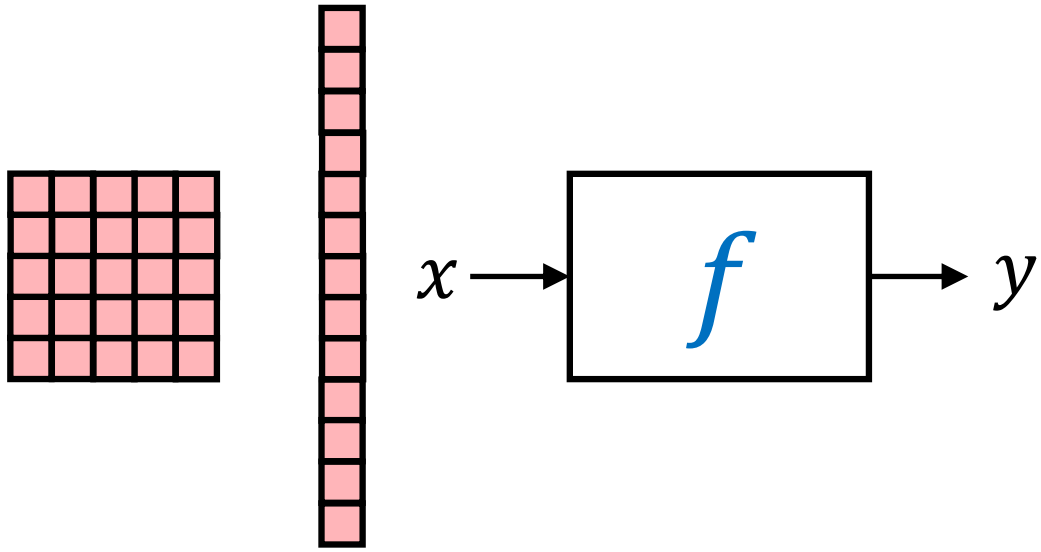
Functional networks

The Curse of Dimensionality

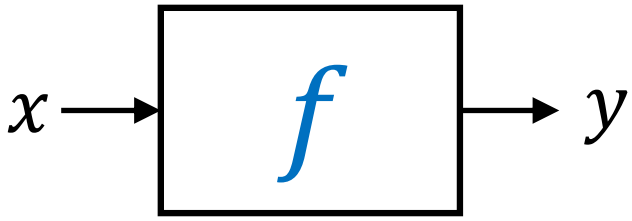


$O(\varepsilon^{-d})$ samples

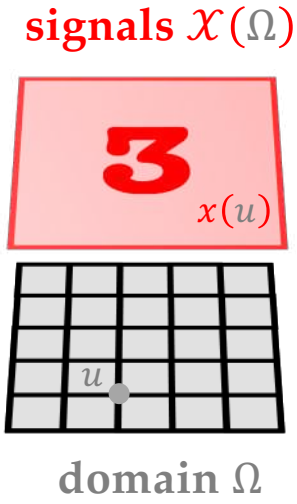
Geometric Priors

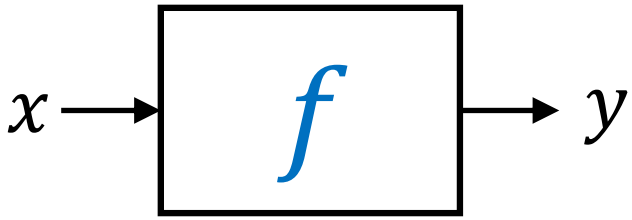


Geometric Priors

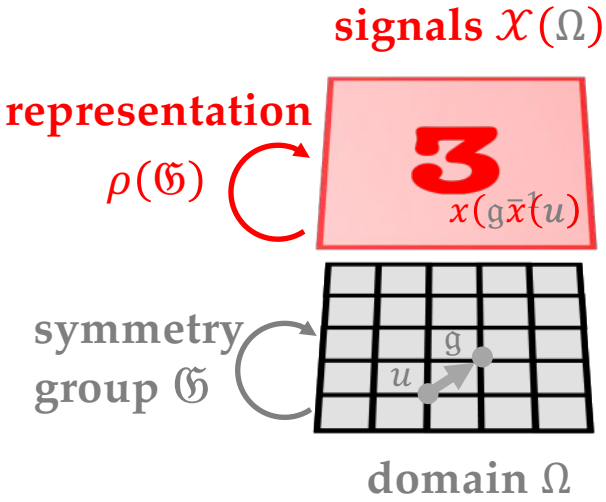


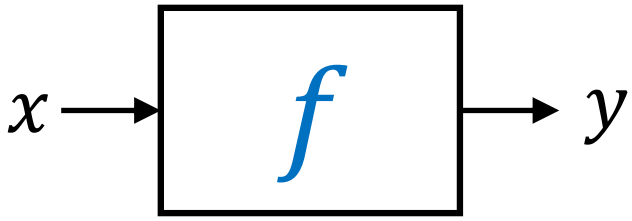
Symmetry Prior





Symmetry Prior

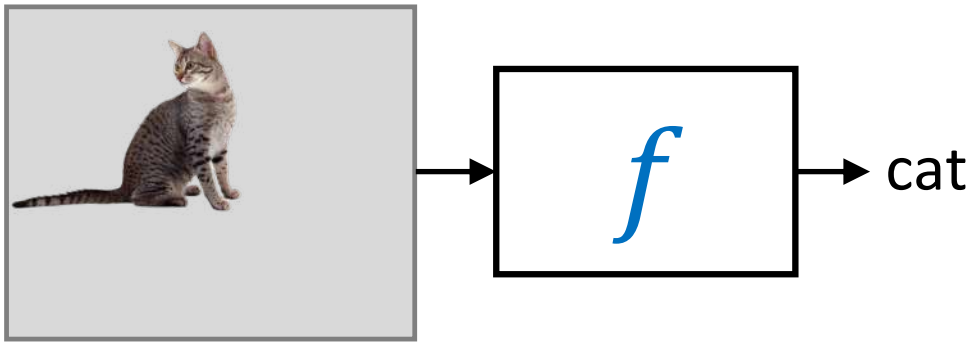




$$\rho(\mathfrak{g})x(u) = x(\mathfrak{g}^{-1}u)$$

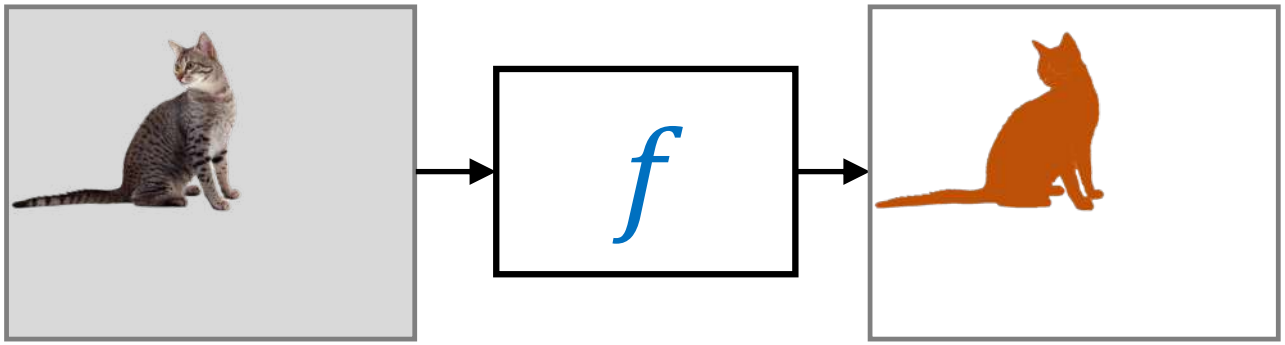
Invariant functions: Image Classification

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

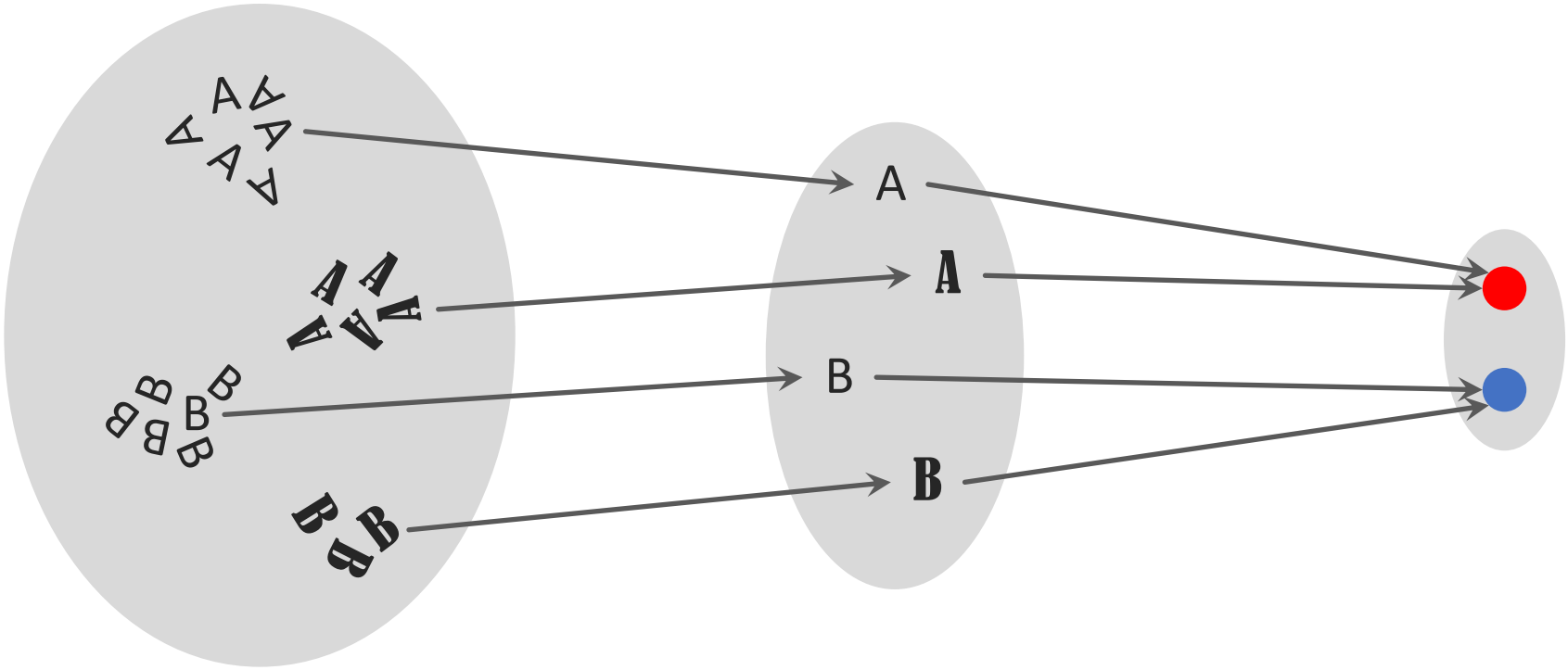


Equivariant functions: Image Segmentation

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



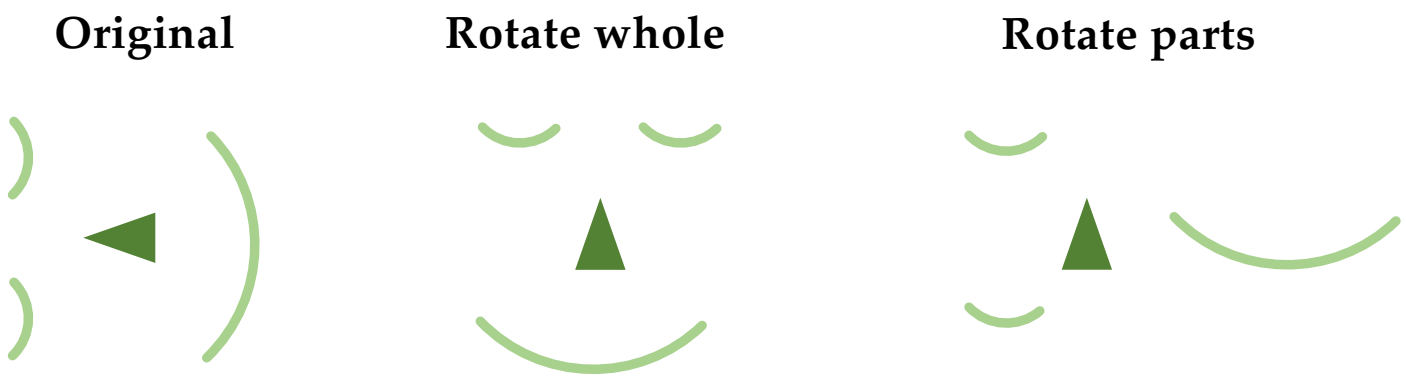
Invariant Representations



Problem with Invariant Representations

To recognise whole we must recognize parts. Relative positions of parts are important.

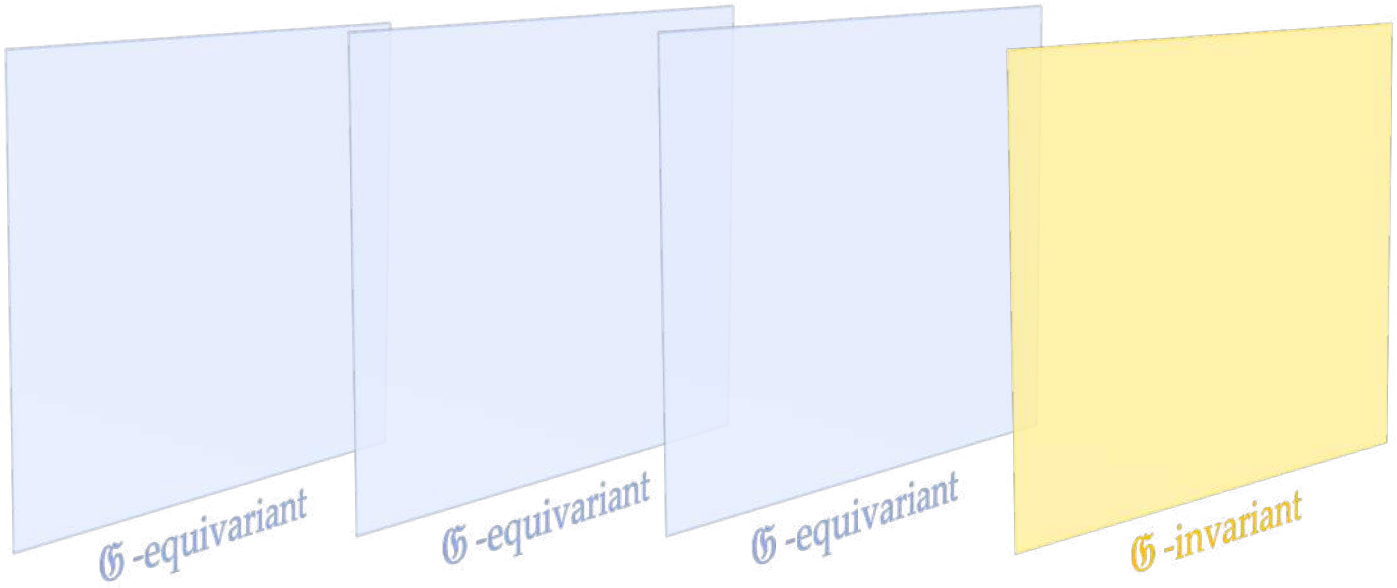
—Hinton



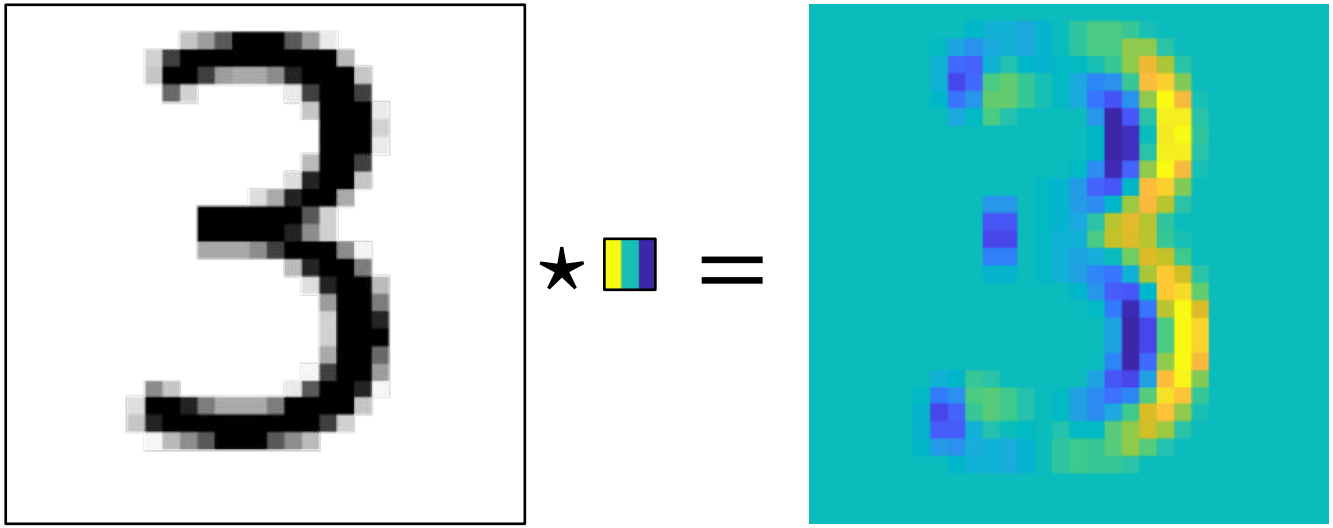
loose critical information!

Hinton et al. 2011

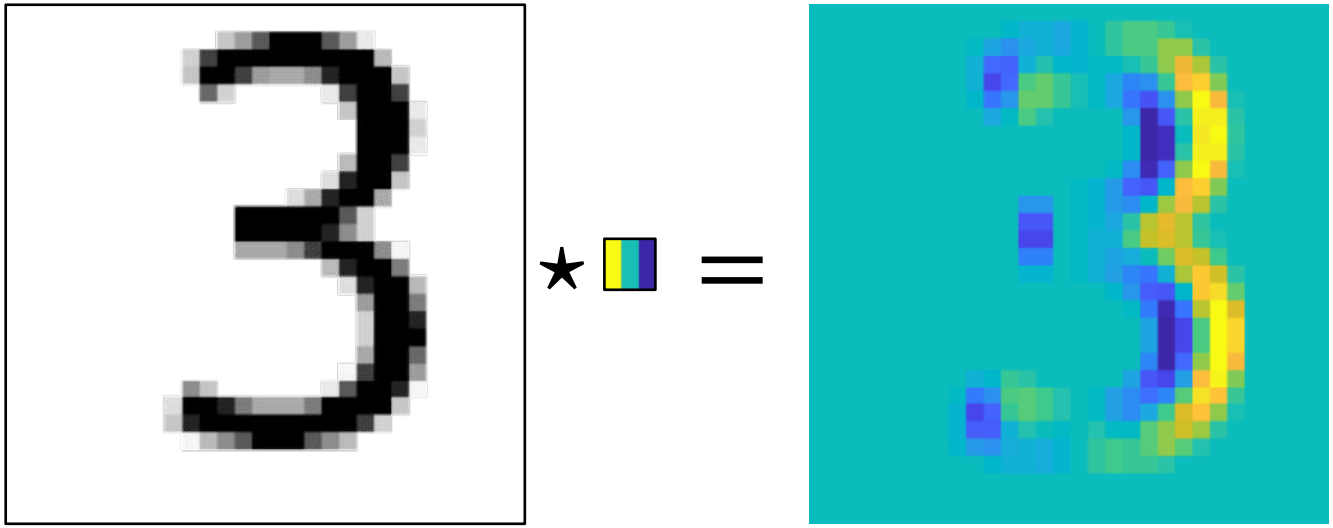
Geometric Deep Learning Blueprint



Translation Equivariance in CNNs

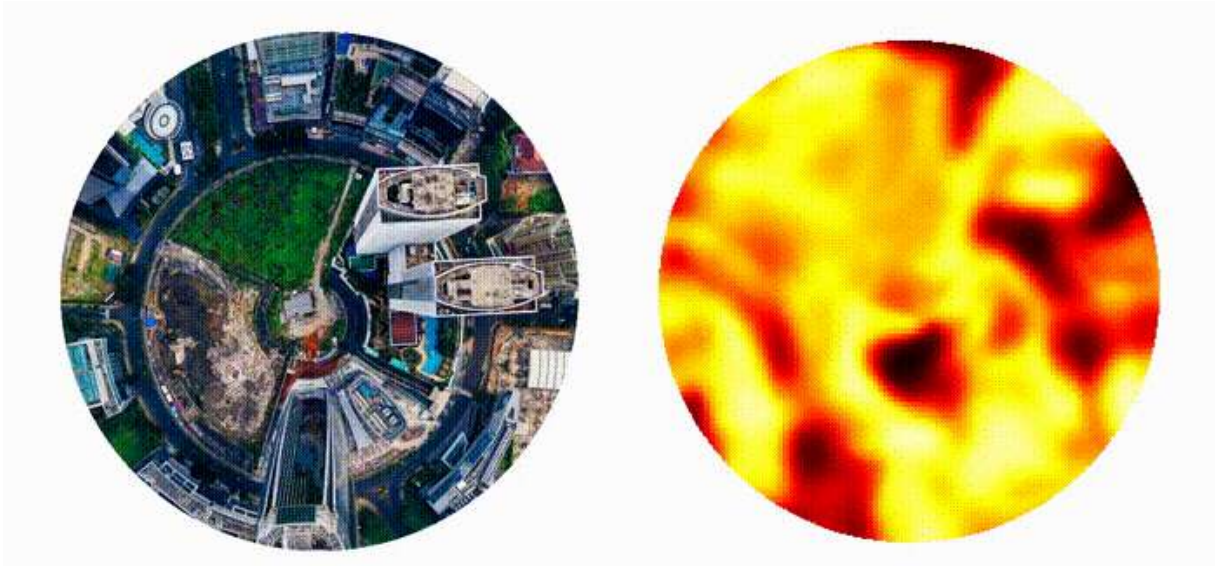


Translation Equivariance in CNNs



Rotation Equivariance in CNNs

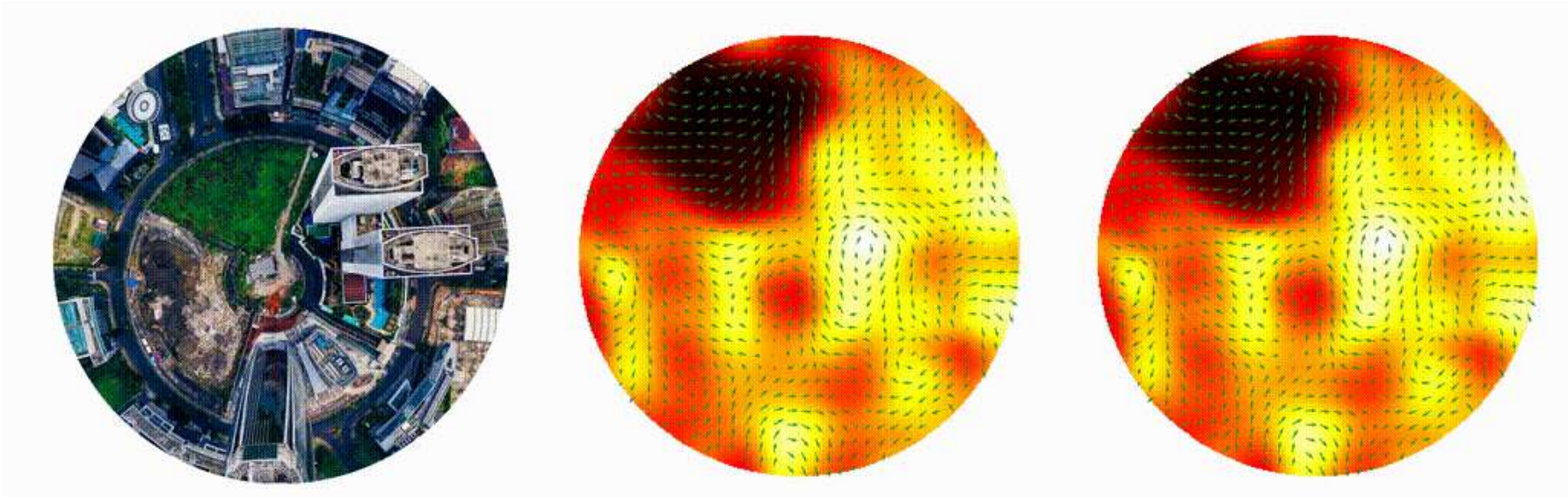
input **feature map**



no rotation equivariance!

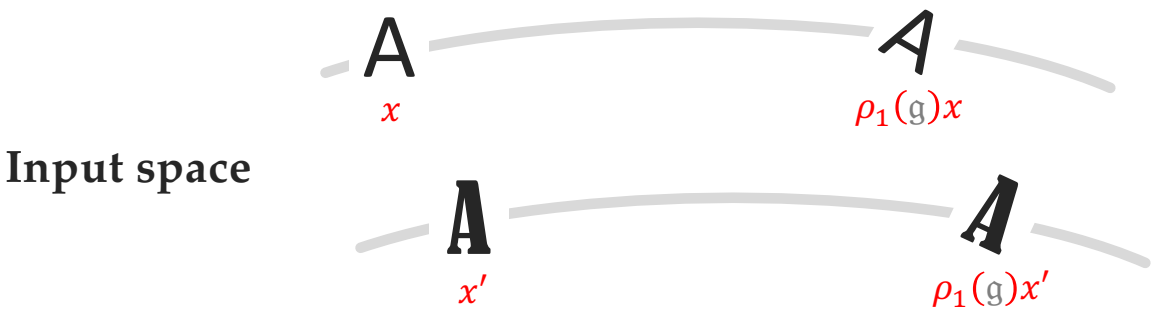
Rotation Equivariance in Group CNNs

input **feature map** **stabilised view**

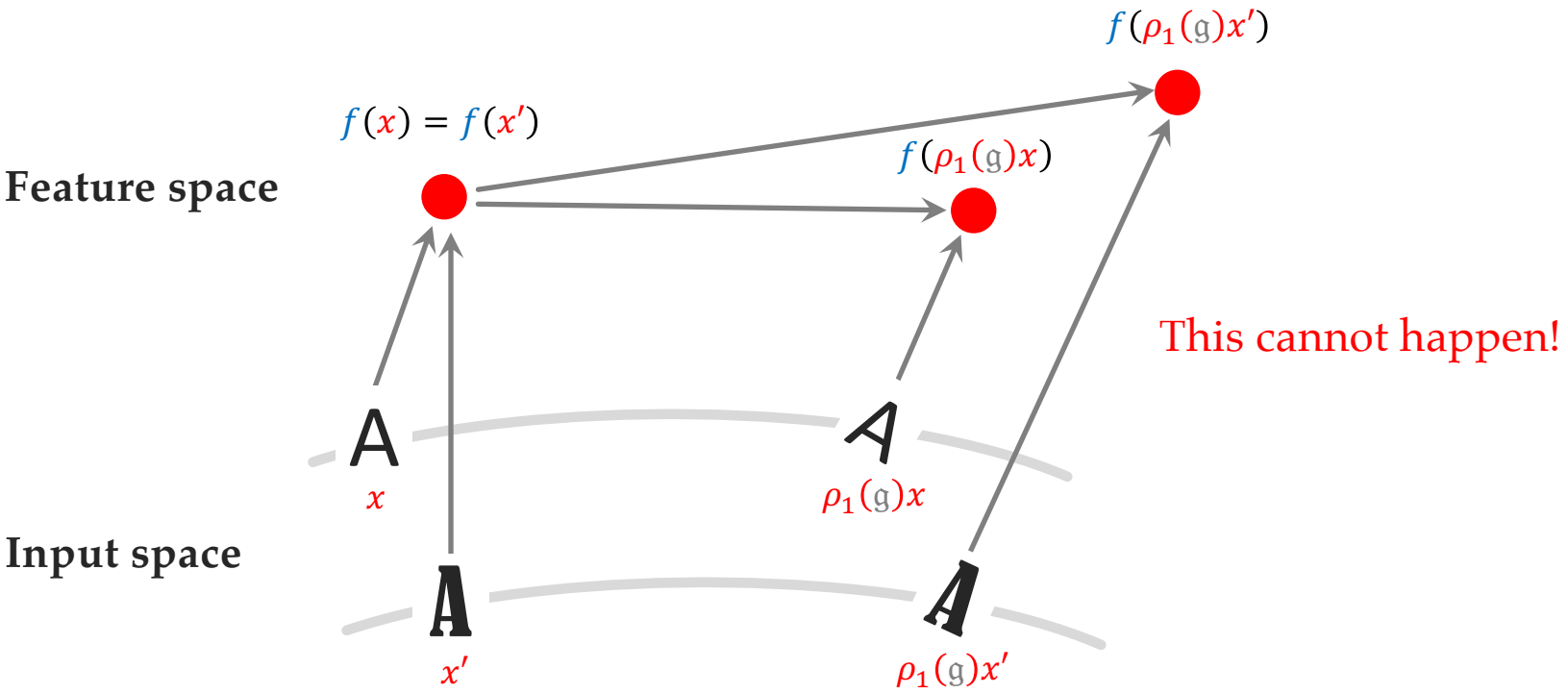


rotation equivariance!

Equivariance as Symmetry-Consistent Generalisation

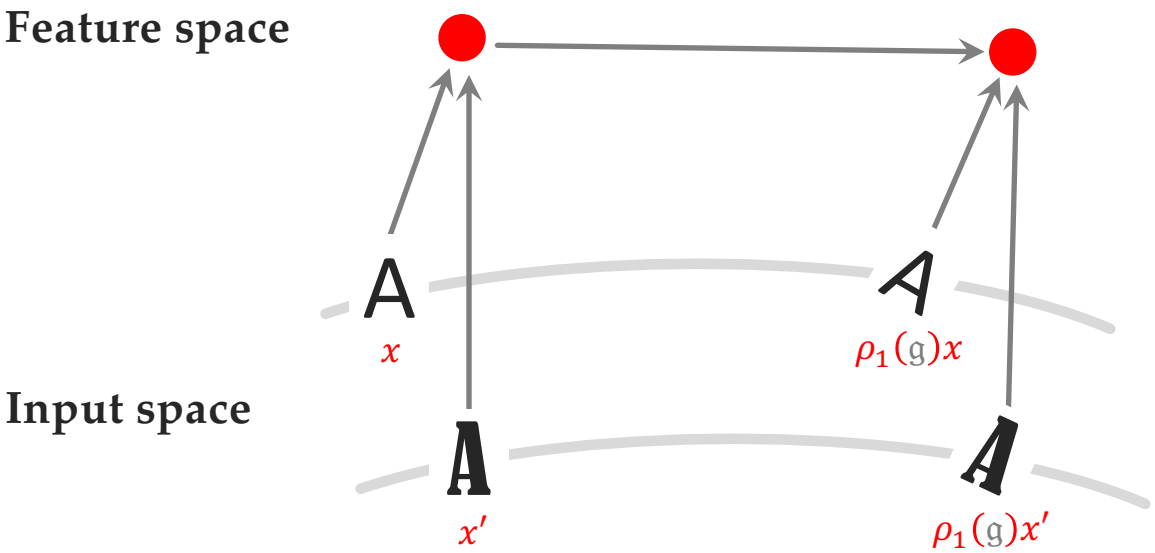


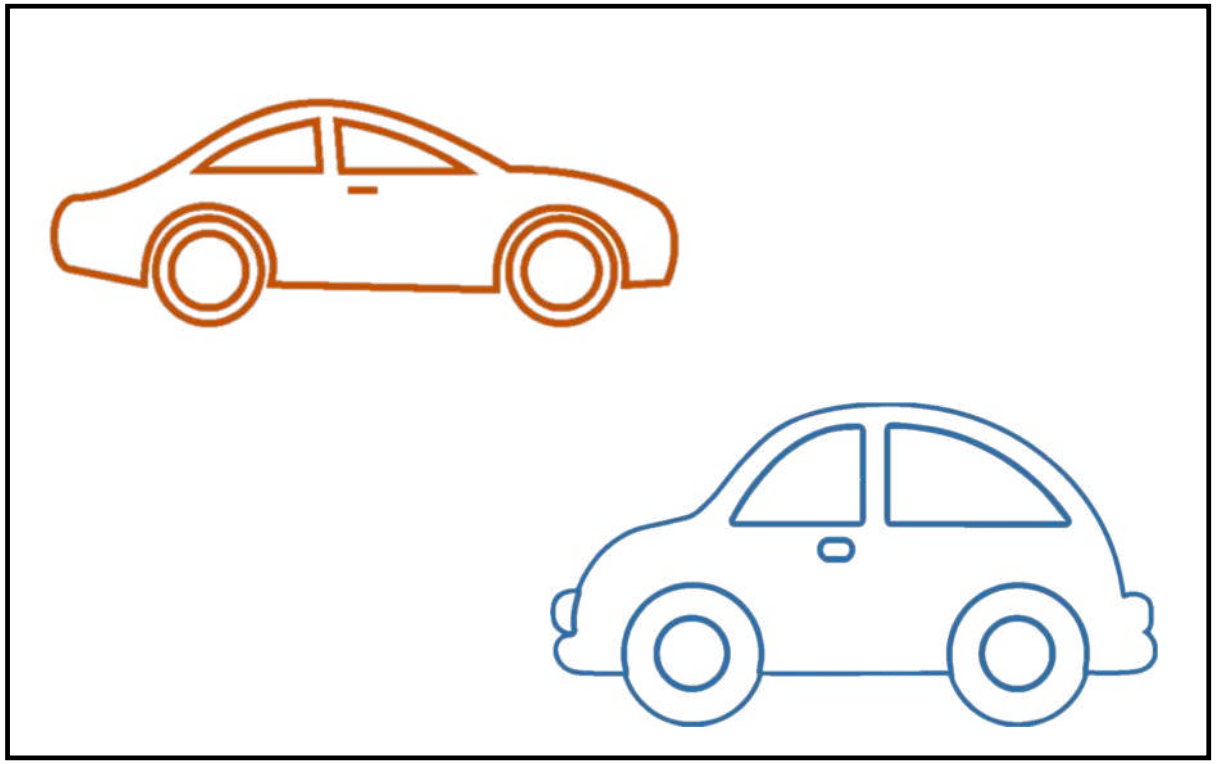
Equivariance as Symmetry-Consistent Generalisation



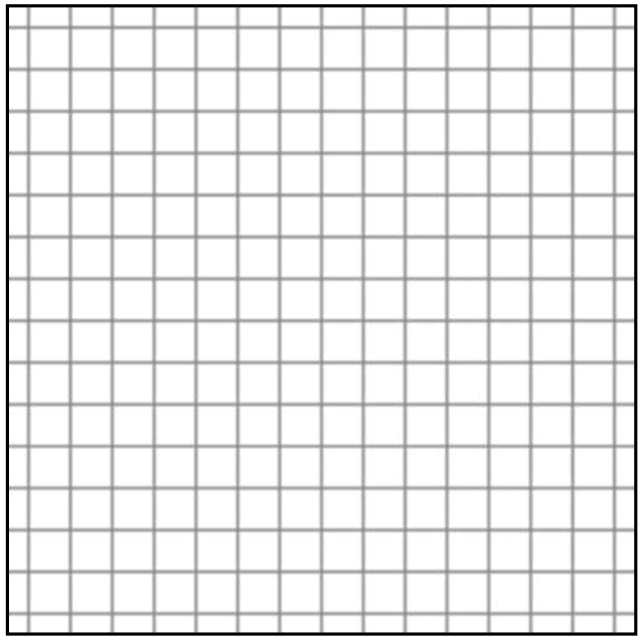
Equivariance as Symmetry-Consistent Generalisation

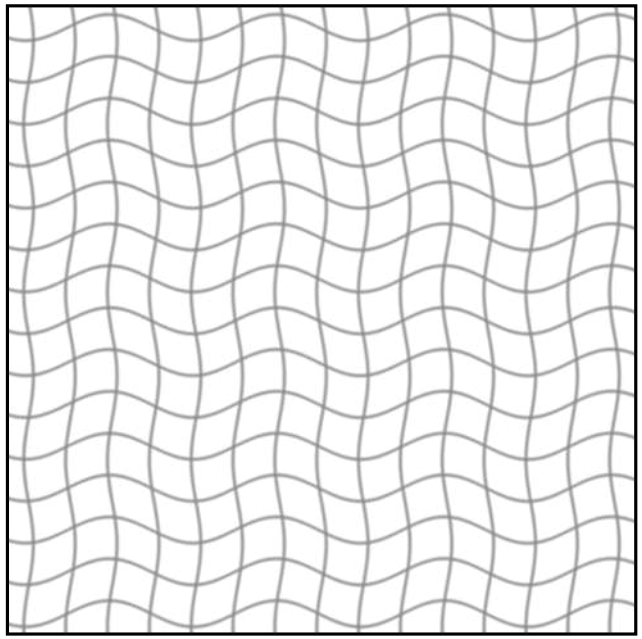
$$f(\rho_1(\mathrm{g})x) = \rho_2(\mathrm{g})f(x)$$



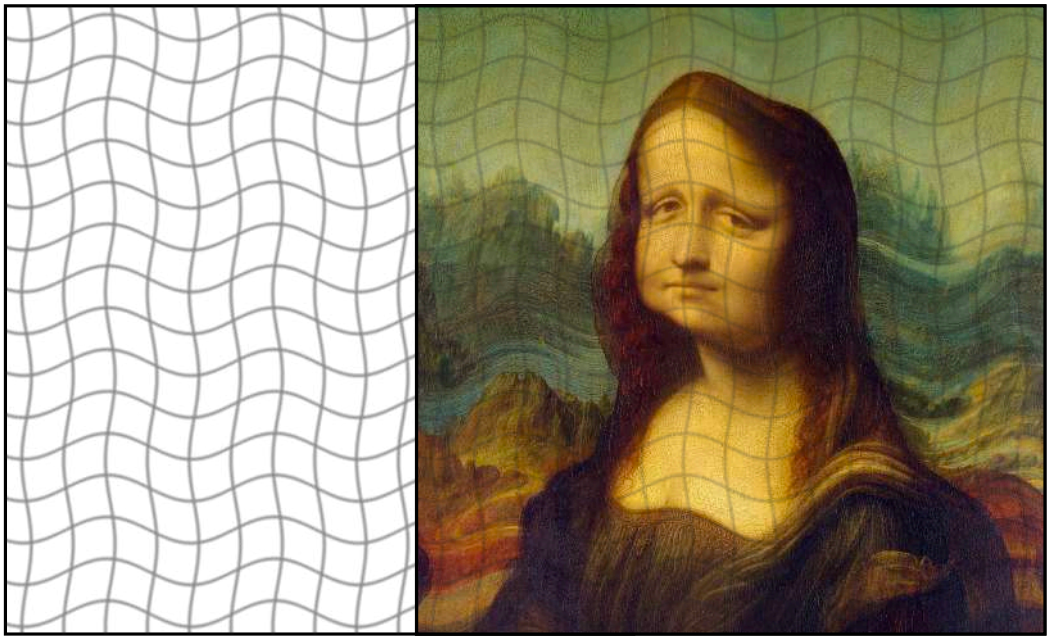


no global translation!



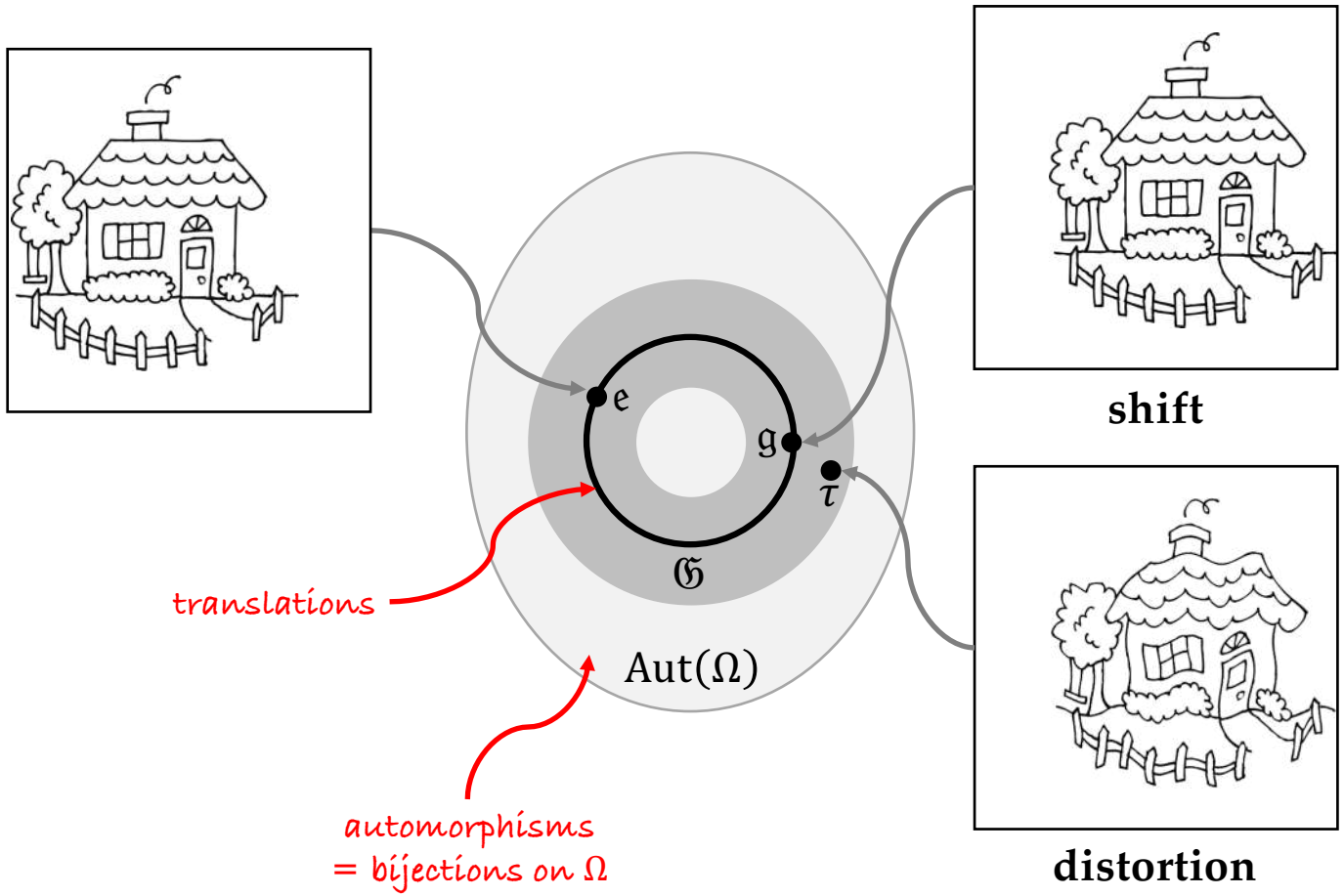


$\tau: \Omega \to \Omega$



$\tau: \Omega \rightarrow (\Omega_\tau x)(u) = x(\tau^{-1} u)$

Geometric Stability



Geometric Stability

"deformation insensitivity"

$$\|f(L_\tau x) - f(x)\| \leq c\|\nabla\tau\| \cdot \|x\|$$

stability under deformation

"Dirichlet energy"

Bruna, Mallat 2012

Geometric Stability

generalization of shift invariance

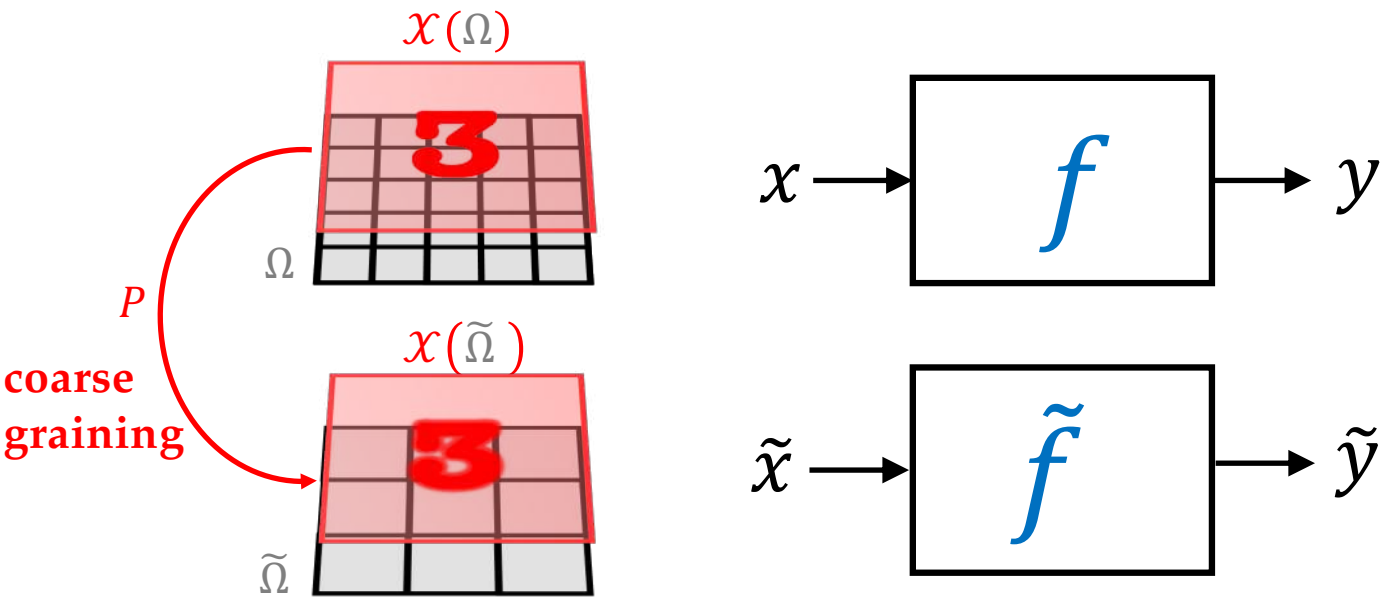
$$\|f(T_v x) - f(x)\| = 0$$

shift $\tau(u) = u - v$

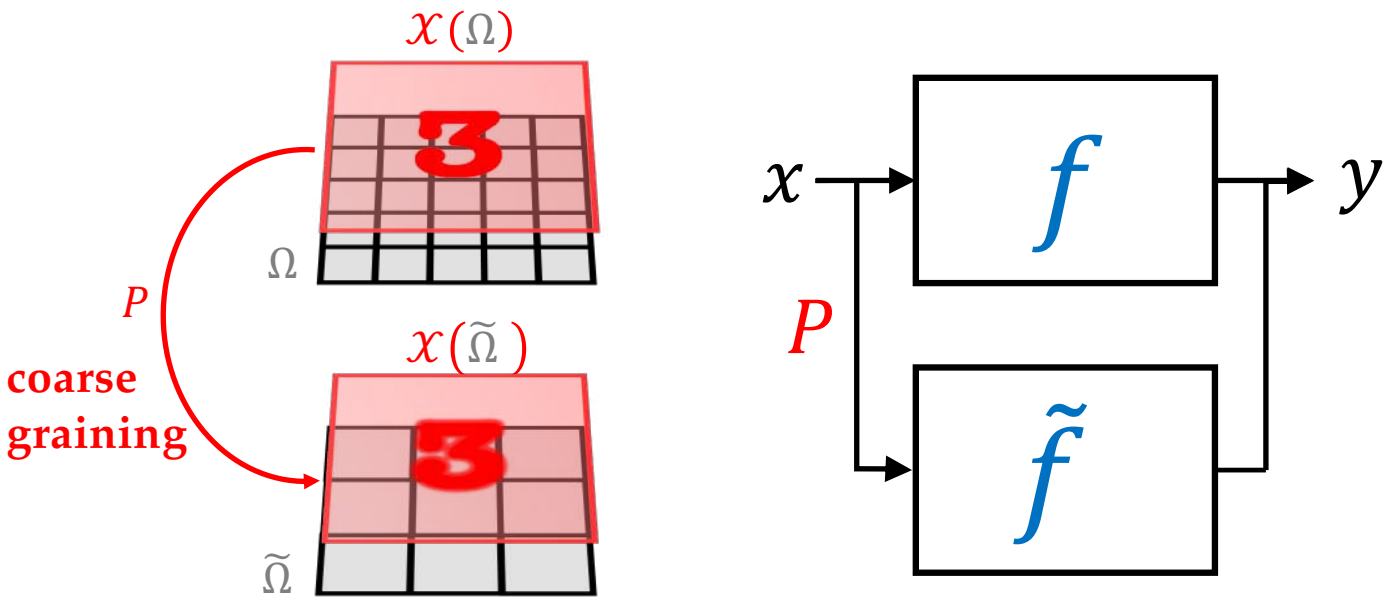
$\nabla\tau = 0$

Bruna, Mallat 2012

Scale Separation Prior

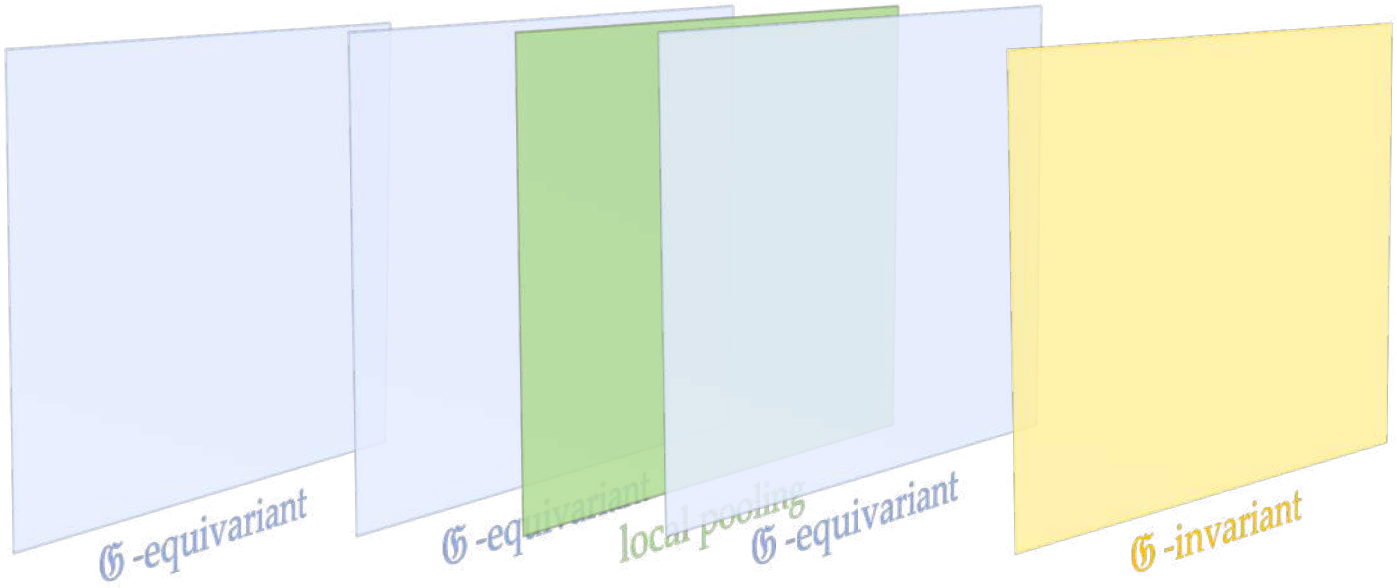


Scale Separation Prior

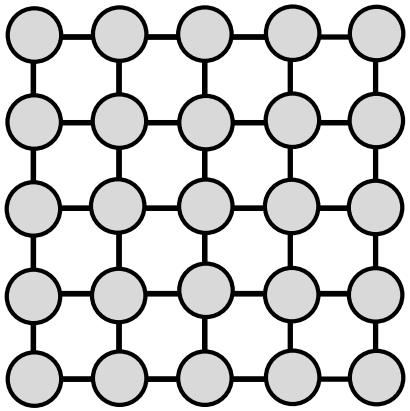


$$f \approx \tilde{f} \circ P$$

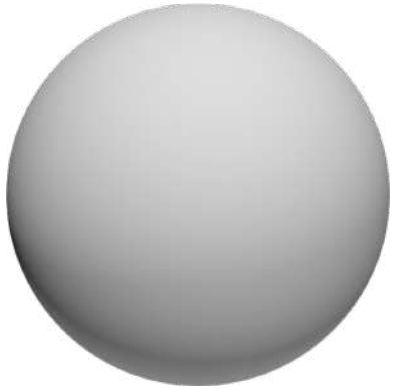
Geometric Deep Learning Blueprint



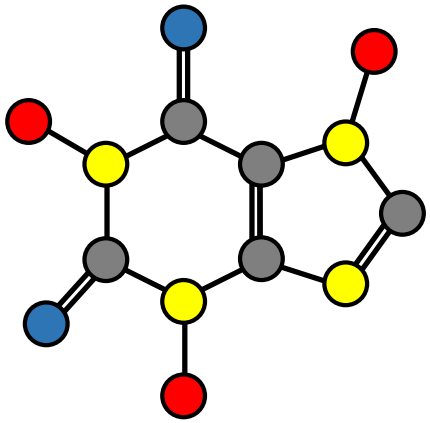
The "5G" of Geometric Deep Learning



Grids



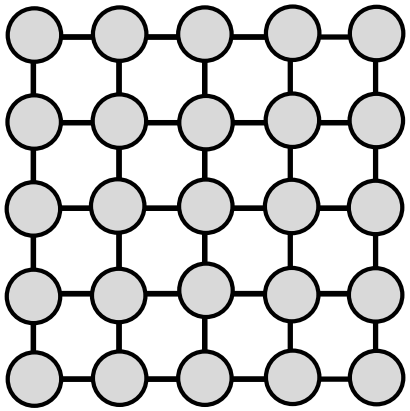
Groups



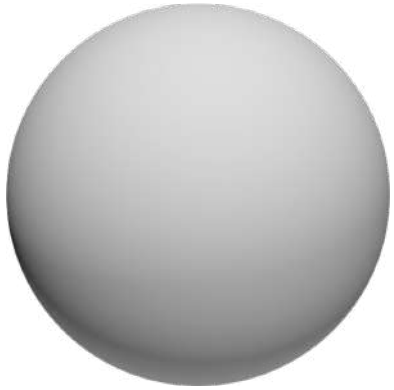
Graphs

Geodesics & Gauges

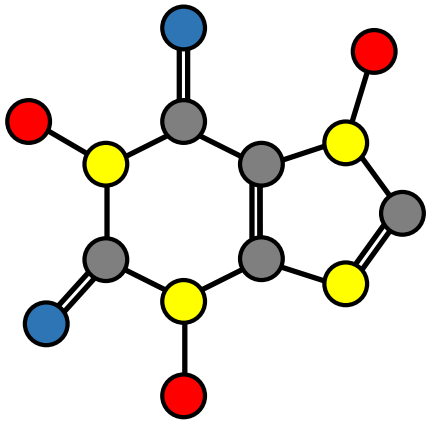
The "5G" of Geometric Deep Learning



Images & Sequences

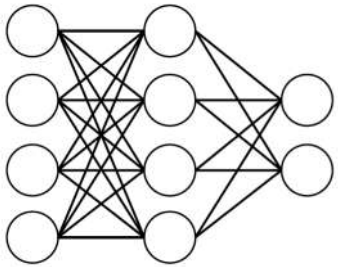


Homogeneous spaces

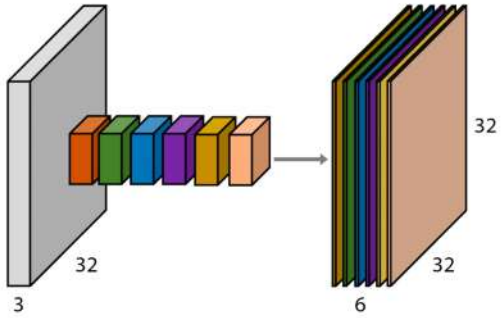


Graphs & Sets

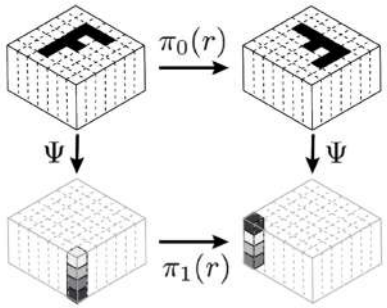
Manifolds, Meshes & Geometric graphs



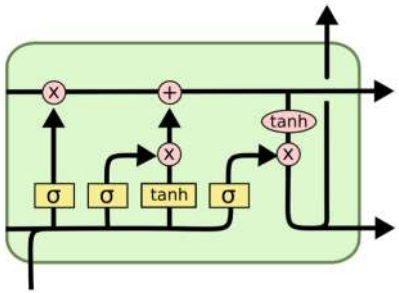
Perceptrons
Function regularity



CNNs
Translation

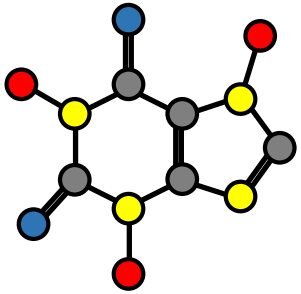


Group-CNNs
Translation+Rotation,
Global groups



LSTMs
Time warping

DeepSets / Transformers
Permutation



GNNs
Permutation

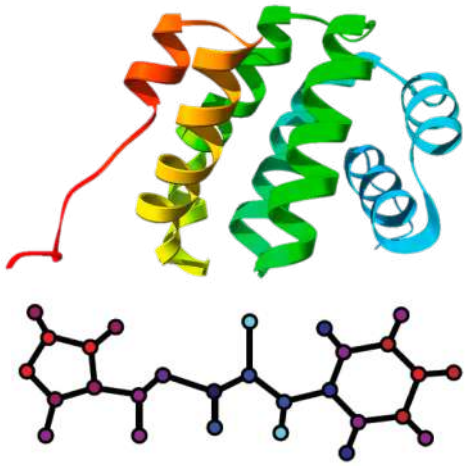
Intrinsic CNNs
Isometry / Gauge choice

Popular architectures as instances of GDL Blueprint

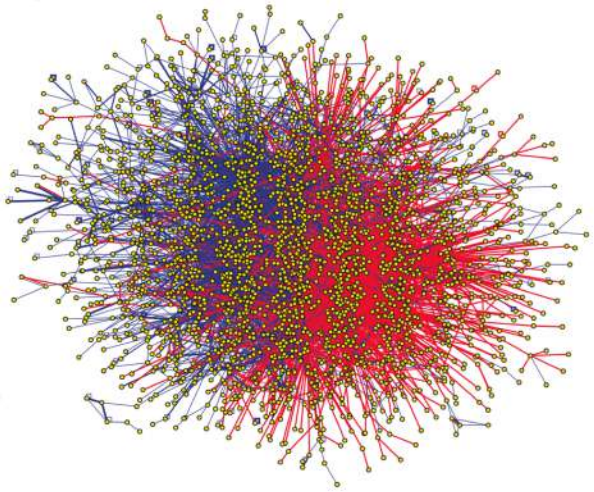
Architecture	**Domain Ω**	**Symmetry Group $\mathfrak{G}$**
CNN	Grid	Translation
Spherical CNN	Sphere / $\mathrm{SO}(3)$	Rotation $\mathrm{SO}(3)$
Intrinsic / Mesh CNN	Manifold	Isometry $\mathrm{Iso}(\Omega)$ / Gauge Symmetry $\mathrm{SO}(2)$
GNN	Graph	Permutation Σ_n
Deep Sets	Set	Permutation Σ_n
Transformer	Complete Graph	Permutation Σ_n
LSTM	1D Grid	Time warping

GRAPHS

Graphs = Systems of Relations and Interactions



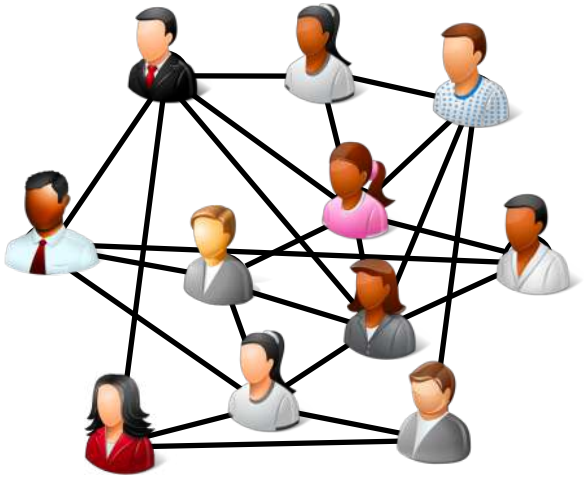
Molecules



Interactomes

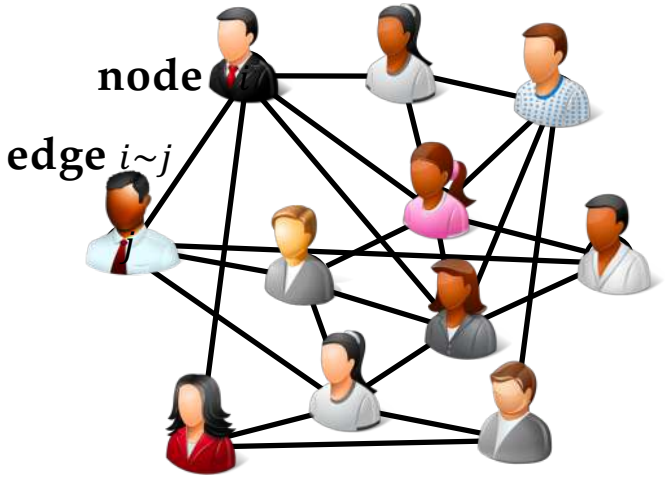
Social networks

Graphs



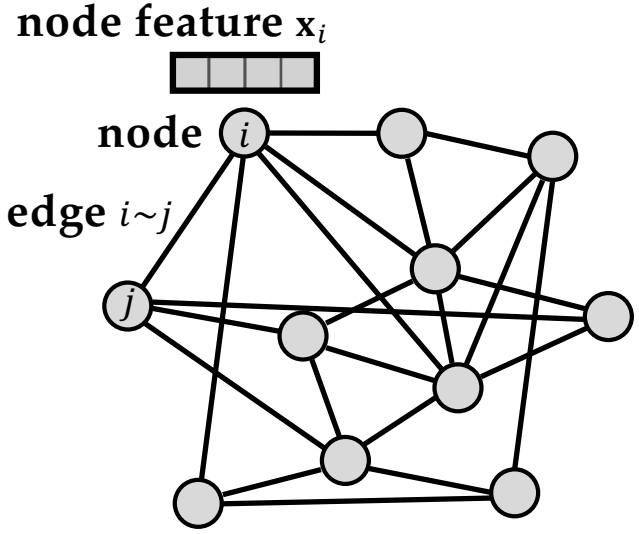
social network

Graphs



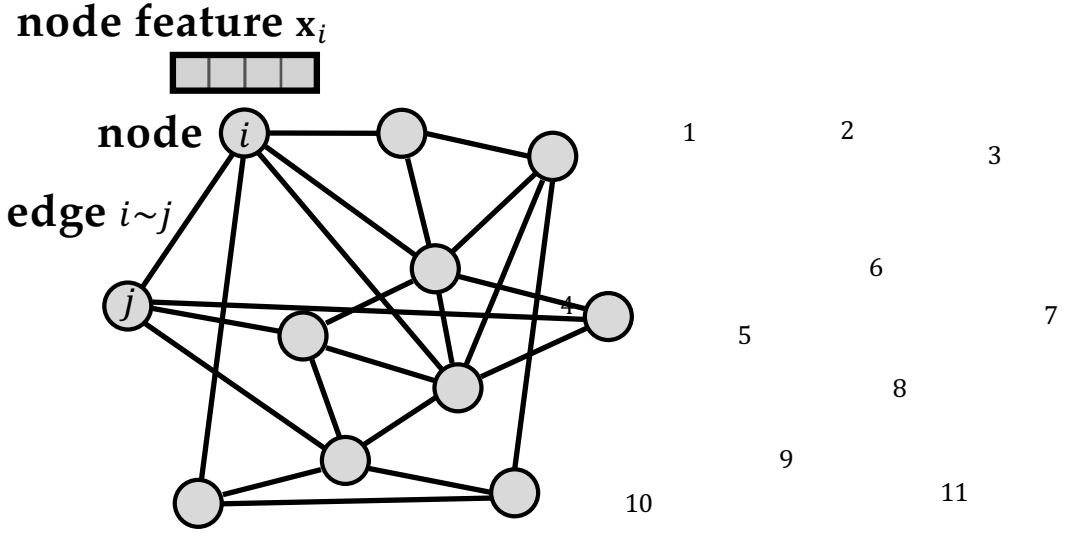
social network graph

Graphs



graph

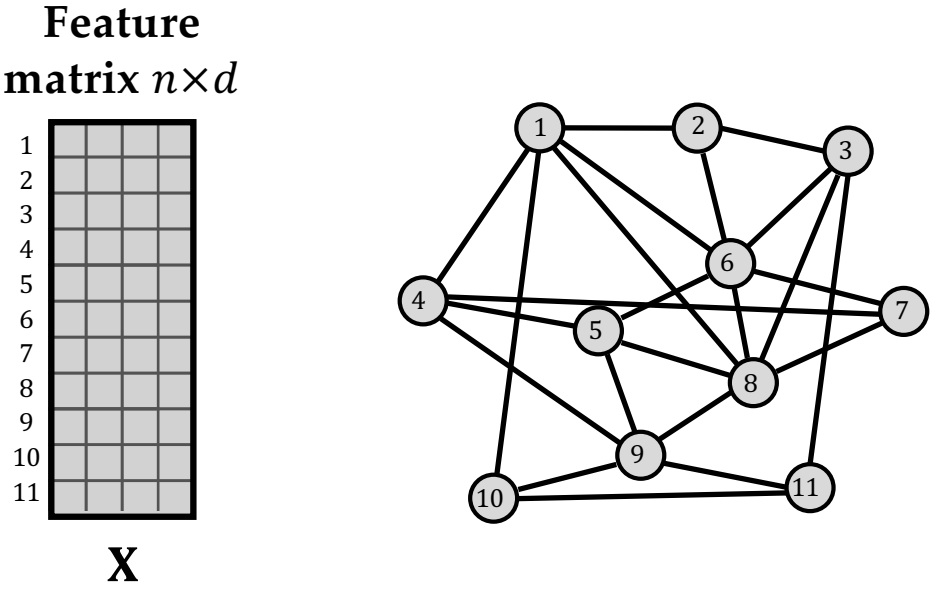
Key Structural Properties of Graphs



graph

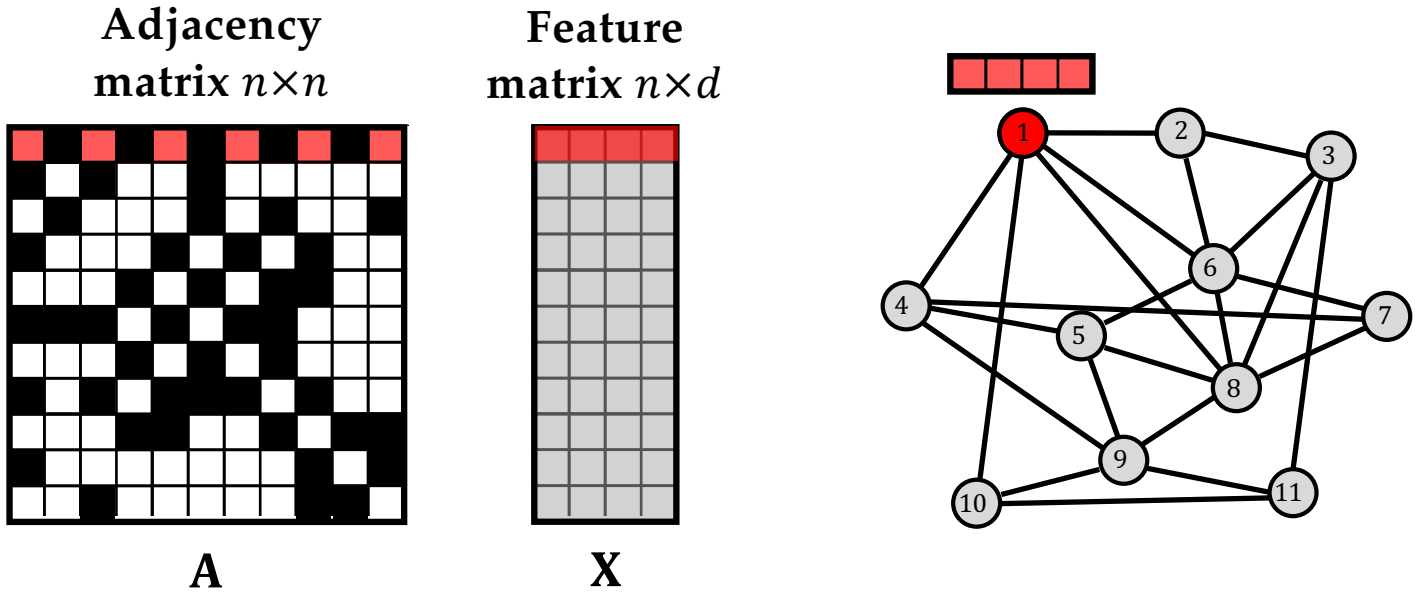
arbitrary ordering of nodes

Key Structural Properties of Graphs



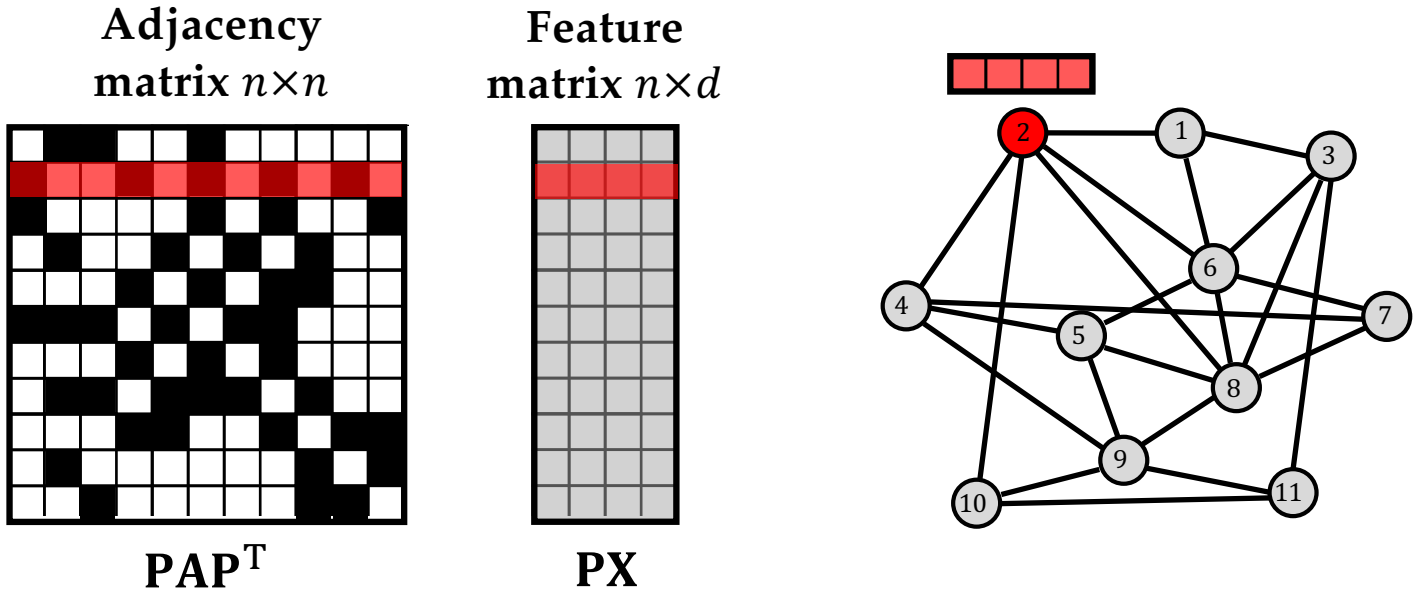
arbitrary ordering of nodes

Key Structural Properties of Graphs



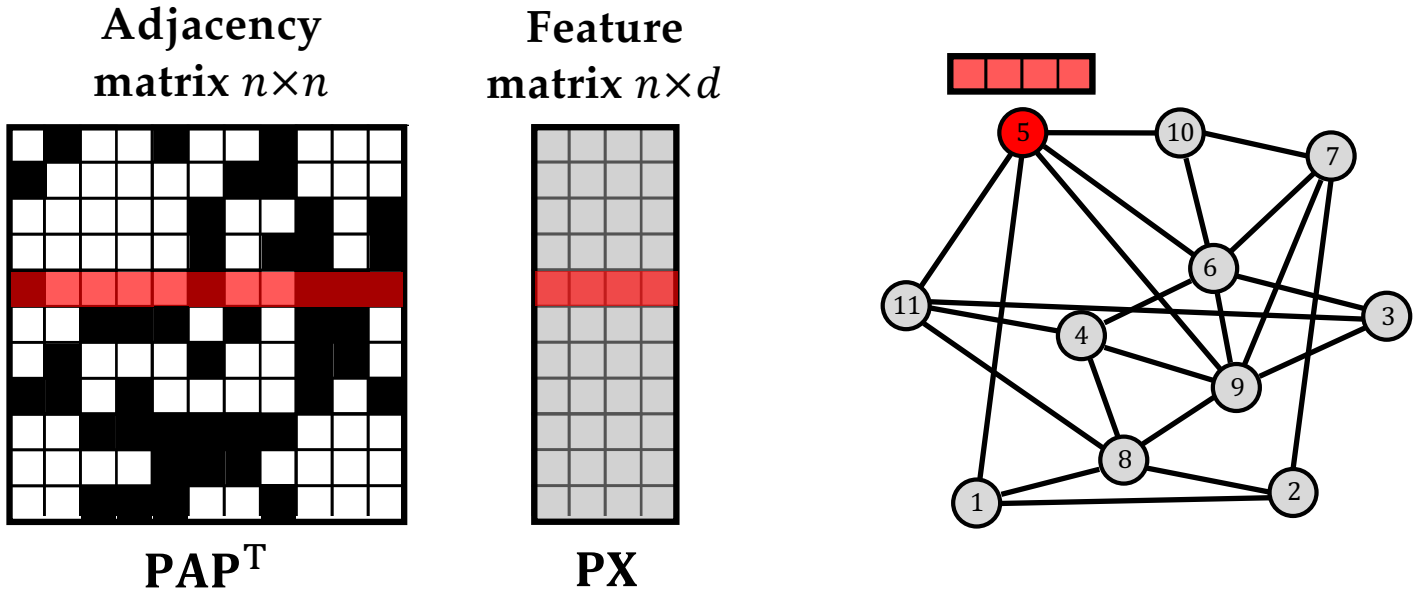
arbitrary ordering of nodes

Key Structural Properties of Graphs



arbitrary ordering of nodes

Key Structural Properties of Graphs

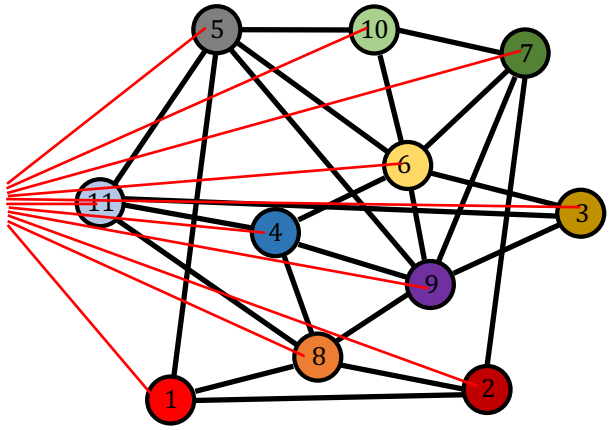


arbitrary ordering of nodes

$n!$ permutations

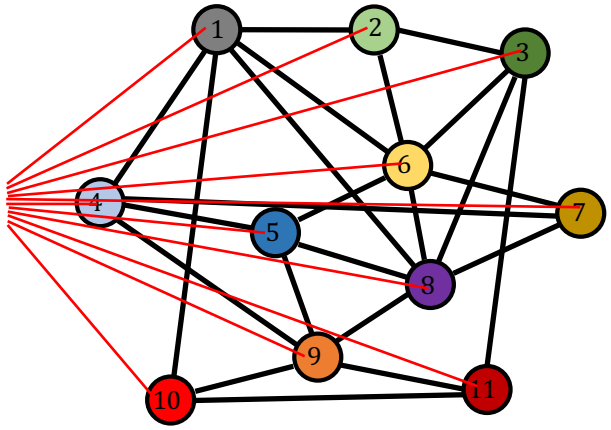
Invariant Graph Functions

graph function $f(\mathbf{X}, \mathbf{A})$



Invariant Graph Functions

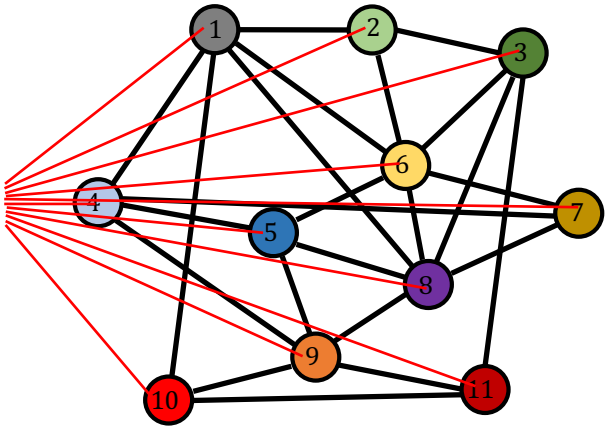
graph function $f(\mathbf{X}, \mathbf{A})$



Invariant Graph Functions

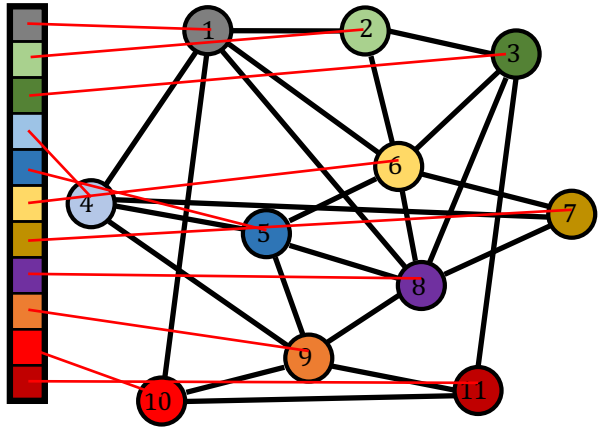
permutation-invariant

$$f\left(\mathbf{PX}, \mathbf{PAP}^{\top}\right) = f(\mathbf{X}, \mathbf{A})$$



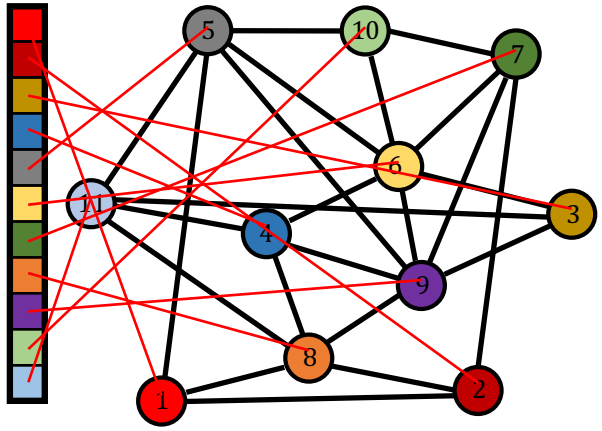
Equivariant Graph Functions

node function $\mathbf{F}(\mathbf{X}, \mathbf{A})$



Equivariant Graph Functions

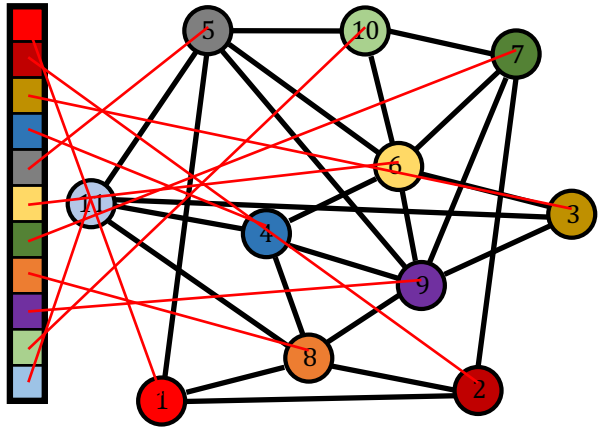
node function $\mathbf{F}(\mathbf{X}, \mathbf{A})$



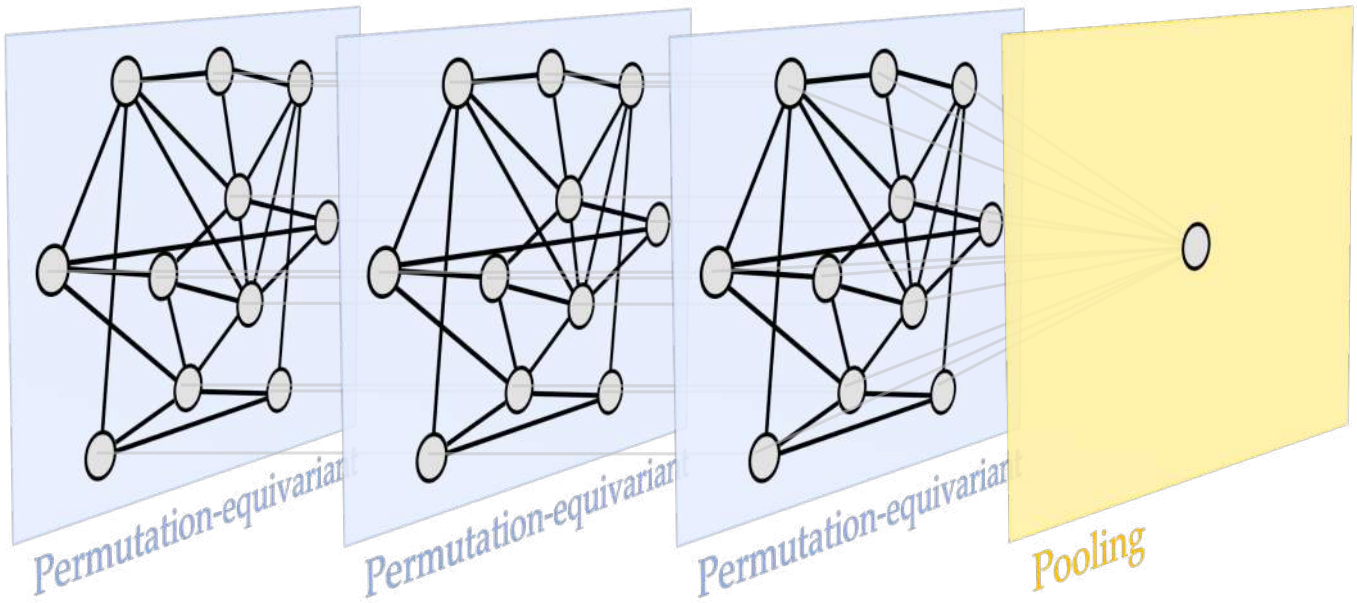
Equivariant Graph Functions

permutation-equivariant

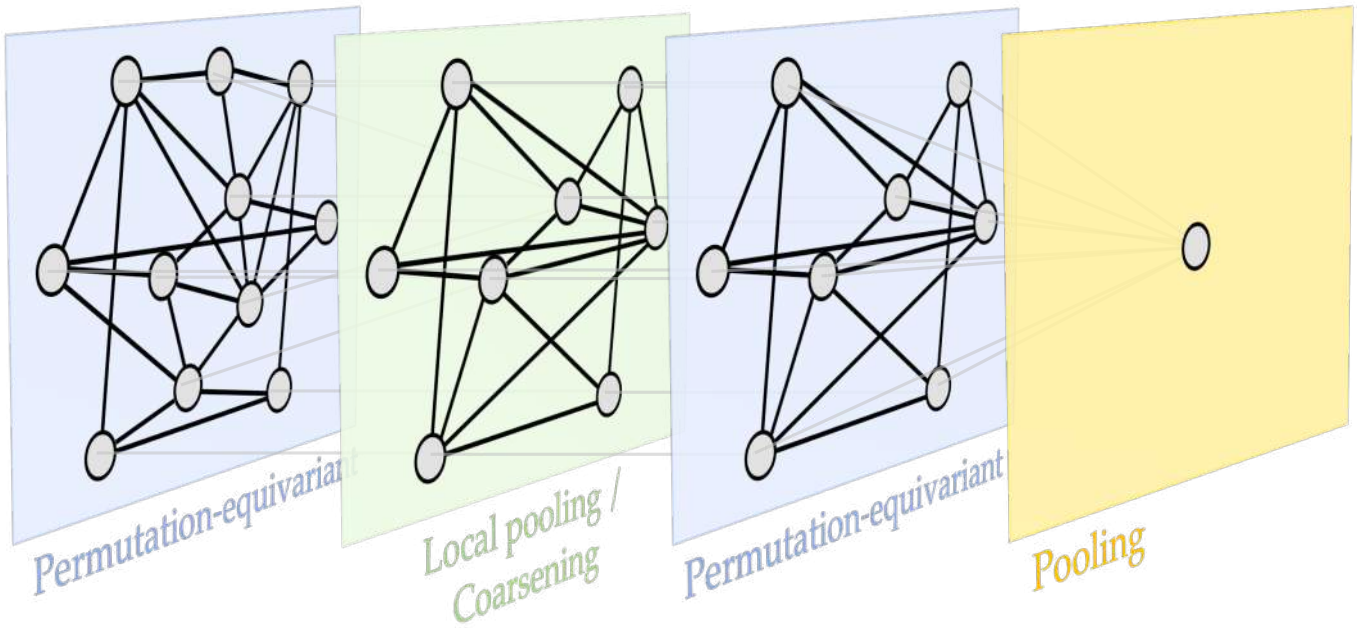
$$\mathbf{F}\left(\mathbf{PX}, \mathbf{PAP}^{\top}\right) = \mathbf{PF}(\mathbf{X}, \mathbf{A})$$



Graph Neural Networks



Graph Neural Networks



History of Graph Neural Networks according to Machine Learners

A. Sperduti

Labeling RAAM

1994

C. Goller

A. Küchler

Backprop through structure

1996

M. Gori

F. Scarselli

Graph Neural Networks

2005, 2008

Y. Li

Gated GNN

2015

History of Graph Neural Networks according to Graph Theorists

N. Shervashidze

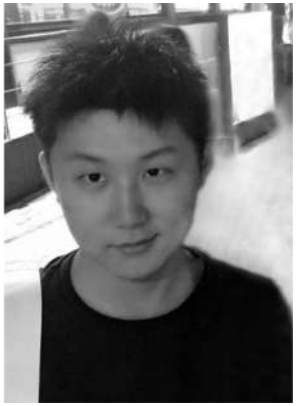
K. Borgwardt

Weisfeiler-Lehman kernels

2009

C. Morris

k-GNN



K. Xu

GIN

2019

H. Maron

Provably powerful GNN

Weisfeiler-Lehman Test

УДК 519.1

ПРИВЕДЕНИЕ ГРАФА К КАНОНИЧЕСКОМУ ВИДУ И ВОЗНИКАЮЩАЯ ПРИ ЭТОМ АЛГЕБРА

Б. Ю. ВЕЙСФЕЙЛЕР, А. А. ЛЕМАН

Рассматривается алгоритм приведения заданного конечного мультиграфа Γ к каноническому виду. В процессе такого приведения возникает новый инвариант графа — алгебра $\mathfrak{A}(\Gamma)$. Изучение свойств алгебры $\mathfrak{A}(\Gamma)$ оказывается полезным при решении некоторых задач теории графов.

Выдвигаются и обсуждаются некоторые предположения относительно связи между свойствами алгебры $\mathfrak{A}(\Gamma)$ и группой автоморфизмов графа $Aut(\Gamma)$. Построен пример неориентированного графа Γ, алгебра $\mathfrak{A}(\Gamma)$ которого совпадает с групповой алгеброй некоторой некоммутативной группы.

An algorithm is considered, reducing the specified finite multigraph Γ to canonical form. In the course of this reduction, a new invariant of the graph is generated — algebra $\mathfrak{A}(\Gamma)$. Study of the properties of the algebra $\mathfrak{A}(\Gamma)$ proves helpful in solving a number of graph-theoretic problems. Some propositions concerning the relationships between the properties of the algebra $\mathfrak{A}(\Gamma)$ and the graph's automorphism group $Aut(\Gamma)$ are discussed. An example of non-oriented graph Γ is constructed whose algebra $\mathfrak{A}(\Gamma)$ coincides with the group algebra of a non-commutative group.

1. Рассмотрим произвольный конечный граф Γ и его матрицу смежности $A(\Gamma)=\|a_{ij}\|$; здесь a_{ij} — число ребер, ведущих из i-й вершины графа в j-ую; $i, j=1, 2, \ldots, n$. В случае неориентированного графа полагаем $a_{ij}=a_{ji}$. Каноническим видом графа мы будем называть его матрицу смежности при канонической нумерации вершин, т. е. при таком частичном упорядочении множества вершин, при котором из того, что $a \neq b$ несравнимы следует, что существует автоморфизм графа, переводящий вершину a в b и сохраняющий отношение смежности.

В п. 5, 7 описан процесс приведения графа к каноническому виду, состоящий в перестановке переупорядоченных строк и столбцов матрицы $A(\Gamma)$, который, грубо говоря, сводится к следующему.

Рассмотрим для простоты неориентированный граф без кратных связей. Сначала каждой вершине графа сопоставим характеристический вектор, единственная компонента которого равна числу соседей данной вершины. Затем разобьем вершины на классы, так чтобы вершины с одинаковым характеристическим вектором принадлежали одному классу; классы при этом упорядочим в соответствии с естественным порядком в множестве характеристических векторов. Далее, каждой вершине сопоставим характеристический вектор $v_i=(l, v_{i1}, v_{i2}, \ldots)$, где v_{ik} — число соседей k-го класса у i-ой вершины, l — номер класса, к которому принадлежит i-ая вершина. Теперь снова разобьем вершины на классы в соответствии с новыми характеристическими векторами, упорядоченными лексикографически, и т. д. Заметим, что если вершины a и b на некотором шаге принадлежали разным классам и было выполнено условие $a<b$, то и в дальнейшем это условие всегда будет выполнено. Отсюда следует, что описанный процесс обрывается не позже, чем через n шагов — и либо все вершины относятся к разным классам (т. е. построена каноническая нумерация), либо дальнейшего разбиения на классы не происходит.

В случае, если Γ — ориентированный мультиграф, возьмем в качестве характеристического вектора v_i упорядоченную i-ую строку матрицы $A(\Gamma)$ (считая при этом, что диагональный элемент предшествует всем остальным). Вместо различных элементов a_{ij} введем различные независимые переменные $x_0, x_1, \ldots$, упорядочив их в соответствии с порядком среди a_{ij}. Полученную таким способом матрицу обозначим $X=X(\Gamma)$. При очередном разбиении вершин на классы к одному классу, как и прежде, отнесем вершины с одинаковым характеристическим вектором; при этом k-ая компонента вектора v_i есть по определению сумма элементов i-ой строки матрицы $X(\Gamma)$, соответствующих вершинам k-го класса (предыдущего разбиения). Матрица $X(\Gamma)$ разбивается, таким образом, на блоки, в каждом из которых мы можем ввести свои независимые переменные и т. д. (точное определение этих операций см. п. 6, операции α_0, β_0).

Заметим, что описанное до сих пор процедура аналогична методам, изложенным в [1] и [2].

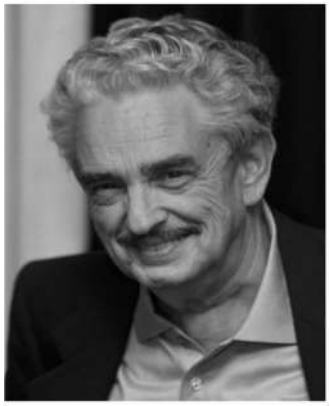
Для дальнейшего разбиения вершин на классы рассмотрим элемент μ_{ij} матрицы $U=X\cdot X'$, где X' — матрица, полученная из X заменой переменных $x_1, x_2 \ldots$ переменными $x_1', x_2', \ldots$, причем все переменные $x_1, x_2, \ldots, x_1', x_2' \ldots$ независимы. Элемент μ_{ij} является многочленом второй степени от $x_1, x_2, \ldots, x_1', x_2' \ldots$. Если теперь обозначить различные многочлены различными новыми независимыми переменными, то к полученной матрице снова можно будет применять все описанные выше операции и т. д., до тех пор, пока и этот процесс не оборвется (см. п. 6, операции α_2, β_2, β_3).

2. Геометрически введение независимых переменных в матрицу $U=X\cdot X'$ означает следующее. Если наряду с ребрами данного графа Γ рассматривать ребра дополнительного графа $\bar{\Gamma}$, то на первом же шаге нашей процедуры этим ребрам будут соответствовать наверники разные переменные — т. е. произойдет раскраска ребер данного и дополнительного графов в разные цвета. В дальнейшем каждое новое введение дополнительных переменных вводит новую раскраску ребер. Эритем 1) ребра, окрашенные на каком-то шаге по-разному, в дальнейшем также будут окрашены по-разному; 2) разбиение вершин на классы производится в соответствии с количеством ребер каждого цвета, исходящих из вершины.

Известно, далее, что элемент $a_{ij} \in A^2$, $A=A(\Gamma)$, где Γ — бесориентированный граф без кратных связей, равен числу путей длины 2, ведущих из i-й вершины в j-ую. Аналогично, коэффициент при $x_k x_l'$ в многочлене $\mu_{ij} \in U=X\cdot X'$ равен числу путей, ведущих из i-й вершины в j-ую по ребрам сначала k-го, а затем l-го цвета.

3). Дальнейшая процедура приведения графа к каноническому виду использует применение описанных выше операций к матрице, полученной из $X(\Gamma)$ вычеркиванием некоторого ее столбца и соответствующей строки. Если для матриц порядка $k \leqslant n-1$ приведение к каноническому виду определено, то возникает новая возможность для разбиения вершин на классы: старшей считается та вершина, после вычеркивания которой получается лексикографически старший канонический вид оставшегося графа (см. операции α_3, α_4, β_4, М п. 7). Очевидно, что и такое разбиение за некоторое число шагов оборвется. Доказано, что любые две вершины a и b, отнесенные к одному классу при последнем разбиении, эквивалентны, т. е. существует автоморфизм графа Γ, переводящий a в b и сохраняющий отношение смежности.

Рассмотрим снова такую матрицу $X=X(\Gamma)$, что в матрице $U=X\cdot X'$ на месте одинаковых переменных матрицы X стоят одинаковые многочлены. Матрица $X(\Gamma)$ является тогда общей точной некоторой матричной алгебры $\mathfrak{A}(\Gamma)$, т. е. если задано кольцо K (например, кольцо целых чисел Z или поле рациональных чисел Q), то множество матриц, полученных подстановкой в X вместо ее переменных элементов кольца K образует алгебру $\mathfrak{A}_K(\Gamma)=\mathfrak{A}(\Gamma)\otimes K$. Алгебра $\mathfrak{A}(\Gamma)$ является, очевидно, инвариантом графа. Некоторые соотношения между



A. Lehman

B. Weisfeiler

1968

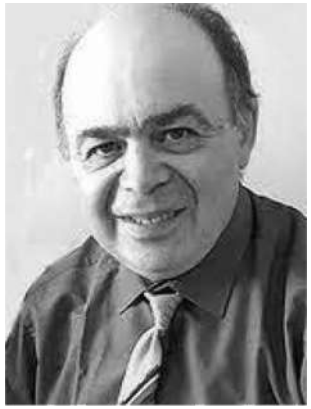
Weisfeiler, Lehman 1968; Ponomarenko 2018 (history)

History of Graph Neural Networks according to Chemists

D. Kireev

ChemNet

1995



I. Baskin

Neural descriptors

1997

C. Merkwirth

Molecular graph net

2005

D. Duvenaud

Molecular fingerprints

2015

J. Gilmer

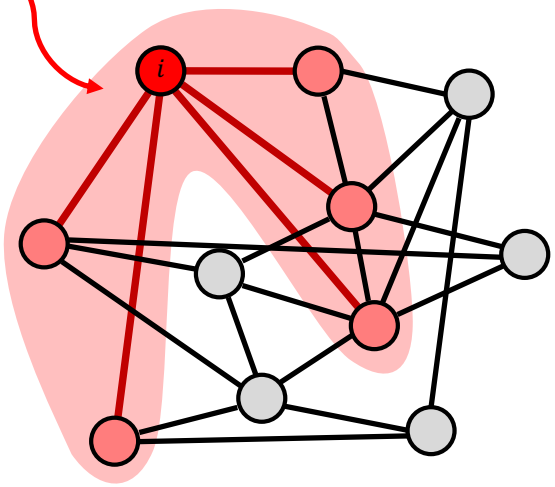
MPNNs

2017

A General Blueprint for Constructing Graph Functions

neighbourhood

$\mathcal{N}_i = \{j : i \sim j \text{ or } j \sim i\}$

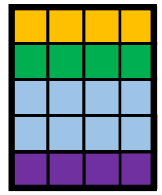


A General Blueprint for Constructing Graph Functions

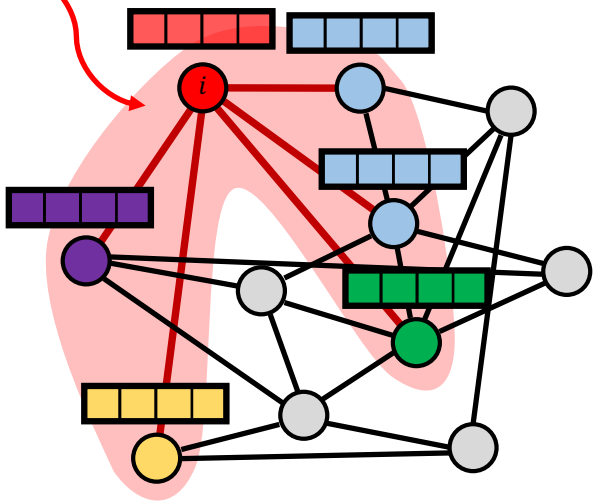
neighbourhood

$\mathcal{N}_i = \{j: i \sim j \text{ or } j \sim i\}$

multiset of
neighbour features



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



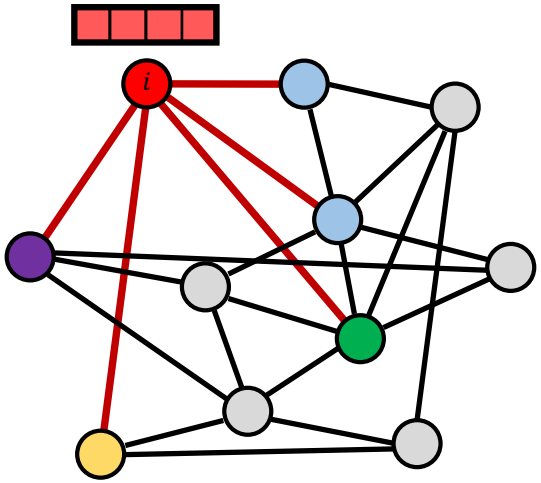
A General Blueprint for Constructing Graph Functions

multiset of neighbour features

local function

$$\phi\left(\mathbf{x}_i, \mathbf{x}_{\mathcal{N}_i}\right)$$

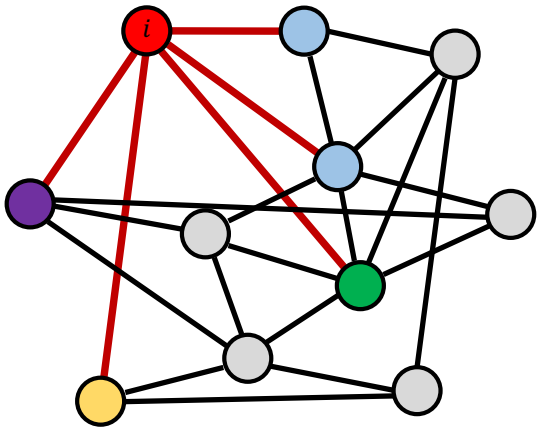
$$\mathbf{X}_{\mathcal{N}_i} = \{\!\{\mathbf{x}_{j \in \mathcal{N}_i}\}\!\}$$



A General Blueprint for Constructing Graph Functions

local function

$$\phi\left(\mathbf{x}_i, \mathbf{x}_{\mathcal{N}_i}\right)$$

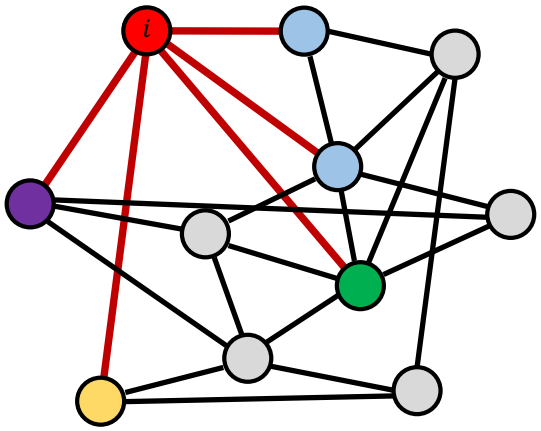


A General Blueprint for Constructing Graph Functions

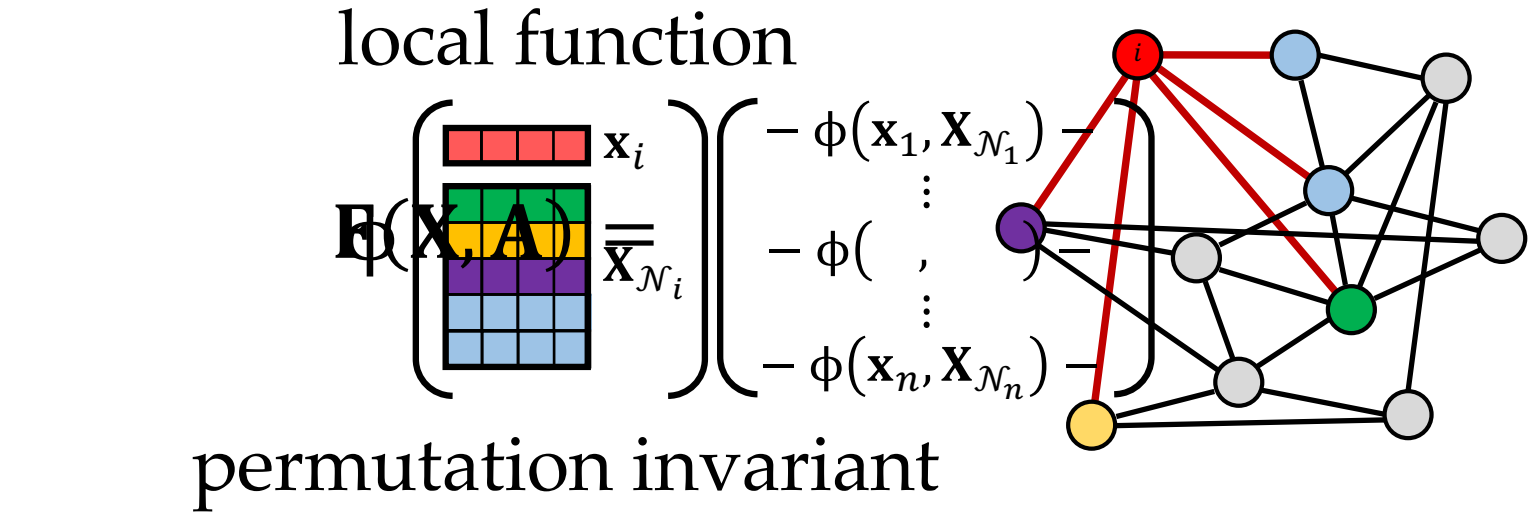
local function

$$\phi\left(\begin{bmatrix}\mathbf{x}_i \\ \mathbf{x}_{\mathcal{N}_i}\end{bmatrix}\right)$$

permutation invariant



A General Blueprint for Constructing Graph Functions

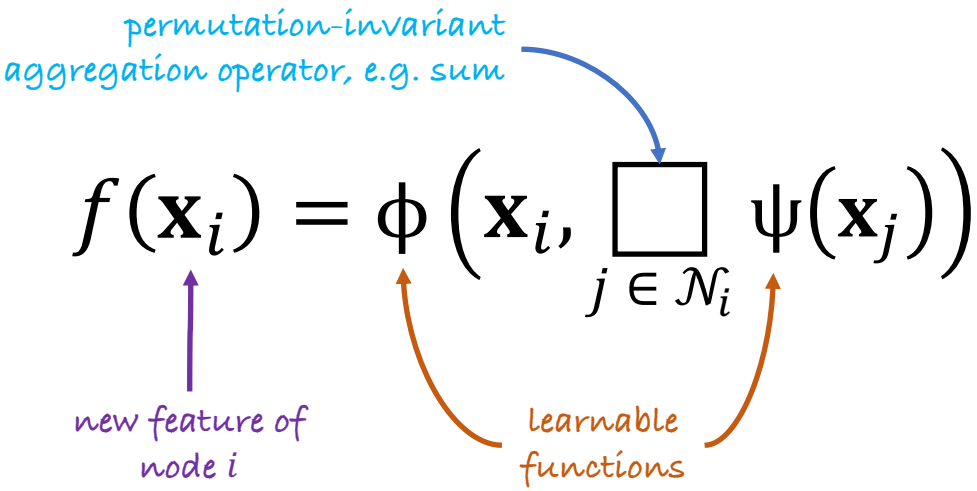


A General Blueprint for Constructing Graph Functions

$$\mathbf{F}(\mathbf{X}, \mathbf{A}) = \begin{pmatrix} - \phi(\mathbf{x}_1, \mathbf{X}_{\mathcal{N}_1}) - \\ \vdots \\ - \phi(\mathbf{x}_i, \mathbf{X}_{\mathcal{N}_i}) - \\ \vdots \\ - \phi(\mathbf{x}_n, \mathbf{X}_{\mathcal{N}_n}) - \end{pmatrix}$$

permutation equivariant

"Flavours" of Graph Neural Networks



"Flavours" of Graph Neural Networks

importance of node j to the representation of i

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

"convolutional"

Defferard et al. 2016; Kipf, Welling 2016 (GCN)

"Flavours" of Graph Neural Networks

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

"attentional"

Monti et al. 2017; Veličković et al. 2018 (GAT)

Message Passing

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

"message passing"

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

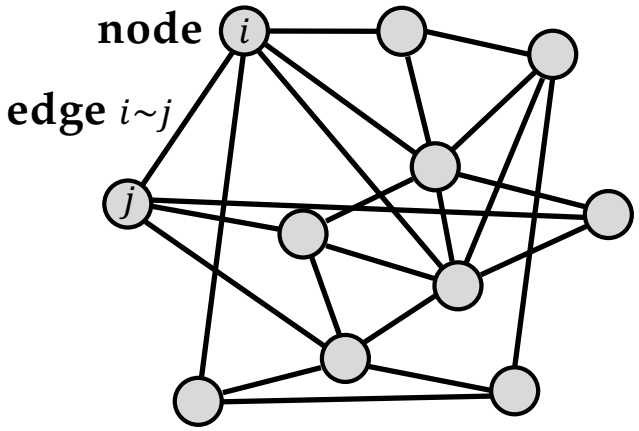
Message Passing

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

"message passing"

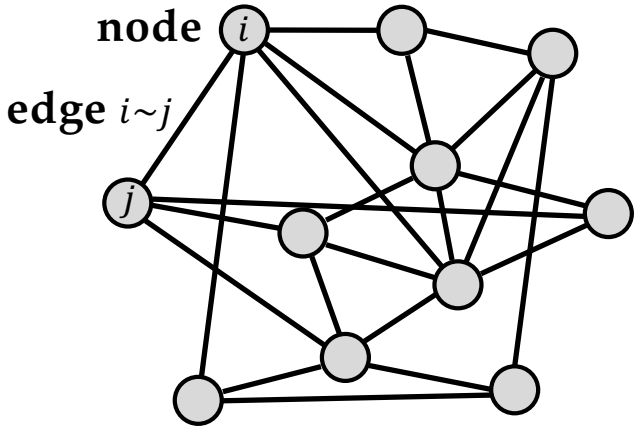
Gilmer et al. 2017 (MPNN); Battaglia et al 2018; Wang et al. 2018

Special Cases of GNNs



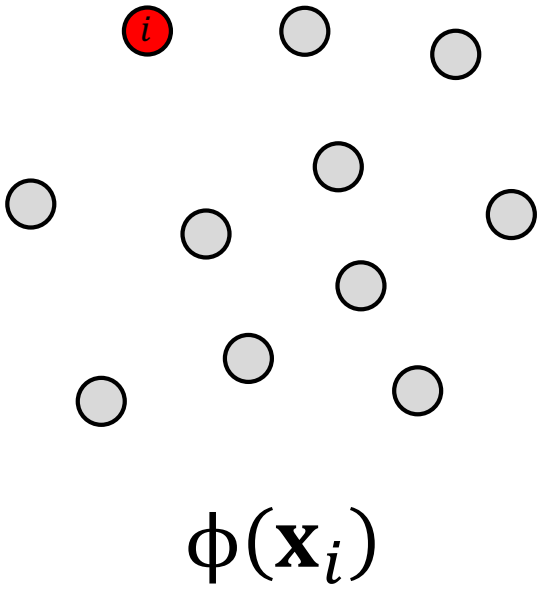
graph

Special Cases of GNNs



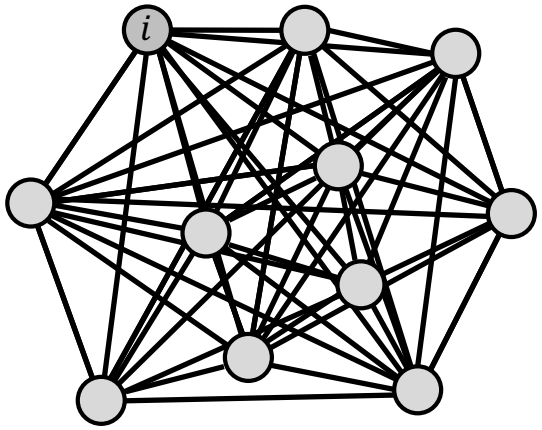
graph

DeepSets



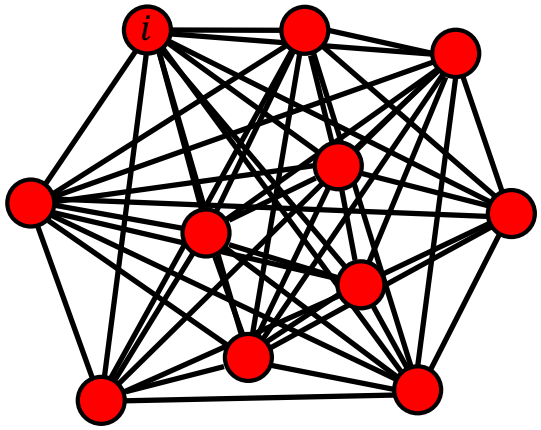
Zaheer et al 2017; Qi et al 2017

Transformers



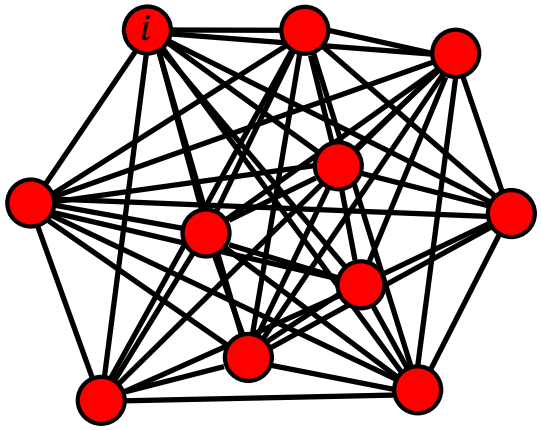
complete graph

Transformers



$$\phi\left(\mathbf{x}_i, \square_{j=1}^{n} c_{ij}\,\psi(\mathbf{x}_j)\right)$$

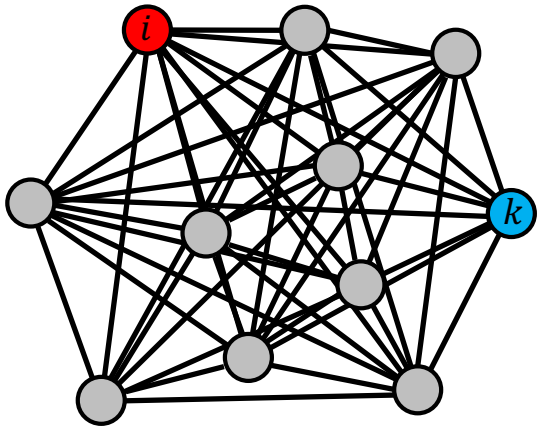
Transformers



$$\phi\left(\mathbf{x}_i, \square_{j=1}^{n} a\left(\mathbf{x}_i, \mathbf{x}_j\right)\psi\left(\mathbf{x}_j\right)\right)$$

Vaswani et al. 2017

Transformers

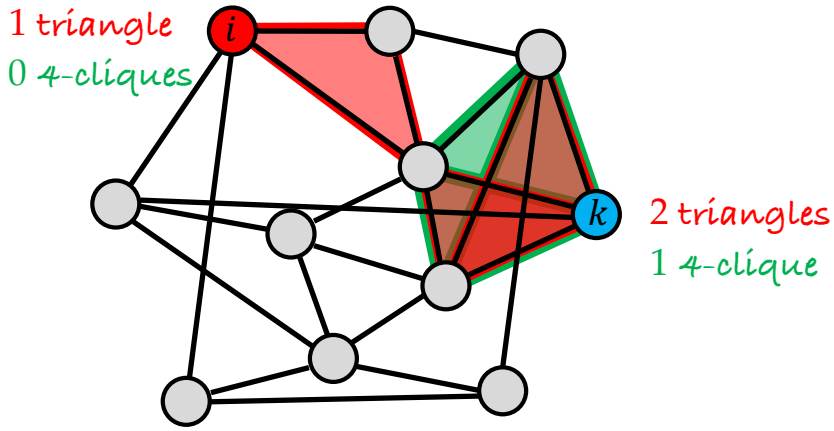


$$\phi\left(\mathbf{x}_i, \square_{j=1}^{n} a\left(\mathbf{x}_i, \mathbf{x}_j, \mathbf{p}_i, \mathbf{p}_j\right)\psi\left(\mathbf{x}_j\right)\right)$$

positional encoding

Vaswani et al. 2017

Graph Substructure Network

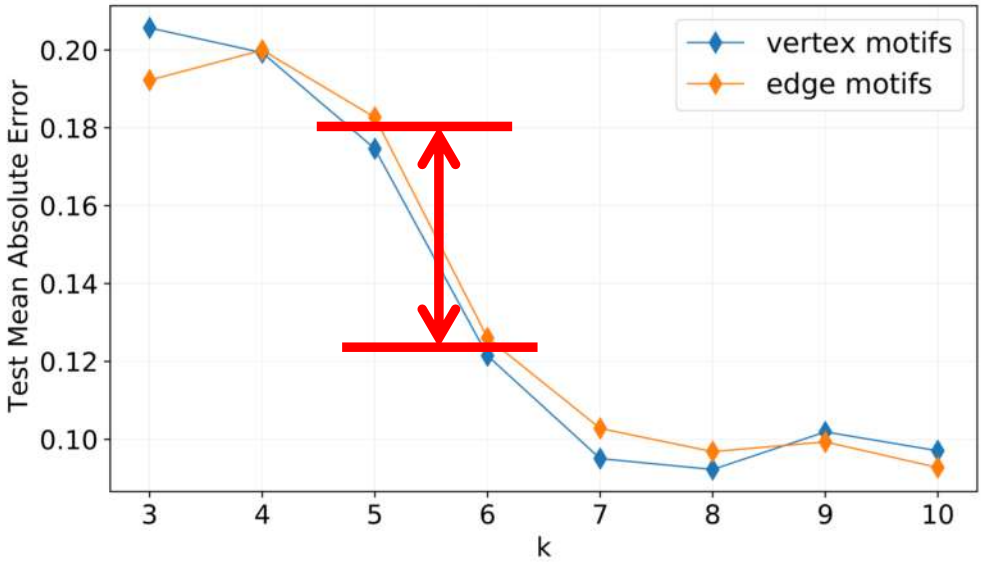


$$\phi\left(\mathbf{x}_i, \square_{j \in \mathcal{N}_i} \psi\left(\mathbf{x}_i, \mathbf{x}_j, \mathbf{p}_i\right)\right)$$

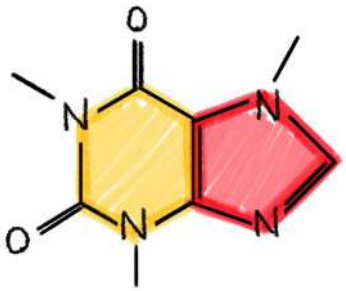
structural encoding

Bouritsas, Frasca et B 2020

Graph Substructure Network



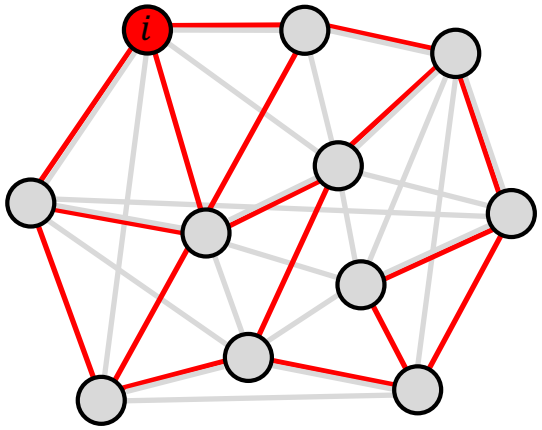
Molecule property prediction on ZINC using GSN with *k*-cycles



Molecule of caffeine

Bouritsas, Frasca et B 2020

decouple computational graph from the input graph



sampling

rewiring

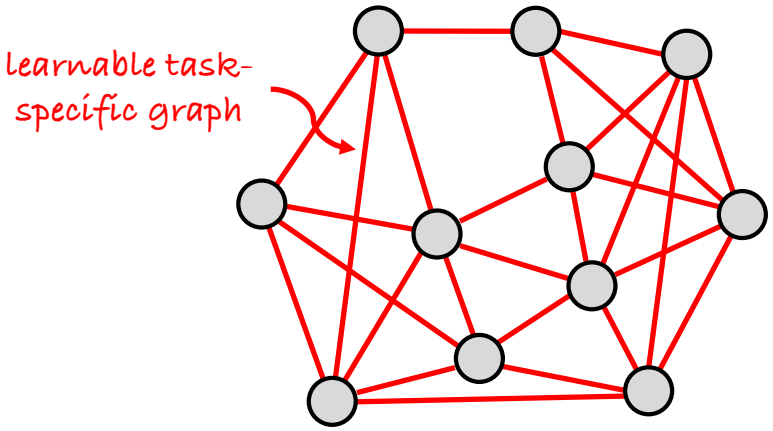
multi-hop filters

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

new graph

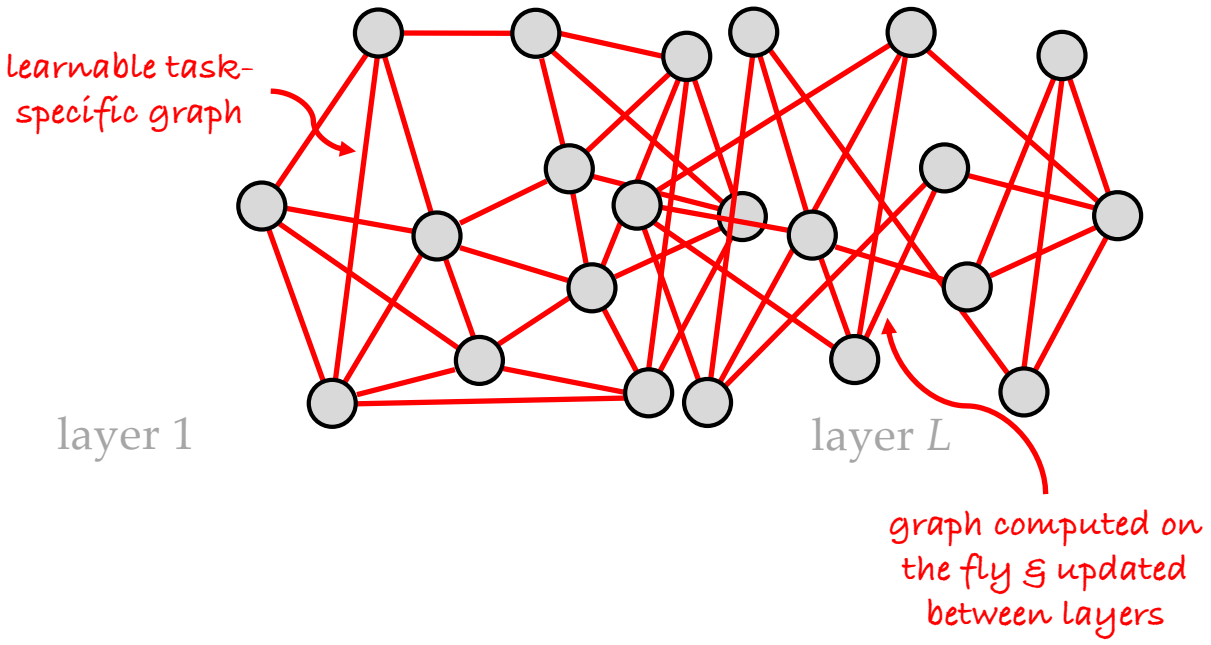
Hamilton et al 2017; Klicpera et al. 2019; Alon & Yahav 2020; Frasca, Rossi et B 2020

Latent Graph Learning

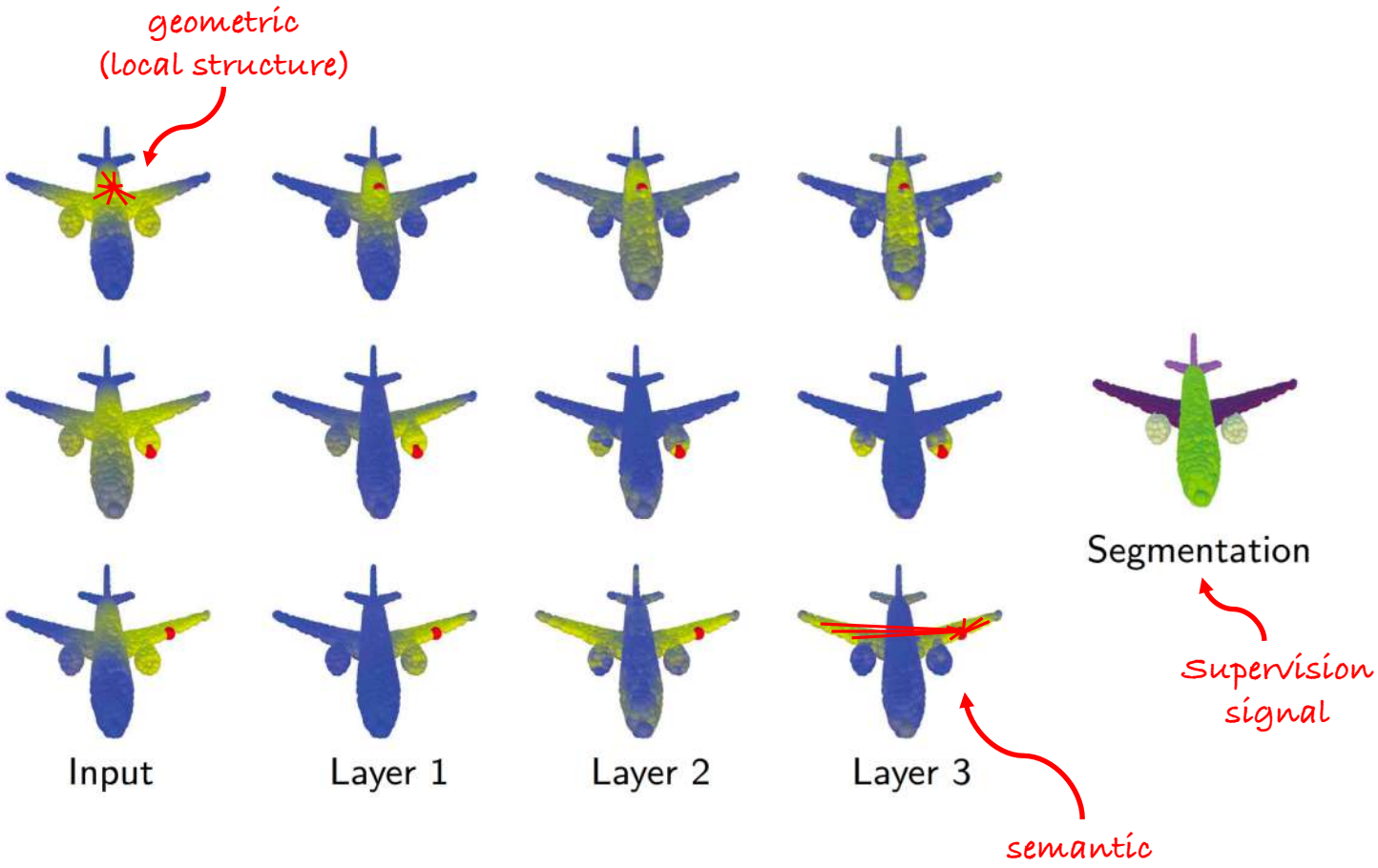


Wang et B 2018; Franceschi et al. 2019; Kipf et al. 2020; Kazi, Cosmo et B 2020; Cranmer et al. 2020

Dynamic Graph CNN



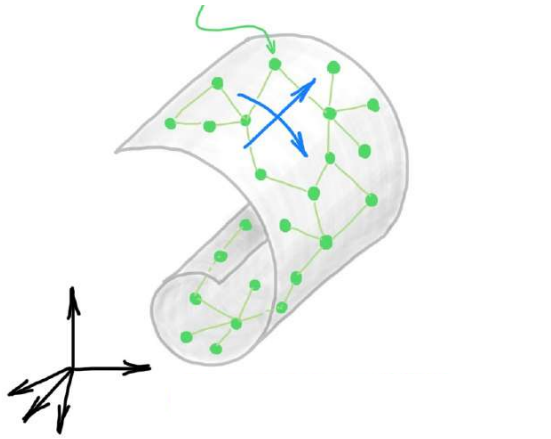
Wang et B 2018; Franceschi et al. 2019; Kipf et al. 2020; Kazi, Cosmo et B 2020; Cranmer et al. 2020



Wang et B 2018

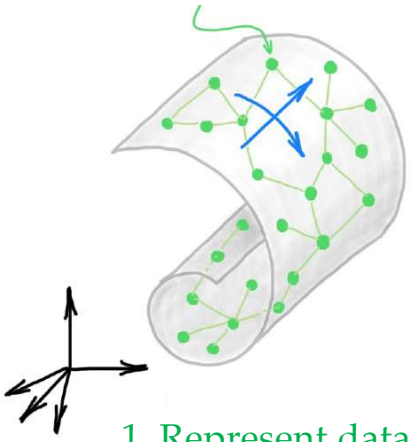
Manifold Learning

intrinsically low-dimensional
data in a high-dimensional space

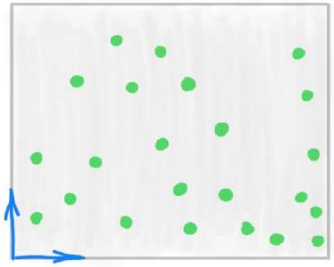


Manifold Learning

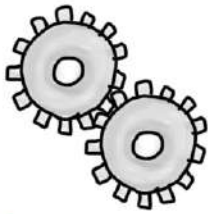
intrinsically low-dimensional
data in a high-dimensional space



1. Represent data structure as a graph



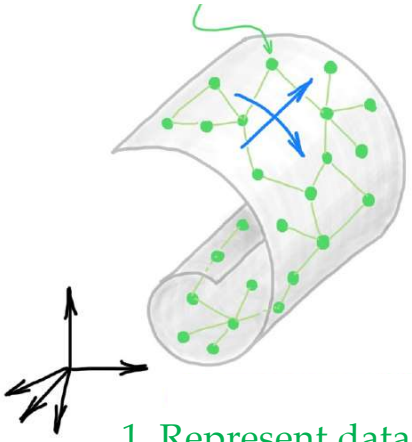
2. Compute low-dimensional embedding



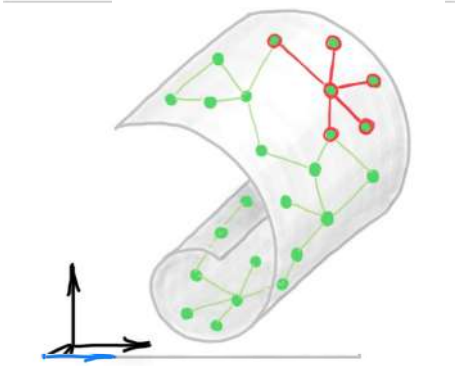
3. Apply ML

Manifold Learning 2.0

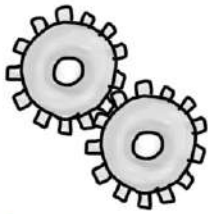
intrinsically low-dimensional
data in a high-dimensional space



1. Represent data structure as a graph



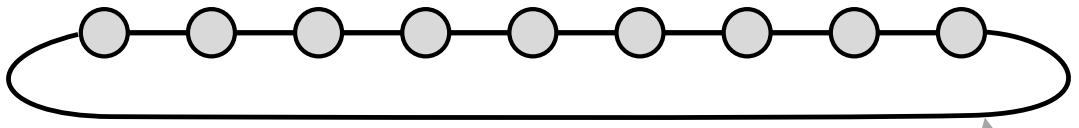
1. Represent data structure as a graph apply ML directly on the graph



3. Apply ML

GRIDS

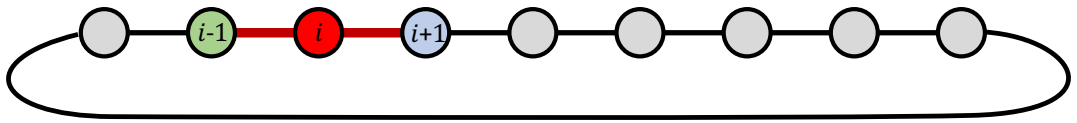
Grids vs Graphs



grid = ring graph

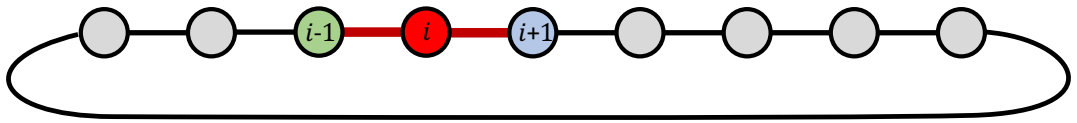
periodic b.c.

Grids vs Graphs



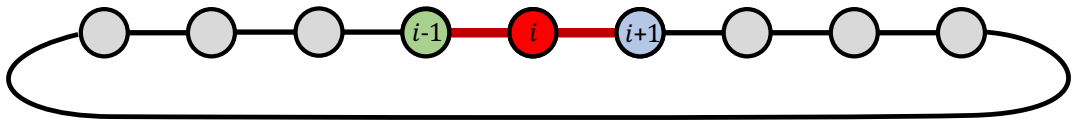
fixed neighbourhood structure

Grids vs Graphs



fixed neighbourhood structure

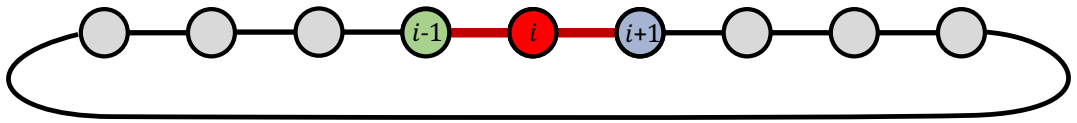
Grids vs Graphs



local aggregation function

$$f(\mathbf{x}_i) = \phi(\mathbf{x}_i, \{\mathbf{x}_{i-1}, \mathbf{x}_{i+1}\})$$

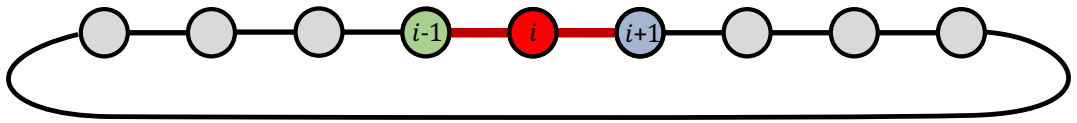
Grids vs Graphs



local aggregation function

$$f(\mathbf{x}_i) = \phi(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1})$$

Grids vs Graphs



linear local aggregation function

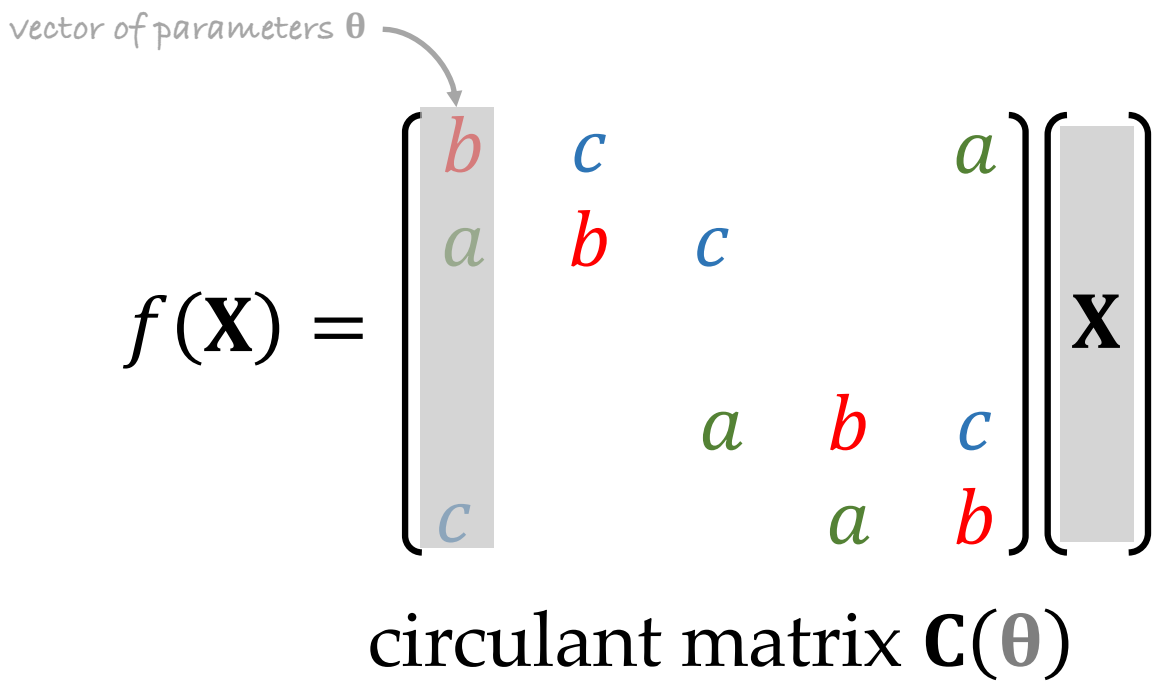
$$f(\mathbf{x}_i) = a\mathbf{x}_{i-1} + b\mathbf{x}_i + c\mathbf{x}_{i+1}$$

Convolution

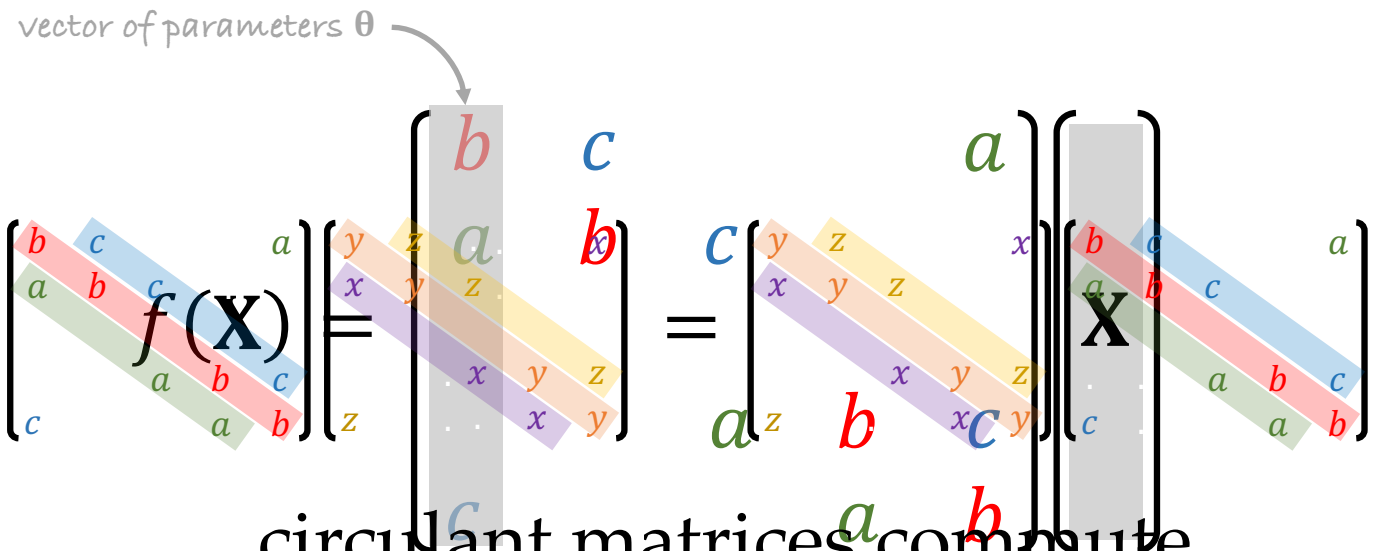
$$f(\mathbf{X}) = \begin{bmatrix} b & c & & & a \\ a & b & c & & \\ & a & b & c & \\ & & a & b & c \\ c & & & a & b \end{bmatrix} \begin{bmatrix} \\ \mathbf{X} \\ \\ \\ \\ \end{bmatrix}$$

circulant matrix = convolution

Convolution



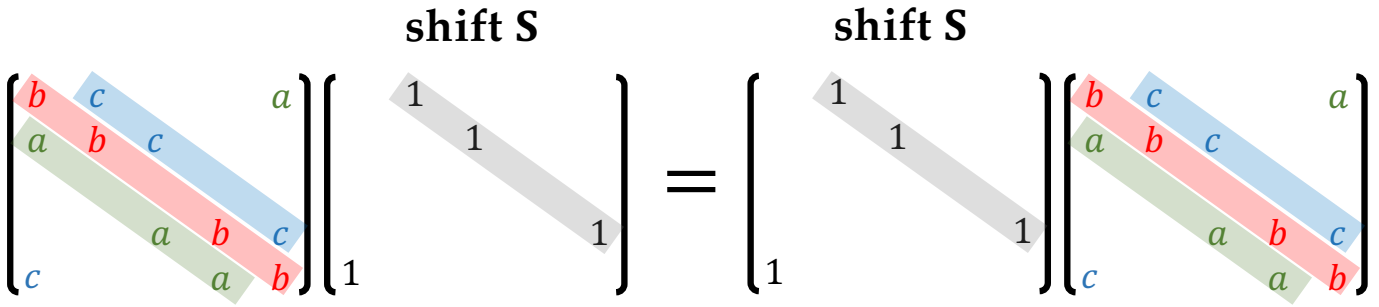
Deriving Convolution from Symmetry



circulant matrices commute

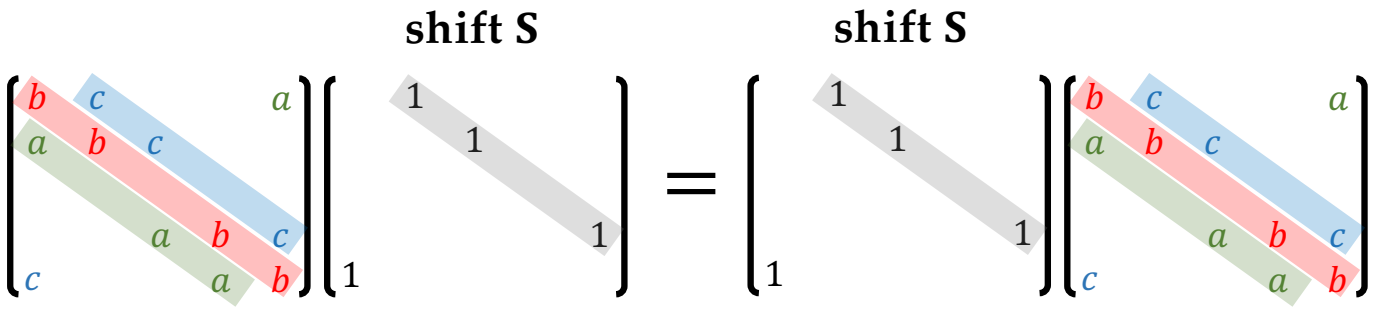
circulant matrix $\mathbf{C}(\theta)$

Deriving Convolution from Symmetry



circulant matrix $\Longrightarrow$ commutes with shift

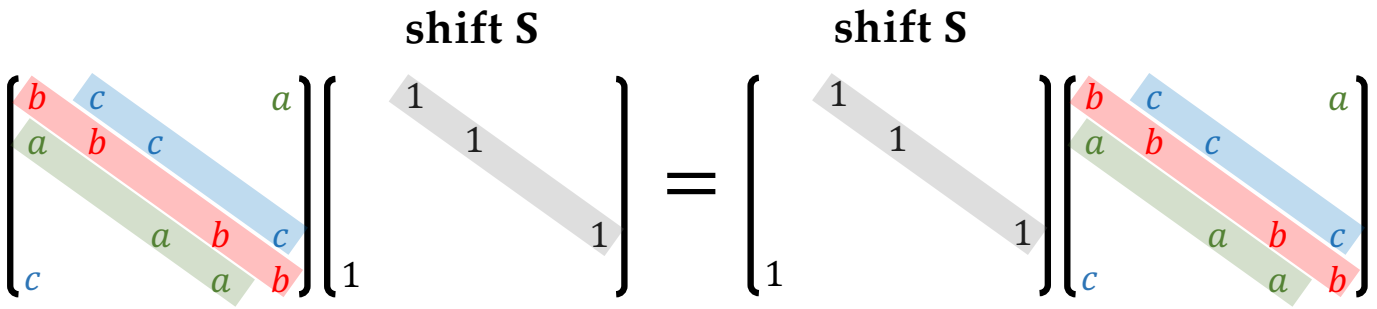
Deriving Convolution from Symmetry



convolution $\Longrightarrow$ shift-equivariant

$$\mathbf{CS = SC}$$

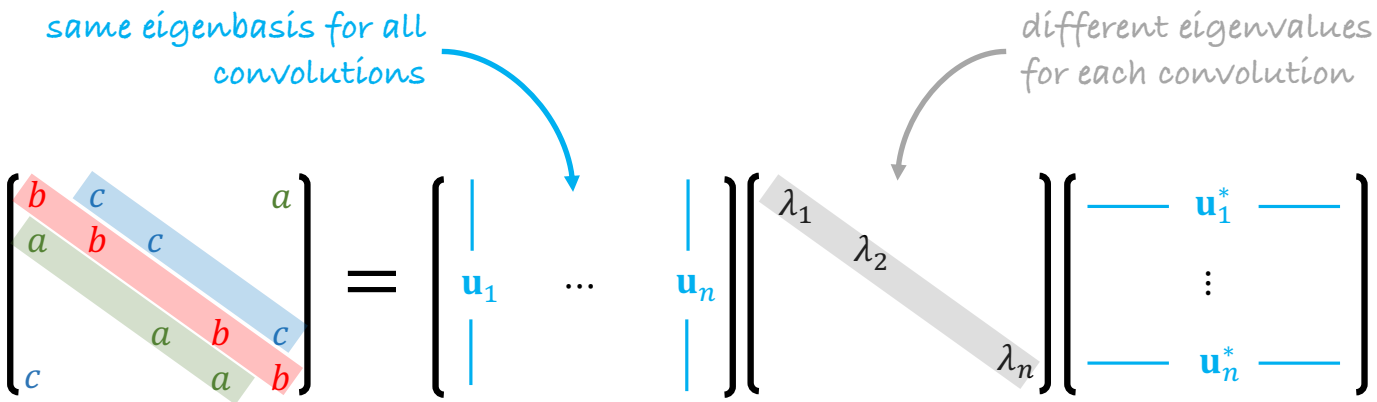
Deriving Convolution from Symmetry



convolution $\Leftrightarrow$ shift-equivariant

convolution **emerges** from translation symmetry

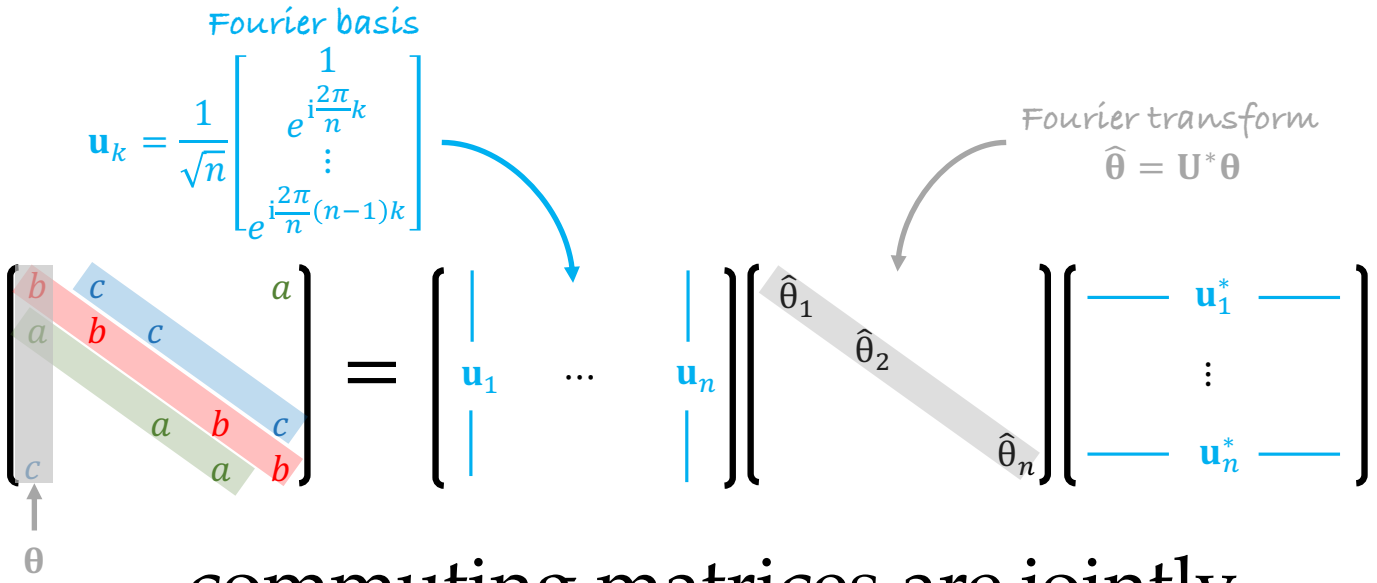
Deriving Fourier Transform from Symmetry



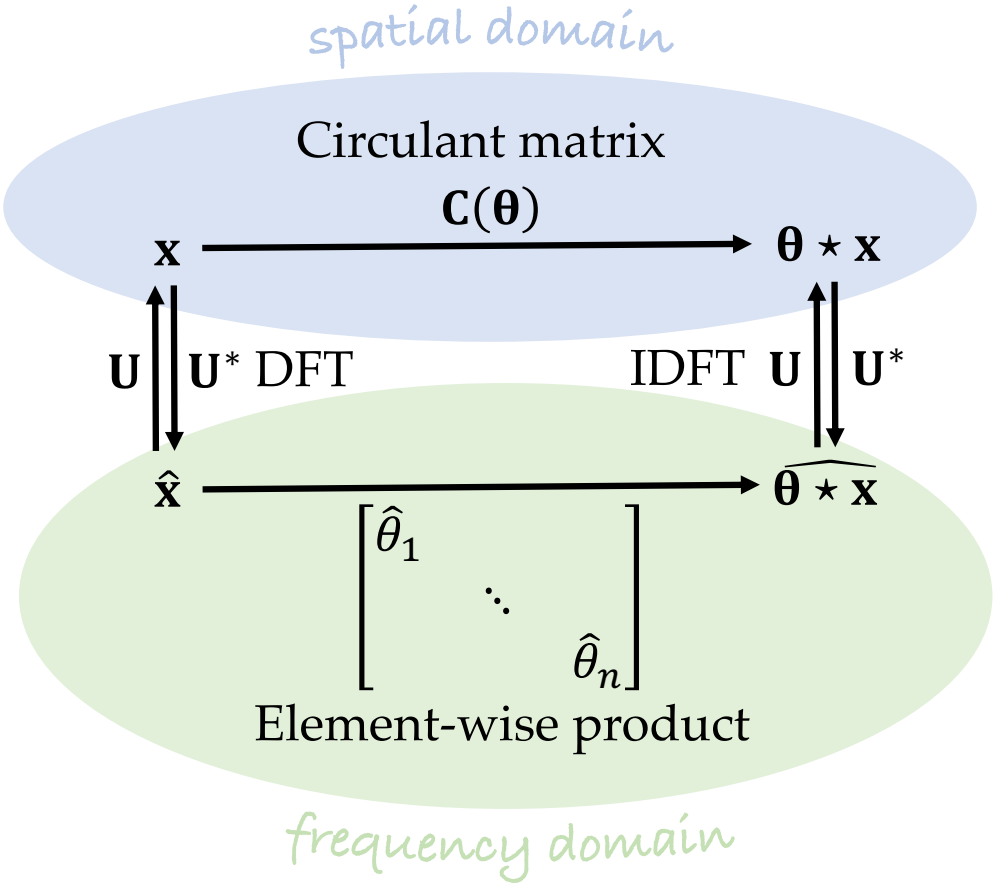
commuting matrices are jointly diagonalisable

by the eigenvectors of **S**

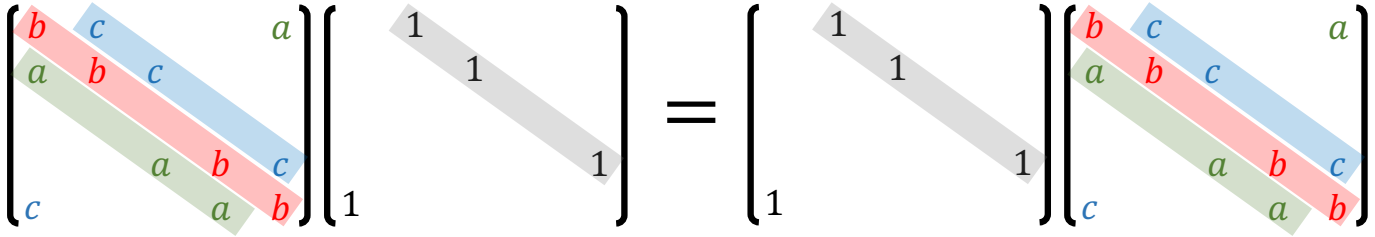
Deriving Fourier Transform from Symmetry



commuting matrices are jointly diagonalisable by Fourier Transform

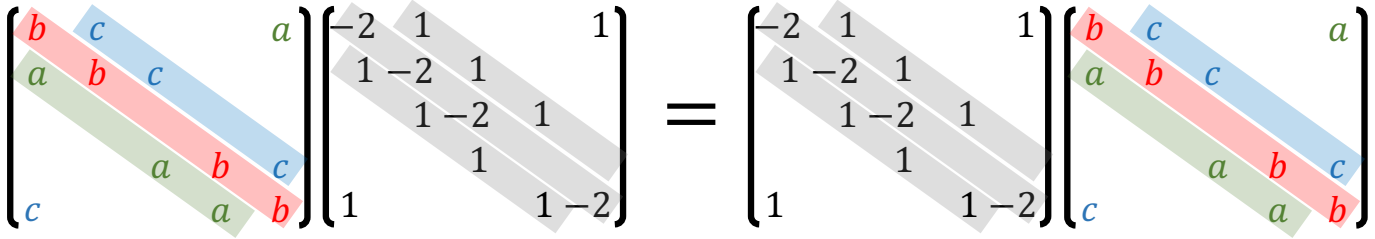

spatial domain
Circulant matrix
$\mathbf{C}(\boldsymbol{\theta})$
$\mathbf{x}$
$\boldsymbol{\theta} \star \mathbf{x}$
$\mathbf{U}$
$\mathbf{U}^*$ DFT
IDFT $\mathbf{U}$
$\mathbf{U}^*$
$\hat{\mathbf{x}}$
$\widehat{\boldsymbol{\theta} \star \mathbf{x}}$
$\begin{bmatrix} \hat{\theta}_1 & & \\ & \ddots &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

Reminder: Euclidean case



- Convolutions commute
- Commuting matrices are jointly diagonalizable (“have same eigenvectors”)
- Pick up the shift operator to show eigenvectors = DFT $\Longrightarrow$ convolutions are diagonalised by the Fourier transform

Reminder: Euclidean case



- Convolutions commute
- Commuting matrices are jointly diagonalizable (“have same eigenvectors”)
- Pick up the *Laplacian operator* whose eigenvectors are the DFT ⟹ convolutions are diagonalised by the Fourier transform

Laplacian Operator

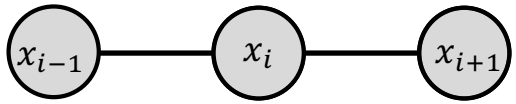
- Fourier basis functions = Laplacian eigenfunctions

$$\Delta e^{+i\omega u} = -\omega^2 e^{+i\omega u}$$

- Fourier basis = orthogonal basis minimising the *Dirichlet energy*

$$\varphi_{k+1} = \underset{\varphi}{\operatorname{argmin}} \int_{-\infty}^{+\infty} \|\nabla \varphi_k(u)\|^2 du \quad \text{s.t.} \quad \|\varphi\| = 1 \quad \text{and} \quad \langle \varphi, \varphi_j \rangle = 0 \text{ where } j = 1, \dots k$$

- "smoothest orthogonal basis"
- "local difference from neighbours"



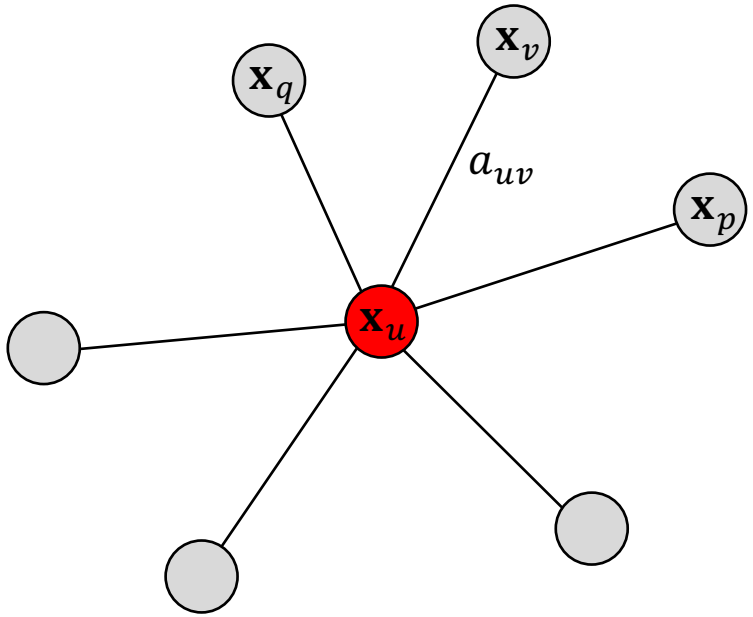
$$(x_{i-1} + x_{i+1}) - 2x_i$$

Graph Laplacian

$$(\mathbf{\Delta x})_u = \sum_{v \in \mathcal{N}_u} a_{uv}(\mathbf{x}_u - \mathbf{x}_v)$$

$$= d_u \mathbf{x}_u - \sum_{v \in \mathcal{N}_u} a_{uv} \mathbf{x}_v$$

$$= \phi\left(\mathbf{x}_u, \mathbf{X}_{\mathcal{N}_u}\right)$$



- Linear local *permutation-invariant* aggregator $\phi\left(\mathbf{x}_u, \mathbf{X}_{\mathcal{N}_u}\right)$
- Node-wise *permutation-equivariant* linear function $\mathbf{F}(\mathbf{X}) = \mathbf{\Delta X}$

Graph Fourier Transform & Convolution

On undirected graph, the Graph Laplacian is a symmetric matrix admitting an orthogonal eigendecomposition $\mathbf{\Delta} = \mathbf{\Phi}\mathbf{\Lambda}\mathbf{\Phi}^{\mathrm{T}}$ with $\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi} = \mathbf{I}$

- *Graph Fourier Transform:* $\hat{\mathbf{x}} = \mathbf{\Phi}^{\mathrm{T}}\mathbf{x}$ $O(n^2)$ complexity
- *Inverse Graph Fourier Transform:* $\mathbf{x} = \mathbf{\Phi}\hat{\mathbf{x}}$ $O(n^2)$ complexity
- *Spectral Convolution:*

$$\begin{aligned}\mathbf{x} \star \boldsymbol{\psi} &= \mathbf{\Phi}\left(\left(\mathbf{\Phi}^{\mathrm{T}}\mathbf{x}\right) \cdot \left(\mathbf{\Phi}^{\mathrm{T}}\boldsymbol{\psi}\right)\right) \\ &= \mathbf{\Phi}\,\mathrm{diag}\left(\hat{\boldsymbol{\psi}}\right)\hat{\mathbf{x}} \qquad O(n^2) \text{ complexity}\end{aligned}$$

ChebNet

- *Polynomial spectral filter* $\hat{p}(\lambda) = \sum_{l=0}^{r} \alpha_l \lambda^l$ applied to the Graph Laplacian

$$\hat{p}(\boldsymbol{\Delta}) = \boldsymbol{\Phi}\hat{p}(\boldsymbol{\Lambda})\boldsymbol{\Phi}^{\mathrm{T}} = \boldsymbol{\Phi}\sum_{l=0}^{r} \alpha_l \boldsymbol{\Lambda}^l \ \boldsymbol{\Phi}^{\mathrm{T}}$$

Defferrard et al. 2016

ChebNet

- *Polynomial spectral filter* $\hat{p}(\lambda) = \sum_{l=0}^{r} \alpha_l \lambda^l$ applied to the Graph Laplacian

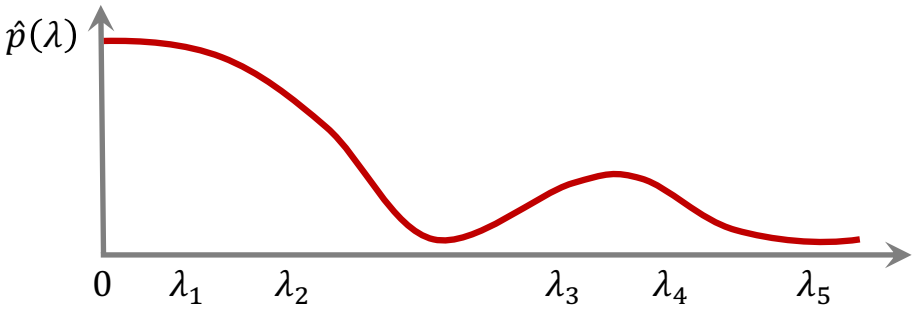
$$\hat{p}(\boldsymbol{\Delta}) = \boldsymbol{\Phi}\hat{p}(\boldsymbol{\Lambda})\boldsymbol{\Phi}^{\mathrm{T}} = \sum_{l=0}^{r} \alpha_l \, \boldsymbol{\Phi}\boldsymbol{\Lambda}^l\boldsymbol{\Phi}^{\mathrm{T}} = \sum_{l=0}^{r} \alpha_l \, \boldsymbol{\Delta}^l$$

- No explicit eigendecomposition!
- Apply powers of the Graph Laplacian – complexity $O(r|\mathcal{E}|) \sim O(n)$
- Stable under graph deformations
- ”Spectral” filter boils down to simple local averaging – “convolutional flavor” of GNN

Defferrard et al. 2016

Spectral vs Spatial Filters

$$(\hat{p}(\Delta)x)(u) = \sum_{k\geq 0} \underbrace{\hat{p}(\lambda_k)}_{\hat{\psi}_k} \underbrace{\langle x, \varphi_k \rangle}_{\hat{x}_k} \varphi_k(u)$$

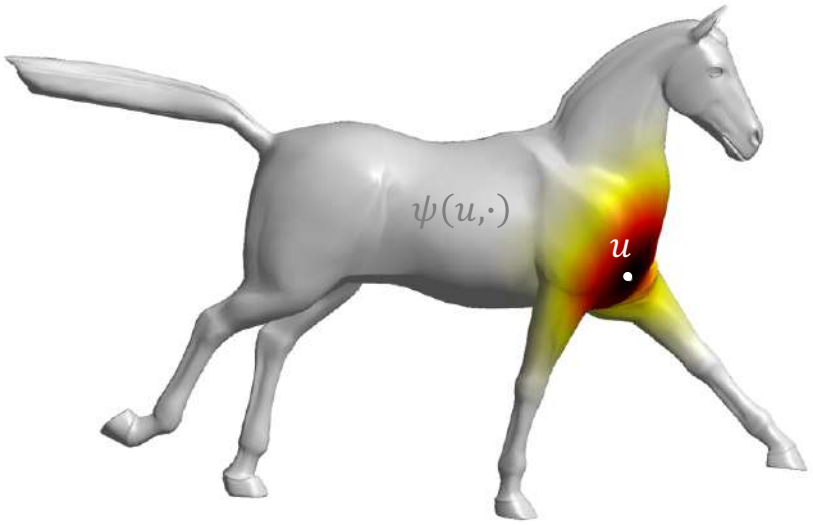


spectral transfer function

Stable Spectral Filters

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

$$= \int_\Omega x(u) \underbrace{\sum_{k\geq 0} \hat{p}(\lambda_k)\varphi_k(v)\varphi_k(u)}_{\psi(u,v)}\, \mathrm{d}v$$

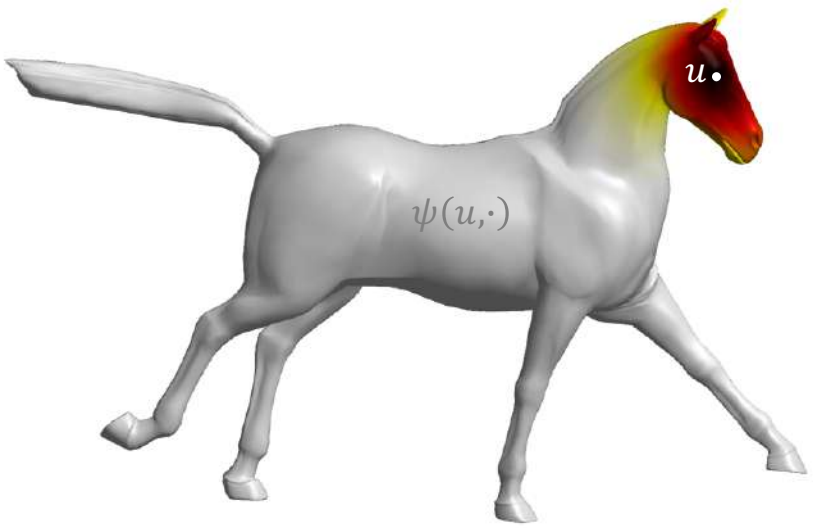


position-dependent spatial kernel

Stable Spectral Filters

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

$$= \int_\Omega x(u) \underbrace{\sum_{k\geq 0} \hat{p}(\lambda_k)\varphi_k(v)\varphi_k(u)}_{\psi(u,v)}\, \mathrm{d}v$$

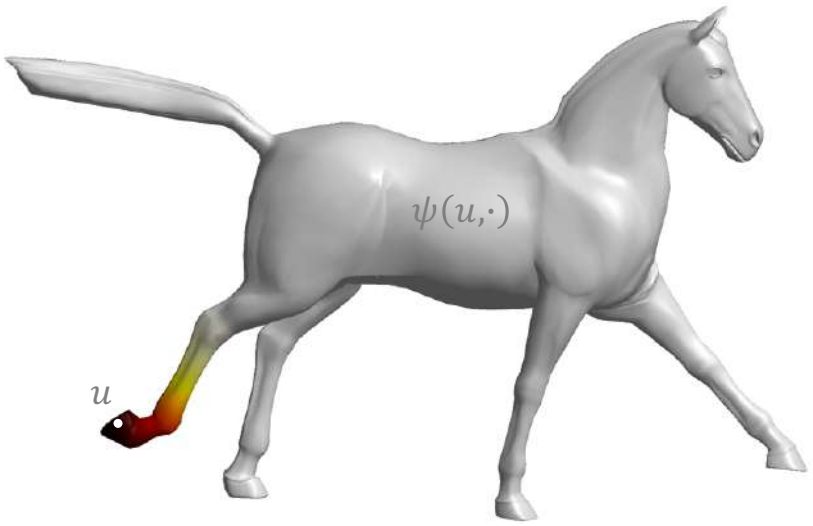


position-dependent spatial kernel

Stable Spectral Filters

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

$$= \int_\Omega x(u) \underbrace{\sum_{k\geq 0} \hat{p}(\lambda_k)\varphi_k(v)\varphi_k(u)}_{\psi(u,v)}\, \mathrm{d}v$$



position-dependent spatial kernel

Sanity Check

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

$$= \int_{-\pi}^{+\pi} x(u) \underbrace{\sum_{k\geq 0} \hat{p}(k)\mathrm{e}^{-\mathrm{i}kv}\mathrm{e}^{+\mathrm{i}ku}}_{\psi(u-v)}\, \mathrm{d}v$$

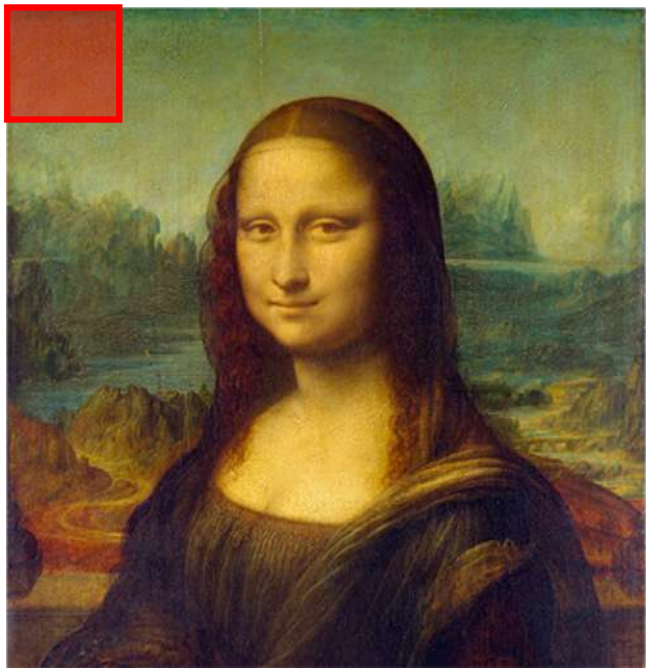
standard convolution in the Euclidean case

"Spectral GNNs"

- Spectral mesh filters [Taubin et al. 1996; Karni, Gotsman 2000]
- Graph Signal Processing [Shuman et al. 2013; Sandryhaila, Moura 2013]
- *Graph Fourier Transform* [Bruna et al. 2013]
  - Revival of the interest in Graph Neural Networks
  - High complexity – $O(n^2)$
  - Graph FT unstable under perturbations
- *Polynomial spectral filters a.k.a. ChebNet* [Defferrard et al. 2016]: $\hat{p}(\lambda) = \sum_{l=0}^{r} \alpha_l \lambda^l$
- *Rational spectral filters a.k.a. CayleyNet* [Levie et al. 2018]
- Spectral filter stability results [Gama et al. 2019; Levie et al. 2019; Kenlay et al. 2021]

GROUPS

Convolution, revisited



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

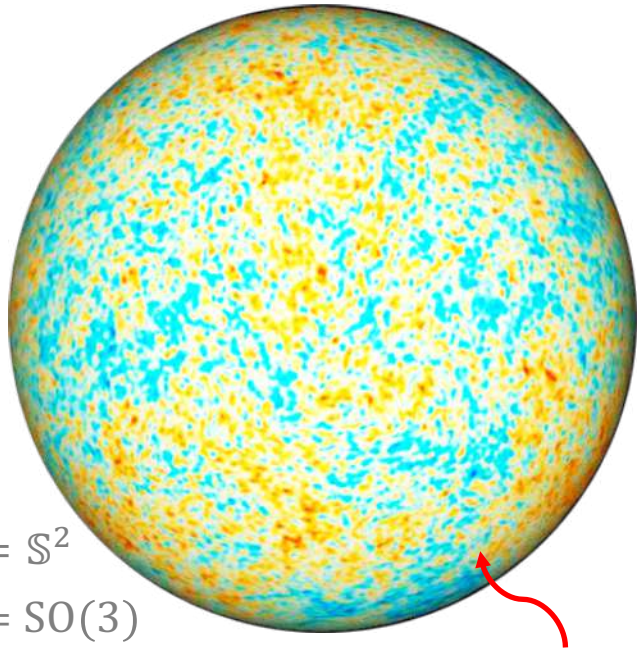
convolution = matching transformed filter

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

group element

group representation

Convolution on the Sphere



sphere $\Omega = \mathbb{S}^2$

rotation group $\mathfrak{G} = \mathrm{SO}(3)$

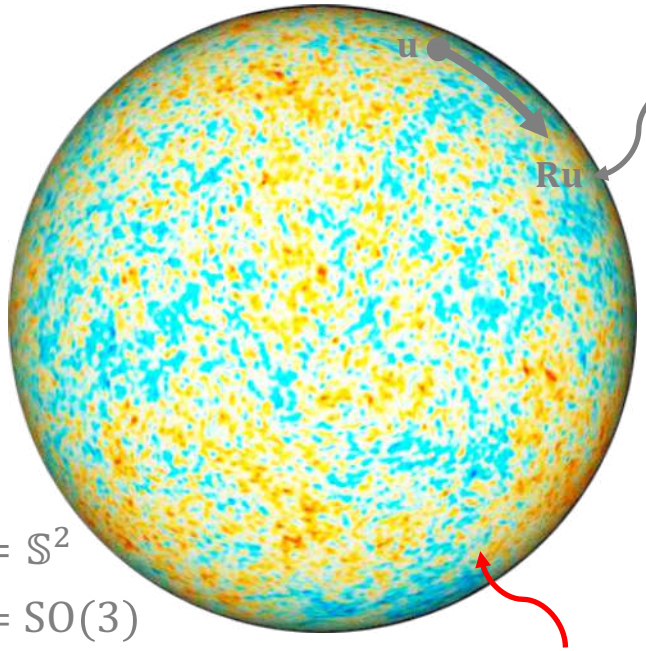
spherical signal x

Cohen, Welling 2016

Convolution on the Sphere

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

signal on $\mathrm{SO}(3)$

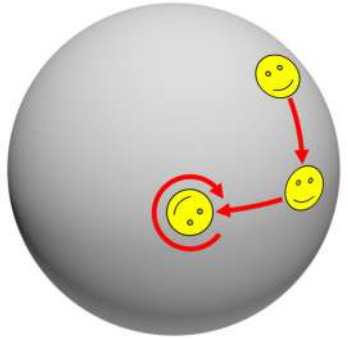


rotation transformation

sphere $\Omega = \mathbb{S}^2$

rotation group $\mathfrak{G} = \mathrm{SO}(3)$

spherical signal x

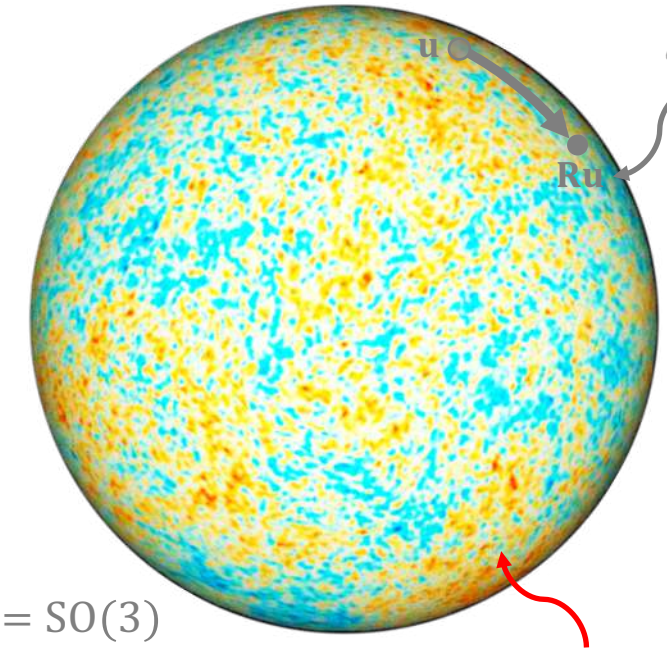


Cohen, Welling 2016

Convolution on the Sphere

$$\big((x \star \psi) \star \phi\big)(\mathbf{R}) = \int_{\mathrm{SO}(3)} (x \star \psi)(\mathbf{Q})\,\phi(\mathbf{R}^{-1}\mathbf{Q})\,\mathrm{d}\mathbf{Q}$$

convolution on SO(3)



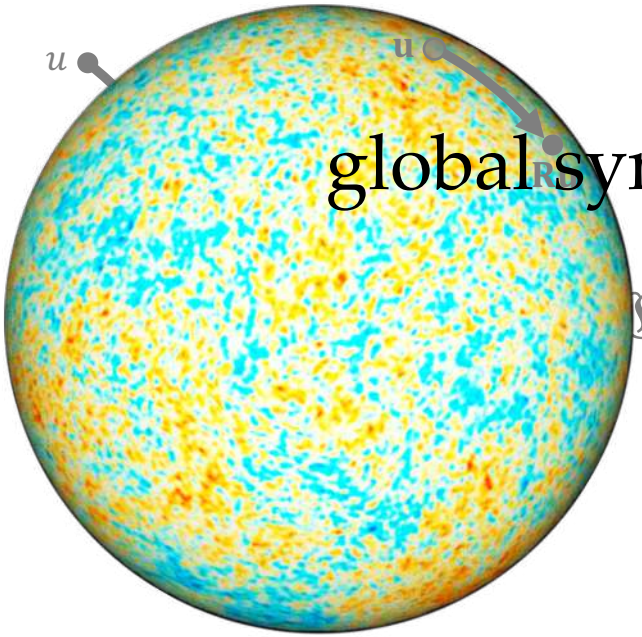
rotation transformation

$\Omega = \mathfrak{G} = \mathrm{SO}(3)$

spherical signal x

Cohen, Welling 2016

Homogeneous Spaces

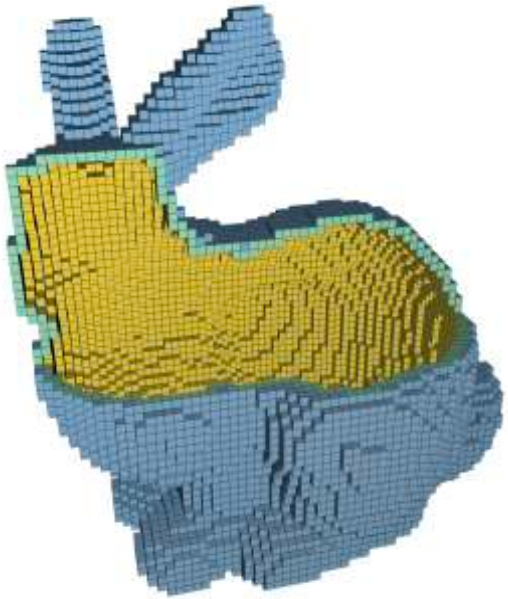


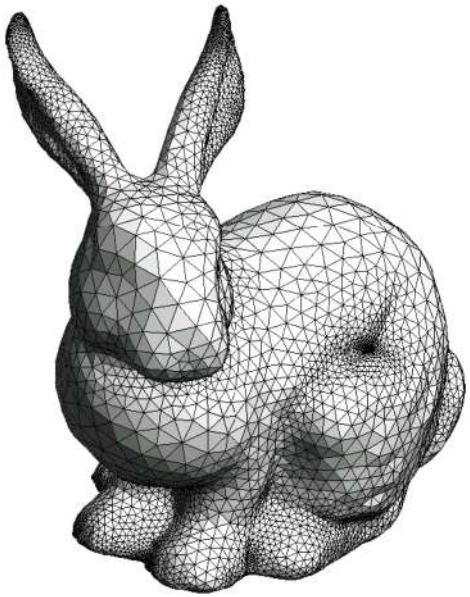
global symmetry group

$\mathfrak{H}$ s.t. $gu = v$

GEODESICS & GAUGES

Why manifolds?





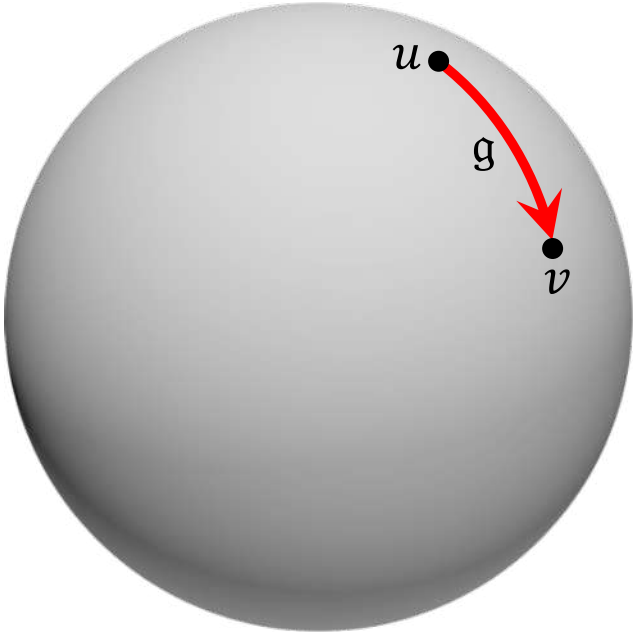
More efficient representation: no “waste” for internal structures

Natural model for deformable shapes

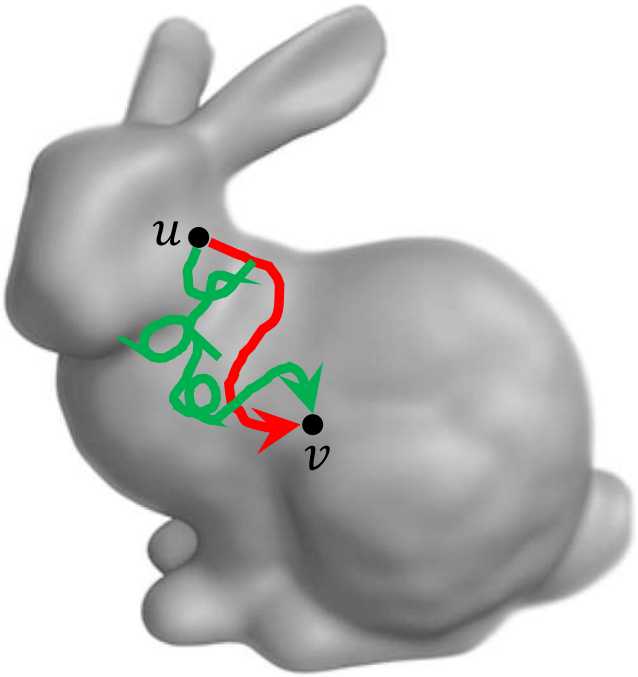
Why manifolds?

In protein modeling, abstract out internal structure that is irrelevant for interactions + allow some conformation changes

Homogeneous Spaces

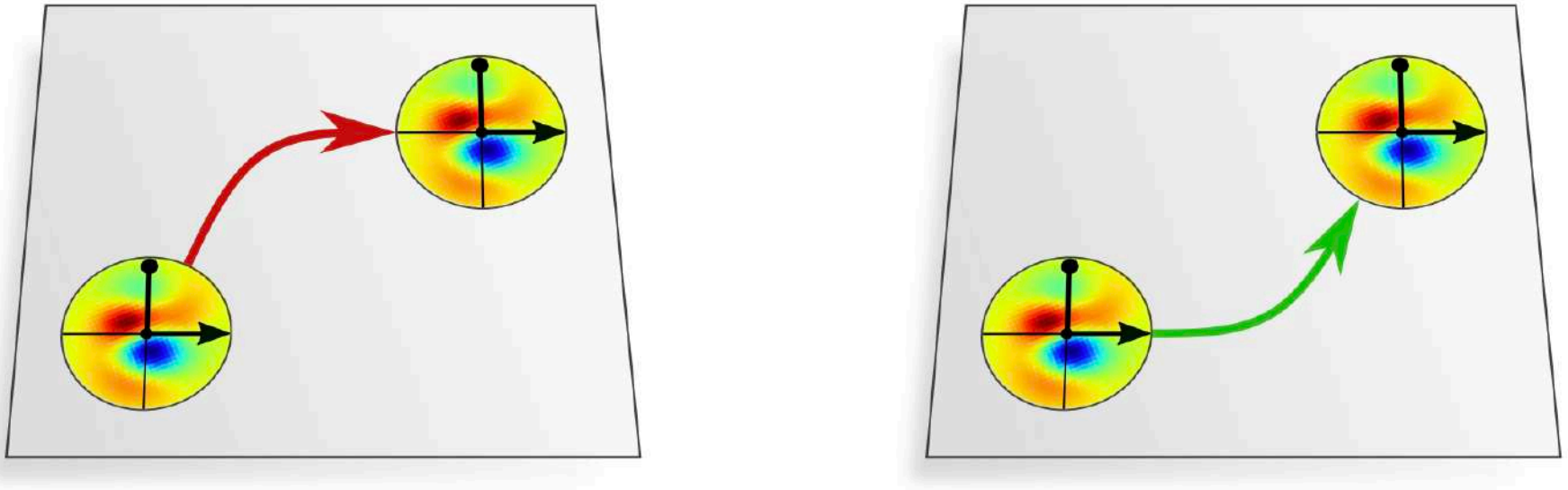


global symmetry group
$\exists \mathfrak{g} \in \mathfrak{G}$ s.t. $\mathfrak{g}u = v$



no global symmetry group

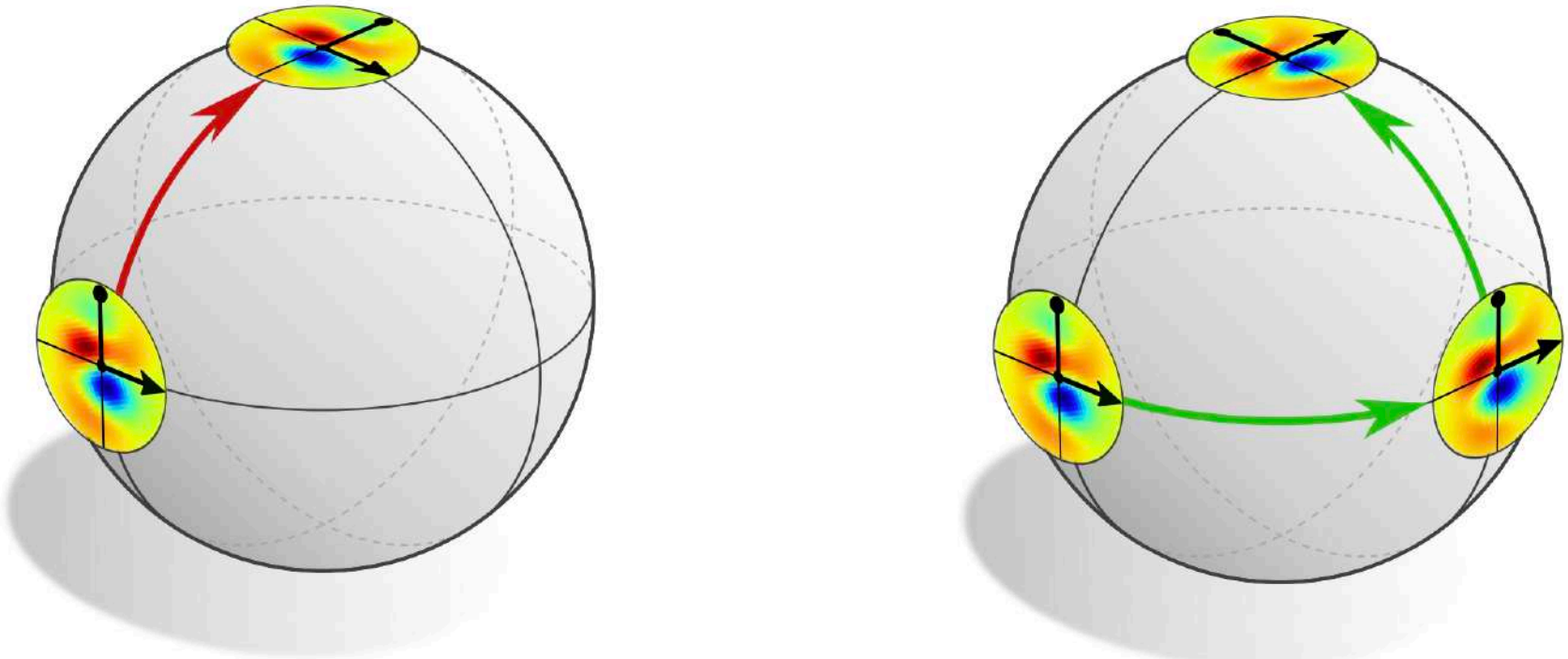
Euclidean Convolution



Euclidean space: Transport the filter around the domain

Figure: M. Weiler et al. 2021

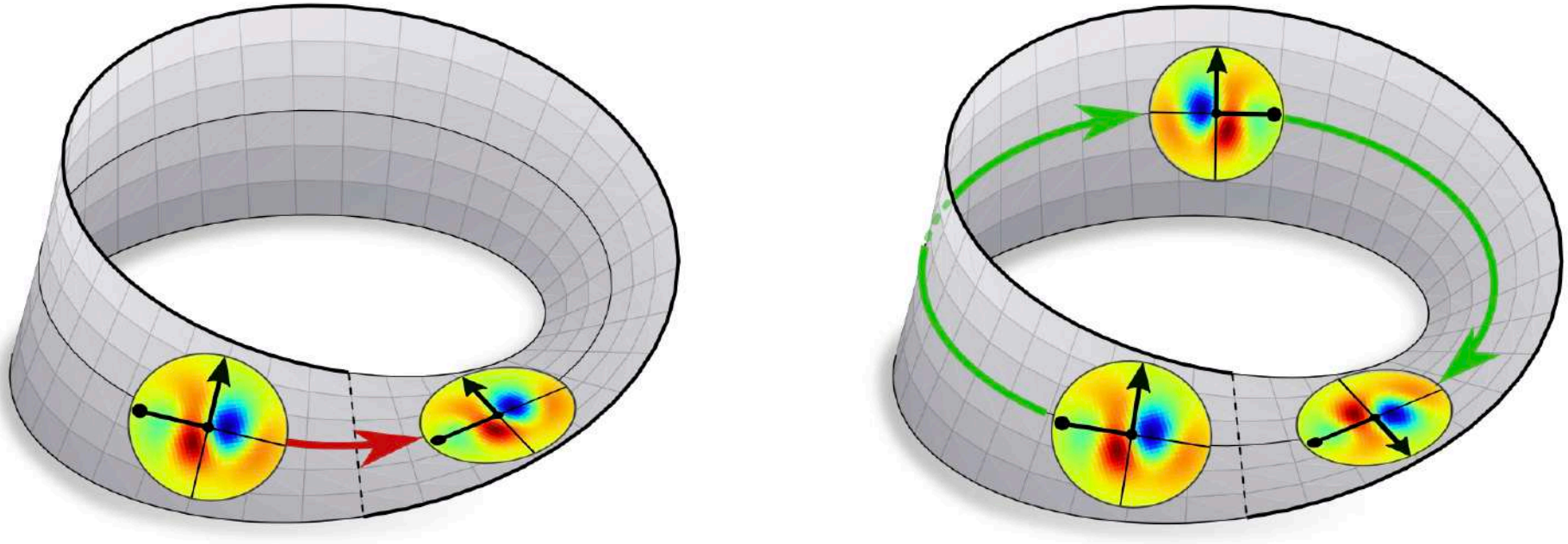
Non-Euclidean Convolution



Manifold: Result of transport is *path dependent*

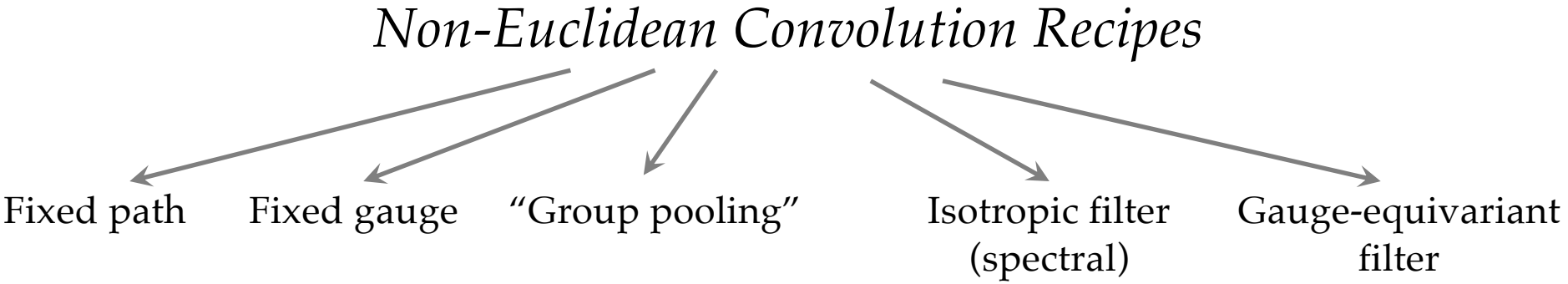
Figure: M. Weiler et al. 2021

Non-Euclidean Convolution

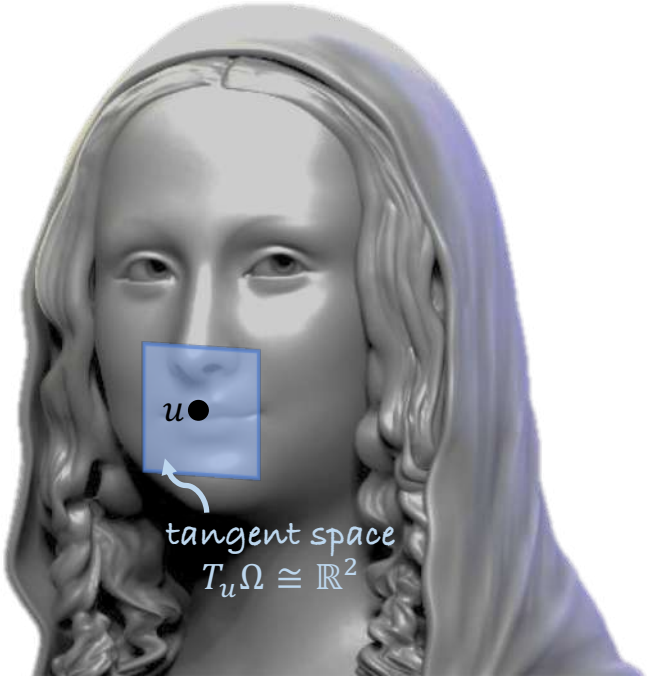


Manifold: Result of transport is *path dependent*

Figure: M. Weiler et al. 2021


Non-Euclidean Convolution Recipes
Fixed path
Fixed gauge
"Group pooling"
Isotropic filter (spectral)
Gauge-equivariant filter

Manifolds



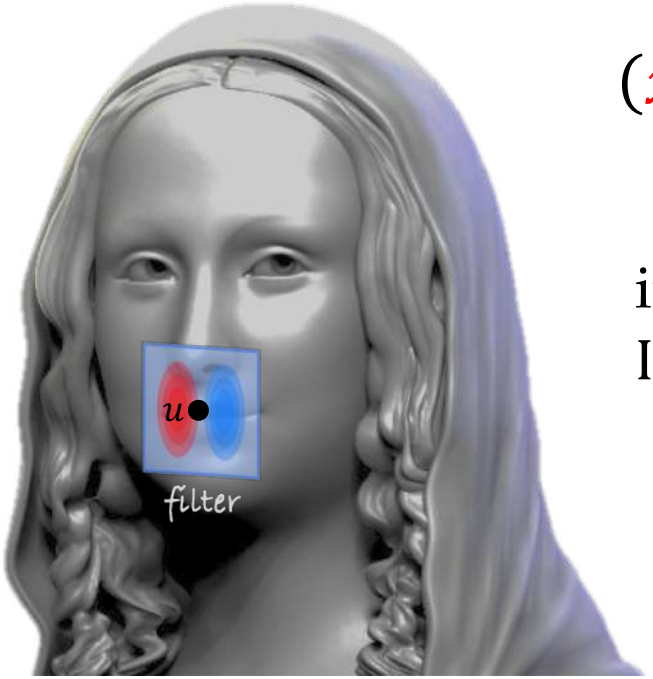
manifold Ω

manifold = locally Euclidean space

Riemannian metric = local length / direction

Isometry = metric-preserving deformation

Geodesic CNNs



manifold Ω
isometry group $\mathrm{Iso}(\Omega)$

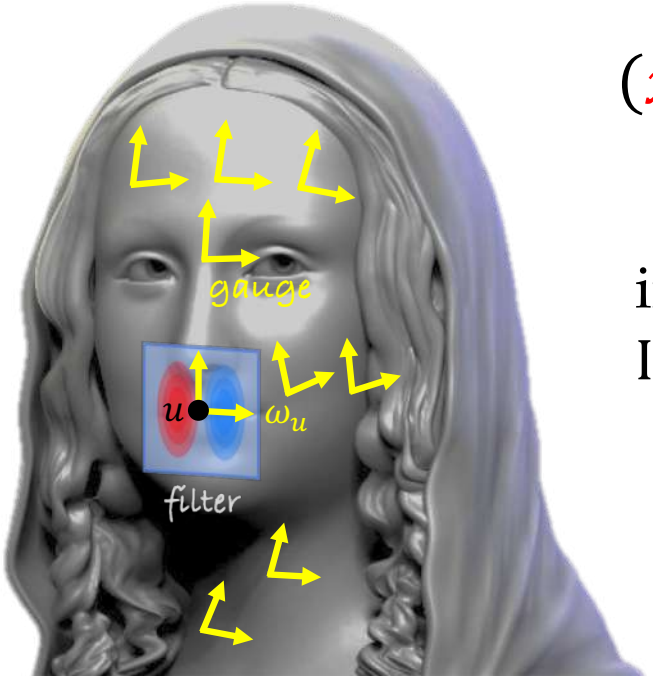
Masci et al. 2015; Boscaini et al. 2016; Monti et al. 2017

Exponential map
$\exp_u : T_u\Omega \to \Omega$

$$(x \star \psi)(u) = \int_{T_u\Omega} \psi(v)\, x(\exp_u v)\, \mathrm{d}v$$

intrinsic filter = invariant to isometry group
$\mathrm{Iso}(\Omega)$ of metric-preserving deformations

Geodesic CNNs



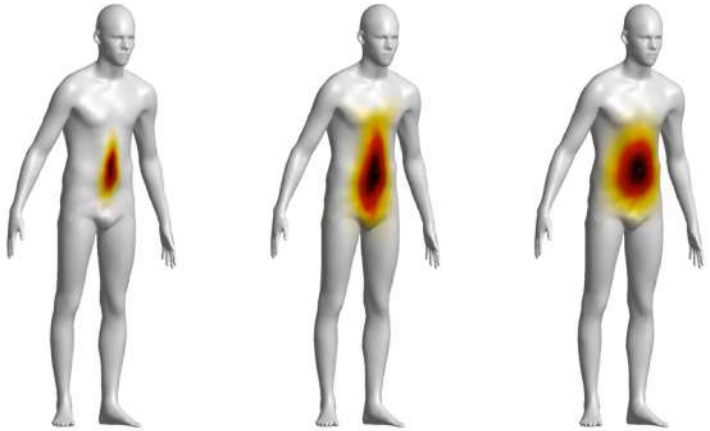
manifold Ω
isometry group $\mathrm{Iso}(\Omega)$

Masci et al. 2015; Boscaini et al. 2016; Monti et al. 2017

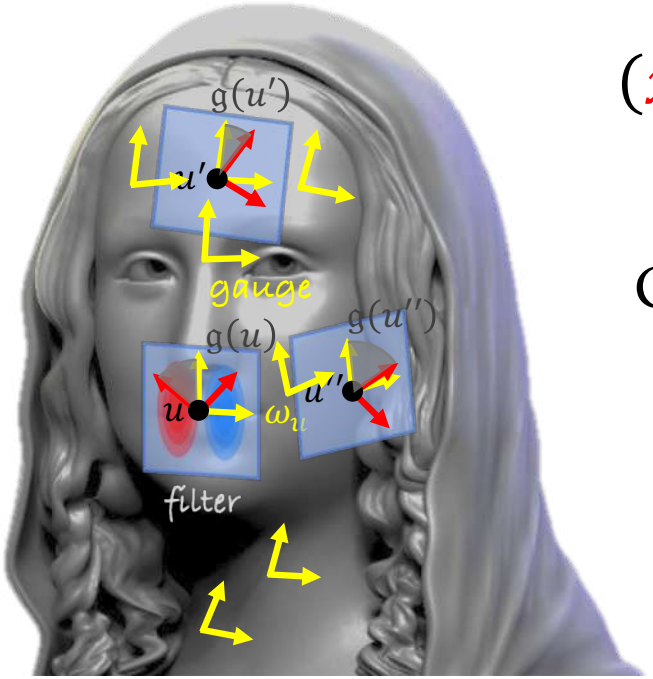
local reference frame
$\omega_u : \mathbb{R}^2 \to T_u\Omega$

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

intrinsic filter = invariant to isometry group $\mathrm{Iso}(\Omega)$ of metric-preserving deformations



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

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

Structure Group

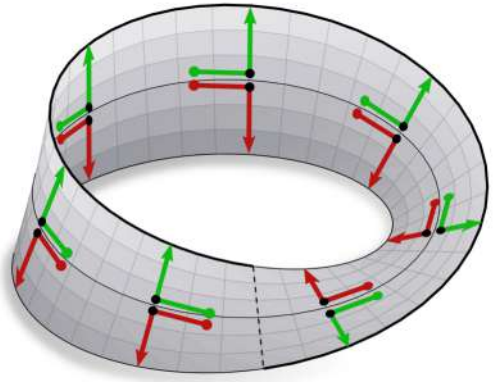
A gauge is defined up to a *gauge transformation* $\mathfrak{g}: \Omega \to \mathfrak{G}$

• "Naked" manifold	$\mathrm{GL}(s)$	invertible matrices
• Manifold+orientation	$\mathrm{GL}^+(s)$	invertible matrices with det>0
• Manifold+volume	$\mathrm{SL}(s)$	matrices with det=1
• Manifold+metric	$\mathrm{O}(s)$	orthogonal matrices
• Manifold+metric+orientation	$\mathrm{SO}(s)$	orthogonal matrices with det=1
• Manifold+frame field	{id}	identity (no ambiguity)

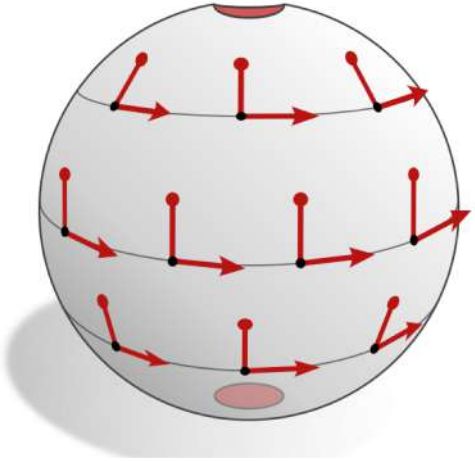
Weiler et al. 2021

Structure Group

rotation
SO(2)



reflection
R



fixed gauge
{id}

Weiler et al. 2021

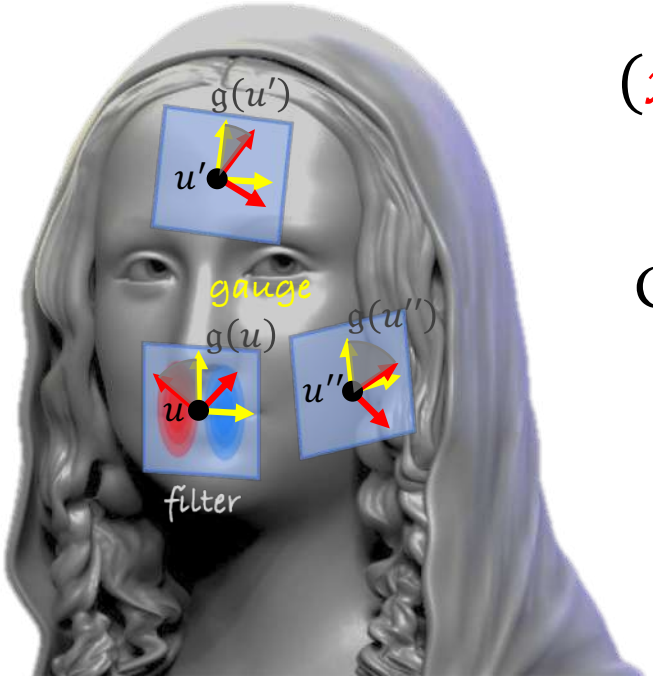
Structure Group

A gauge is defined up to a *gauge transformation* $\mathfrak{g}: \Omega \to \mathfrak{G}$

• "Naked" manifold	$\mathrm{GL}(s)$	invertible matrices
• Manifold+orientation	$\mathrm{GL}^+(s)$	invertible matrices with det>0
• Manifold+volume	$\mathrm{SL}(s)$	matrices with det=1
• Manifold+metric	$\mathrm{O}(s)$	orthogonal matrices
• Manifold+metric+orientation	$\mathrm{SO}(s)$	orthogonal matrices with det=1
• Manifold+frame field	{id}	identity (no ambiguity)

Weiler et al. 2021

Gauge-equivariant CNNs



manifold Ω
structure group $\mathfrak{G} = \mathrm{SO}(2)$

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

Gauge defined up to gauge transformation

$$\mathfrak{g}: \Omega \to \mathrm{SO}(2)$$

gauge-equivariant filter

$$\psi(\mathfrak{g}^{-1}\mathbf{v}) = \rho(\mathfrak{g}^{-1})\psi(\mathbf{v})\rho(\mathfrak{g})$$

Cohen et al. 2019

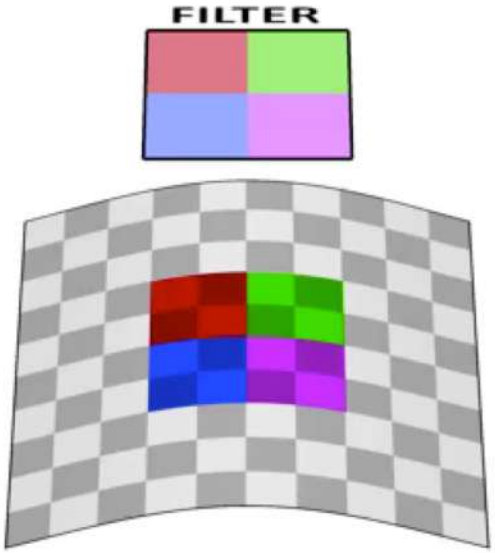
Two Types of Symmetry

Local gauge transformation

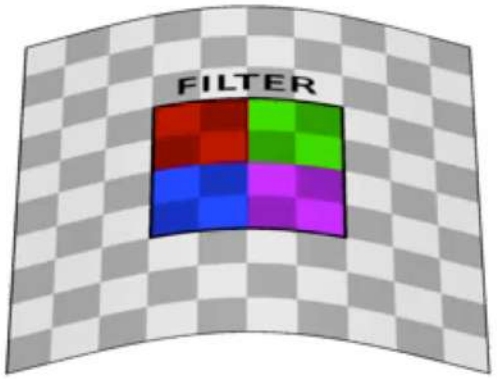
Two Types of Symmetry

Local gauge transformation

Global deformation

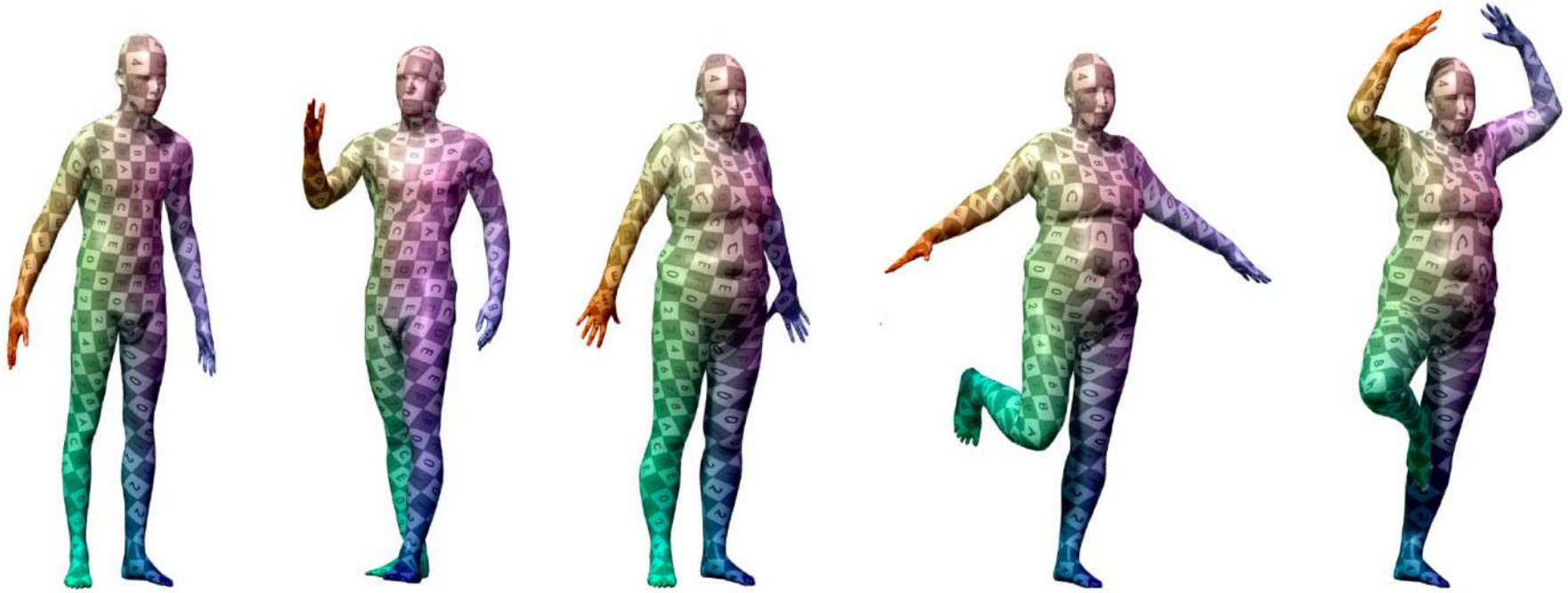


Euclidean (extrinsic) convolution



Geometric (intrinsic) convolution

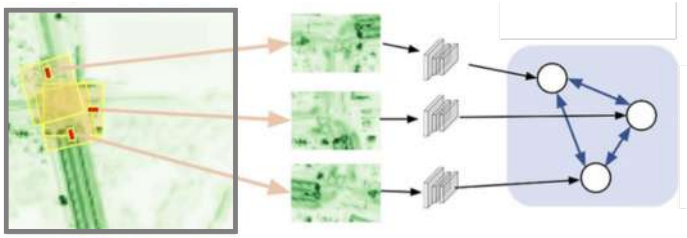
Applications in Computer Graphics



Correspondence of deformable 3D shapes

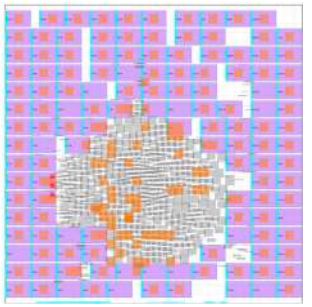
Monti et al. 2017

Wild zoo of Applications!

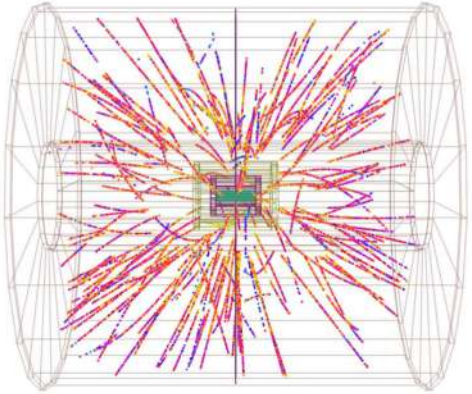


Self-driving cars

Navigation



Chip design



Particle physics

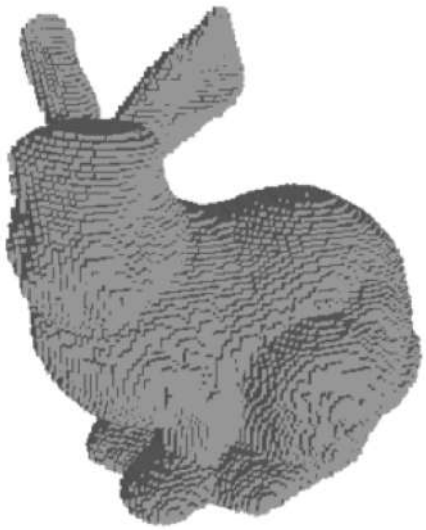
Protein science

Drug discovery

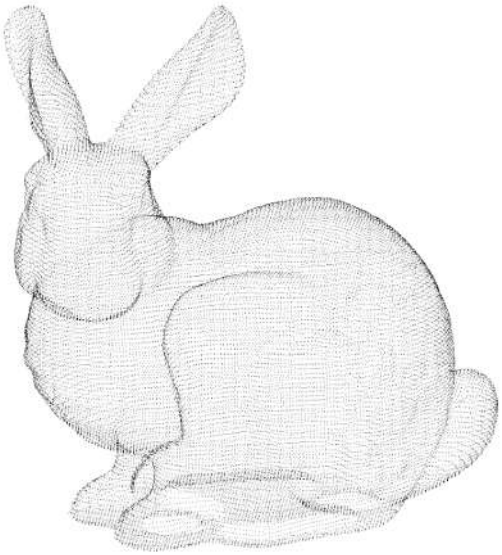
3D VISION & GRAPHICS

Geometric Data

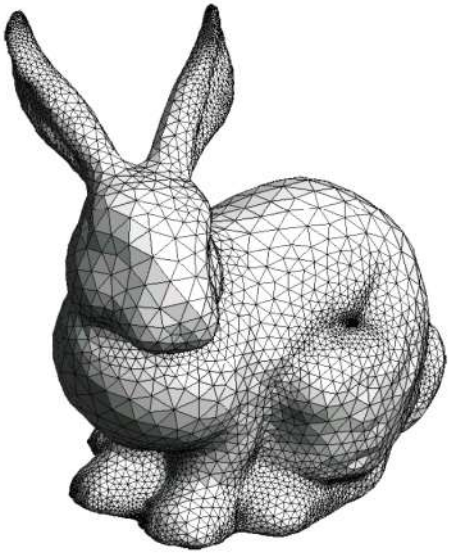
Image-based



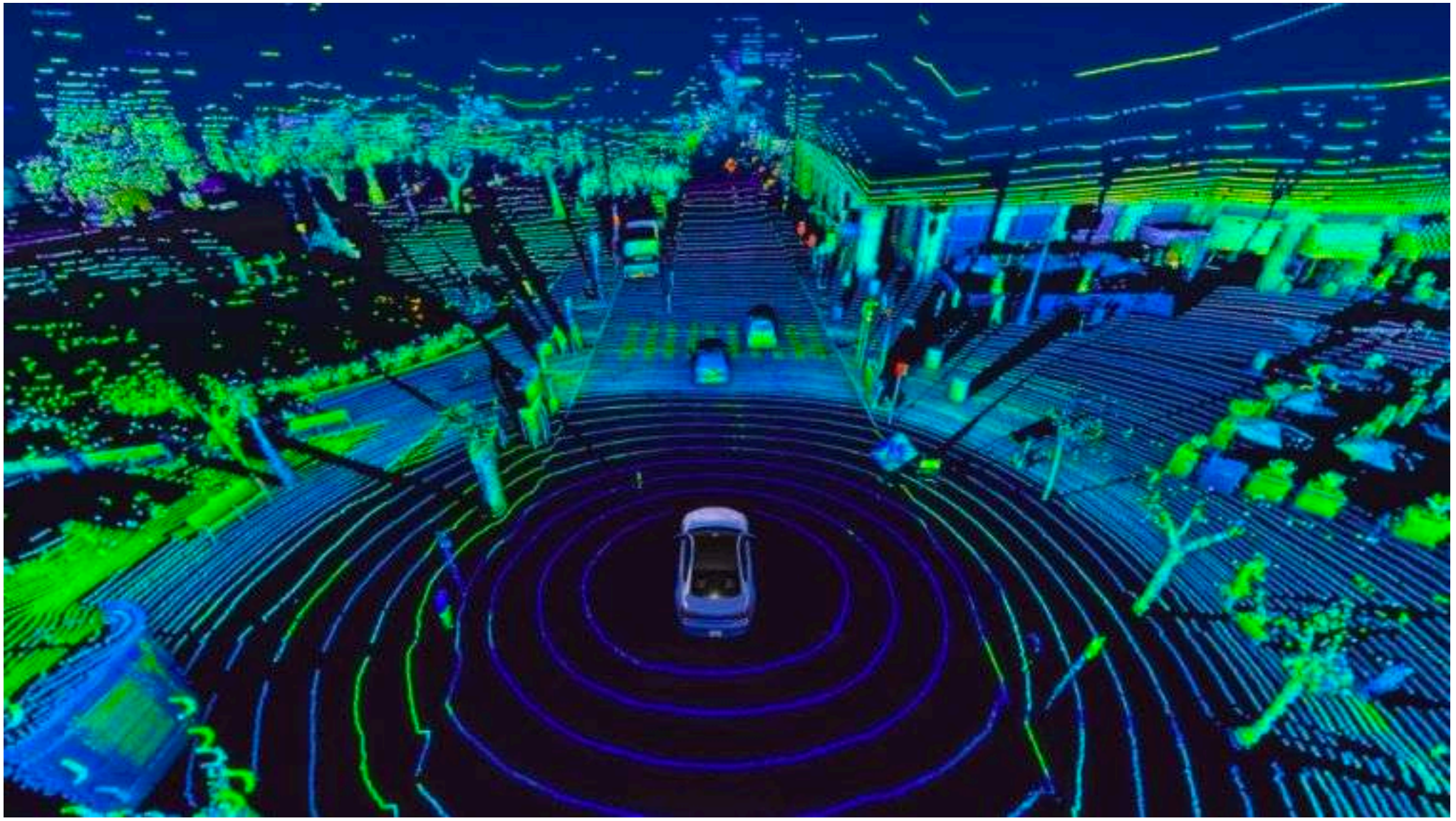
Volumetric



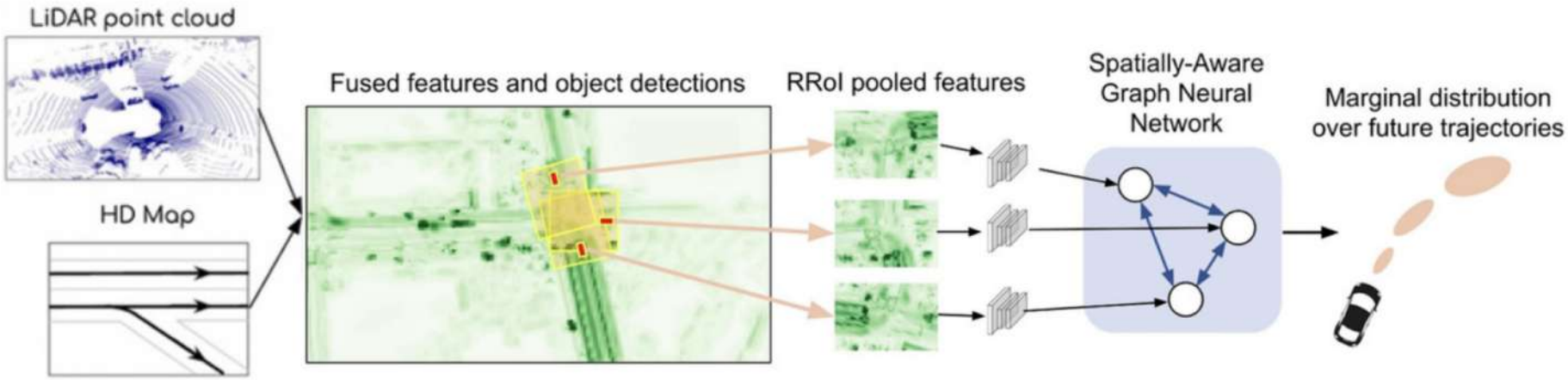
Point cloud



Mesh-based



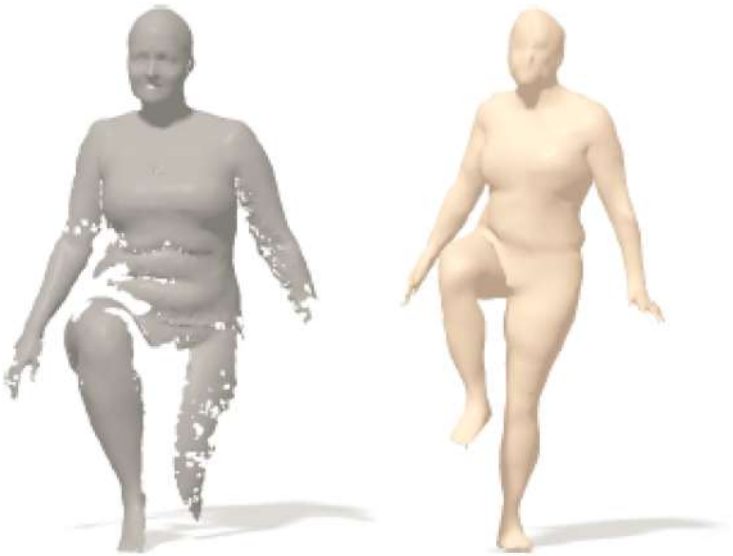
Self-Driving Cars



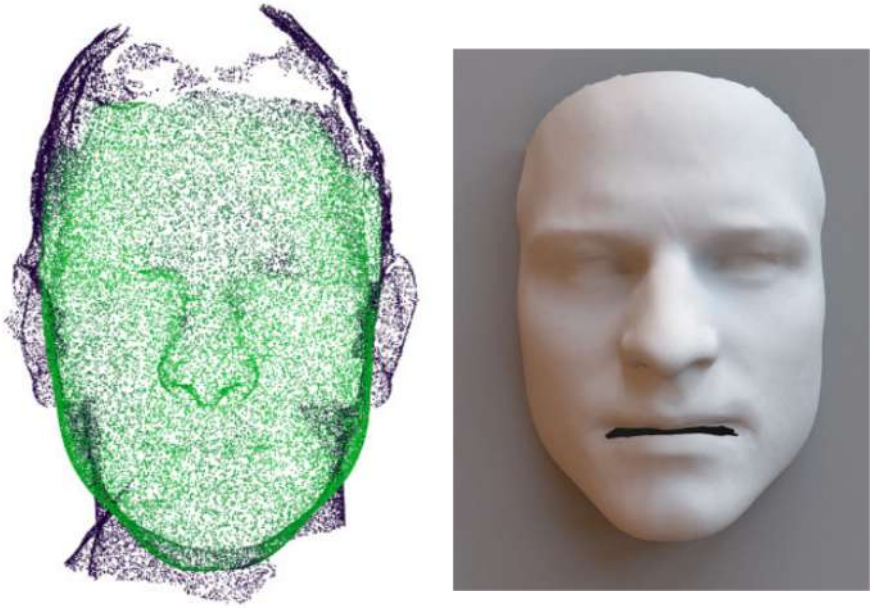
Casas et al. 2020 (from Raquel Urtasun's talk)


FaceShift 2015
GDC

3D Avatars

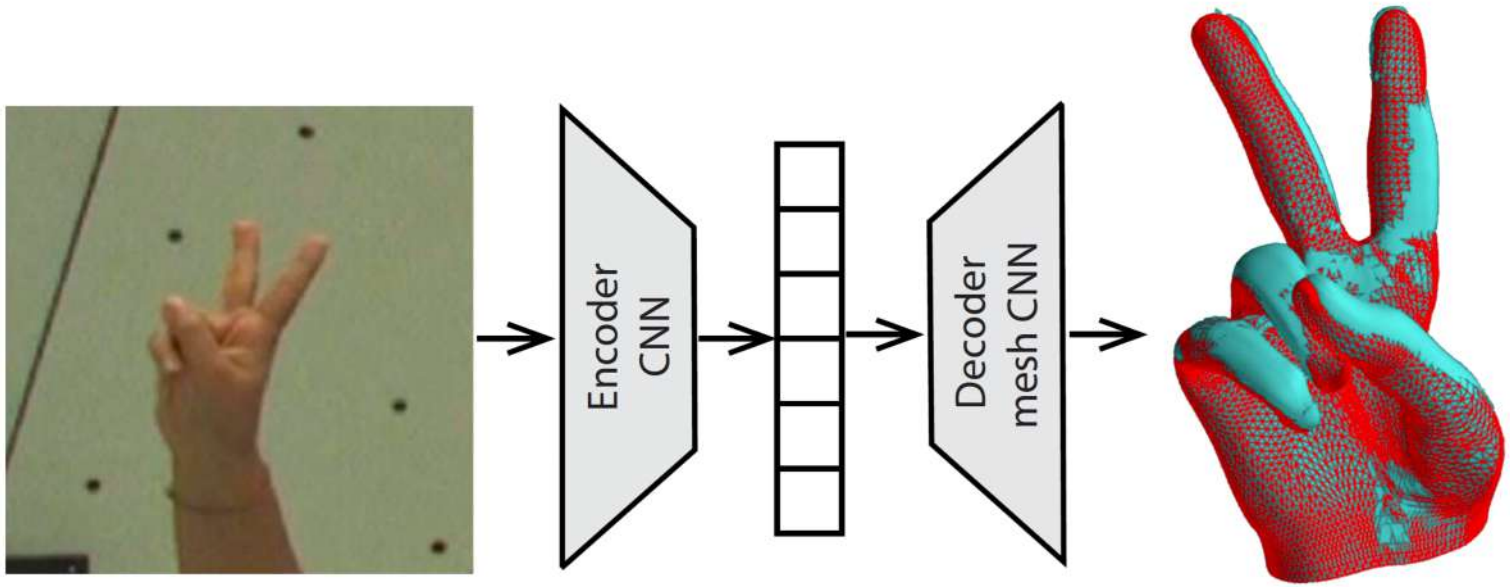


Litany et B 2018

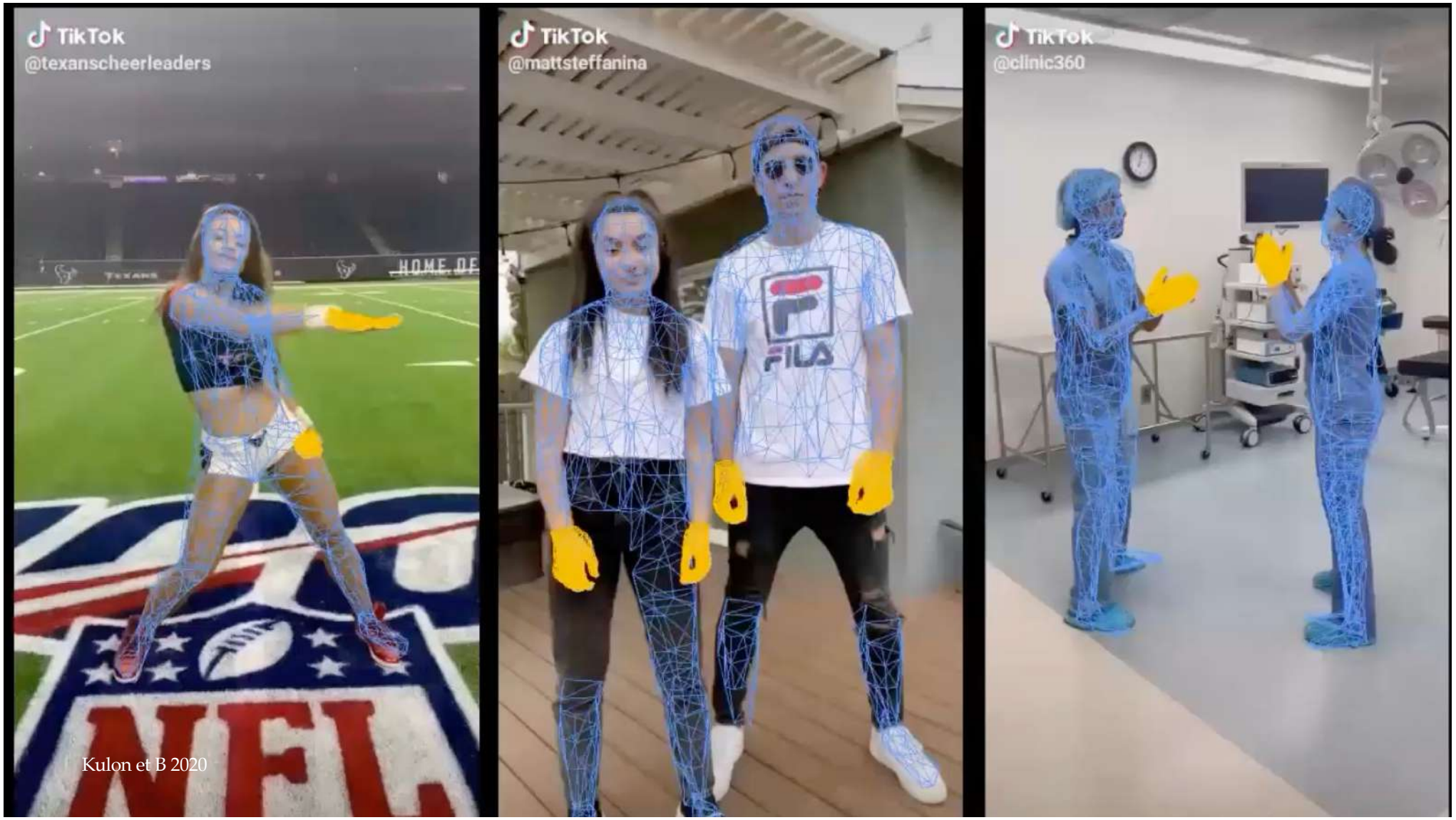


Bahri et B 2020

3D Hand Reconstruction

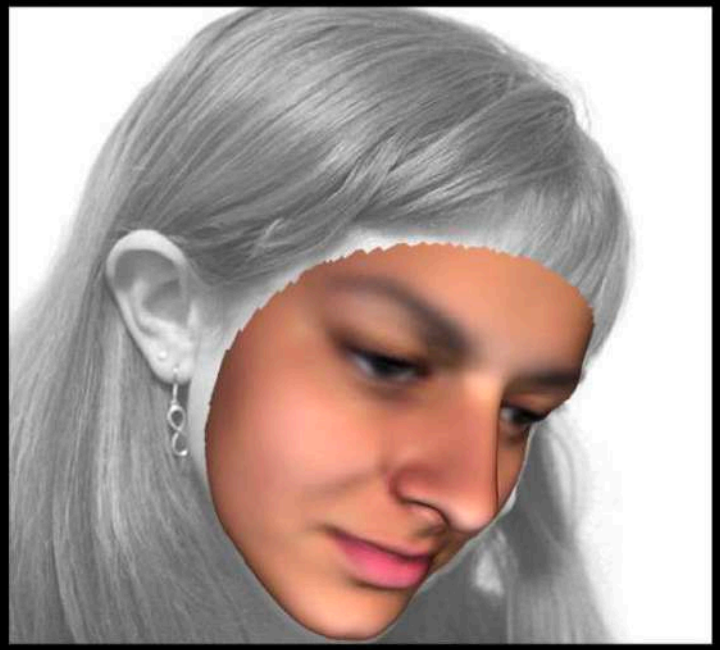


Kulon et B 2020


TikTok
@texanscheerleaders
TikTok
@mattsteffanina
FILA
TikTok
@clinic360
NFL
Kulon et B 2020

Snap Acquires Ariel AI To Enhance AR Features

Face from DNA



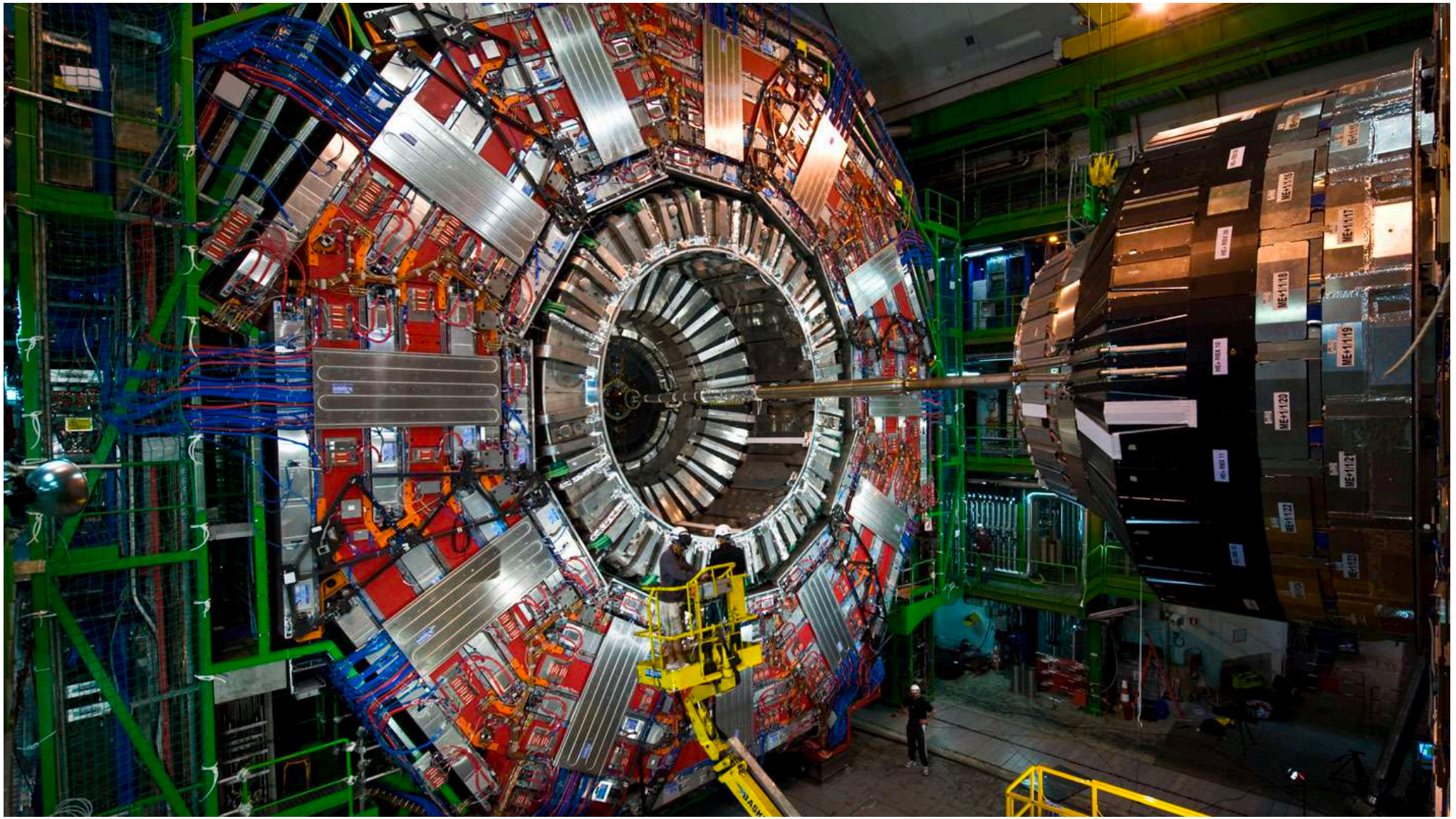
collaboration with P. Claes

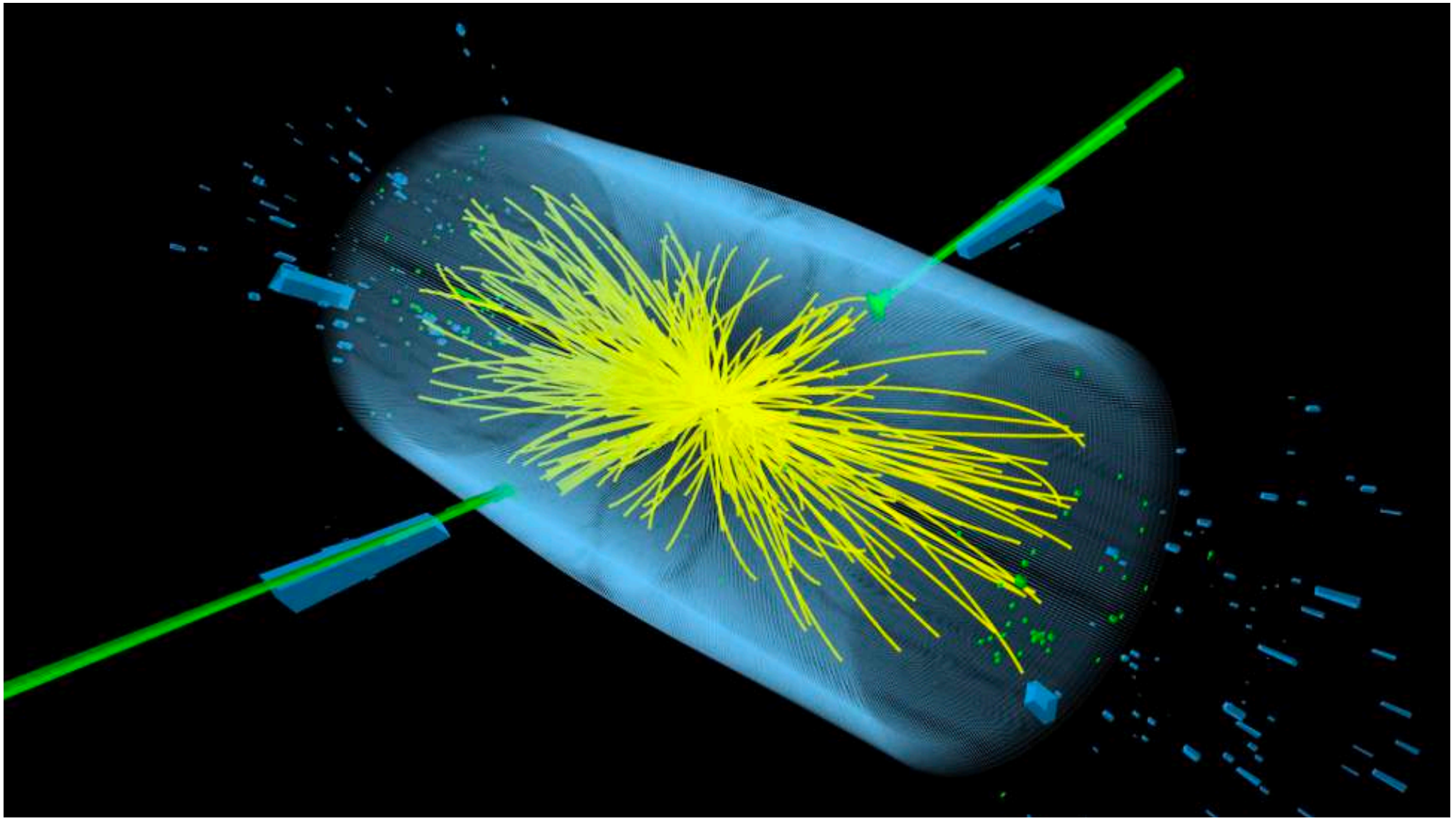
PHYSICS

“It has been amazing to see how in the last two years Graph ML has become very popular in the field of physics.”

—Kyle Cranmer (NYU)

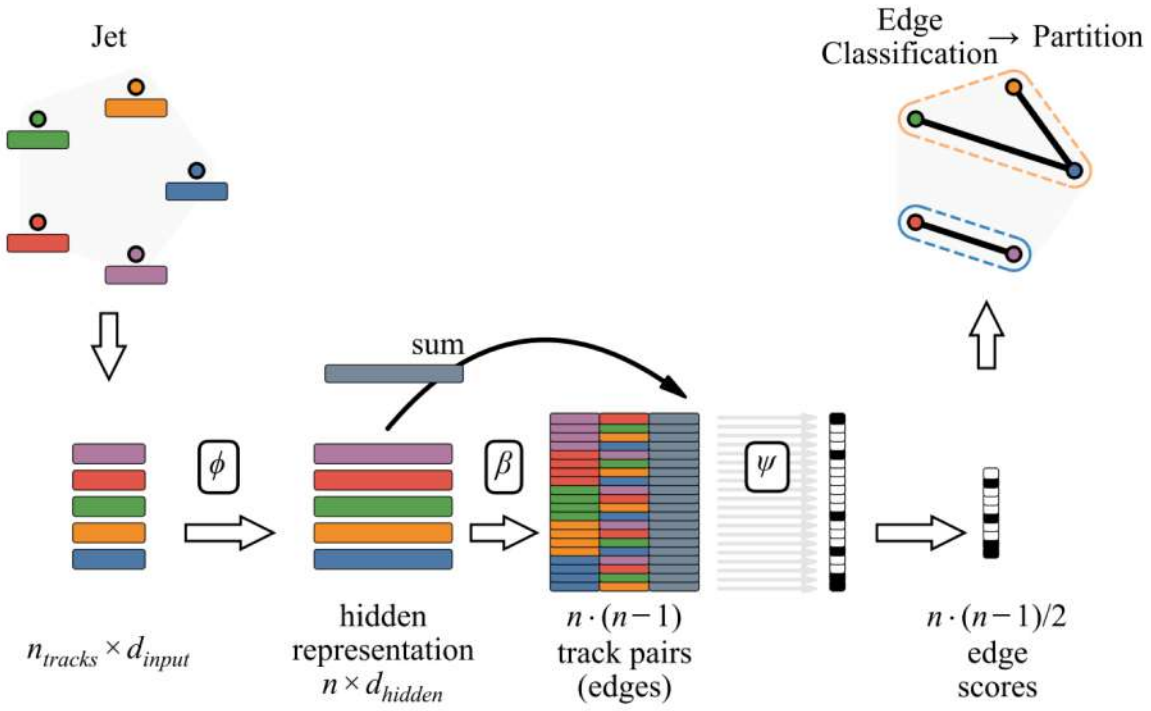
one of the discoverers of the Higgs boson


ME+1/1/15
ME+1/1/17
ME+1/1/18
ME+1/1/19
ME+1/1/20
ME+1/1/21



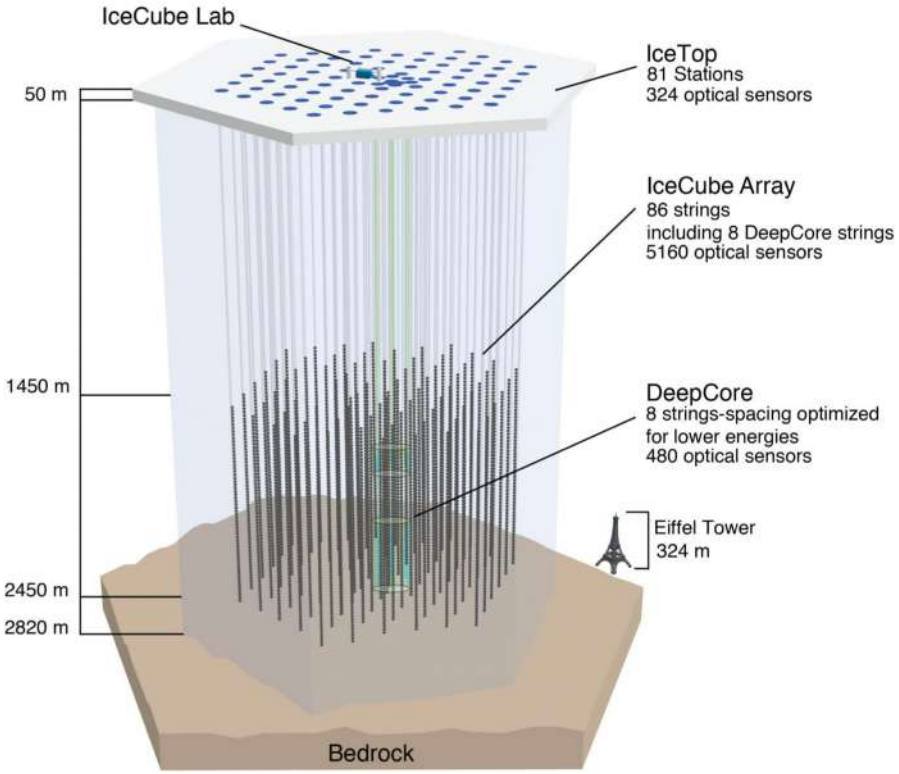
Particle Jet Reconstruction & Segmentation

Set2graph architecture for partitioning observed particles based on their point of origin

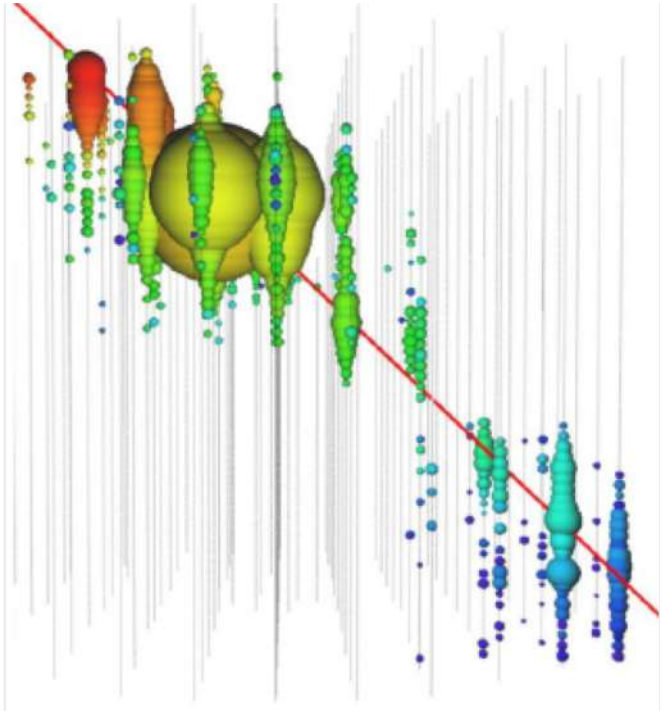


Serviansky et al. 2020

Detector Modeling



Choma et B B 2018



Light deposition for a high-energy single muon in IceCube detector

SOCIAL NETWORKS

Recommender Systems

PinSage: A new graph convolutional neural network for web-scale recommender systems

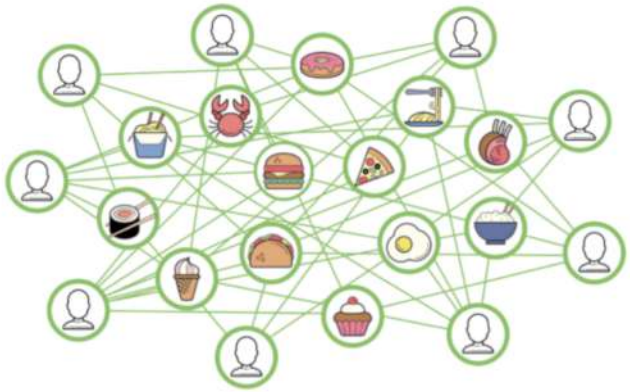
Pinterest Engineering Follow
Aug 15, 2018 · 8 min read

Ruining He | Pinterest engineer, Pinterest Labs

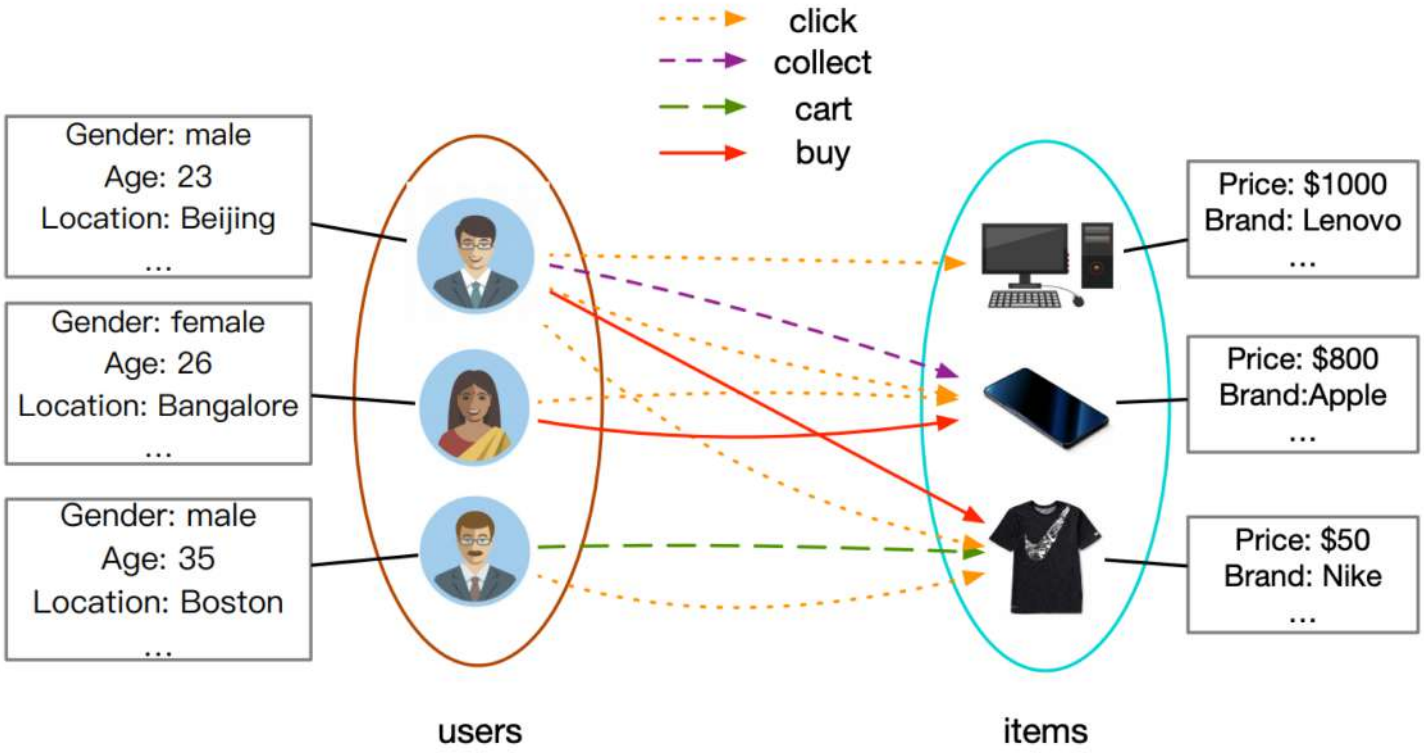
Food Discovery with Uber Eats: Using Graph Learning to Power Recommendations

Ankit Jain, Isaac Liu, Ankur Sarda, and Piero Molino
0

December 4, 2019

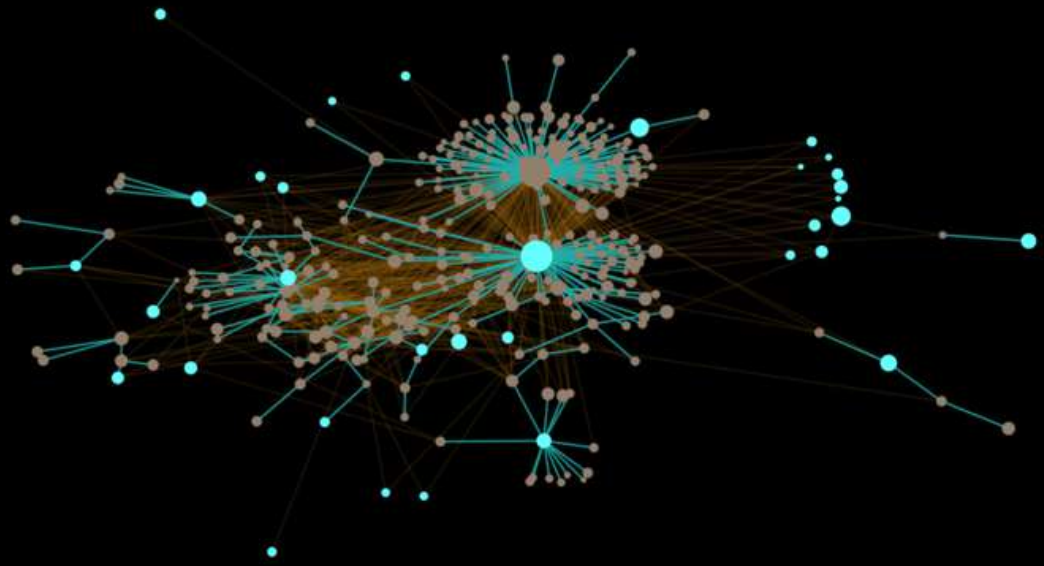


Recommender Systems



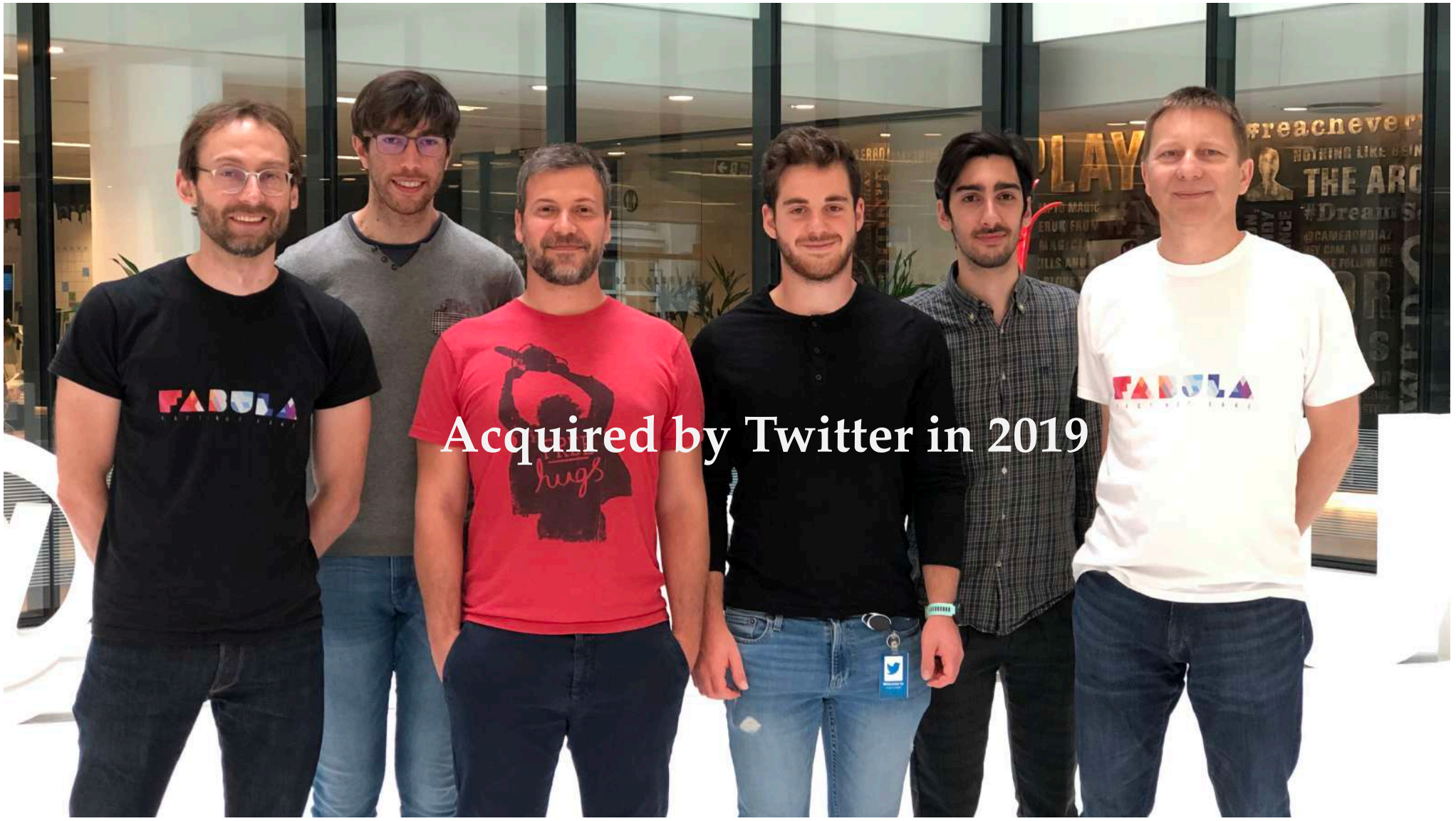
AliGraph (Zhu et al. 2019)

Vosoghi et al. 2018



FABULA

Monti et B 2019

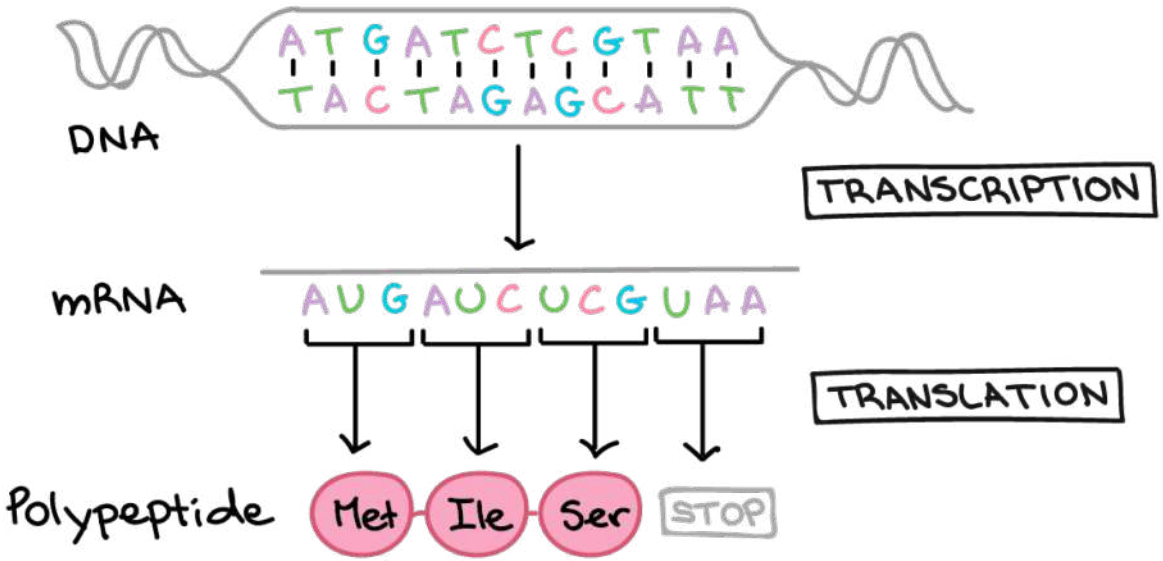

Acquired by Twitter in 2019

STRUCTURAL BIOLOGY

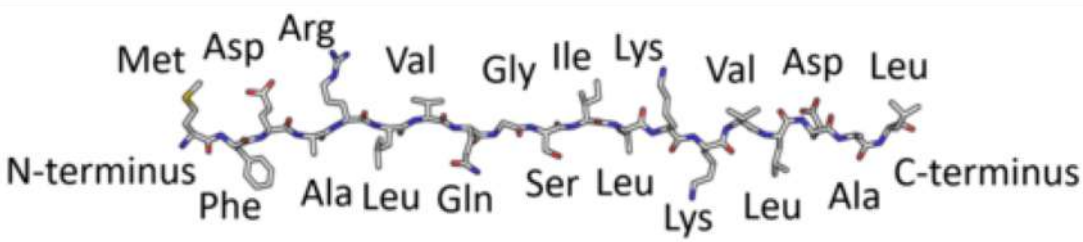
“In 2020, exciting progress has been made in protein structure prediction, a key problem in bioinformatics. Yet, ultimately the chemical and geometric pattern displayed at the surface of these molecules are critical for protein function.”

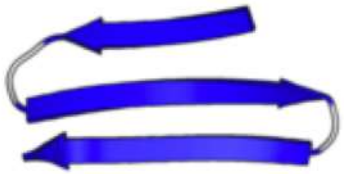
—Bruno Correia (EPFL Lab of Protein Design)

"Central Dogma"



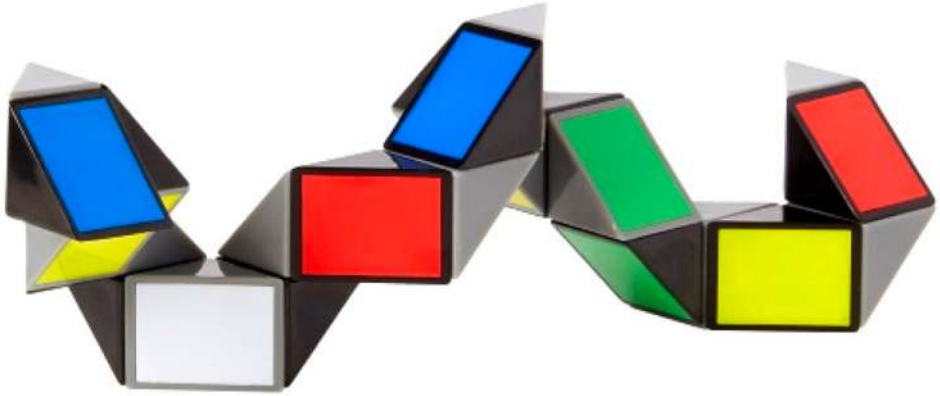
TTT Phe	TCT Ser	TAT Tyr	TGT Cys
TTC Phe	TCC Ser	TAC Tyr	TGC Cys
TTA Leu	TCA Ser	TAA stop	TGA stop
TTG Leu	TCG Ser	TAG stop	TGG Trp
CTT Leu	CCT Pro	CAT His	CGT Arg
CTC Leu	CCC Pro	CAC His	CGC Arg
CTA Leu	CCA Pro	CAA Gln	CGA Arg
CTG Leu	CCG Pro	CAG Gln	CGG Arg
ATT Ile	ACT Thr	AAT Asn	AGT Ser
ATC Ile	ACC Thr	AAC Asn	AGC Ser
ATA Ile	ACA Thr	AAA Lys	AGA Arg
ATG Met	ACG Thr	AAG Lys	AGG Arg
GTT Val	GCT Ala	GAT Asp	GGT Gly
GTC Val	GCC Ala	GAC Asp	GGC Gly
GTA Val	GCA Ala	GAA Glu	GGA Gly
GTG Val	GCG Ala	GAG Glu	GGG Gly

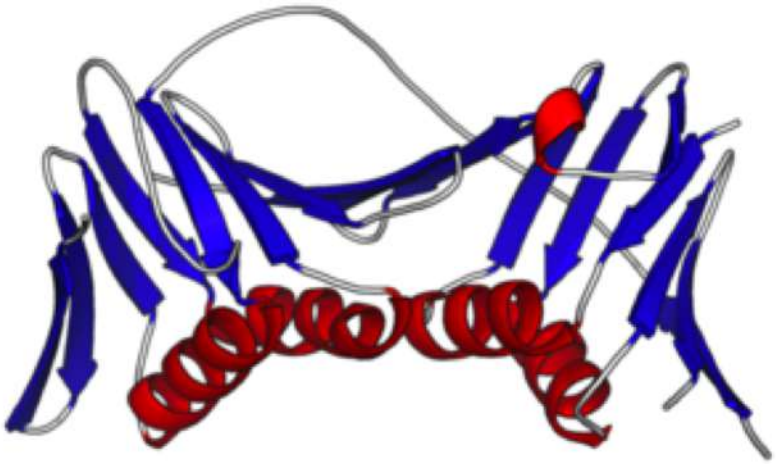




β-Sheet (3 strands)

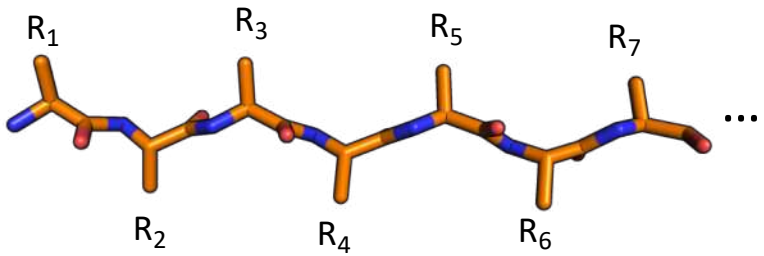
α-helix



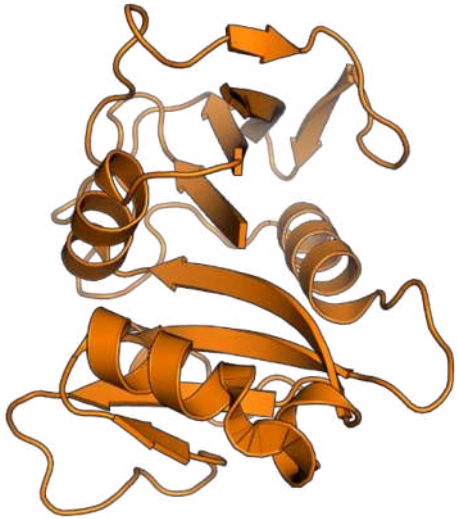


Protein Folding

Protein folding



Primary protein structure

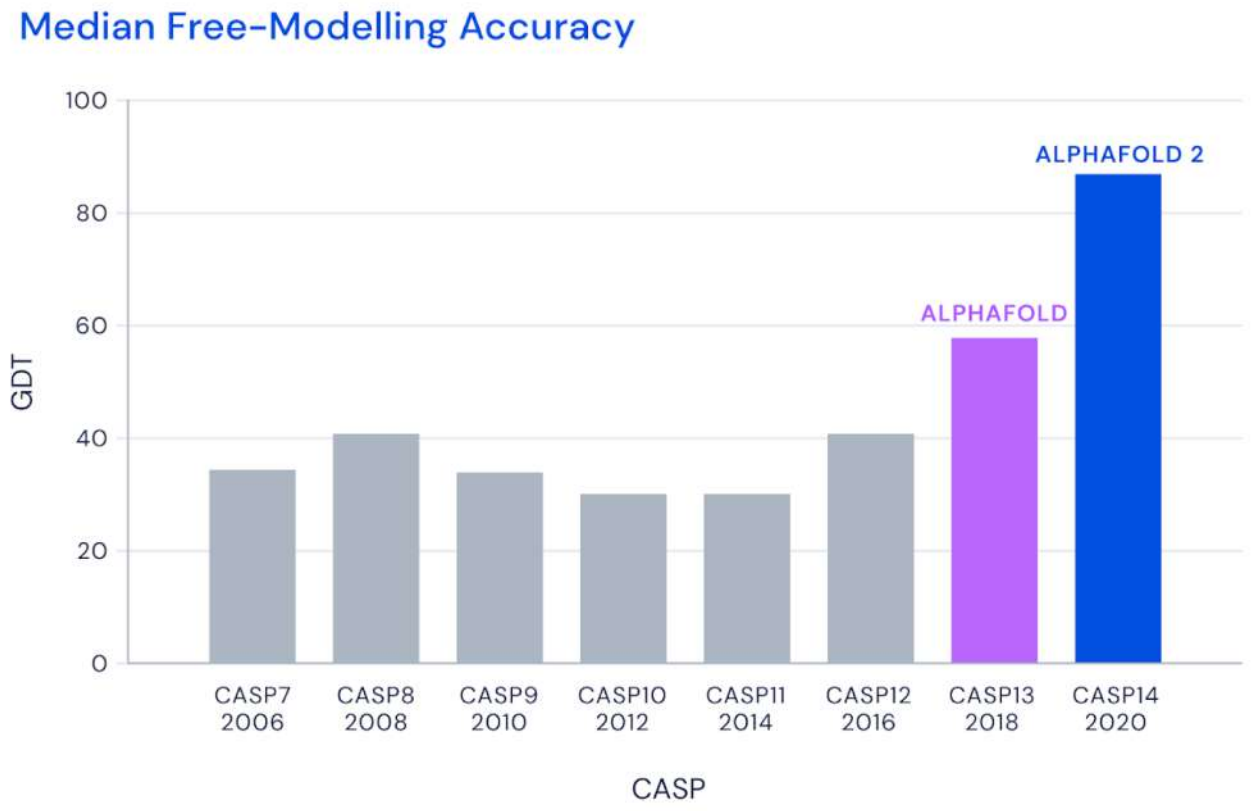


Tertiary protein structure

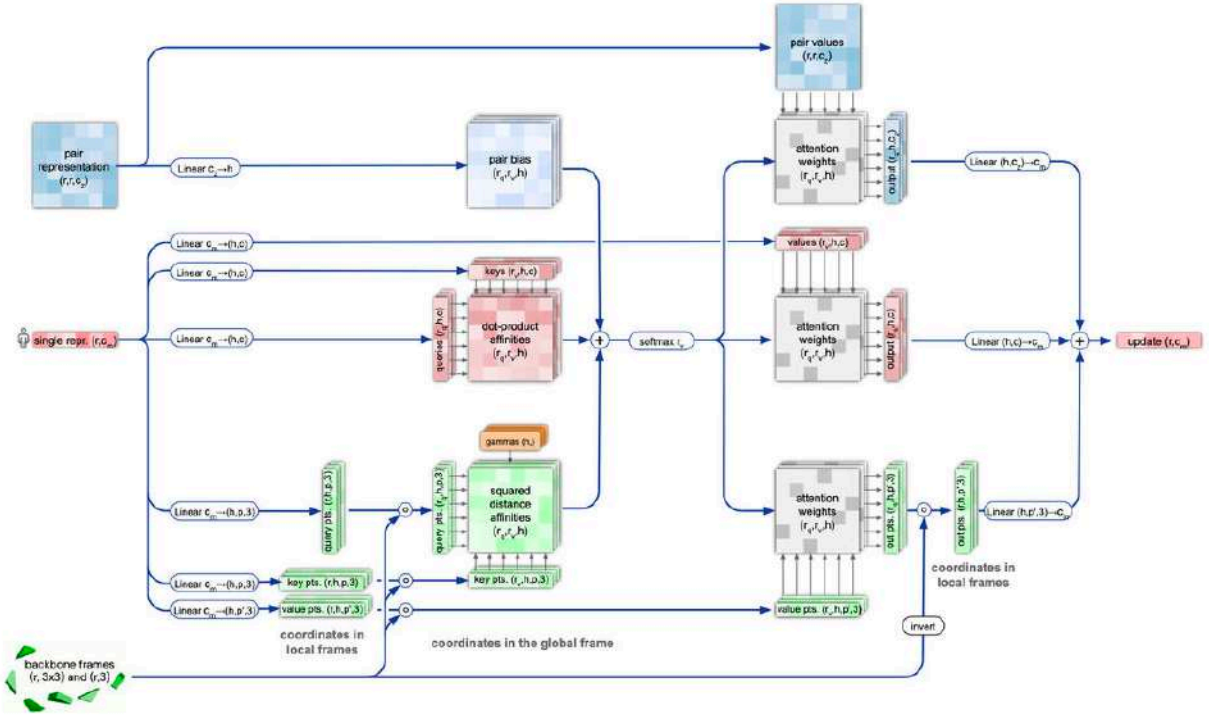
“[protein folding] is determined by […] the aminoacid sequence in a given environment”

—Christian Anfinsen (1972 Nobel Laureate in Chemistry)

"ImageNet Moment" of Structural Biology



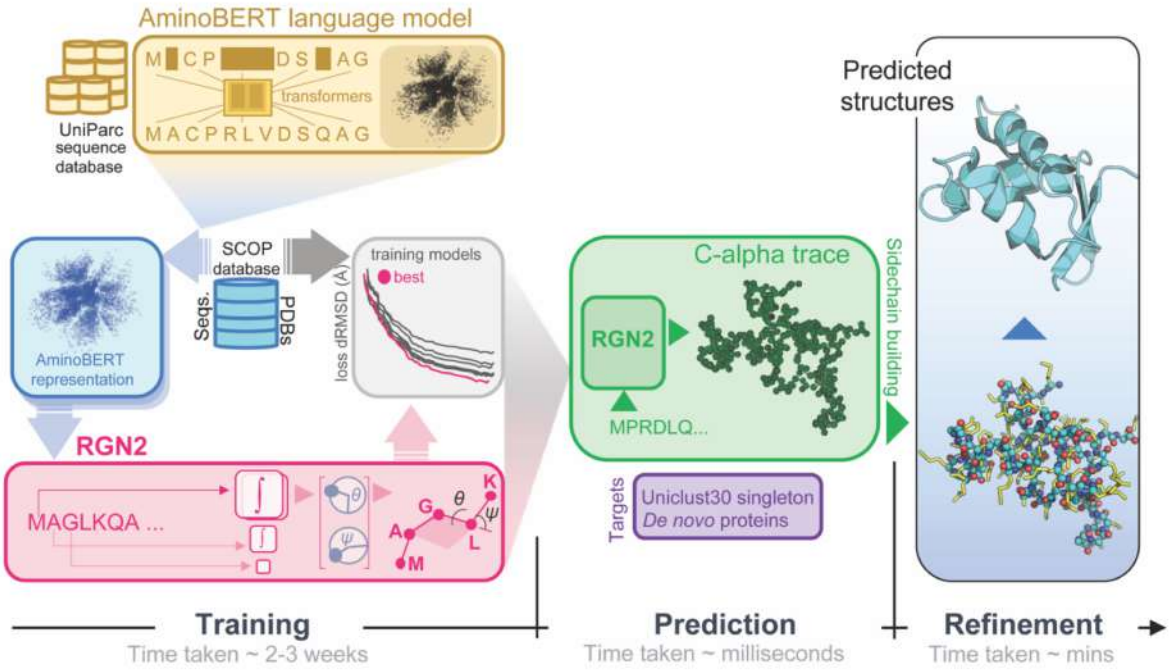
AlphaFold 2



Invariant Point Attention

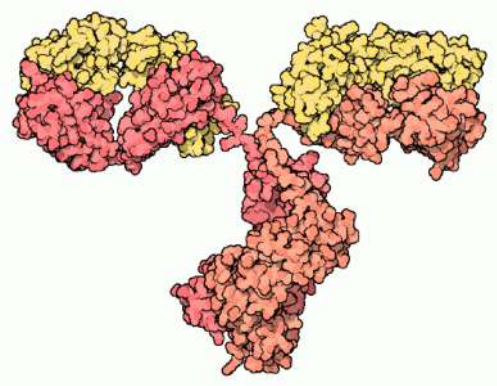
Jumper et al. 2021

Recurrent Geometric Network

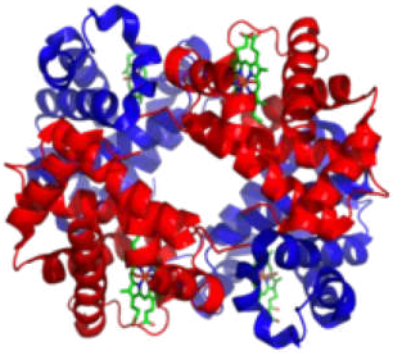


Single sequence protein structure prediction

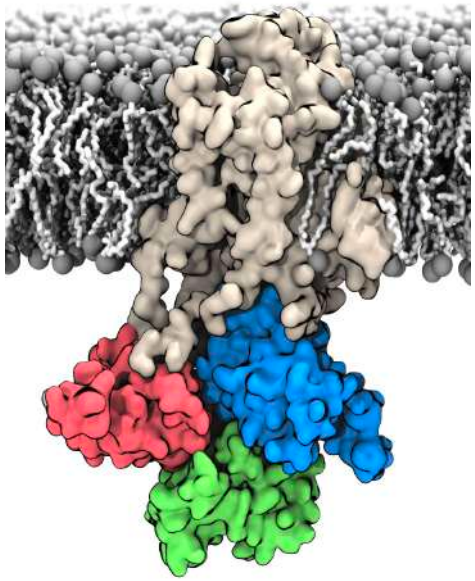
Chowdhury et al. 2021



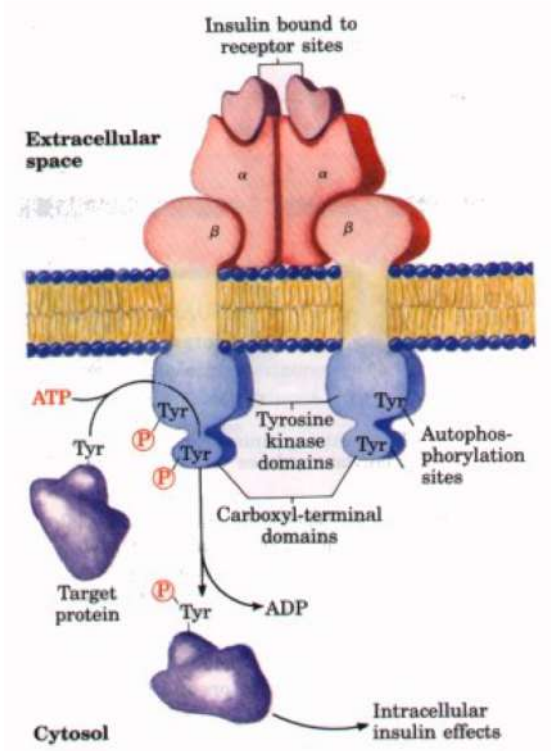
Defense (antibody)



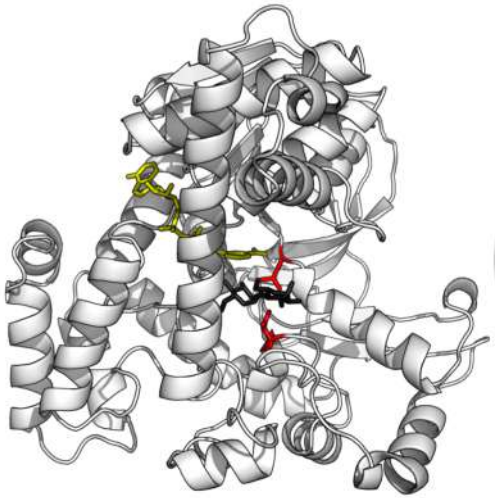
Storage (haemoglobin)

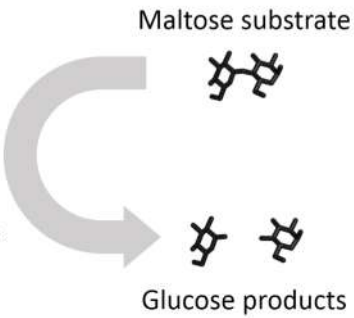


Transport (calcium pump)



Communication (insulin)

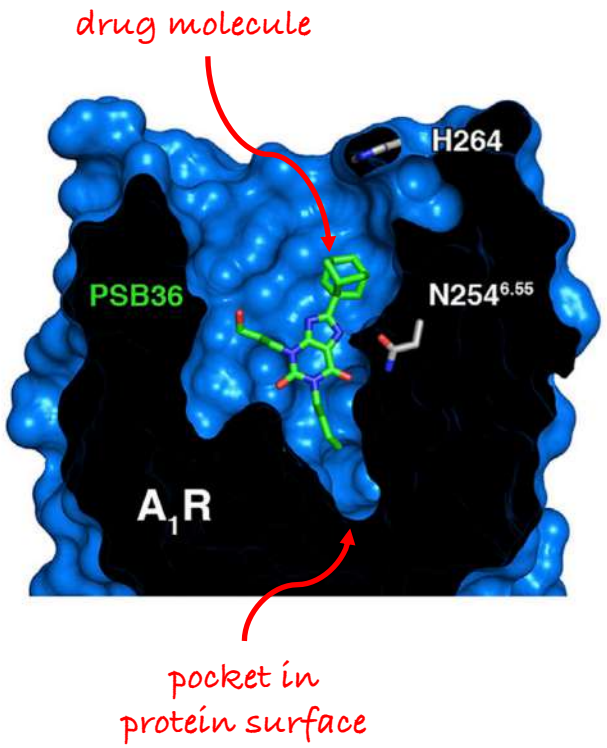




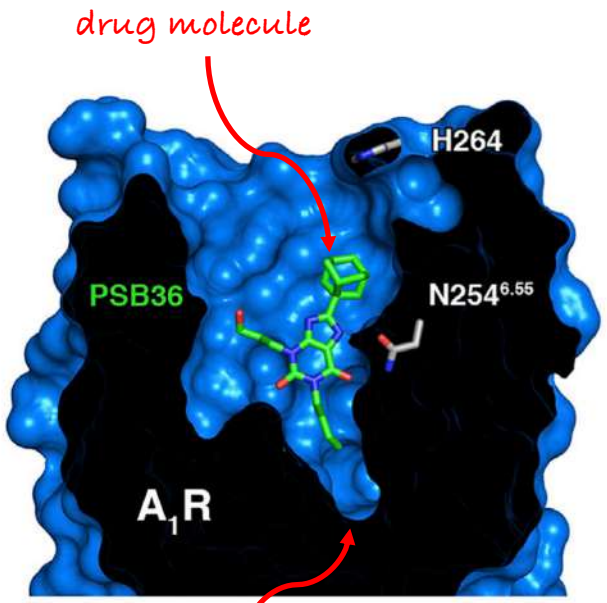
Catalysis (enzyme)

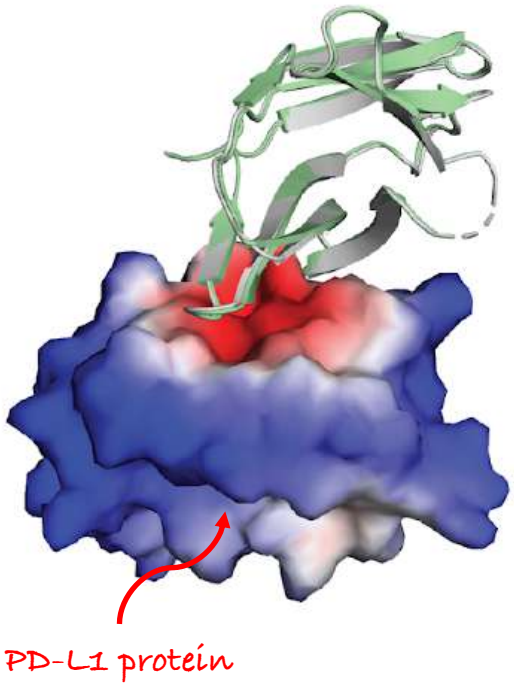
Structure (collagen)

Small Molecule Drugs

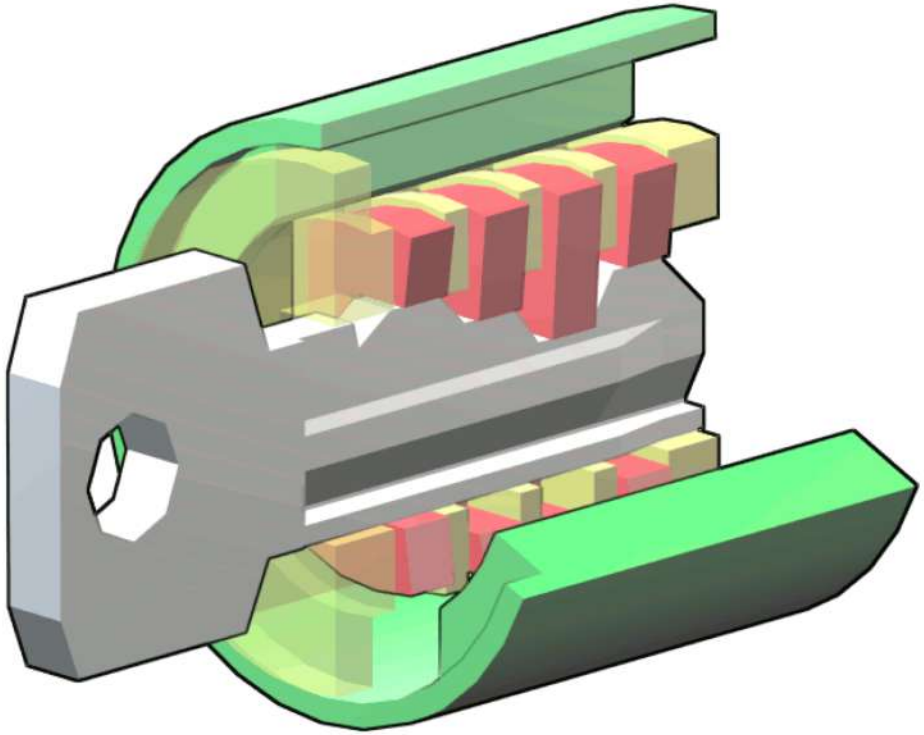


Protein-Protein Interactions



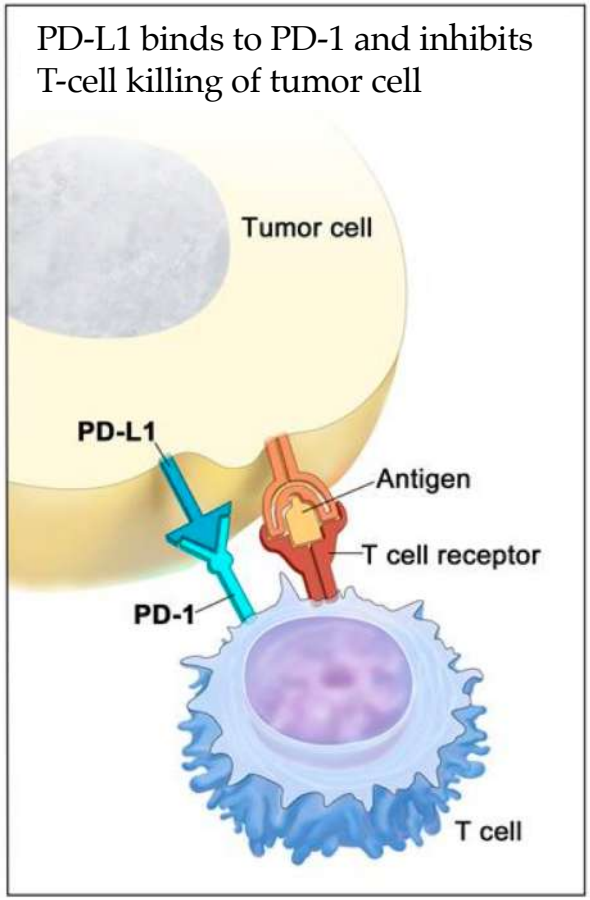


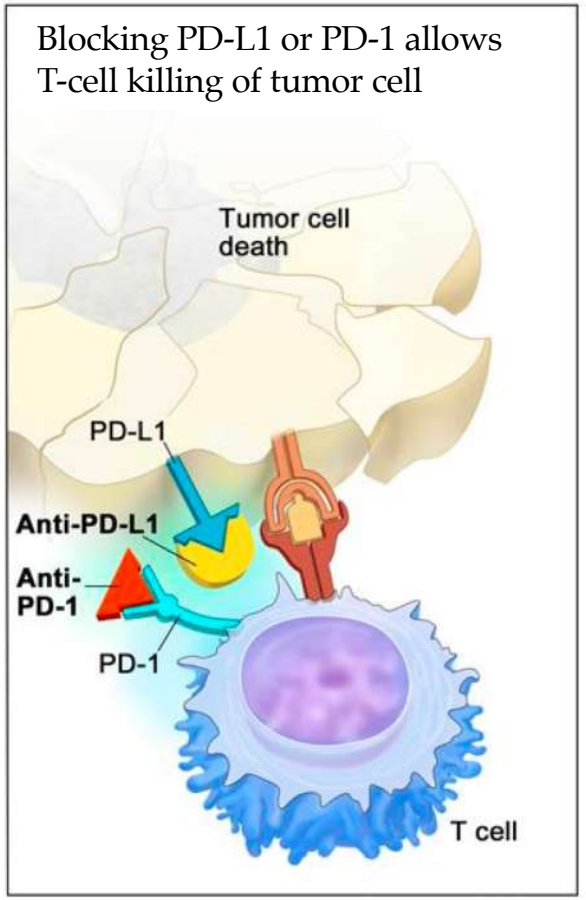
Lock-Key Metaphor



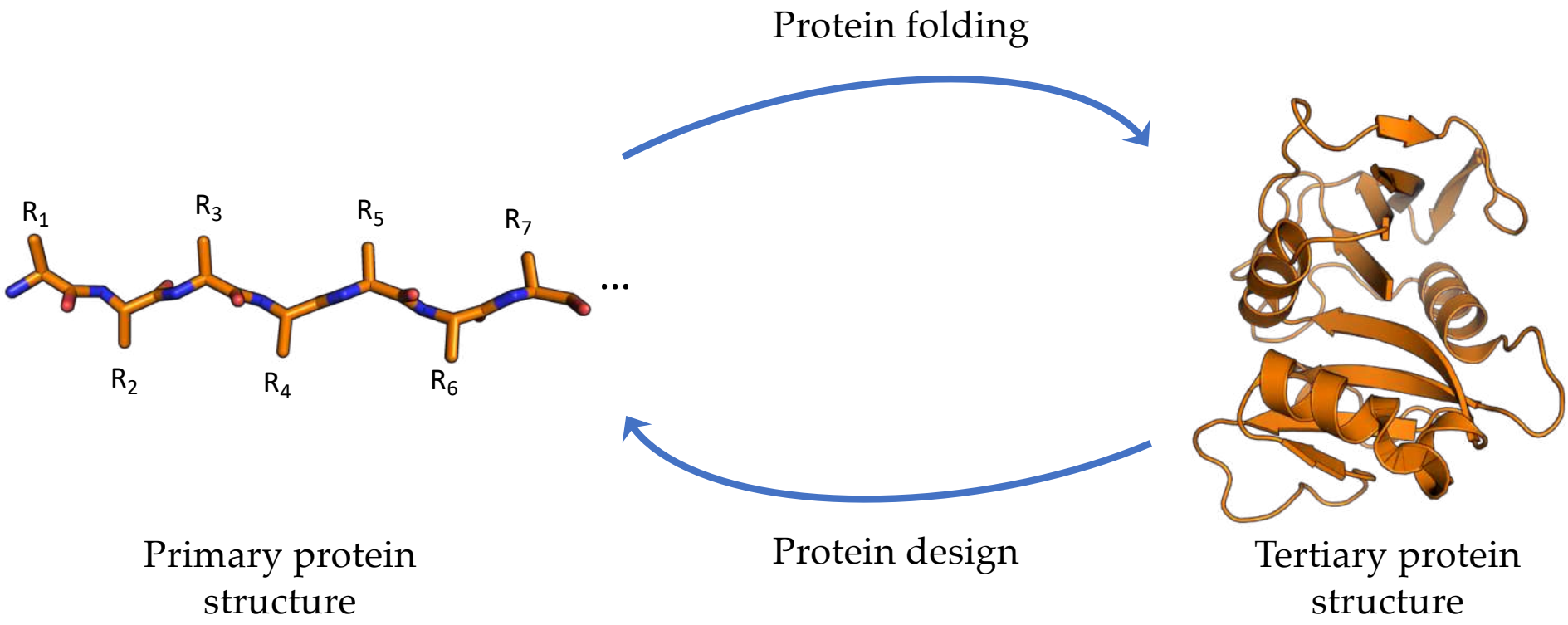
Emil Fischer “Schlüssel-Schloss-Prinzip” 1894

2018 Nobel Prize
PD-proteins role in
immunotherapy



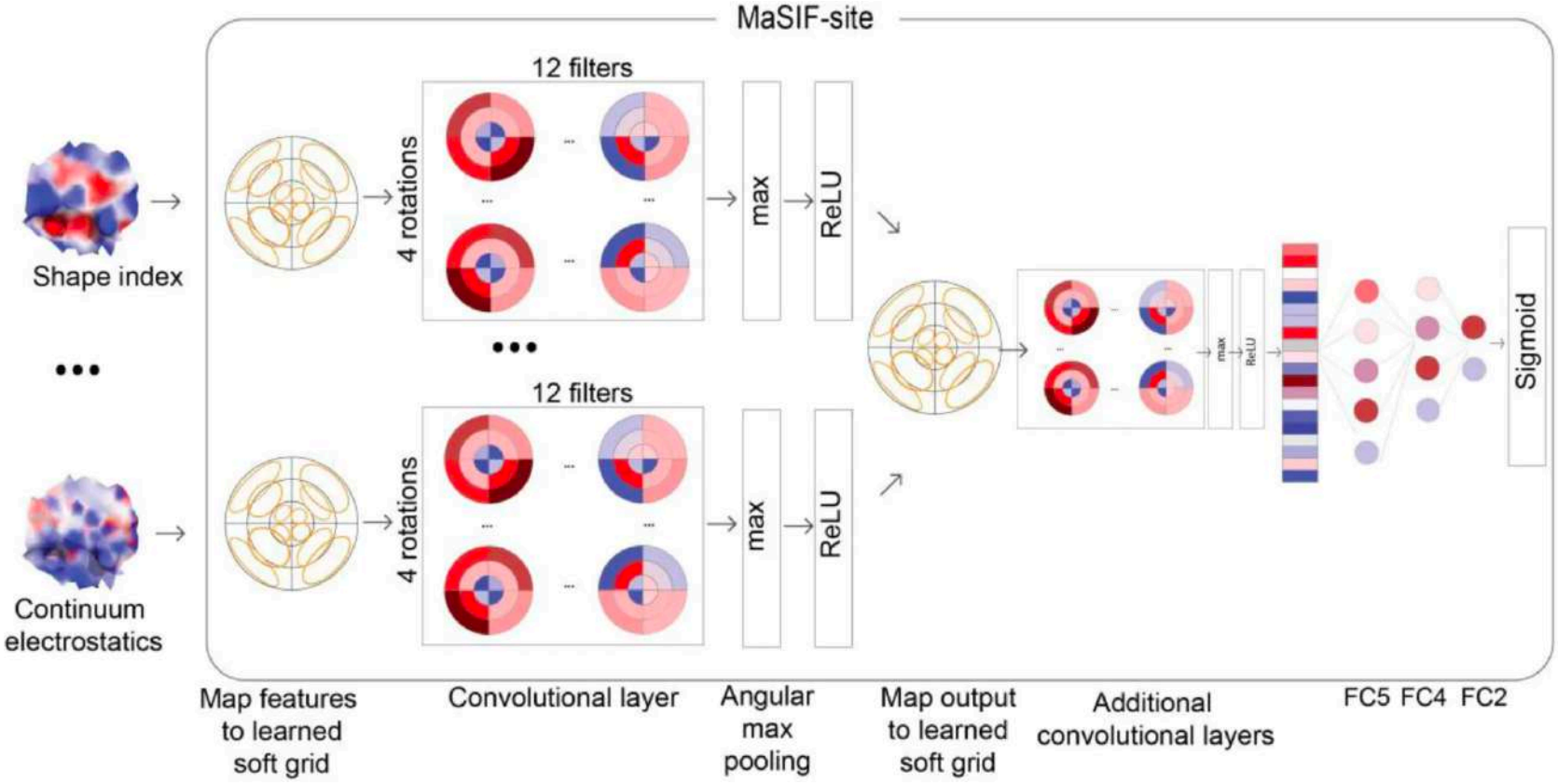


Protein Design = "Inverse Folding"



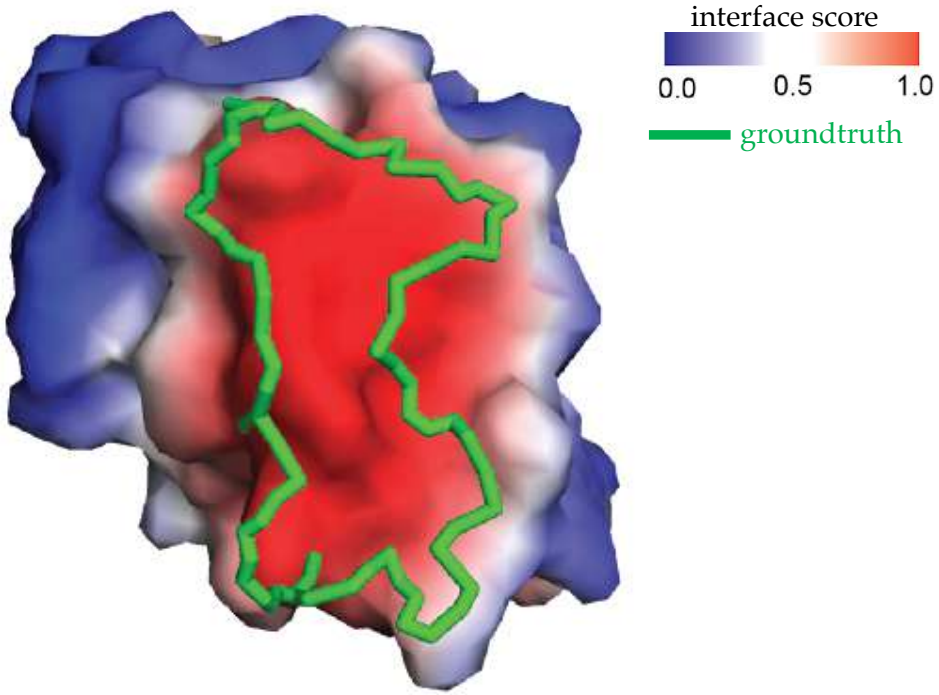

Sequence
Structure
Function

MaSIF: Protein Function Prediction



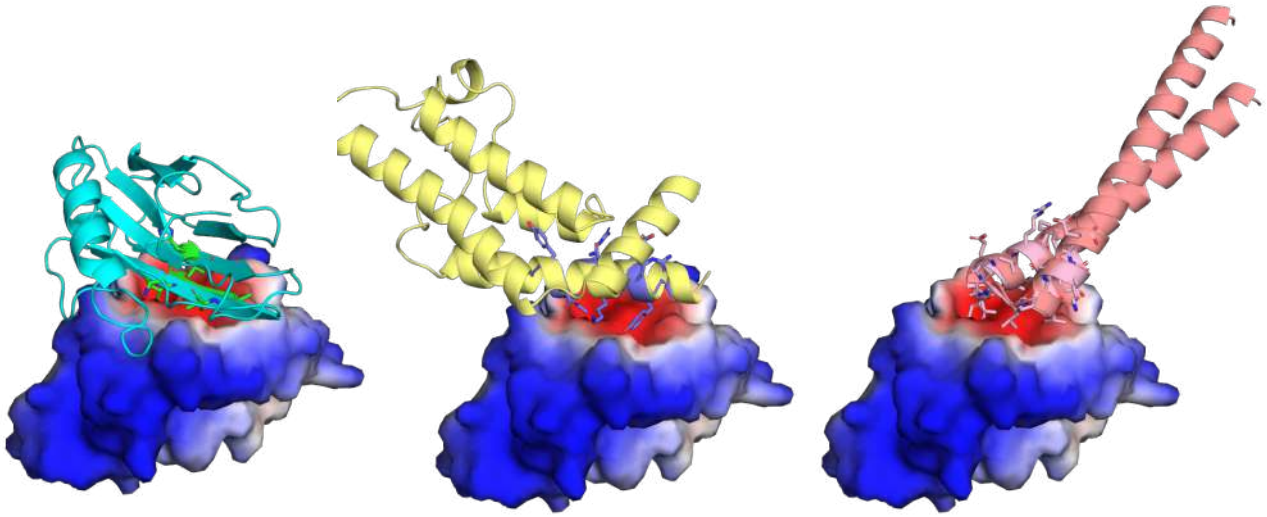
Gainza, Sverrisson et B 2020

MaSIF: Protein Function Prediction

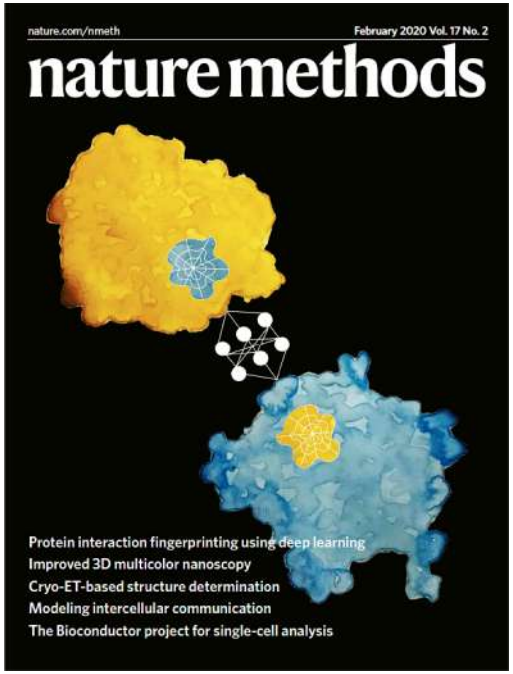


Gainza, Sverrisson et B 2020

De novo *Protein Design*



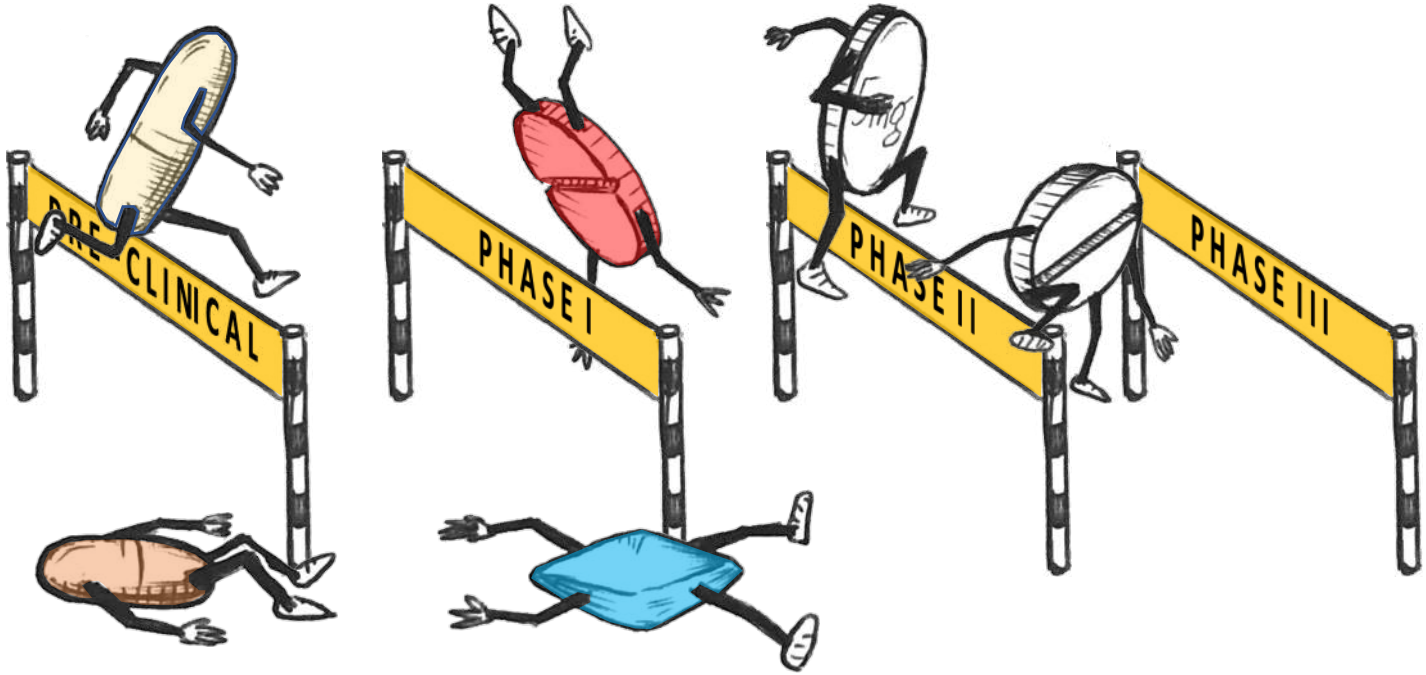
Binding site identification + designed binders
for the oncological target PD-L1



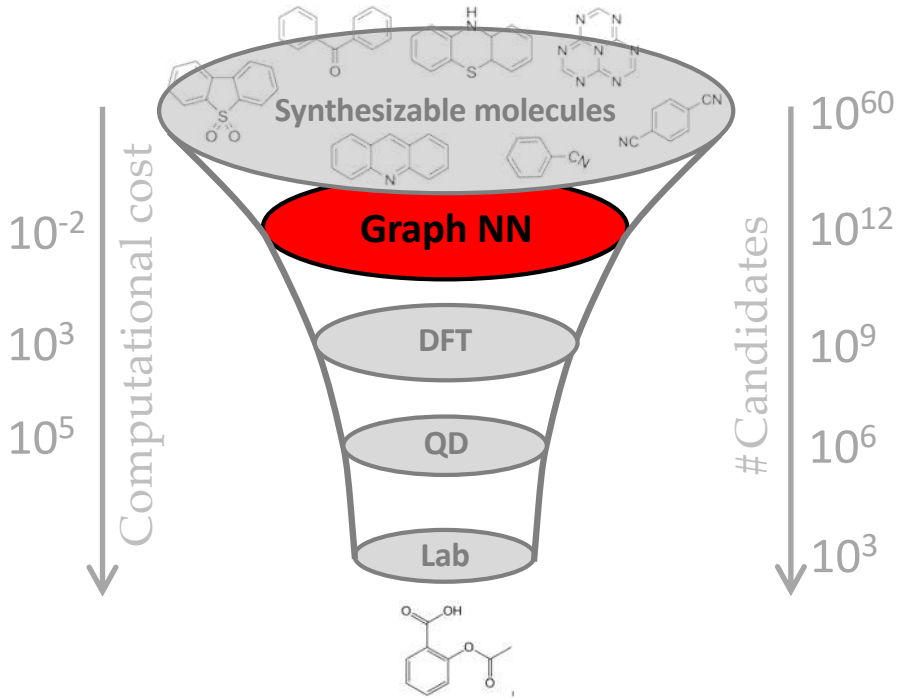
Gainza, Sverrisson et B 2020; Sverrisson, Feydy et B 2021

CHEMISTRY & DRUG DESIGN

Drug Discovery & Design



Virtual Drug Screening



"Computational funnel"

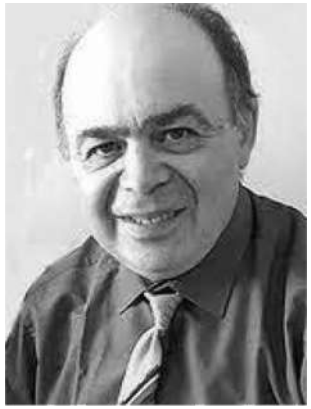
Duvenaud et al. 2015; Gilmer et al. 2017; Jin et al. 2020

History of Graph Neural Networks according to Chemists

D. Kireev

ChemNet

1995



I. Baskin

Neural descriptors

1997

C. Merkwirth

Molecular graph net

2005

D. Duvenaud

Molecular fingerprints

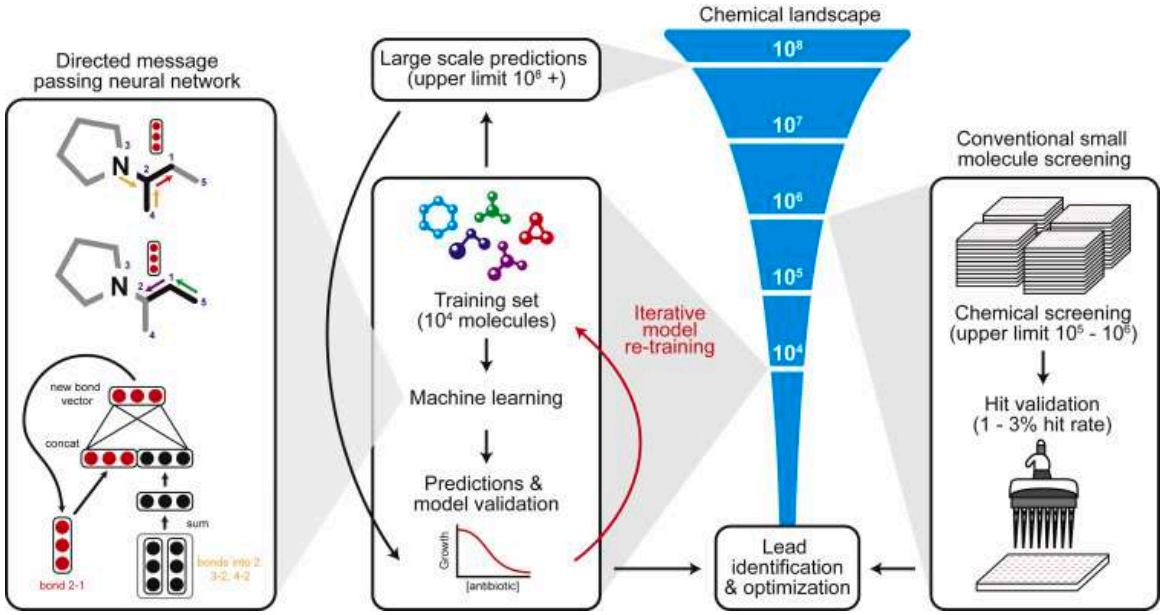
2015

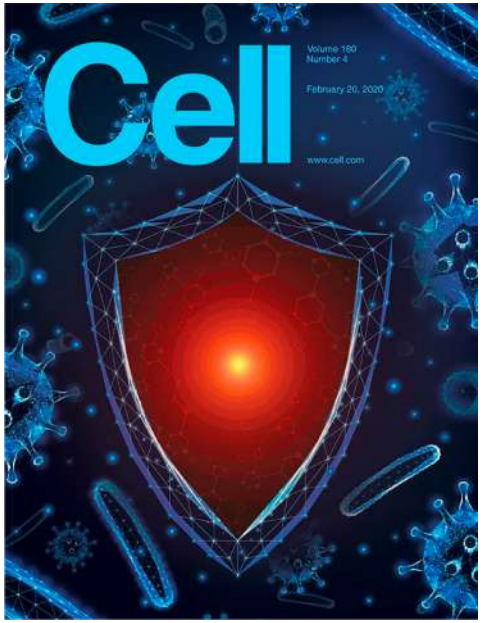
J. Gilmer

MPNNs

2017

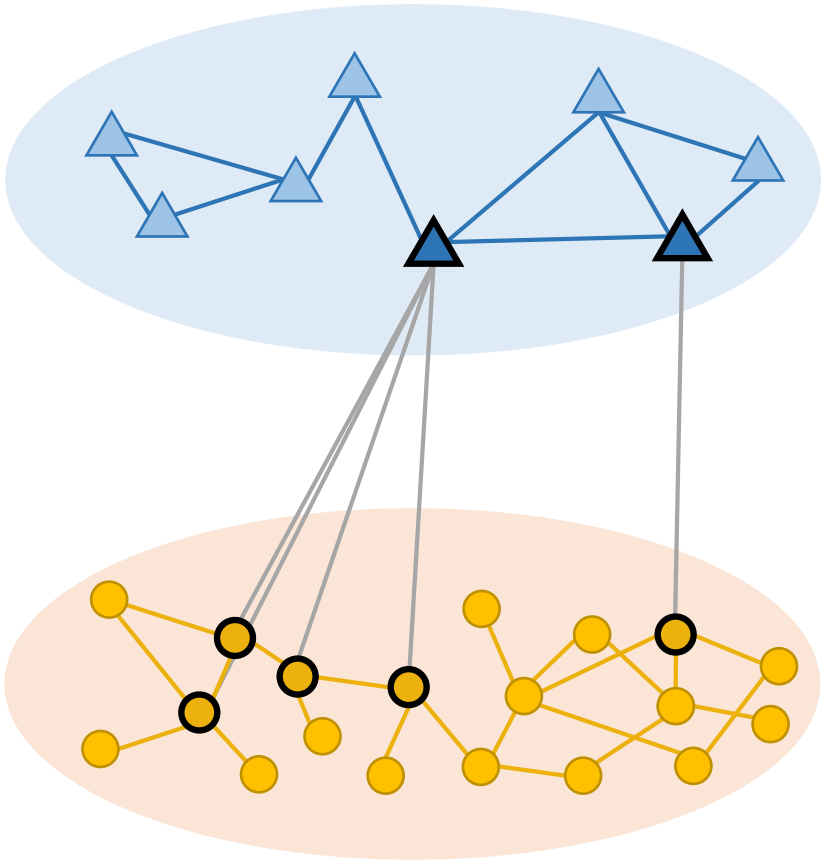
New Antibiotic Discovery



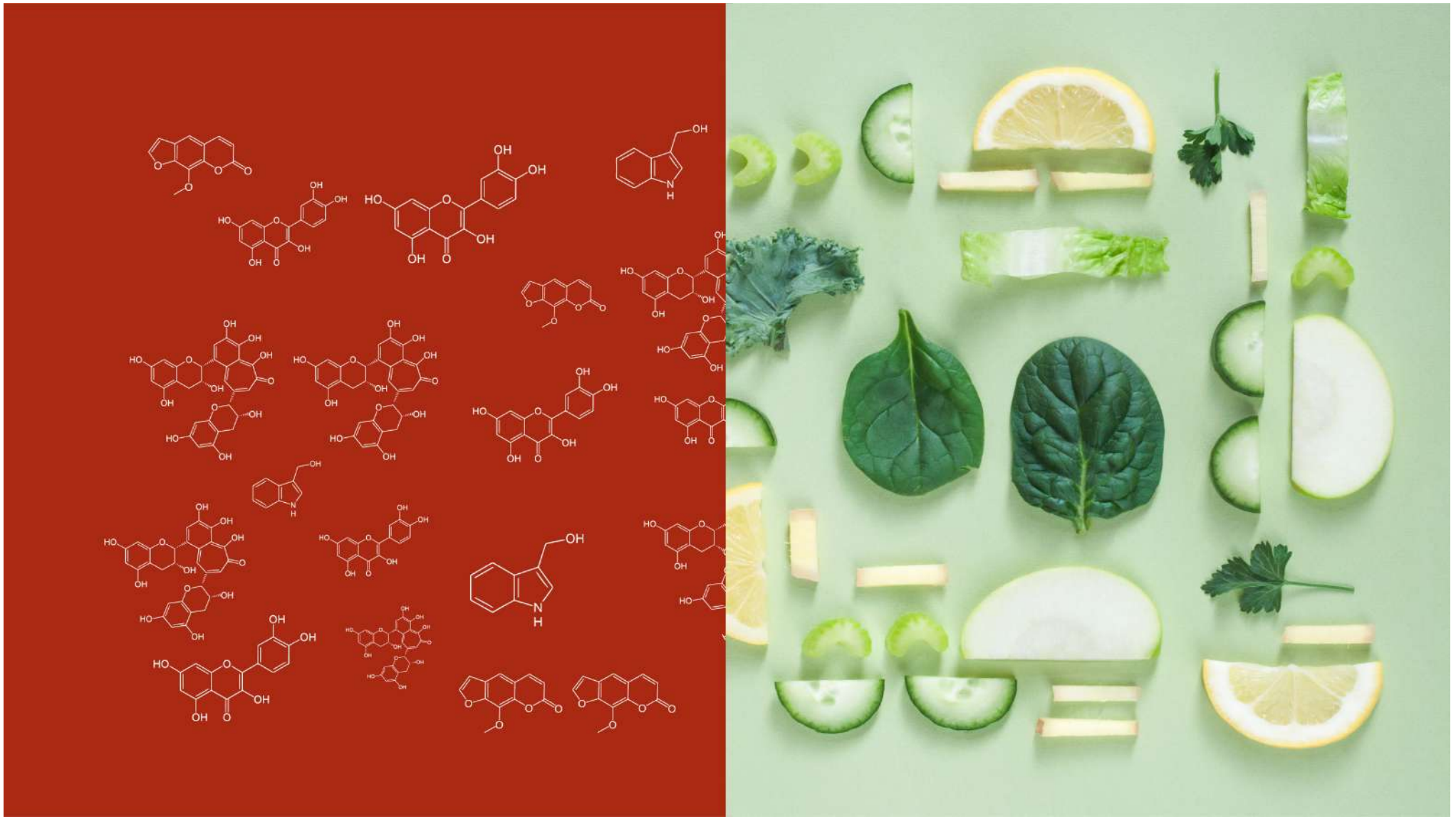


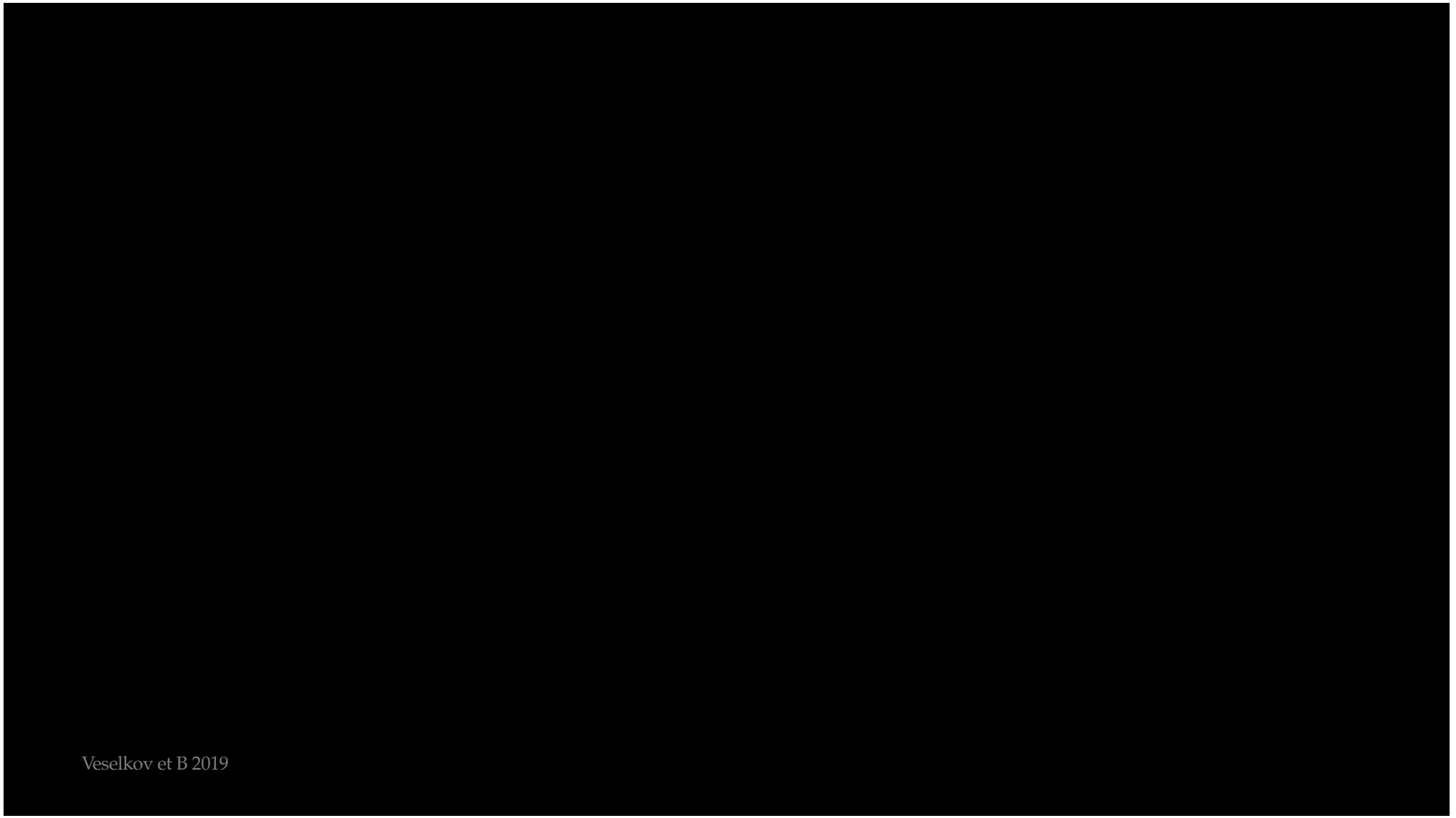
Stokes et al. 2020

Drug repositioning & Combinatorial therapy

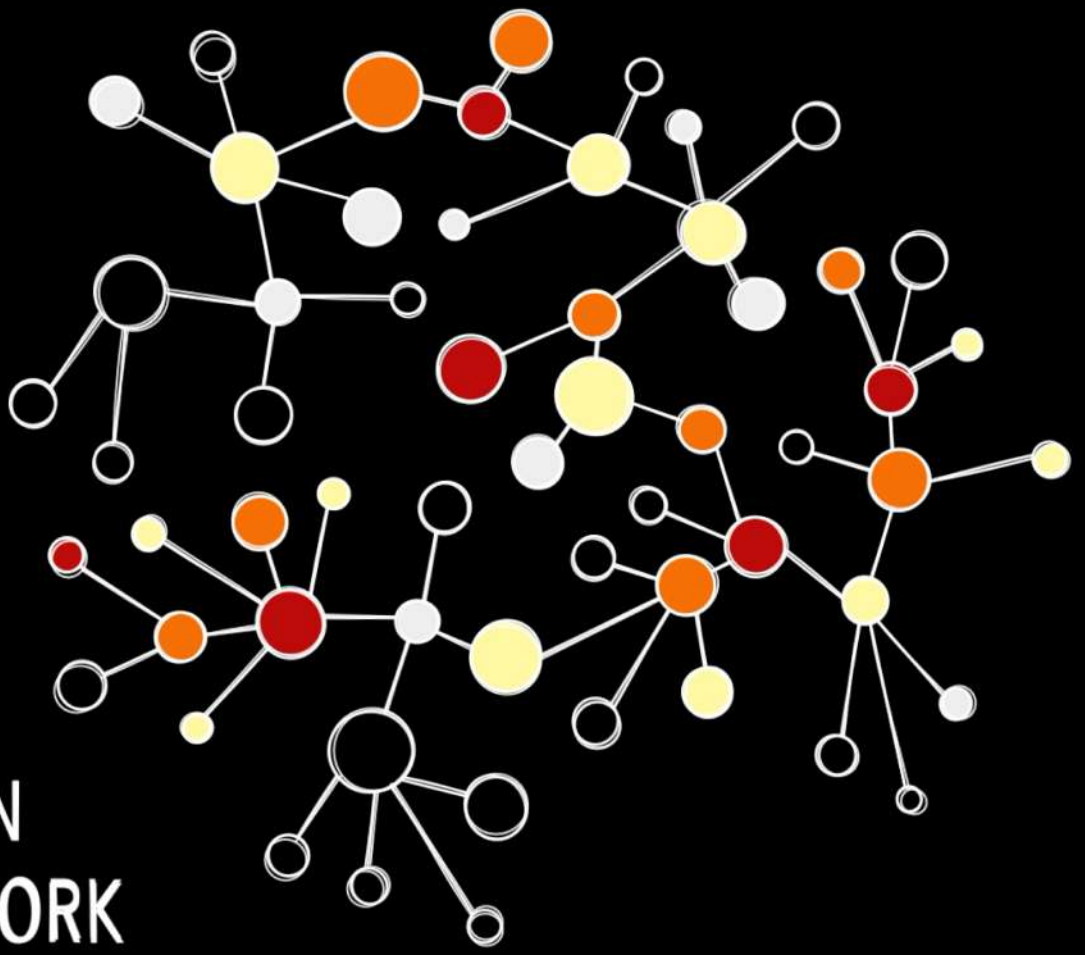


Zitnik et al. 2018



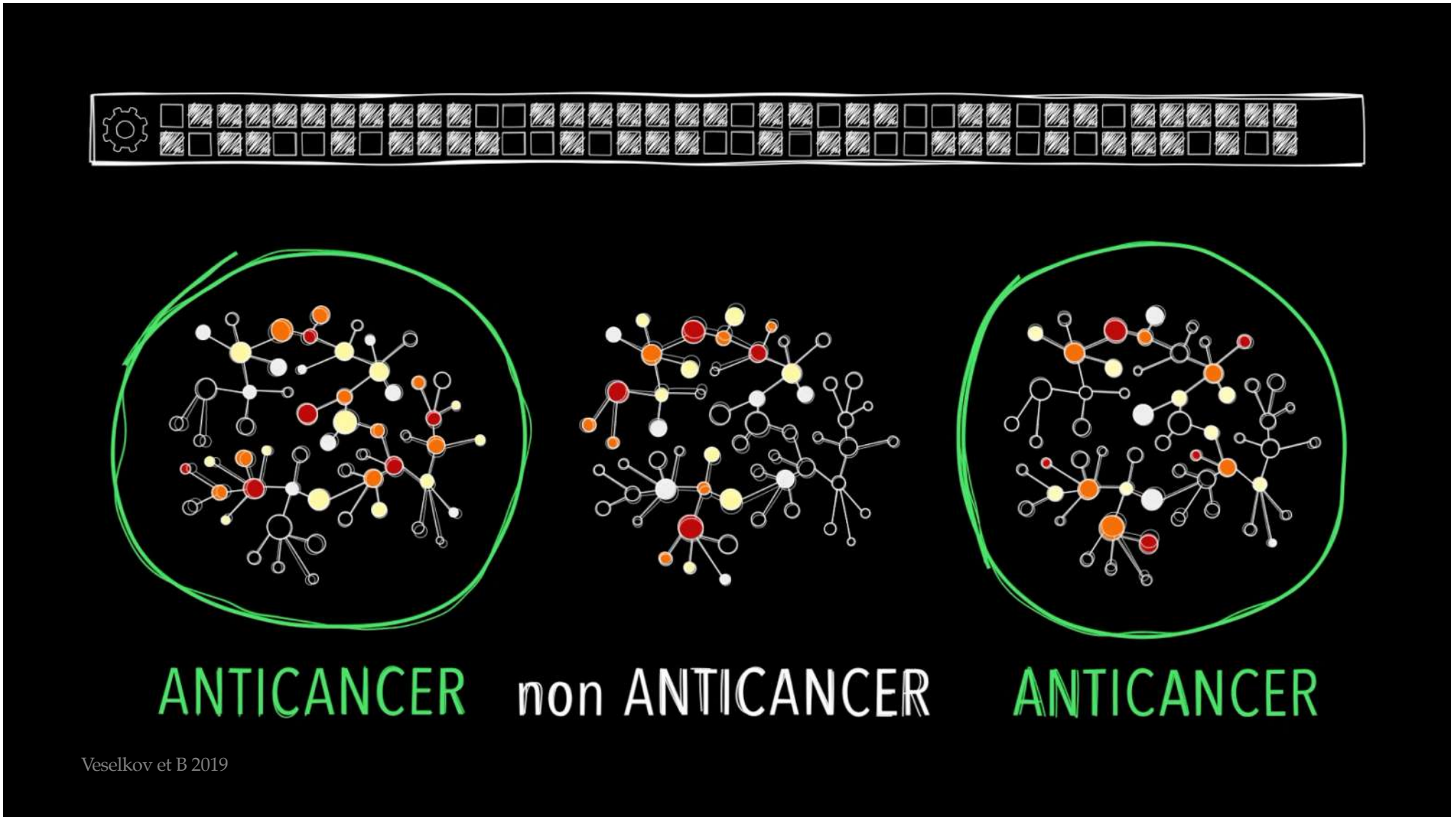

Veselkov et B 2019

Hyperfoods



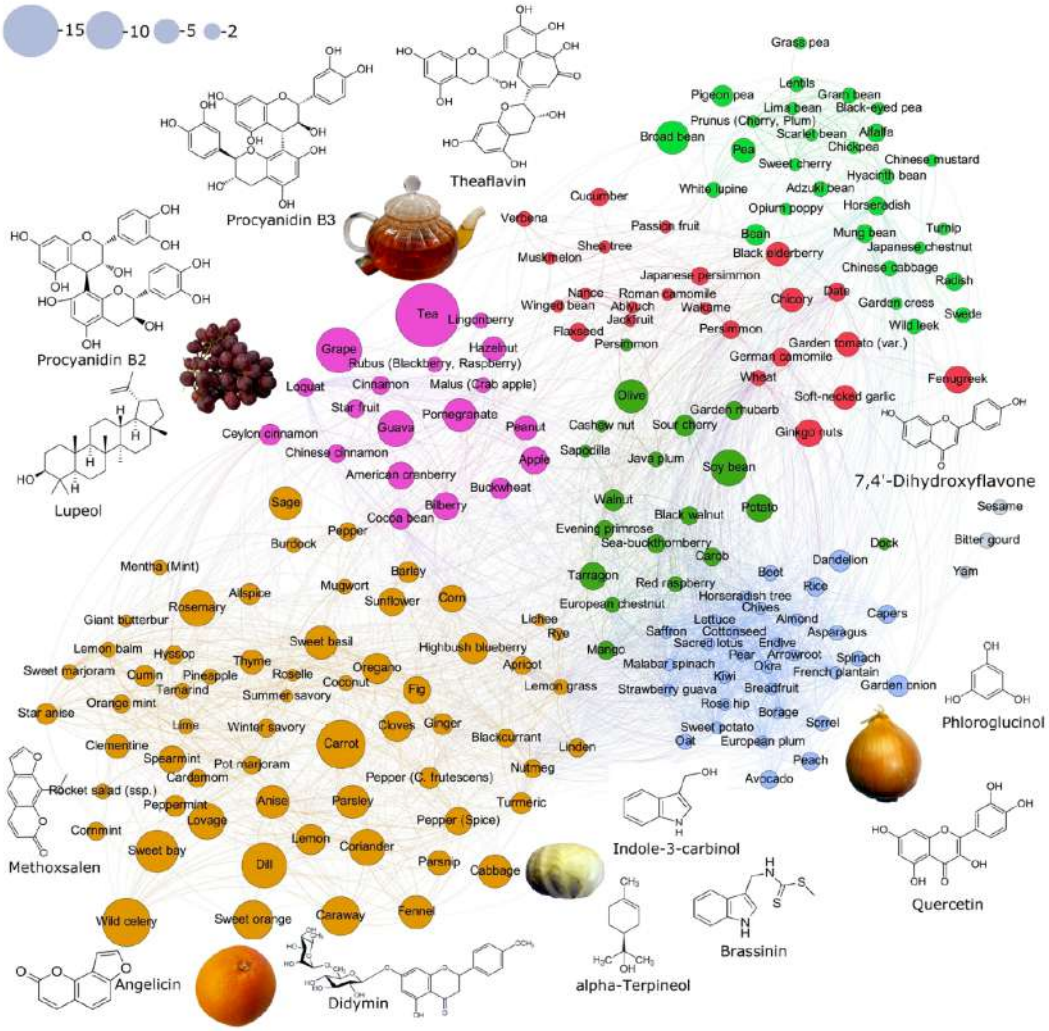
PROTEIN-PROTEIN
INTERACTION NETWORK

Veselkov et B 2019



Veselkov et B 2019

Hyperfoods



Veselkov et B 2019

Video: Vodafone Foundation / Recipes: Bruno Barbieri
Based on Laponogov et B 2020

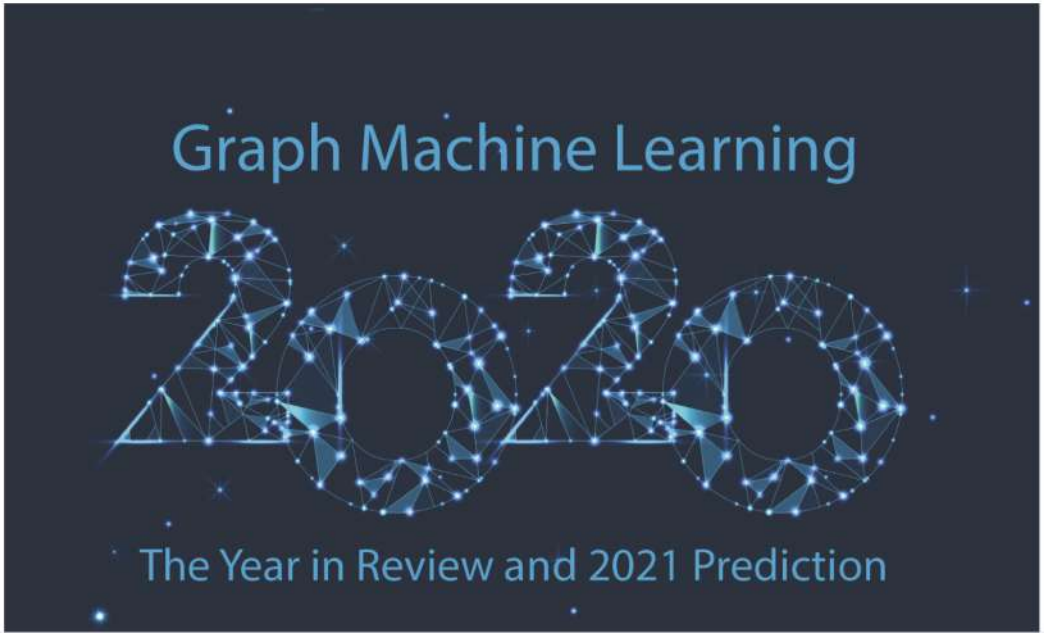
WHAT’S NEXT?

YEAR 2020 IN REVIEW & PREDICTIONS FOR 2021

What 2021 holds for Graph ML?

The end of the year is a good time to recap and make predictions. 2020 has turned Graph ML into a celebrity of machine learning. For this post, I sought the opinion of prominent researchers in the field of graph ML and its applications trying to summarise the highlights of the past year and predict what is in store for 2021.

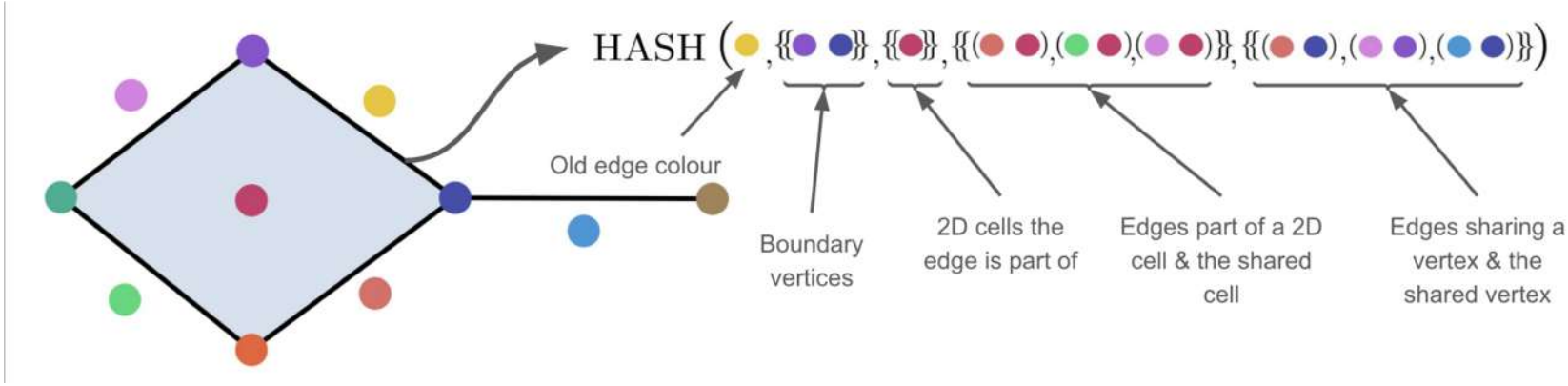
Michael Bronstein Jan 5 · 17 min read ★



“2020 saw the field of Graph ML come to terms with the fundamental limitations of the message-passing paradigm.”

—Will Hamilton (Mila)

Beyond Traditional Message Passing

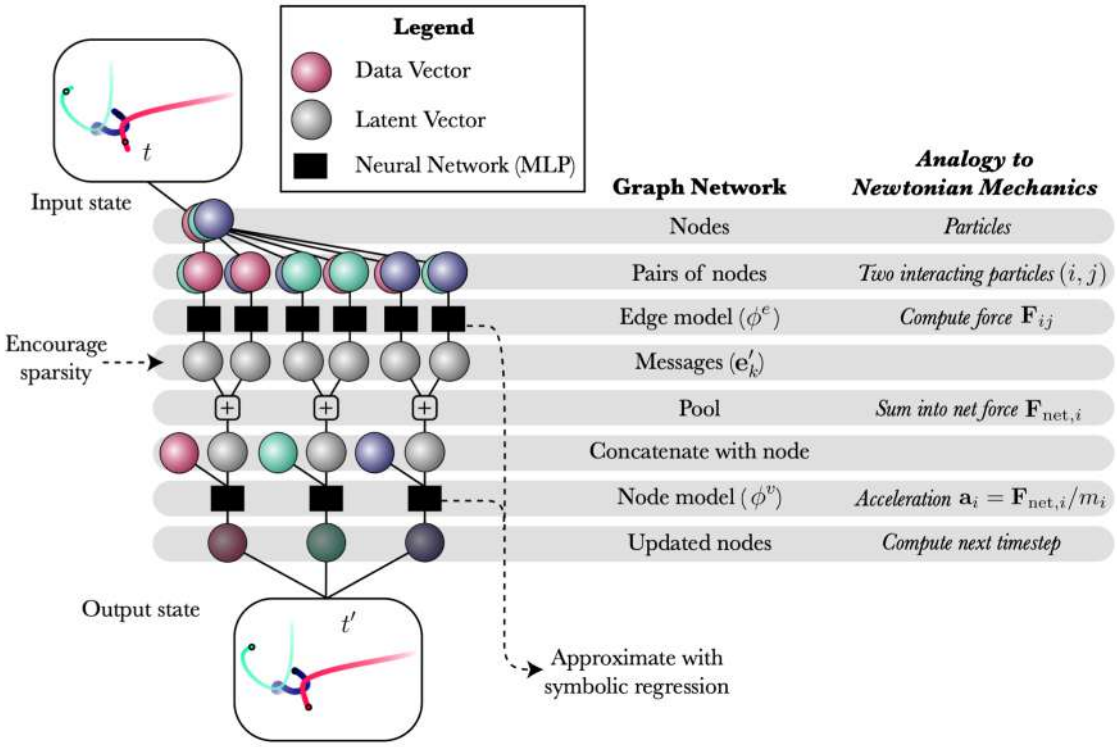


- Weisfeiler-Lehman-like schemes for higher order structures (simplicial/cell complexes)
- Strictly more powerful than MPNNs
- Links to computational topology
- Promising results in chemistry

Bodnar et B 2021

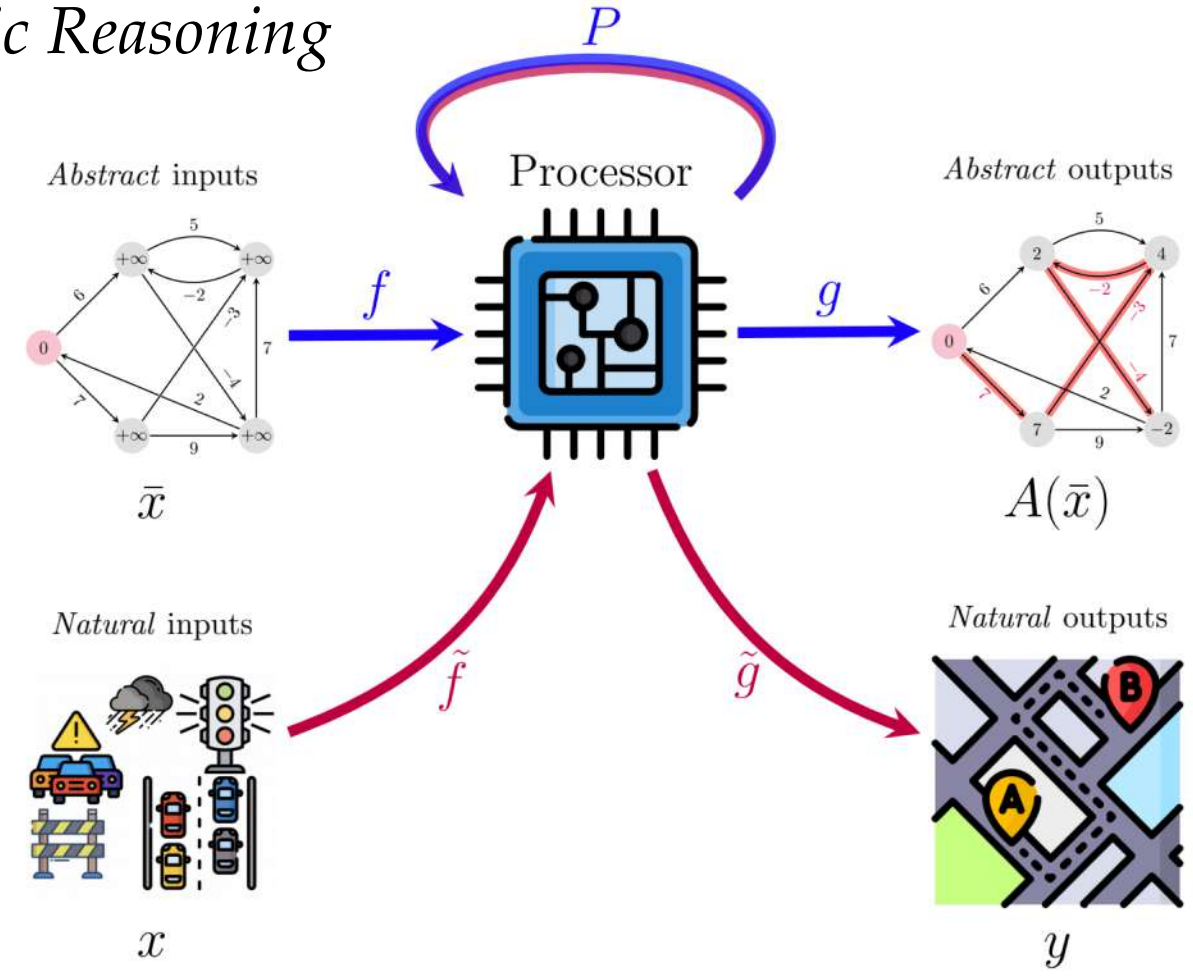
"Explainable" GNNs

Learning equations of motion using GNN to learn a predictor for a dynamical systems + symbolic regression on the messages sent along the edges. Extracted symbolic equations re-introduced into the GNN, replacing the original learned components



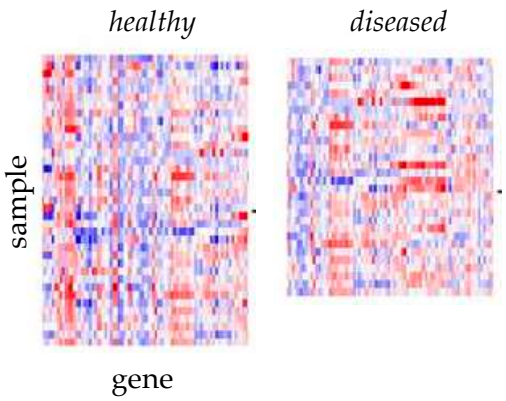
Cranmer et al. 2020

Algorithmic Reasoning

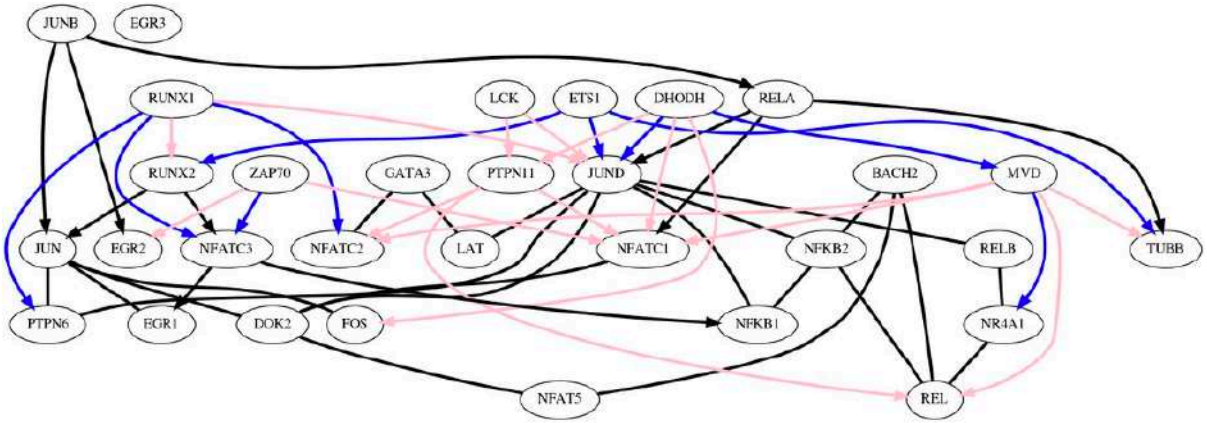


Velickovic et al. 2021

Causal Inference



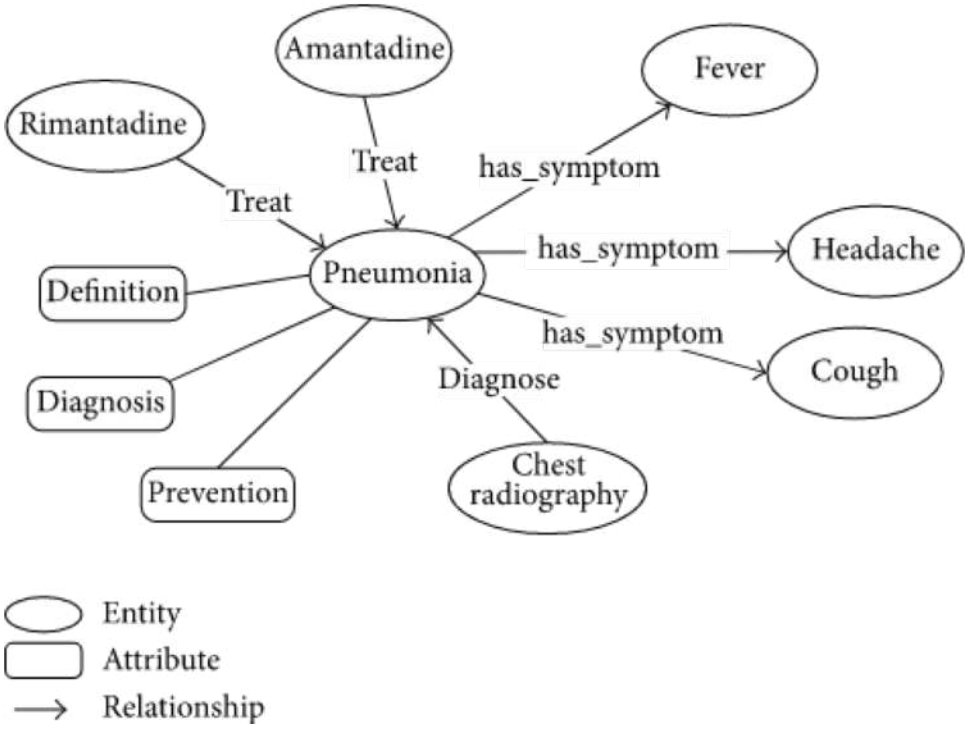
Gene expressions

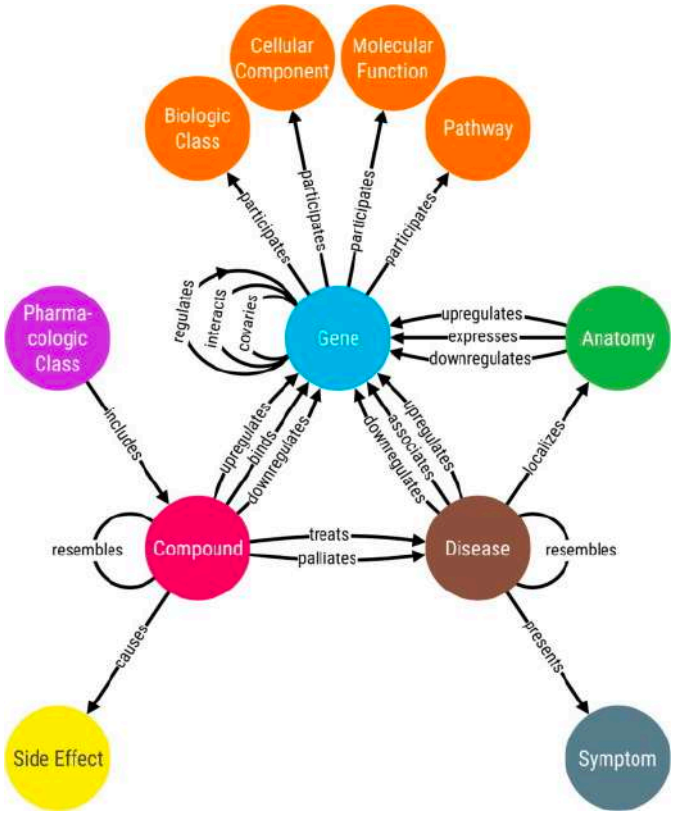


Difference gene regulatory network

Belyaeva et al. 2020

Knowledge Graphs

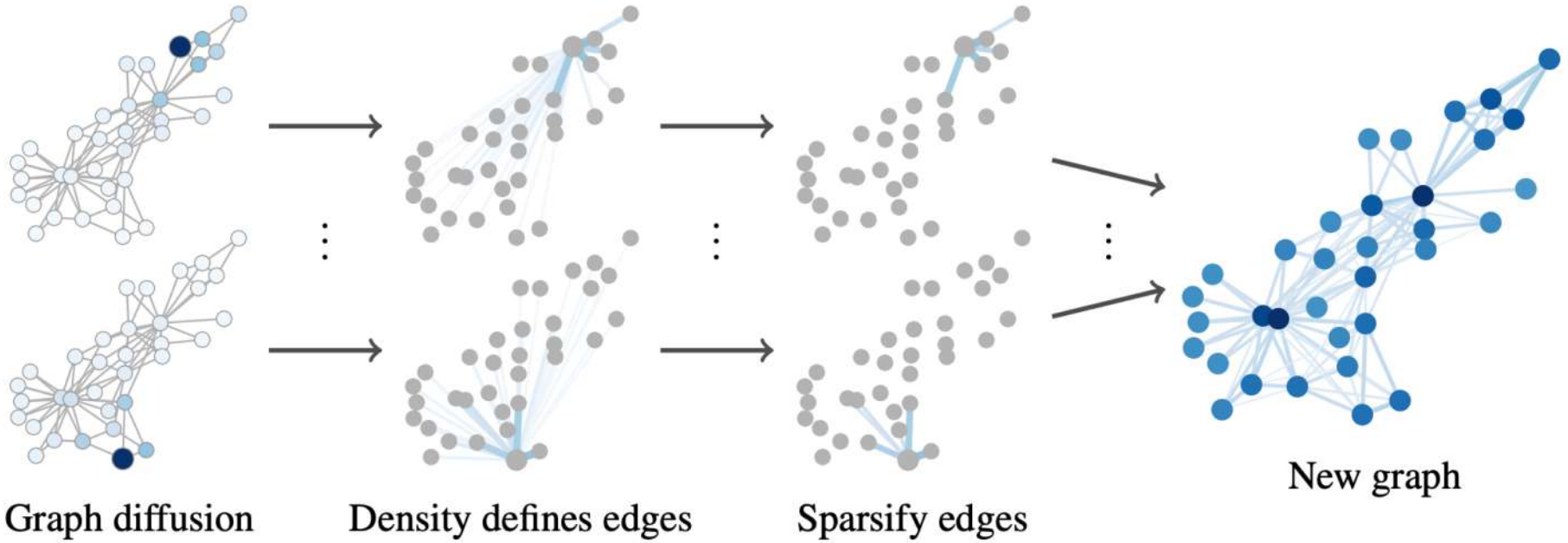




“One particularly noteworthy trend in the Graph ML community since the recent widespread adoption of GNN-based models is the separation of computation structure from the data structure.”

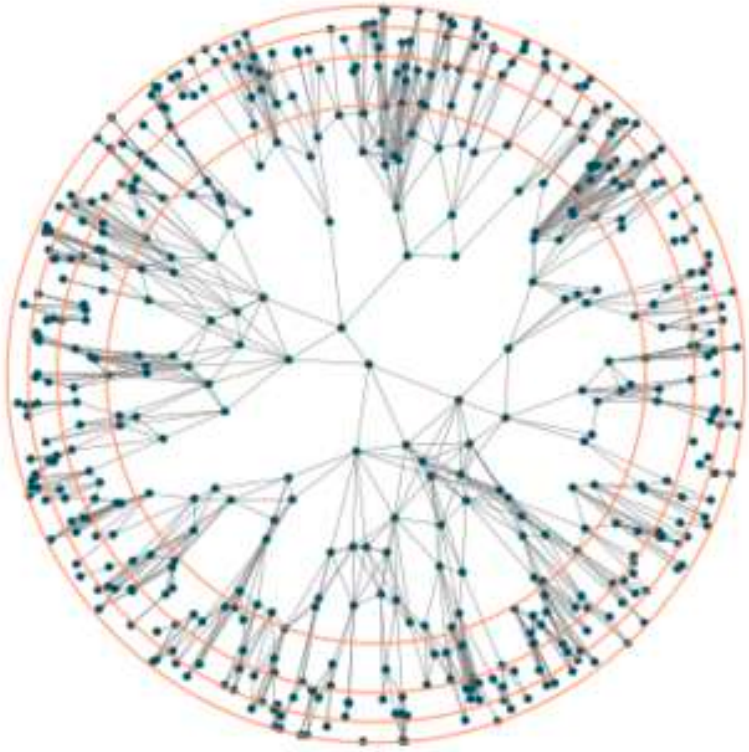
—Thomas Kipf (Google)

Data vs Computational Graphs: "Graph Rewiring"



Klicpera et al. 2019

"Network Geometry"

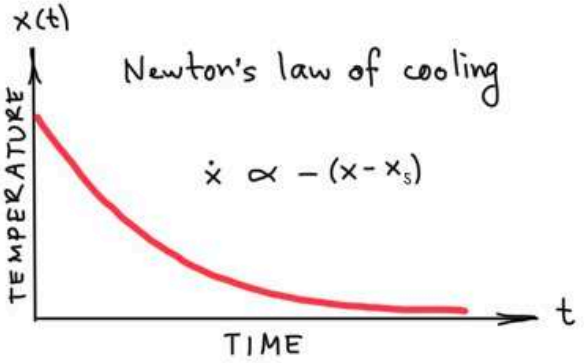


Boguna et al. 2020

Neural PDEs

- GNNs are discretised diffusion PDEs
- Use efficient numerical schemes
- Deep links to differential geometry

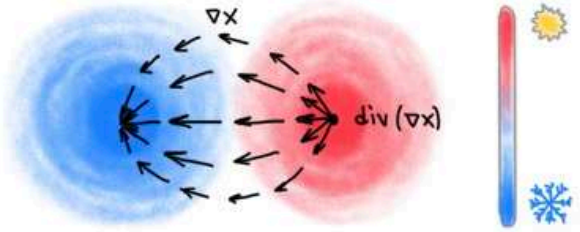
I. Newton



J. Fourier

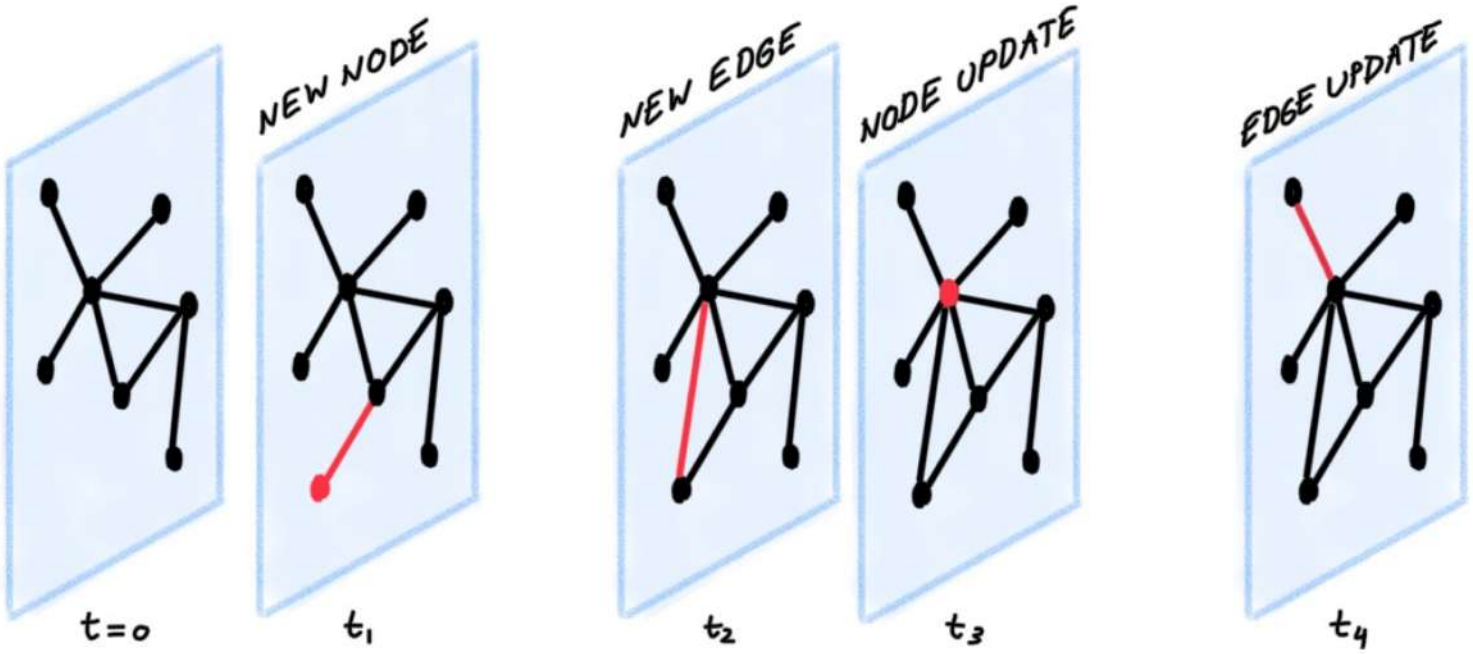
Fourier's heat transfer law

$h = -a\nabla x$



Chamberlain et B 2021

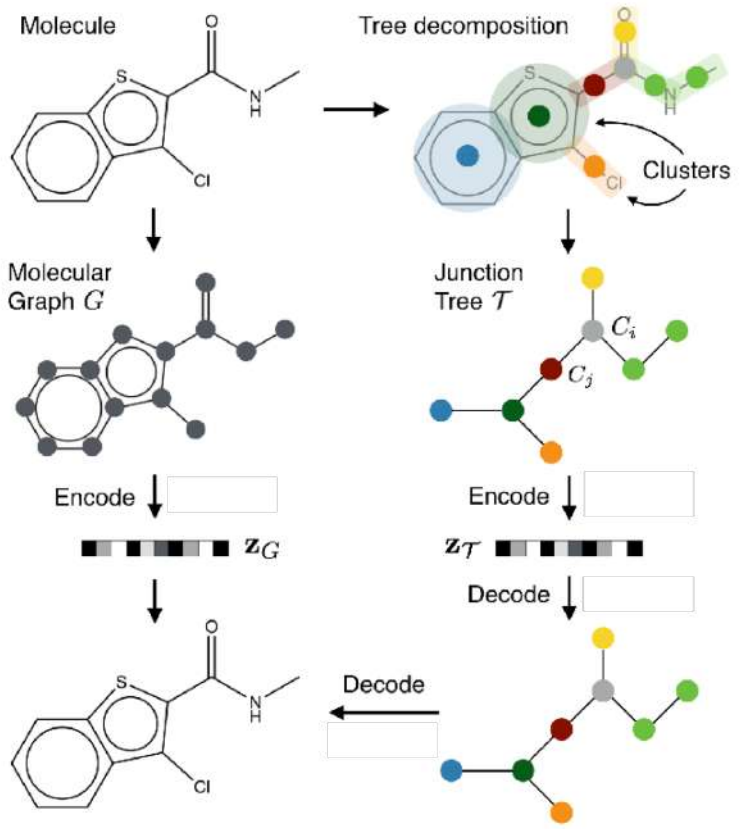
Dynamic Graphs



Rossi et B 2021

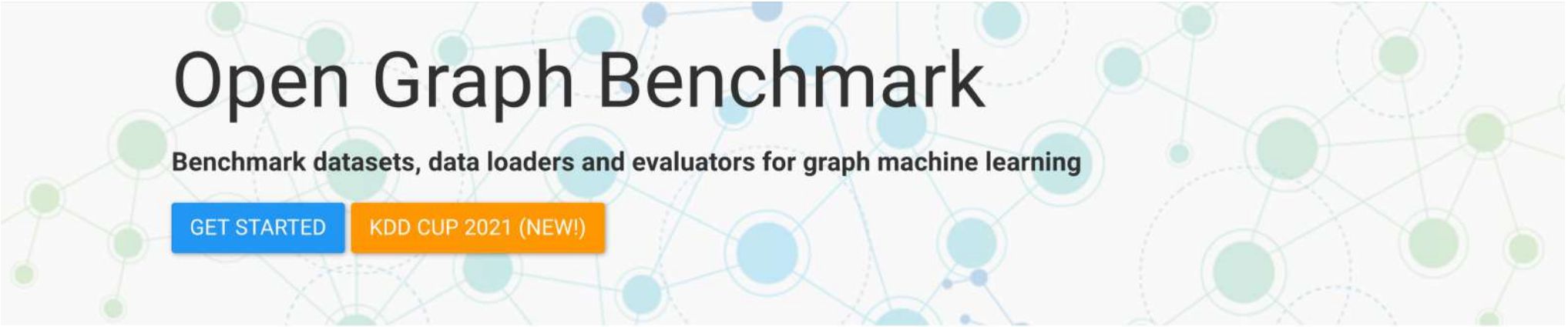
Generative Models

Generative Models for Graphs



Jin et al. 2018

Standardised Benchmarks



The Open Graph Benchmark (OGB) is a collection of realistic, large-scale, and diverse benchmark datasets for machine learning on graphs. OGB datasets are automatically downloaded, processed, and split using the OGB Data Loader. The model performance can be evaluated using the OGB Evaluator in a unified manner.
OGB is a community-driven initiative in active development. We expect the benchmark datasets to

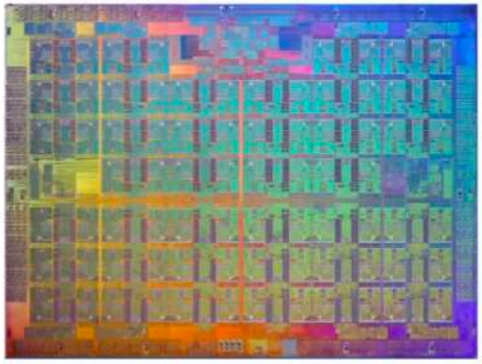
OPEN GRAPH BENCHMARK

Industrial Applications

Slide: Xavier Bresson

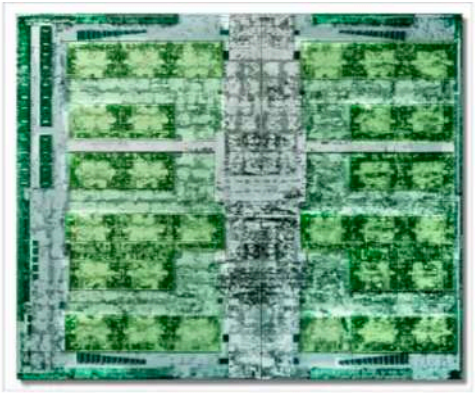
New Hardware: Beyond GPUs

Hardware Lottery: the phenomenon when certain algorithms win not because they are ideally suited to solve a certain problem, but because they run well on existing hardware



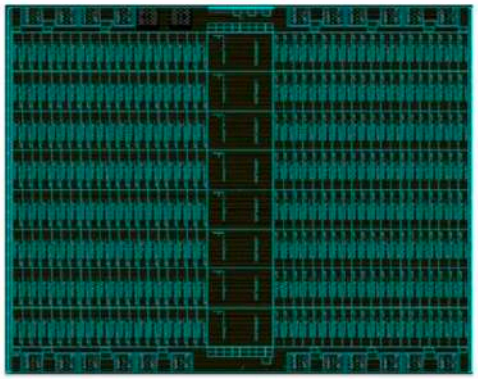
CPU

Scalar



GPU

Vector



IPU

Graph

Thank you!