# Introduction to Geometric Deep Learning

Yan Hu

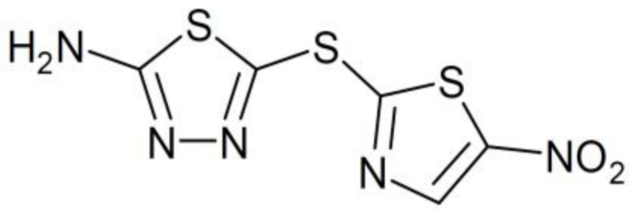
# Background

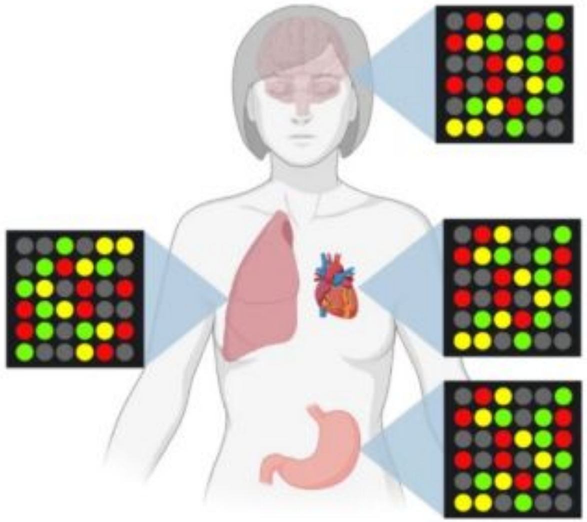
In very broad terms, the data we use to train deep learning models belongs to two main domains:

1. **Euclidean data**:  data represented in multidimensional linear spaces, it obeys Euclidean postulates, e.g, text data or tabular data.

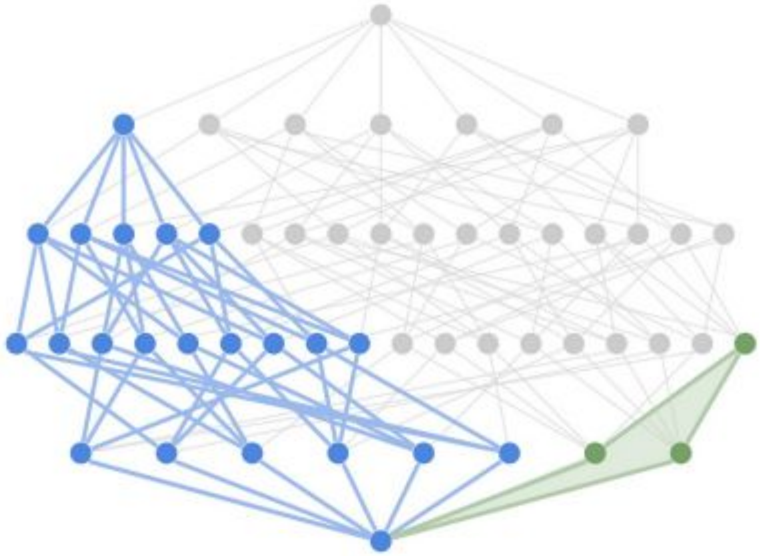
2. **Non-Euclidean data**: in even broader terms, data that doesn’t obey Euclidean postulates, e.g, molecular structures, social network interactions, or meshed 3D surface.

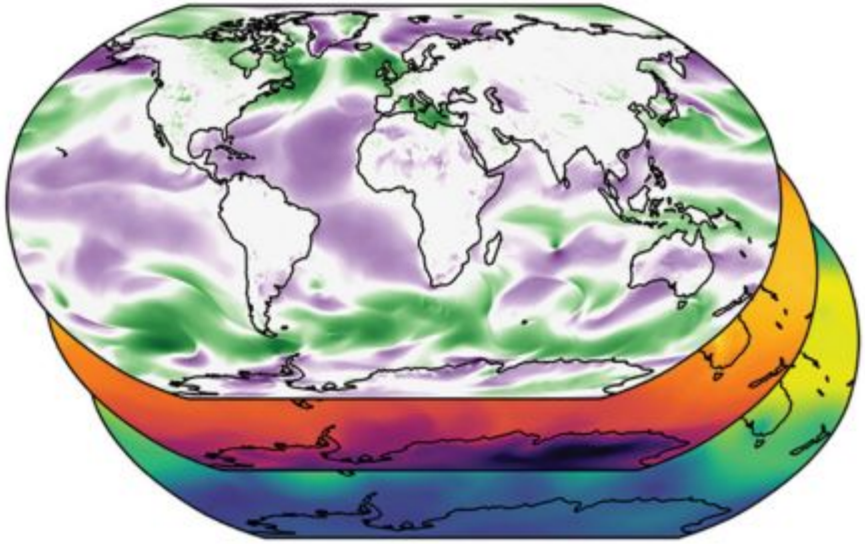
# Data from nature is often geometric



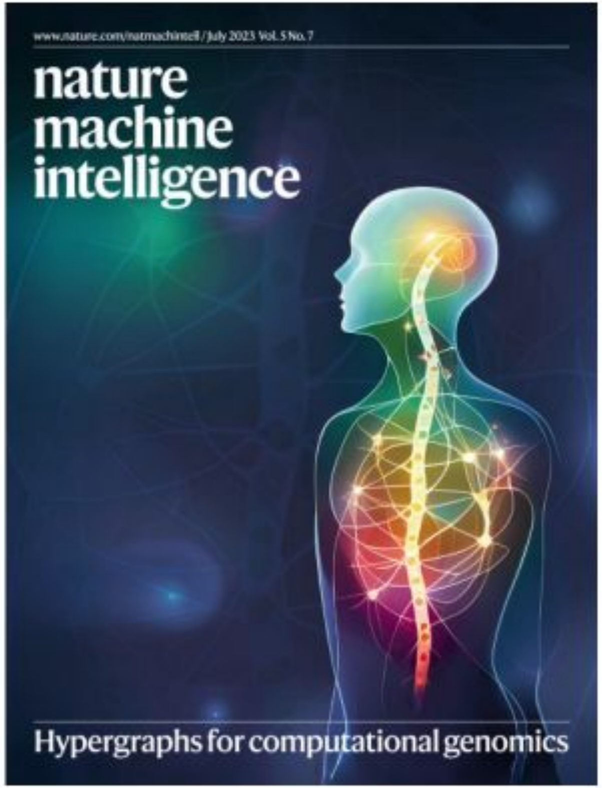
# Data from nature is often geometric





# Models of nature are often geometric


# Models of nature are often geometric




# Non-geometric models have geometric constraints

NLPers may say “no such geometry in language”.  But “geometry” is not just about spatial arrangement!

# Non-geometric models have geometric constraints

NLPers may say “no such geometry in language”.  But “geometry” is not just about spatial arrangement!

It is about constraints: design model to “respect” regularity in data. Models like Transformers touted as “generic”, but significantly constrained.

# Non-geometric models have geometric constraints

NLPers may say “no such geometry in language”.  But “geometry” is not just about spatial arrangement!

It is about constraints: design model to “respect” regularity in data. Models like Transformers touted as “generic”, but significantly constrained.

Generally, GDL offers us a perspective to categorise existing architectures. Based on which data regularity constraints they satisfy.

# Non-geometric models have geometric constraints

NLPers may say “no such geometry in language”.  But “geometry” is not just about spatial arrangement!

It is about constraints: design model to “respect” regularity in data. Models like Transformers touted as “generic”, but significantly constrained.

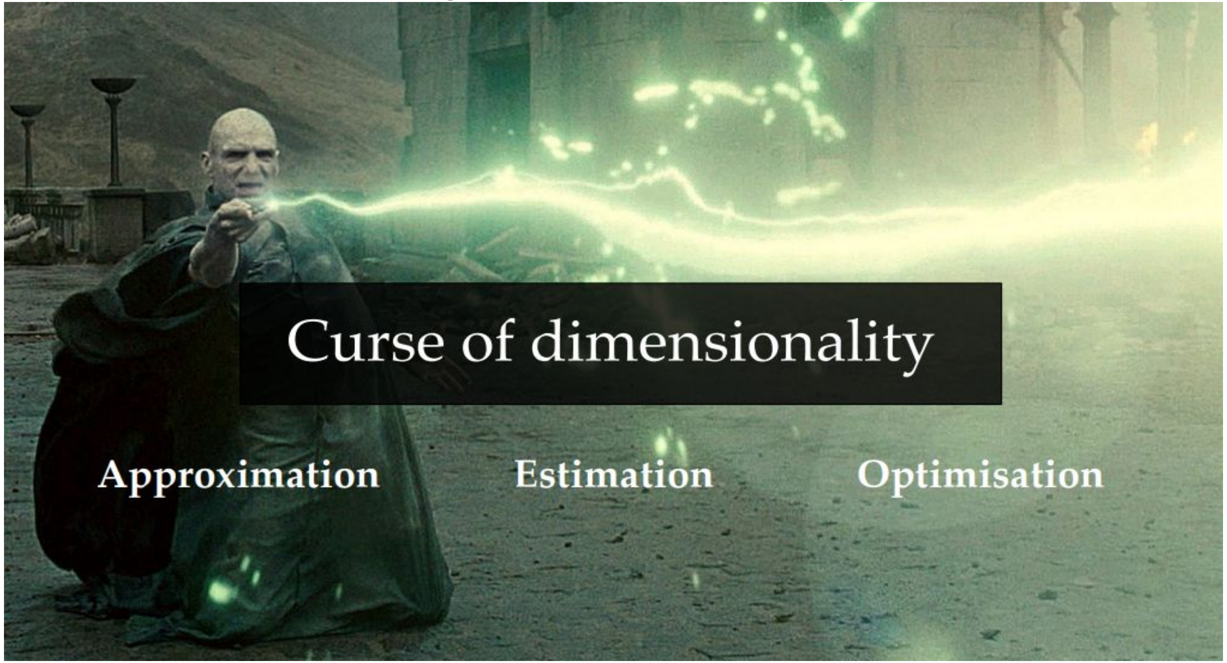
Generally, GDL offers us a perspective to categorise existing architectures. Based on which data regularity constraints they satisfy.

This is a useful perspective even if you never encounter “geometric” data.

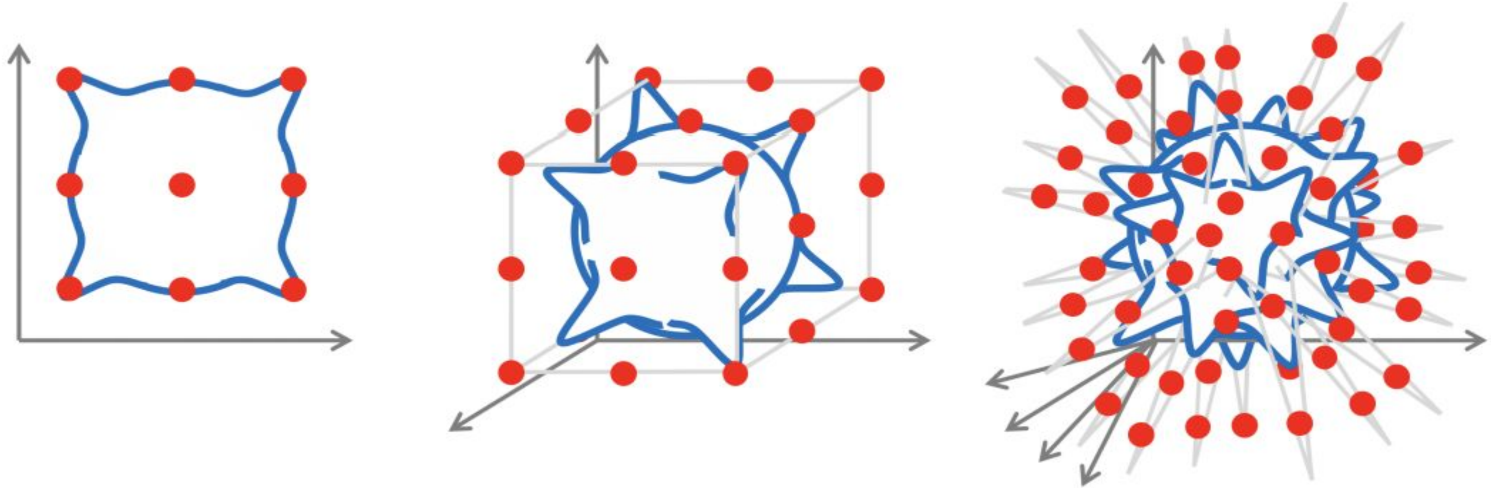
# Learning in High Dimensions

In general, learning functions in high dimensions is intractable.

Number of samples required grows exponentially with dimensions.



# Curse of Dimensionality



# Curse of Dimensionality

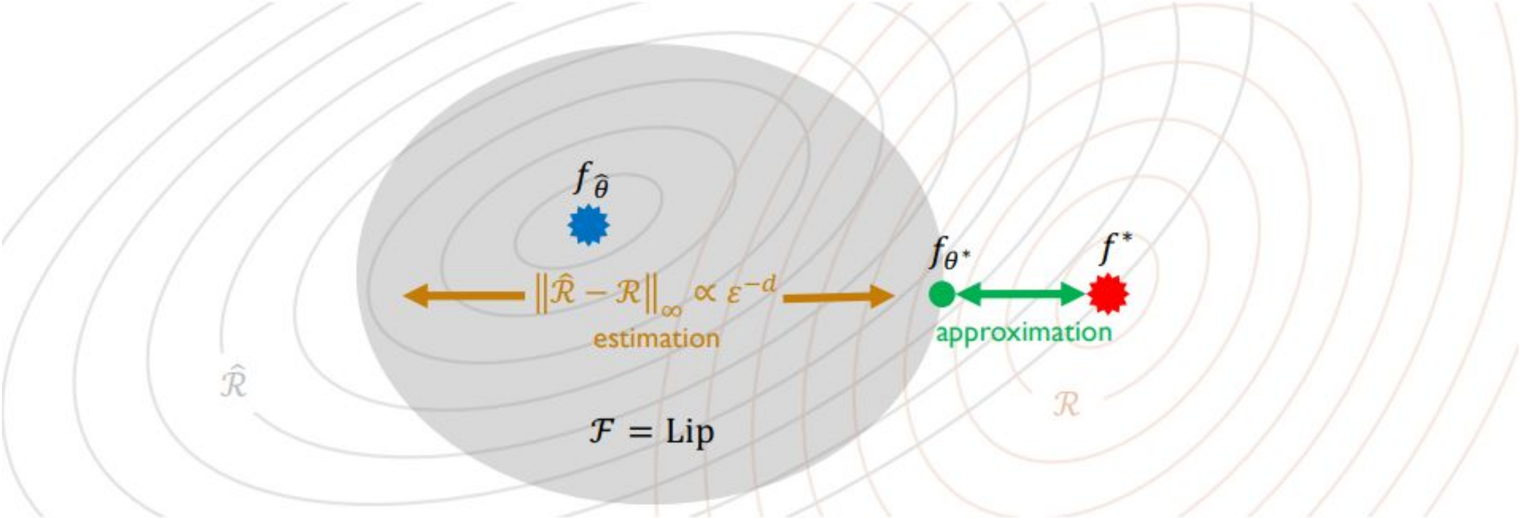
"[dimensionality is] a curse which has hung over the head of the physicist and astronomer for many a year. "

— *Dynamic Programming*

**R. Bellman**

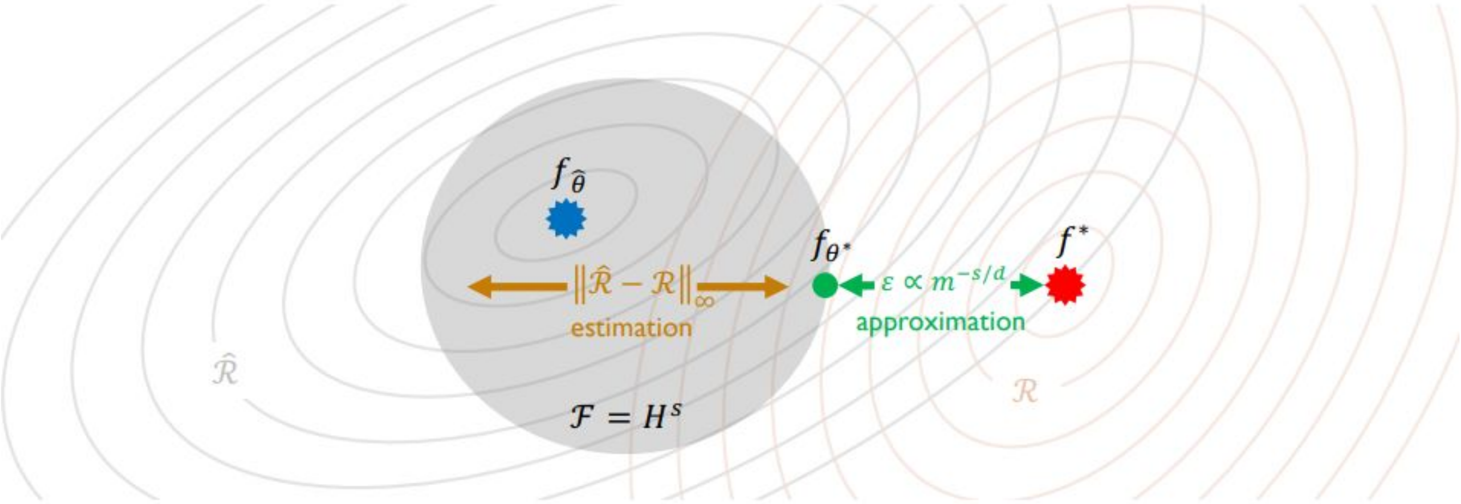
# Classical notions of regularity are of little use

Lipschitz class is too large: estimation error is dimensionality-cursed



# Classical notions of regularity are of little use

Sobolev class is too small: approximation error is dimensionality-cursed



# Takeaways

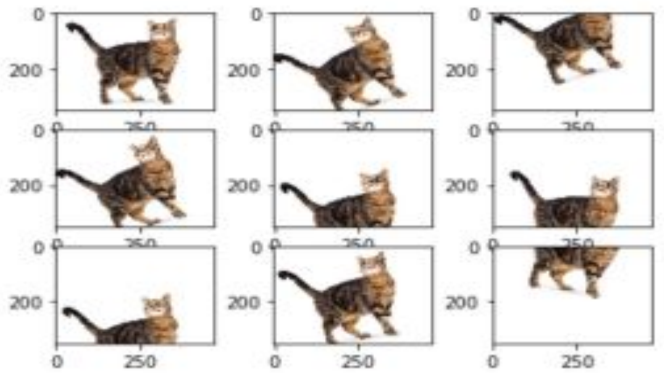
1. Learning in high dimensions is plagued by the curse of dimensionality
2. Impossible without assumptions (“priors”)
3. Classical assumptions of regularity (from low-dimensional analysis) are not appropriate priors
4. Geometric priors: inputs are signals defined over low-dimensional geometric domains

# Geometry to the rescue!

We can inject assumptions about geometry through inductive biases.

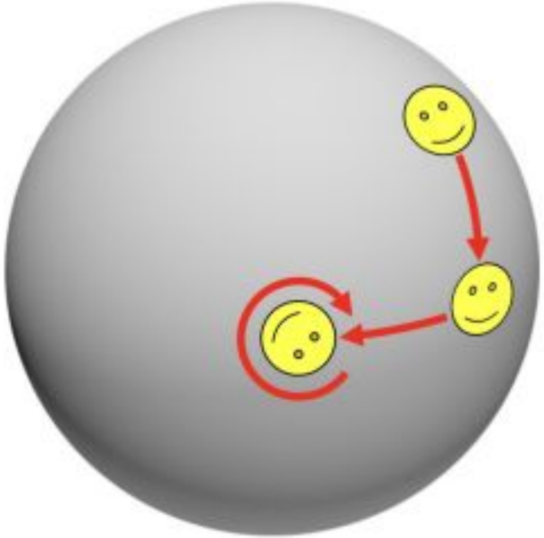
Restrict the functions to ones that respect the geometry. This can make the high-dimensional problem more tractable!

Examples: **Image** data should be processed independently of shifts



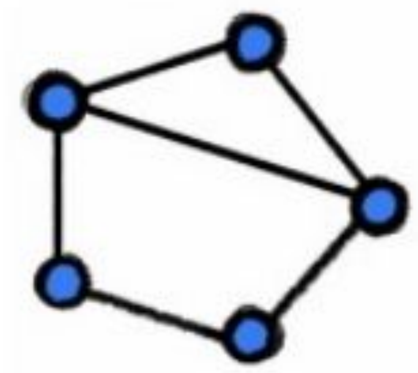
# Continued

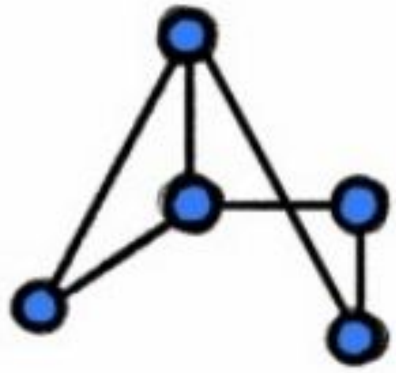
Examples: **Spherical data** should be processed independently of rotations



# Continued

Examples: **Graph data** should be processed independently of isomorphism





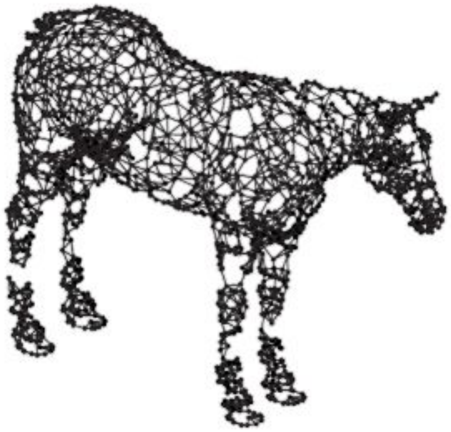
# A Roadmap for Formalisation

1. To handle geometry of data, we need to formalise where the data lives (domain) and how to featurise it (signal)
2. Once we understand data domains, we can then formalise symmetries of those domains (groups)
3. Equipped with groups, we need to formalise how they transform the data domains (group actions)
4. Deep learning concerns itself with linear algebra; we need to be able to talk about group actions as matrix operations (representations)
5. Using representations, we can formalise what it means for a deep learning model to respect symmetries (invariance & equivariance)

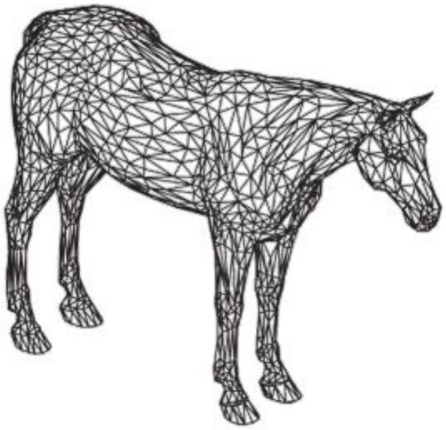
# Geometric Domains

- **Domain** $\Omega$ = set + some structure

**Point cloud**
(bare set)



**Graph**
(local neighbourhood)

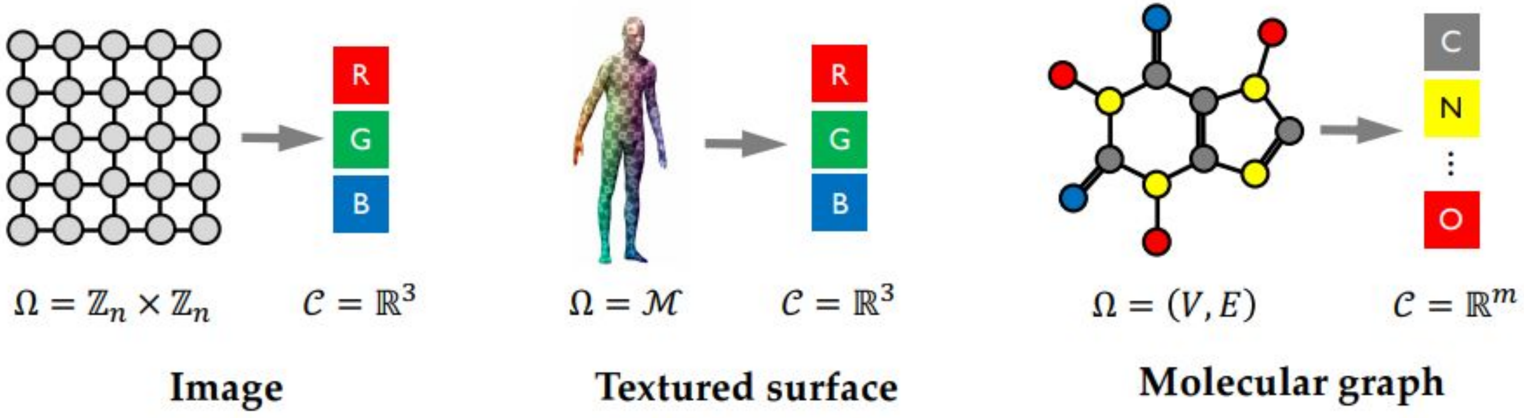


**Mesh**
(local metric)

# Signals on Geometric Domains

Signal $x \in \mathcal{X}(\Omega, C) = \{x: \Omega \rightarrow C\}$ , C-valued function on Ω

- Domain Ω (often no vector space structure, i.e., we cannot add points)
- Vector space C (dimensions referred to as "channels")



# Signals on Geometric domains

The space of signals $\mathscr{X}(\Omega,C)$ is a vector space (possibly infinite-dimensional)

- We can add signals and multiply them by a scalar

# Signals on Geometric Domains

The space of signals $\mathscr{X}(\Omega, C)$ is a vector space (possibly infinite-dimensional)

- Given an inner product $\langle , \rangle$ on C, and a measure $\mu$ on Ω, we can define an inner product on $\mathscr{X}(\Omega, C)$ as:

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

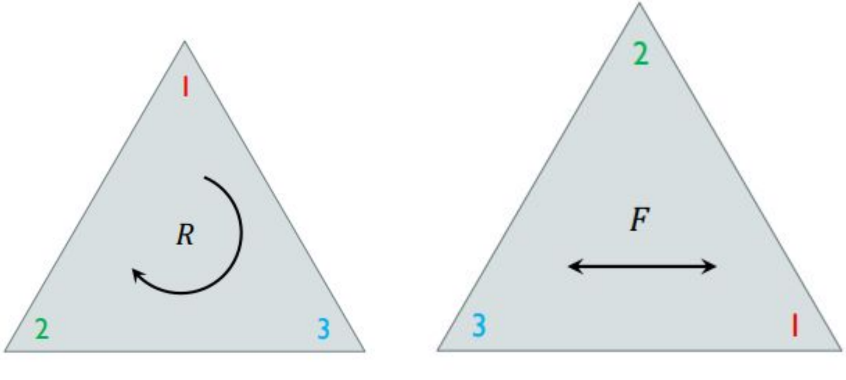
# A Roadmap for Formalisation

1. To handle geometry of data, we need to formalise where the data lives (domain) and how to featurise it (signal)
2. Once we understand data domains, we can then formalise symmetries of those domains (groups)
3. Equipped with groups, we need to formalise how they transform the data domains (group actions)
4. Deep learning concerns itself with linear algebra; we need to be able to talk about group actions as matrix operations (representations)
5. Using representations, we can formalise what it means for a deep learning model to respect symmetries (invariance & equivariance)

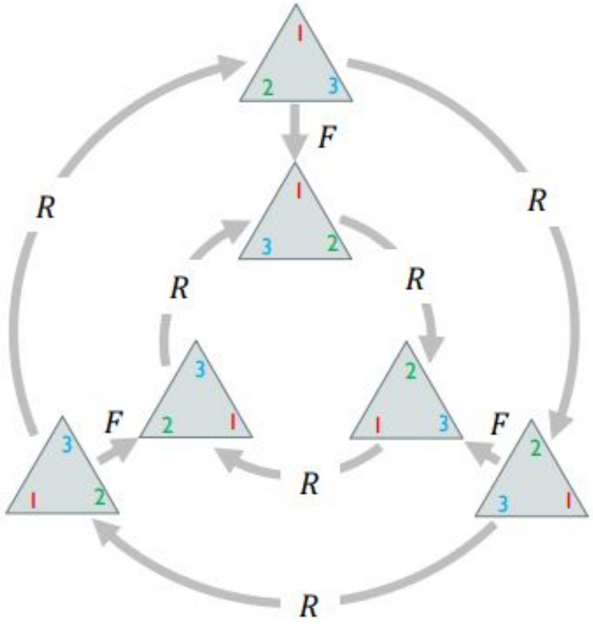
# Symmetries

A symmetry of an object is a transformation of that object that leaves it unchanged

Examples: Symmetry of a triangle




rotation by 120°    reflection



# Symmetries

- The identity transformation is always a symmetry
- Given two symmetry transformations, their composition (doing one after the other) is also a symmetry
- Given any symmetry, it must be invertible
- Moreover, its inverse is also a symmetry

# Groups

A **group** $(G,*)$ is a set $G$ together with binary operation $* : G\times G \to G$ (denoted by juxtaposition $g * h = gh$ for brevity) satisfying the following axioms:

- *Associativity:* $(gh)k = g(hk)$ for all $g, h, k \in G$
- *Identity:* $\exists!\, e \in G$ satisfying $eg = ge = g$ for all $g \in G$
- *Inverse:* $\exists!\, g^{-1} \in G$ for each $g \in G$ satisfying $g^{-1}g = gg^{-1} = e$

# A Roadmap for Formalisation

1. To handle geometry of data, we need to formalise where the data lives (domain) and how to featurise it (signal)
2. Once we understand data domains, we can then formalise symmetries of those domains (groups)
3. Equipped with groups, we need to formalise how they transform the data domains (group actions)
4. Deep learning concerns itself with linear algebra; we need to be able to talk about group actions as matrix operations (representations)
5. Using representations, we can formalise what it means for a deep learning model to respect symmetries (invariance & equivariance)

# Group actions on objects

Point in a plane | Image (function) | Vector field

The type of an object can be defined by the way it transforms by a group

# Group Action

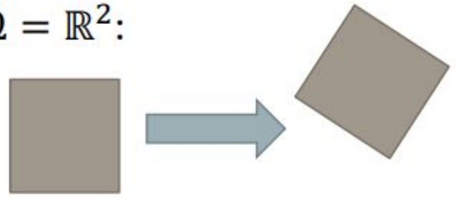
Let $G$ be a group and $X$ a set. A **(left) group action** of $G$ on $X$ (often denoted $gx = \alpha(g,x)$) is a mapping of the form $\alpha : G \times X \to X$ satisfying

- *Identity:* $\alpha(e,x) = x$ for all $x \in X$
- *Composition:* $\alpha(gh,x) = \alpha\big(g,\alpha(h,x)\big)$ for all $g,h \in G$ and $x \in X$

# Group Action

e.g.: Euclidean 2D motions $\mathfrak{G} = \mathbb{R}^3$ (angle + translation) acting on $\Omega = \mathbb{R}^2$:

$$(\theta, t_x, t_y)(x, y) \mapsto \left(x\cos\theta + y\sin\theta + t_x, x\sin\theta + y\cos\theta + t_y\right)$$



**Exercise:** Verify this satisfies the group action axioms

# A Roadmap for Formalisation

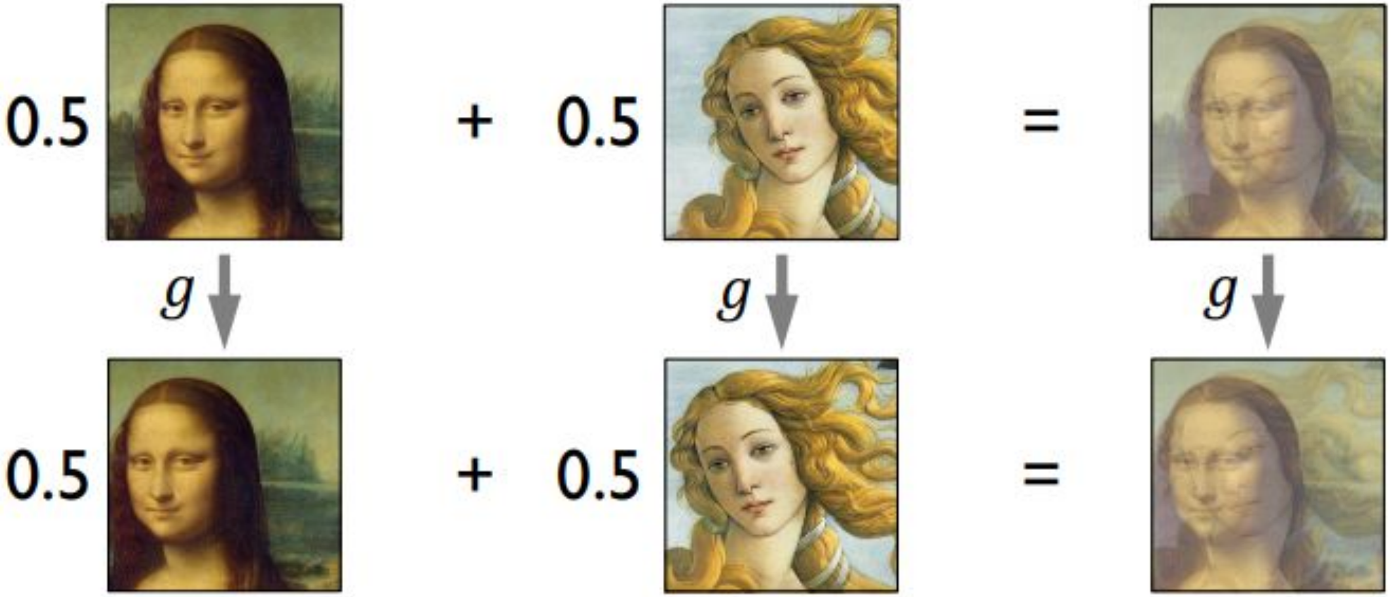
1. To handle geometry of data, we need to formalise where the data lives (domain) and how to featurise it (signal)
2. Once we understand data domains, we can then formalise symmetries of those domains (groups)
3. Equipped with groups, we need to formalise how they transform the data domains (group actions)
4. Deep learning concerns itself with linear algebra; we need to be able to talk about group actions as matrix operations (representations)
5. Using representations, we can formalise what it means for a deep learning model to respect symmetries (invariance & equivariance)

# Linear Group Representation

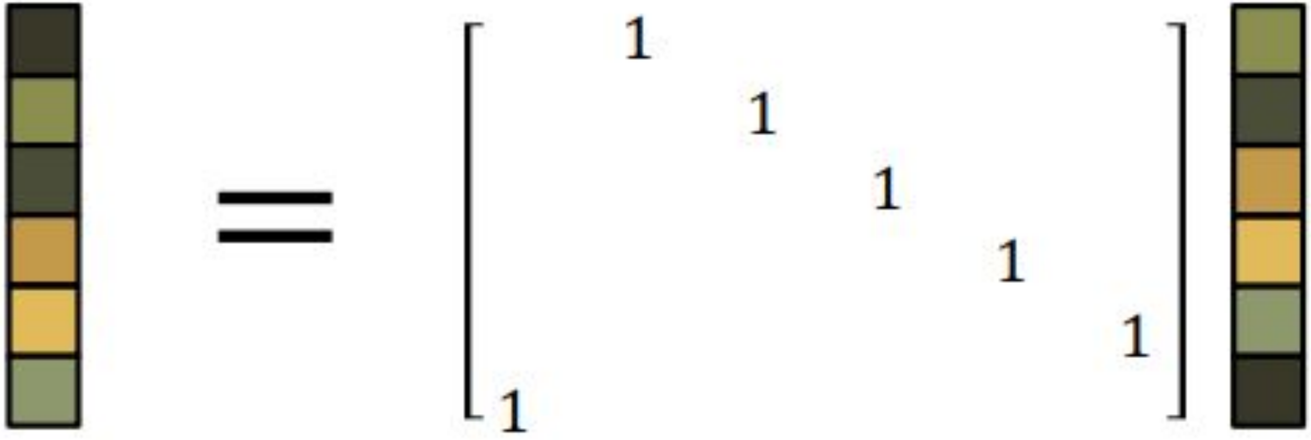
A $d$–**dimensional (linear) representation** of $G$ is a map $\rho\colon G \to \mathbb{R}^{d\times d}$ assigning to each $g \in G$ an **invertible matrix** $\rho(g) \in \mathbb{R}^{n\times n}$ satisfying $\rho(gh) = \rho(g)\rho(h)$ for all $g, h \in G$.

# Group actions on Signals on Geometric Domains

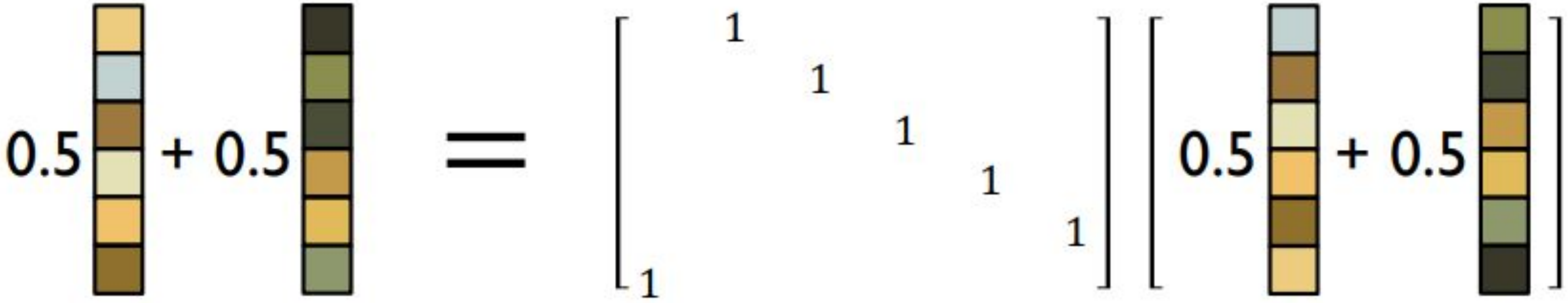
Given a group $G$ acting on a **domain** $\Omega$, we automatically obtain an action of $G$ on the space of signals $\mathcal{X}(\Omega)$ through the **regular representation** $(\rho(g)x)(u) = x(g^{-1}u)$



## Intuition



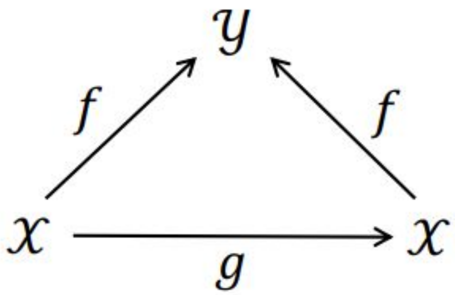
# Intuition



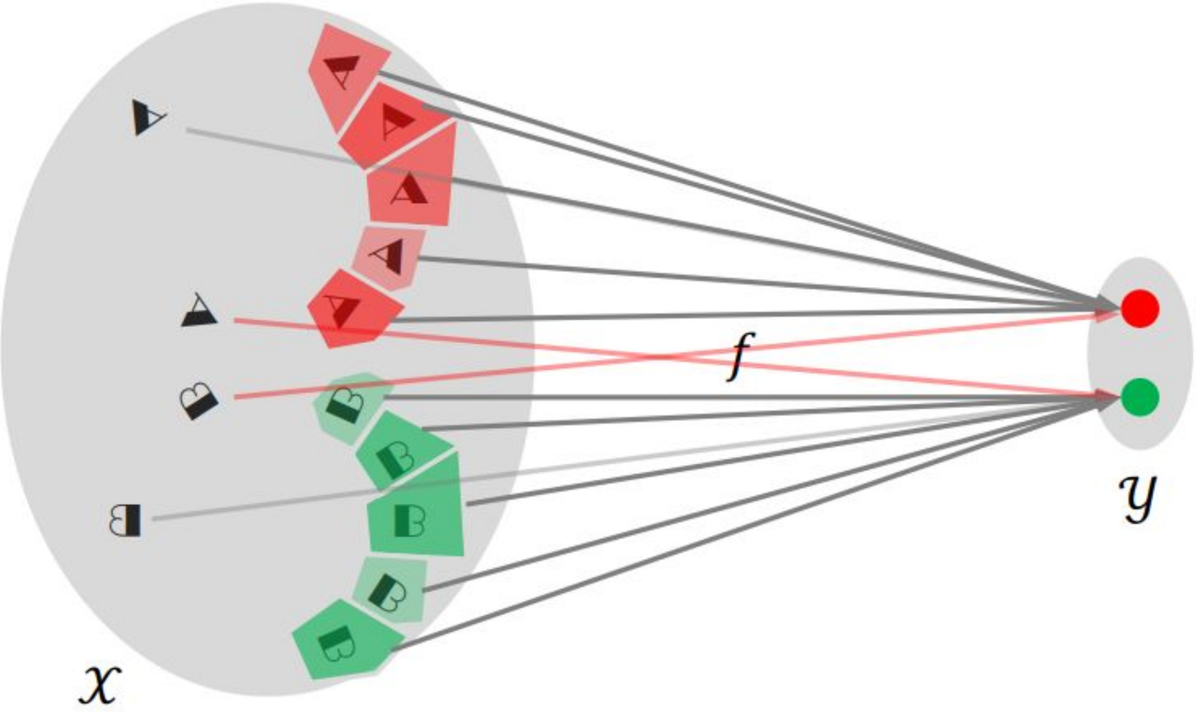
# Symmetry in Learning

Symmetries of the Label Function:

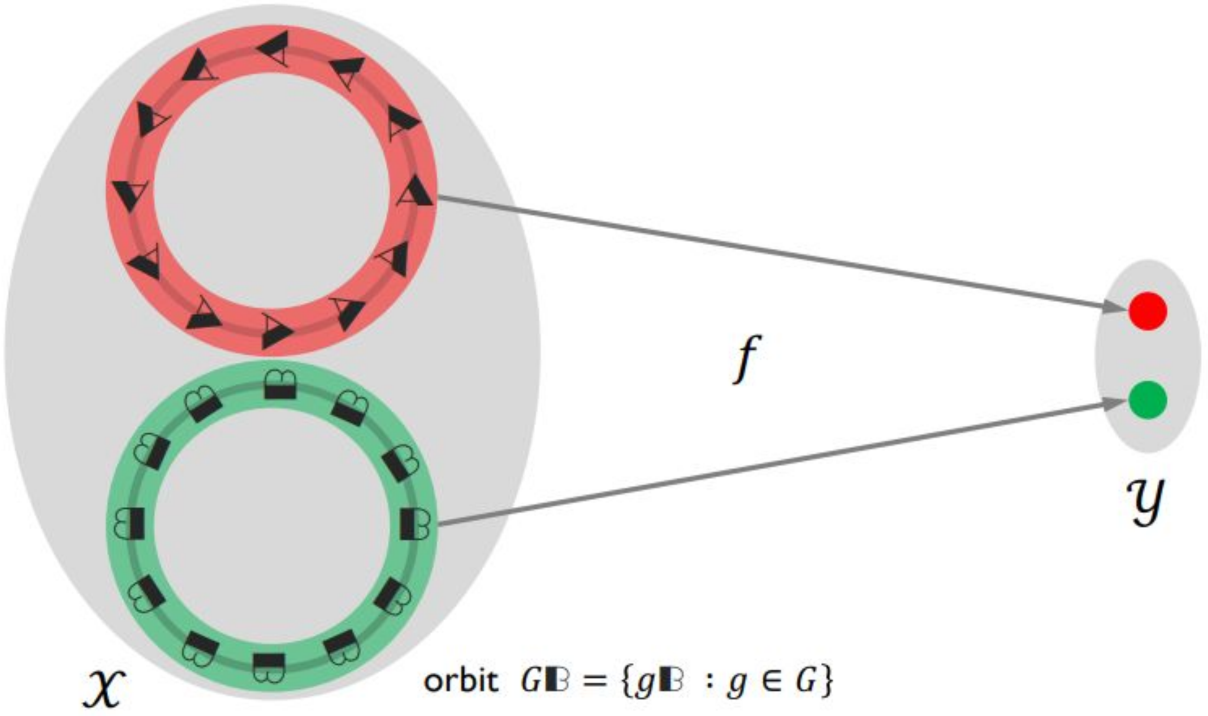
- Label function $f$: X → Y e.g., classification task (Y={1,...,K})
- Symmetry of a label function is an invertible label-preserving map $g$: X → X, i.e. $(f \circ g)(x) = f(x)$ for all $x \in X$



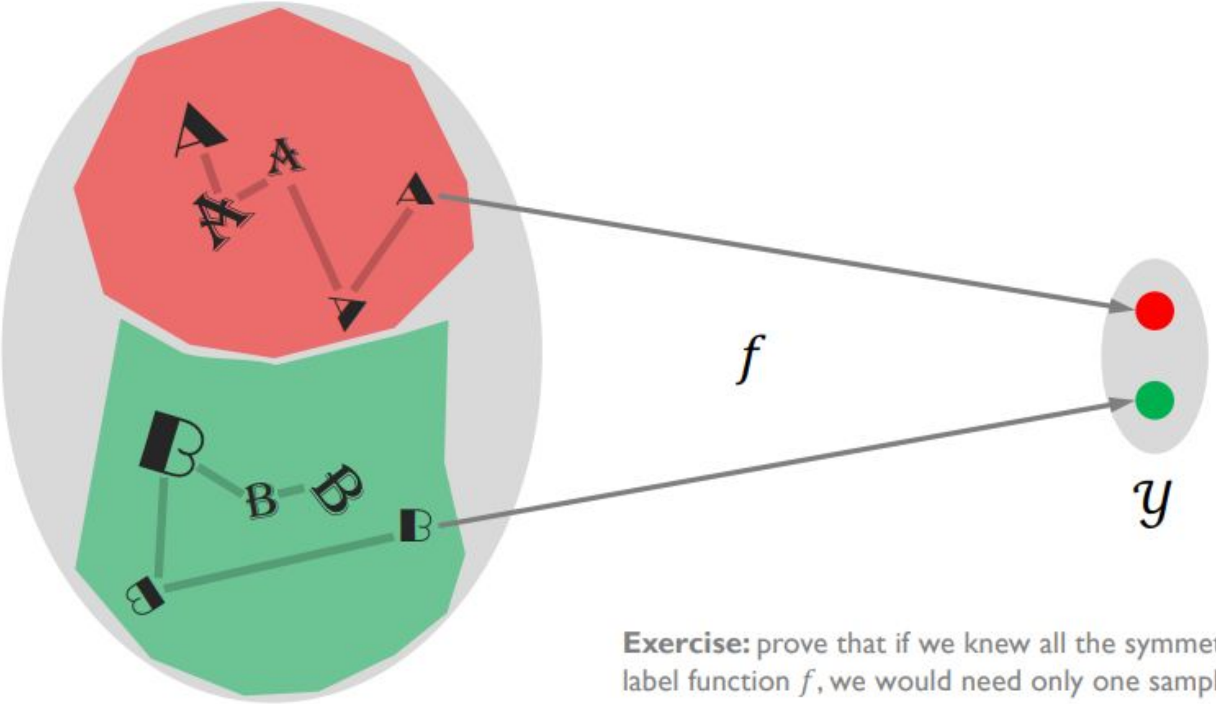
# Symmetries of the Label Function



# Symmetries of the Label Function



# Symmetries of the Label Function



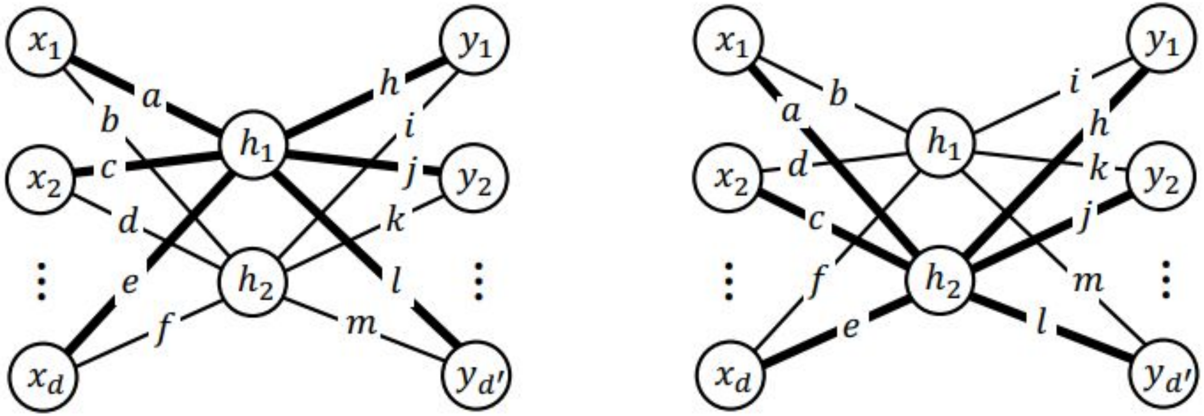

**Exercise:** prove that if we knew all the symmetries of the label function $f$, we would need only one sample per class

# Symmetries of the Weights

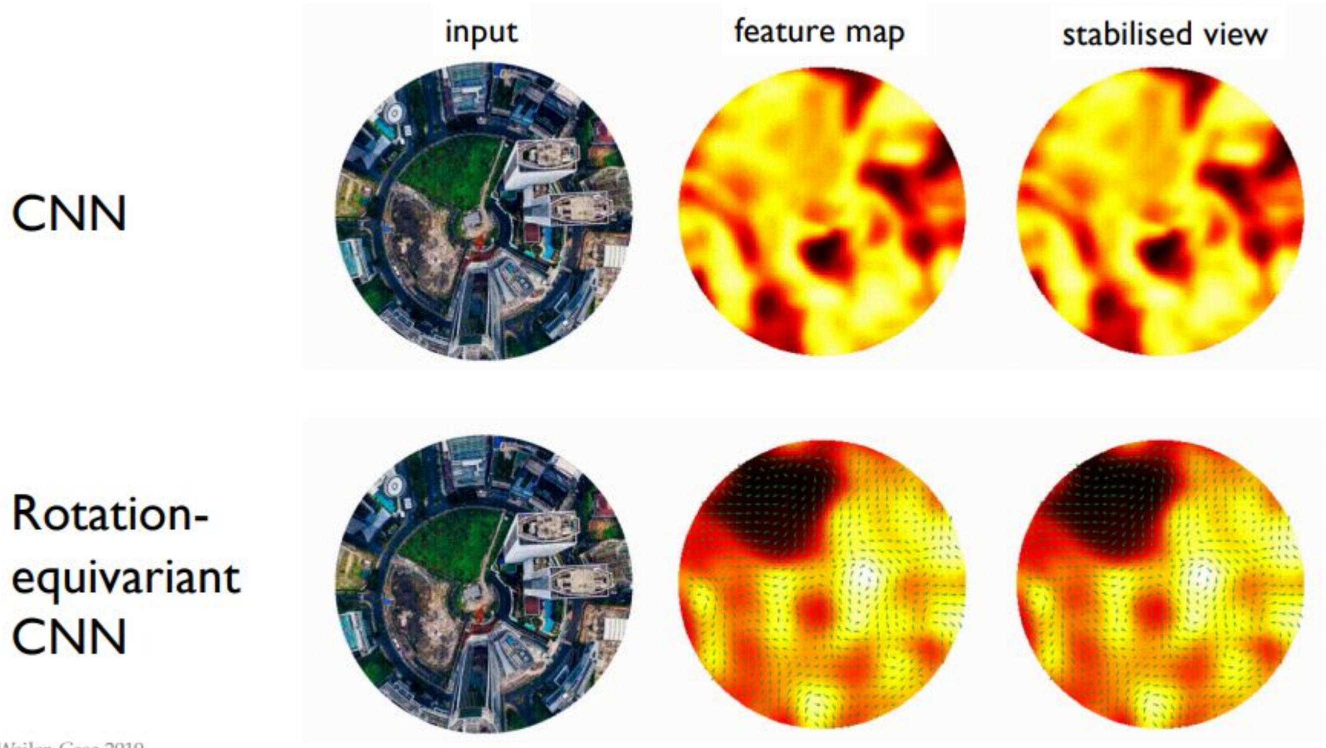
Let $f_\theta : \mathcal{X} \times \Theta \to \mathcal{Y}$ be a parametric model (neural network)

A transformation $h : \Theta \to \Theta$ is a **symmetry of the weights** if, for all $x \in \mathcal{X}$ and $\theta \in \Theta$

$$f_{h\theta}(x) = f_\theta(x)$$



# Example

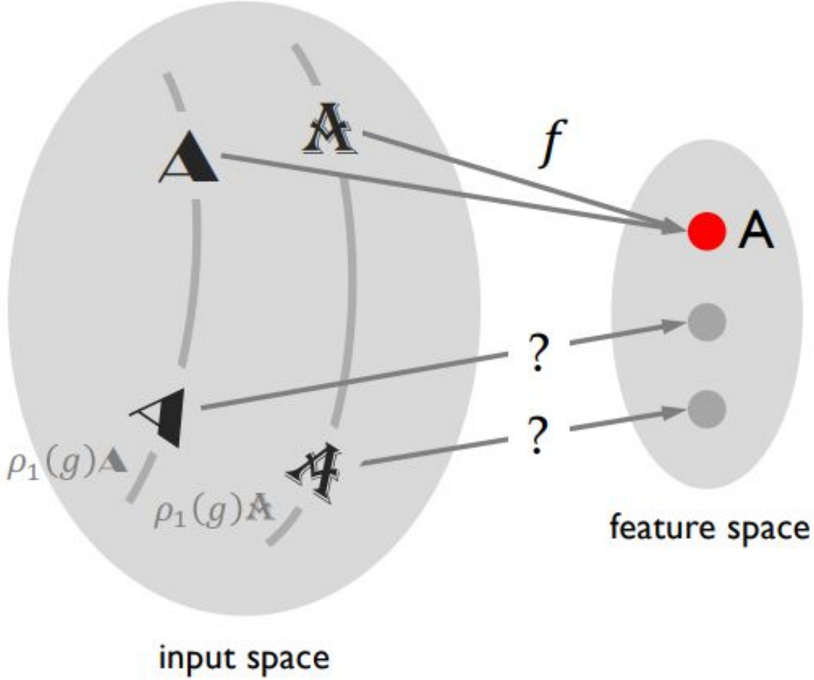



Weiler, Cesa 2019

# A Roadmap for Formalisation

1. To handle geometry of data, we need to formalise where the data lives (domain) and how to featurise it (signal)
2. Once we understand data domains, we can then formalise symmetries of those domains (groups)
3. Equipped with groups, we need to formalise how they transform the data domains (group actions)
4. Deep learning concerns itself with linear algebra; we need to be able to talk about group actions as matrix operations (representations)
5. Using representations, we can formalise what it means for a deep learning model to respect symmetries (invariance & equivariance)

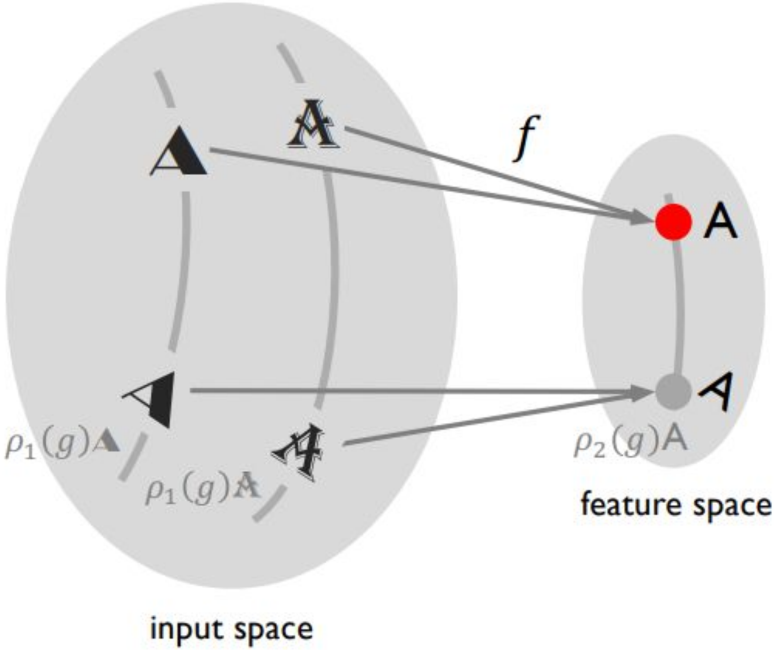
# Equivariance = Symmetry-consistent Generalisation




$$f(\rho_1(g)\mathbf{A}) = \rho_2(g)\, f(\mathbf{A})$$

$$f(\rho_1(g)\mathbf{A}) = \rho_2(g)\, f(\mathbf{A})$$

# Equivariance = Symmetry-consistent Generalisation

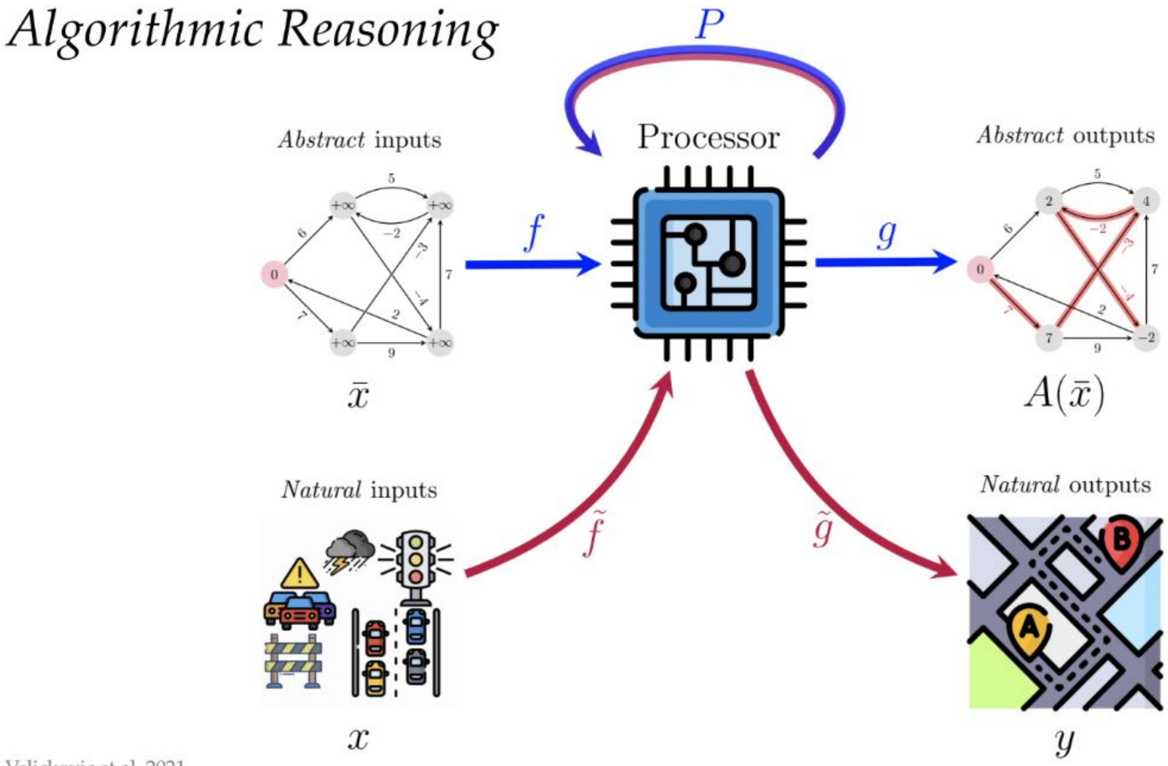



$$f(\rho_1(g)\mathbf{A}) = \rho_2(g)\, f(\mathbf{A})$$

$$\|$$

$$f(\rho_1(g)\mathbf{A}) = \rho_2(g)\, f(\mathbf{A})$$

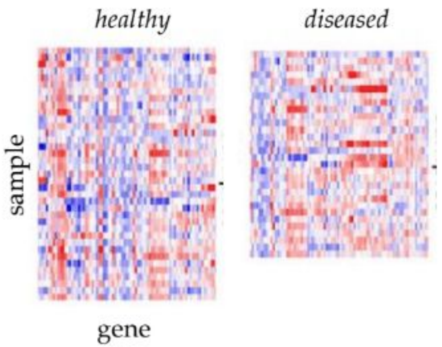
# What's next?



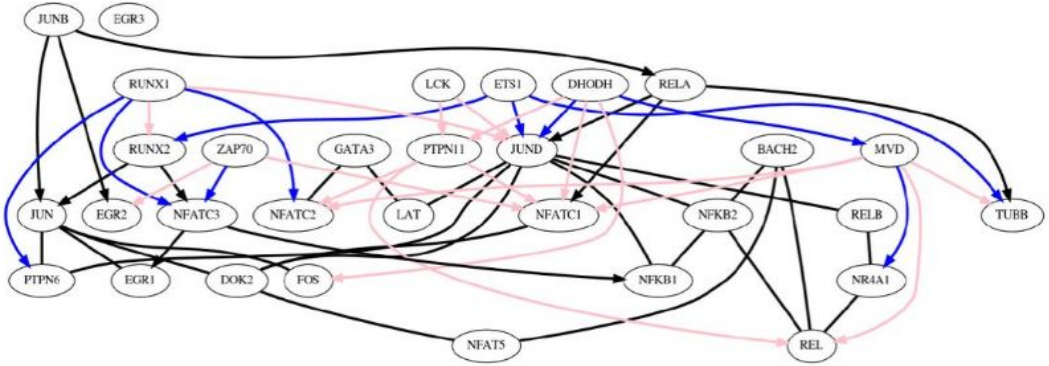

Velickovic et al. 2021

# What’s next?

*Causal Inference*




Gene expressions

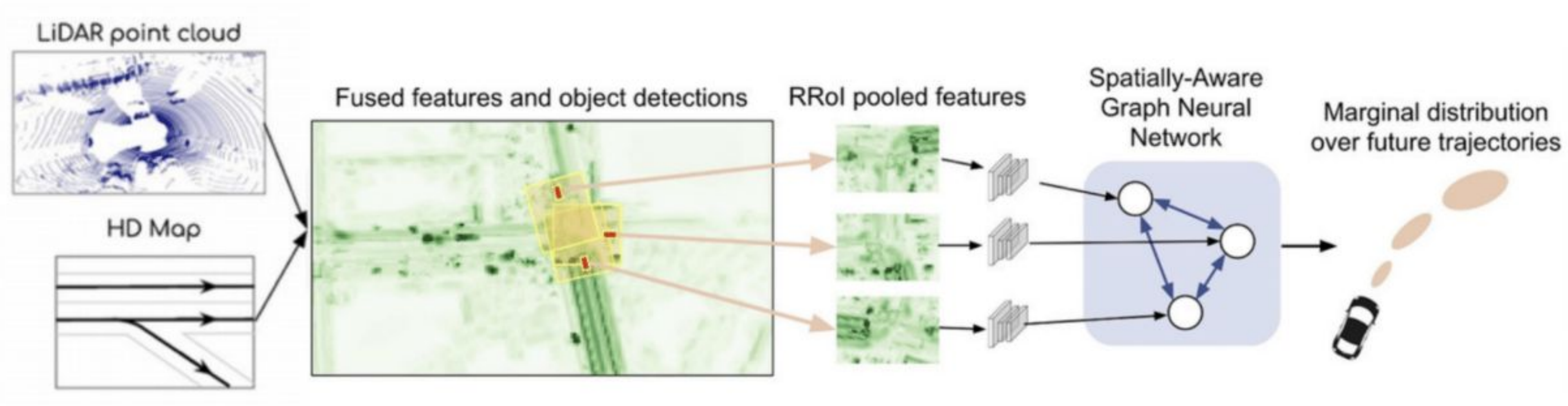



Difference gene regulatory network

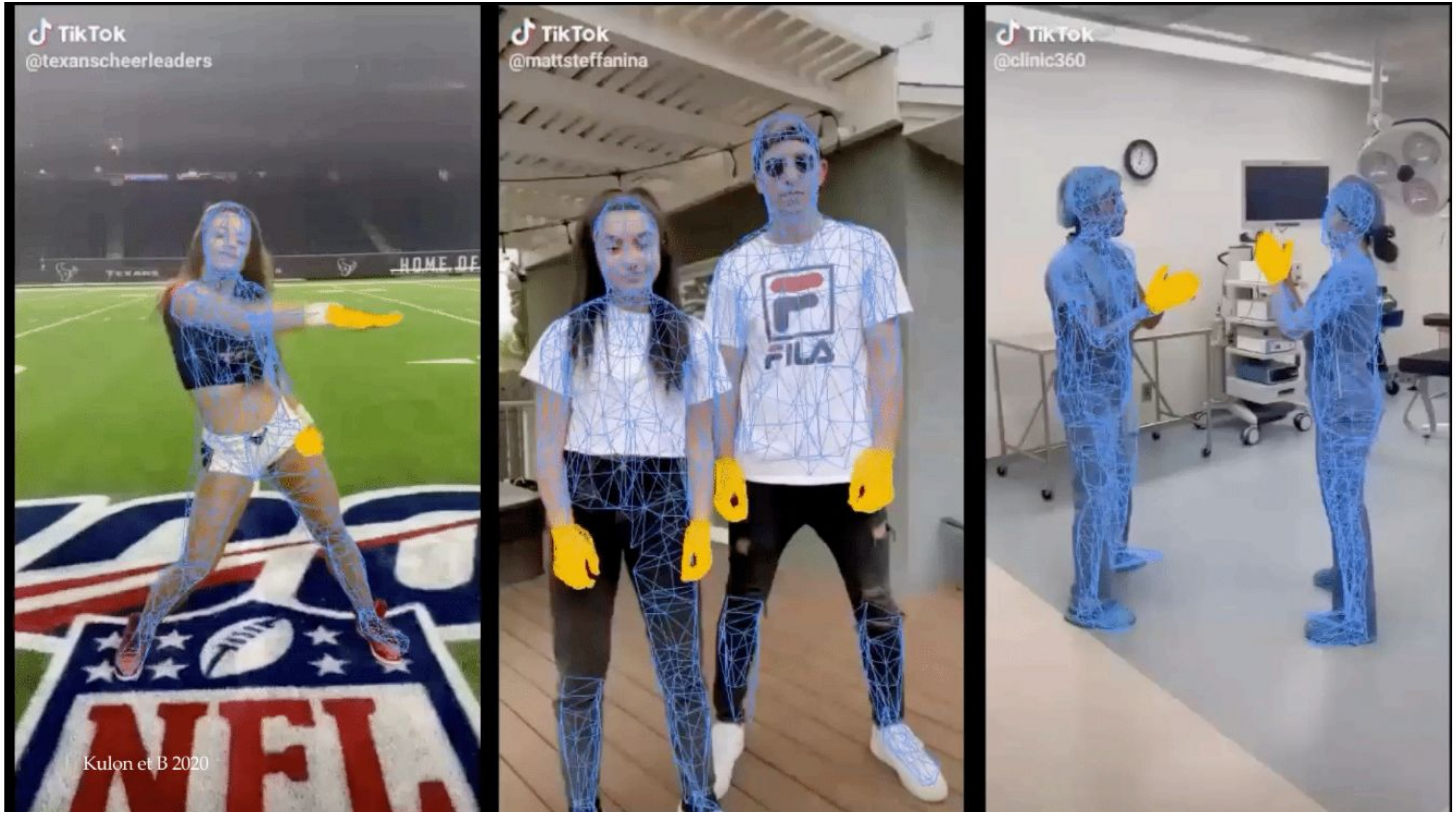
# Social Networks

# 3D VISION & GRAPHICS

## *Self-Driving Cars*

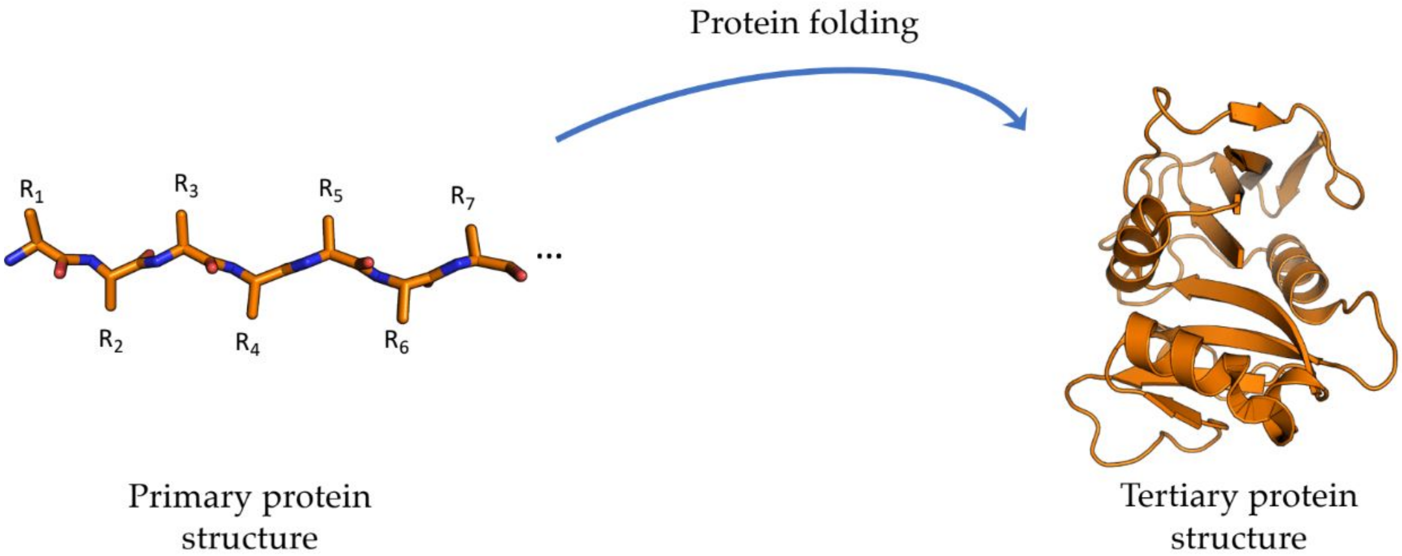


# 3D VISION & GRAPHICS



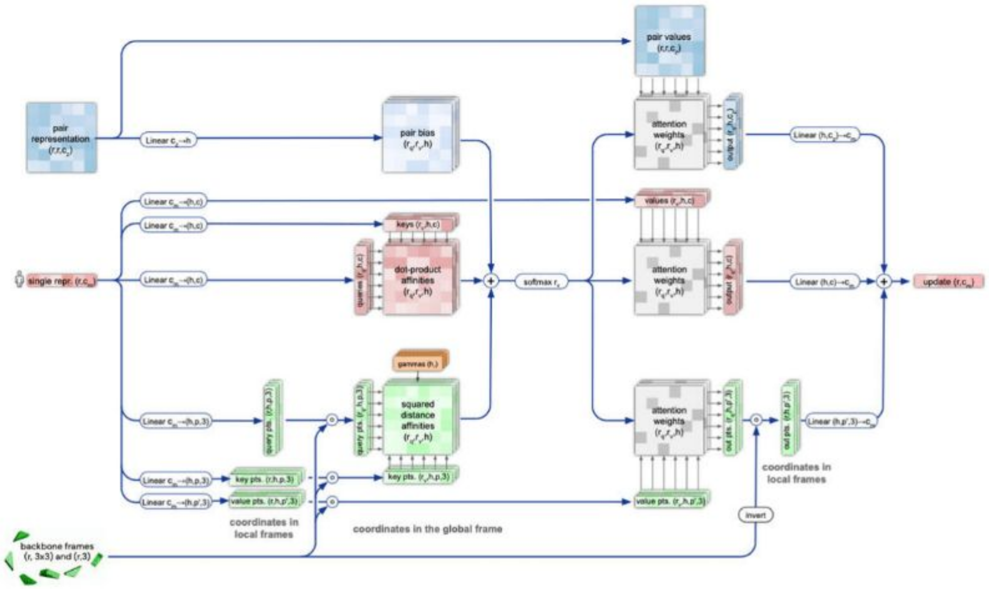
# STRUCTURAL BIOLOGY

*Protein Folding*



# STRUCTURAL BIOLOGY

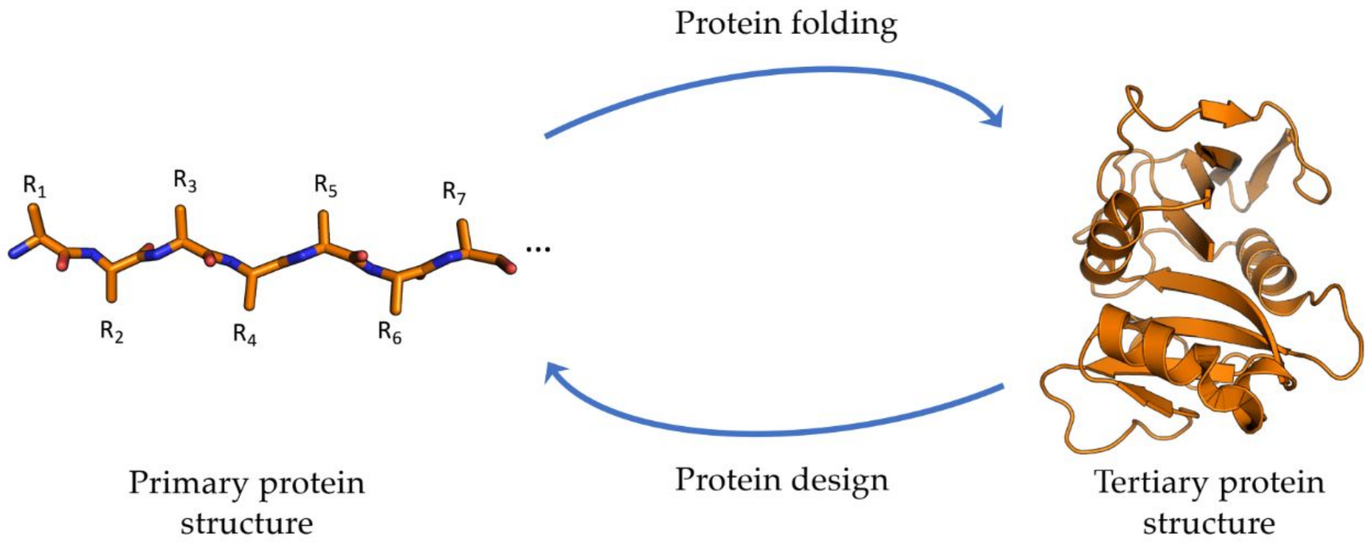
*AlphaFold 2*




Invariant Point Attention

# STRUCTURAL BIOLOGY

*Protein Design = “Inverse Folding”*



# STRUCTURAL BIOLOGY

Sequence ➡ Structure ➡ Function

*MaSIF: Protein Function Prediction*