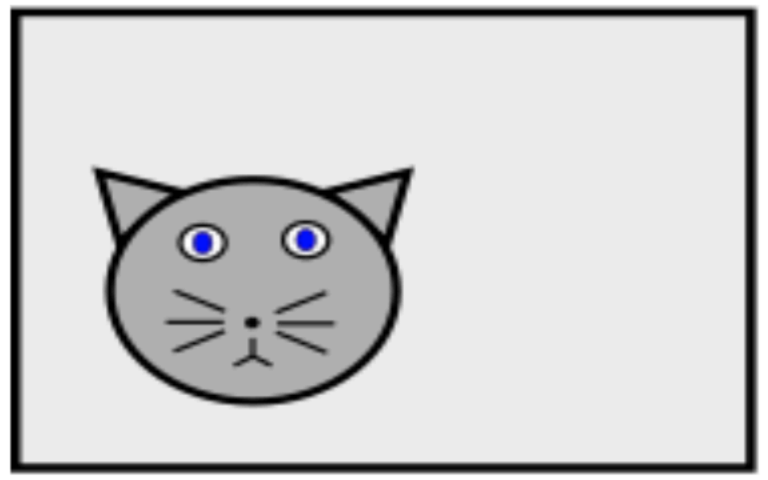
# What is the future of equivariant learning?

Hannah Lawrence, MIT

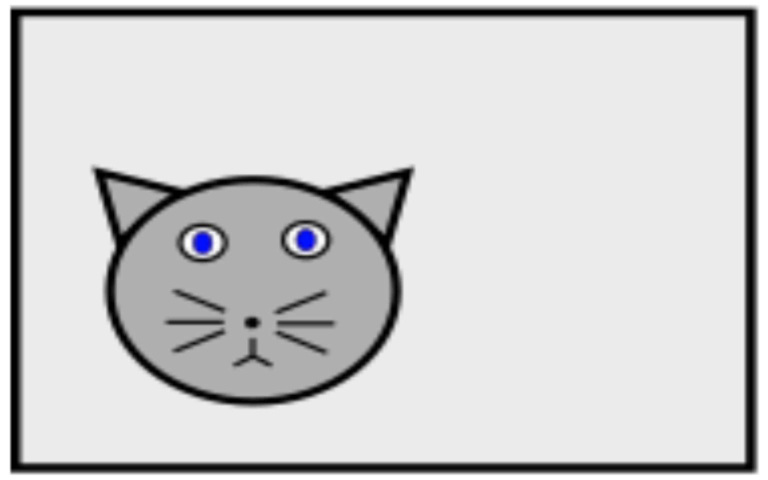
NVIDIA GenAIR Seminar

# What is equivariant learning?

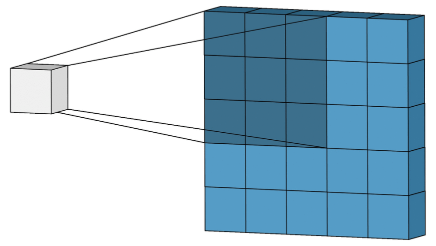
# Convolutions “make sense” for images



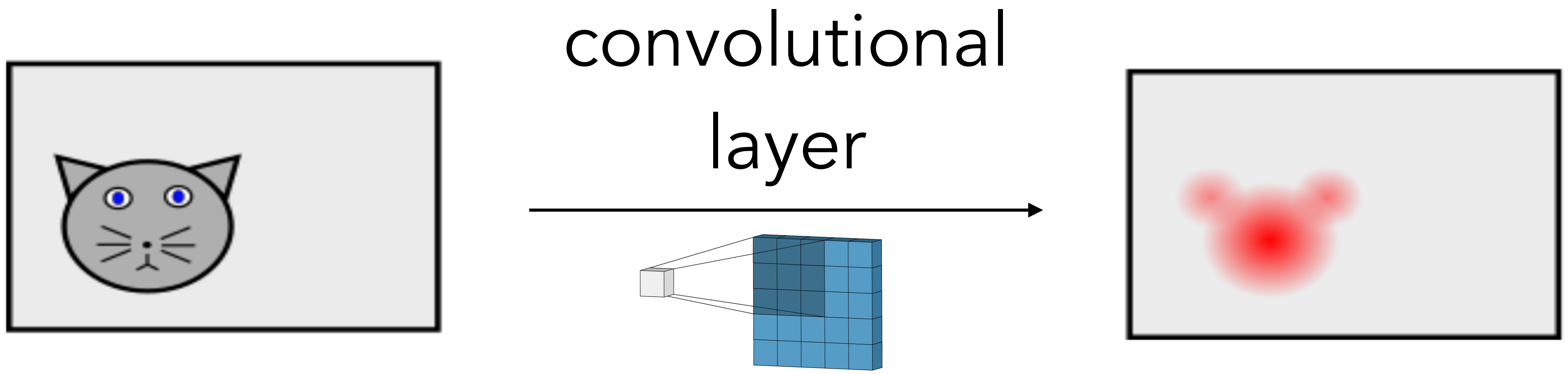
# Convolutions "make sense" for images



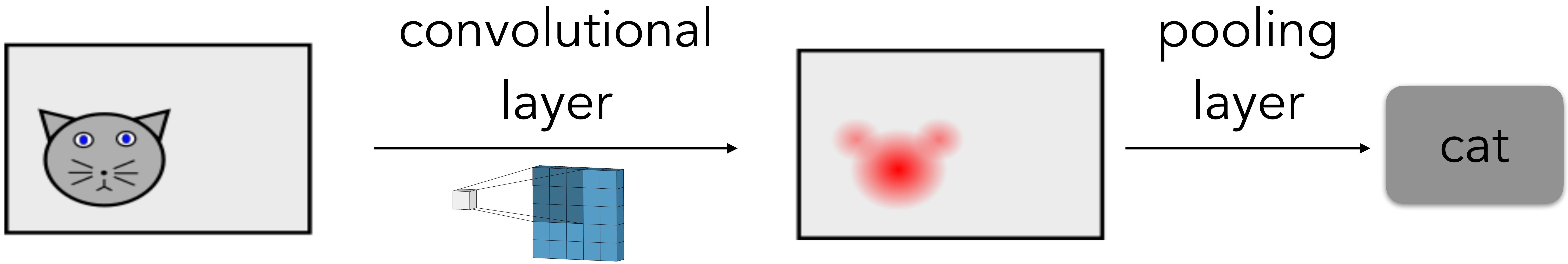
convolutional
layer



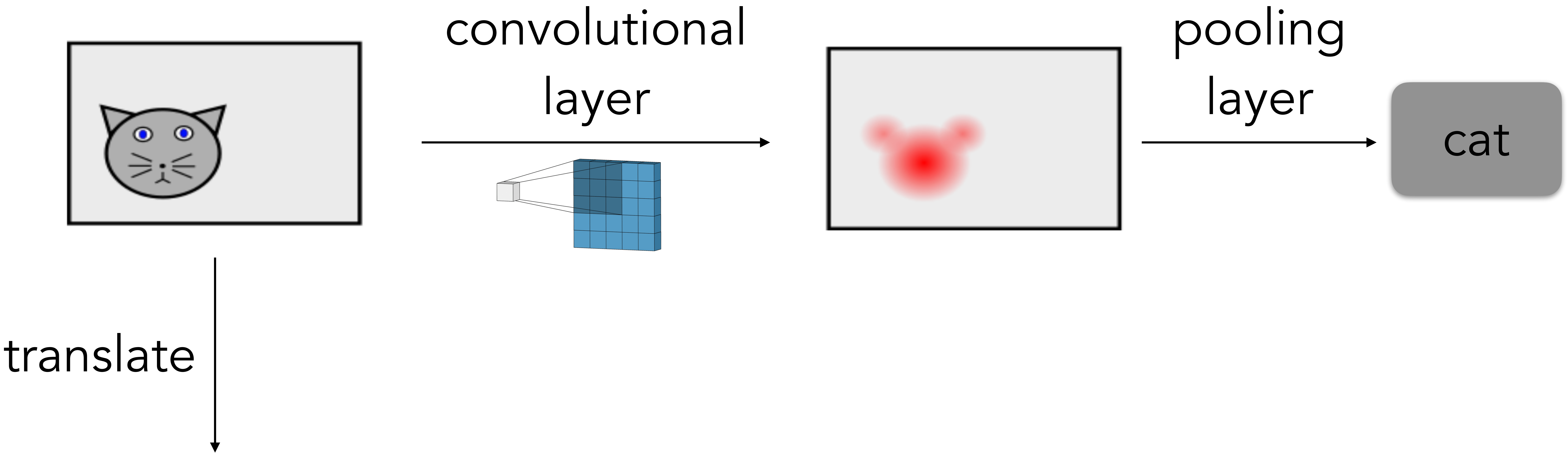
# Convolutions “make sense” for images



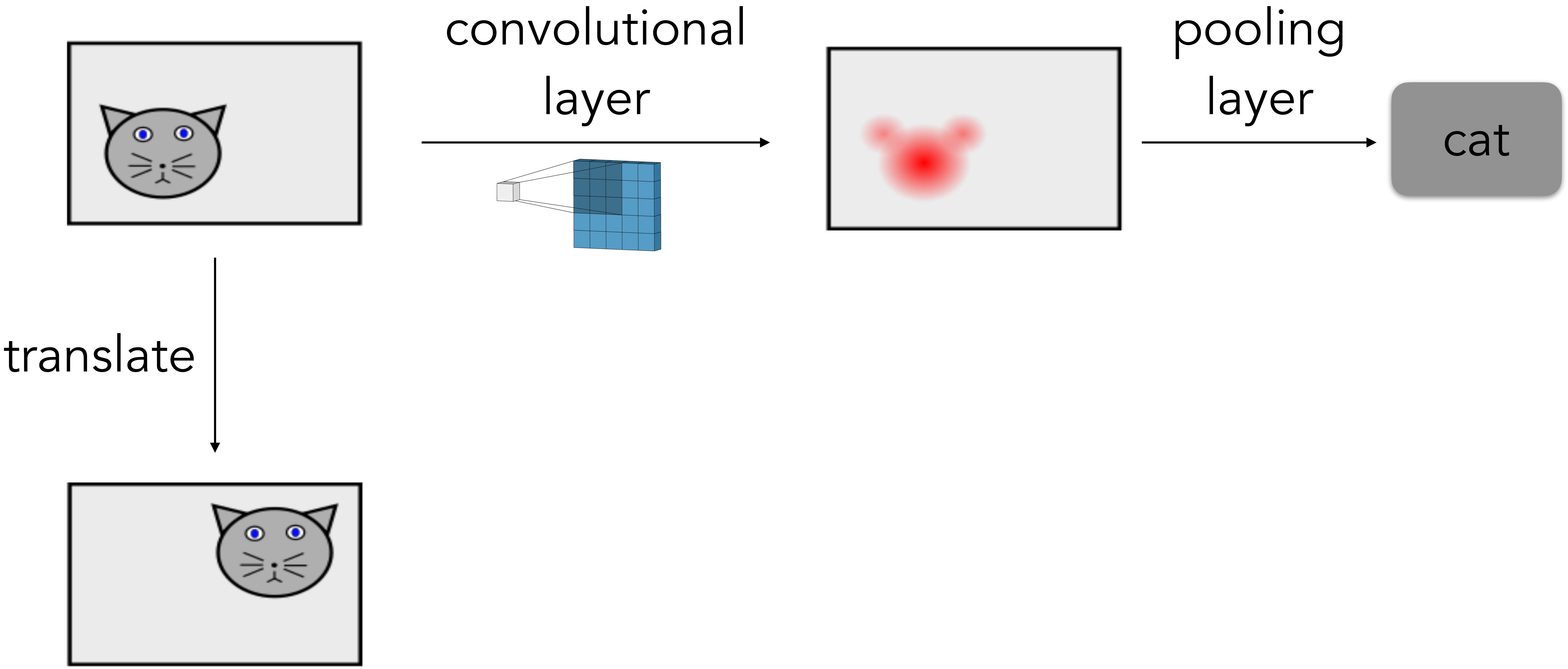
# Convolutions "make sense" for images



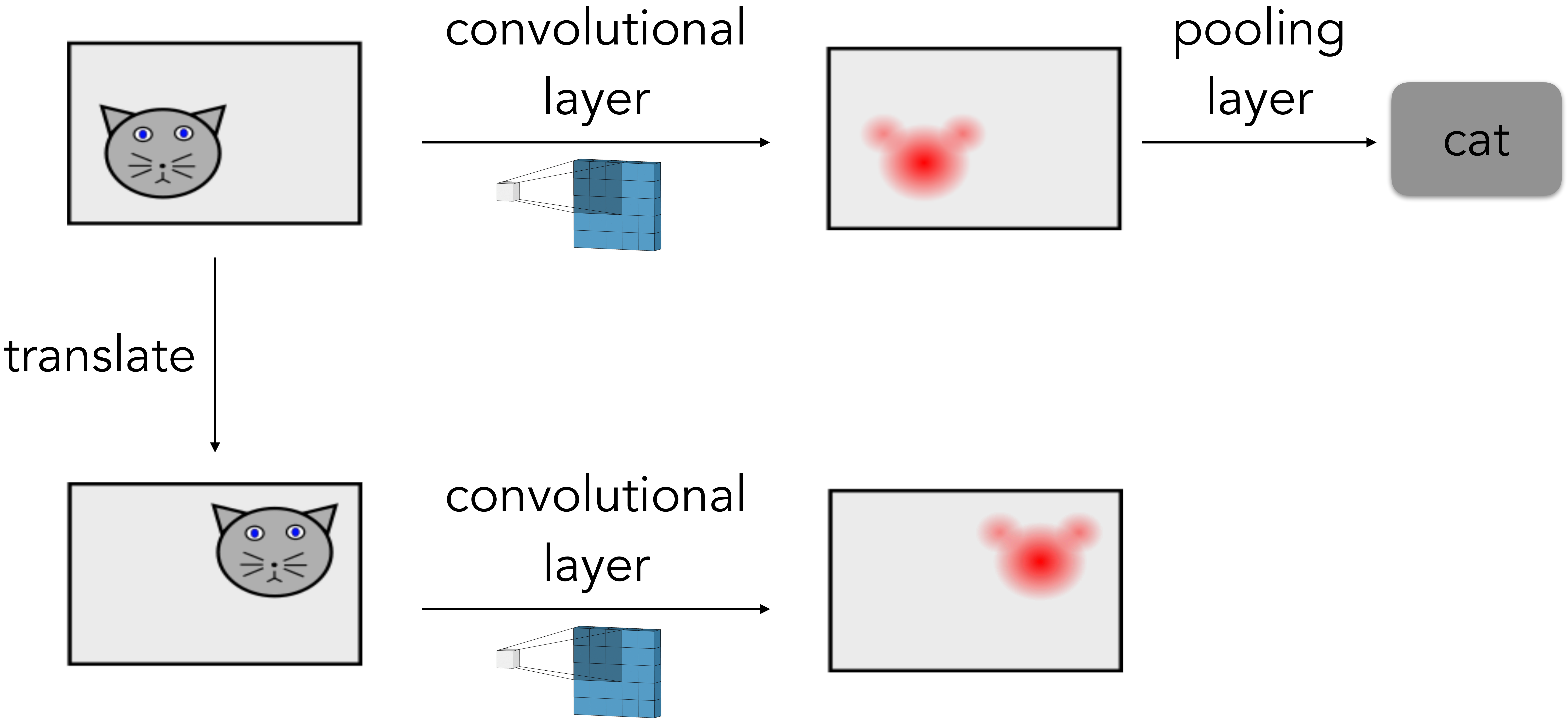
# Convolutions “make sense” for images



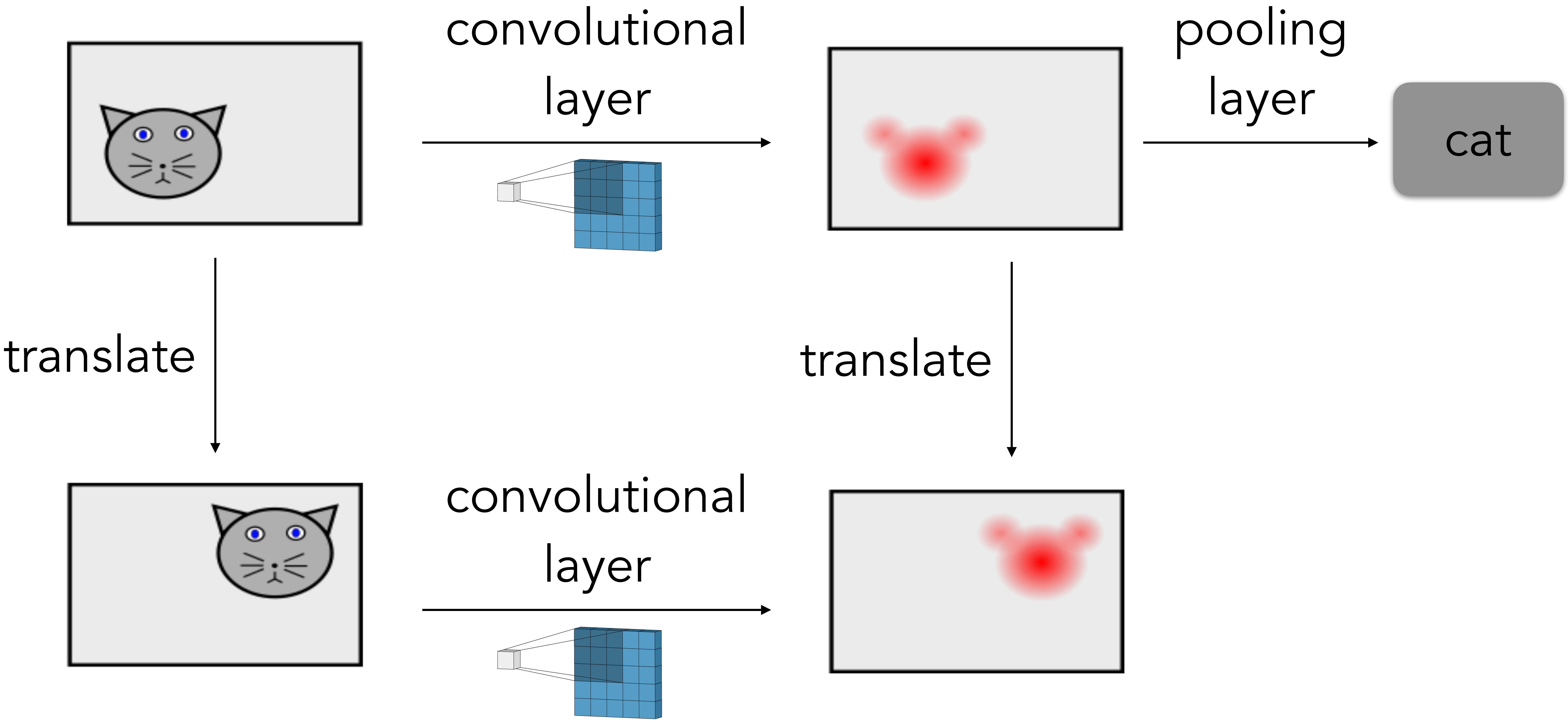
# Convolutions “make sense” for images



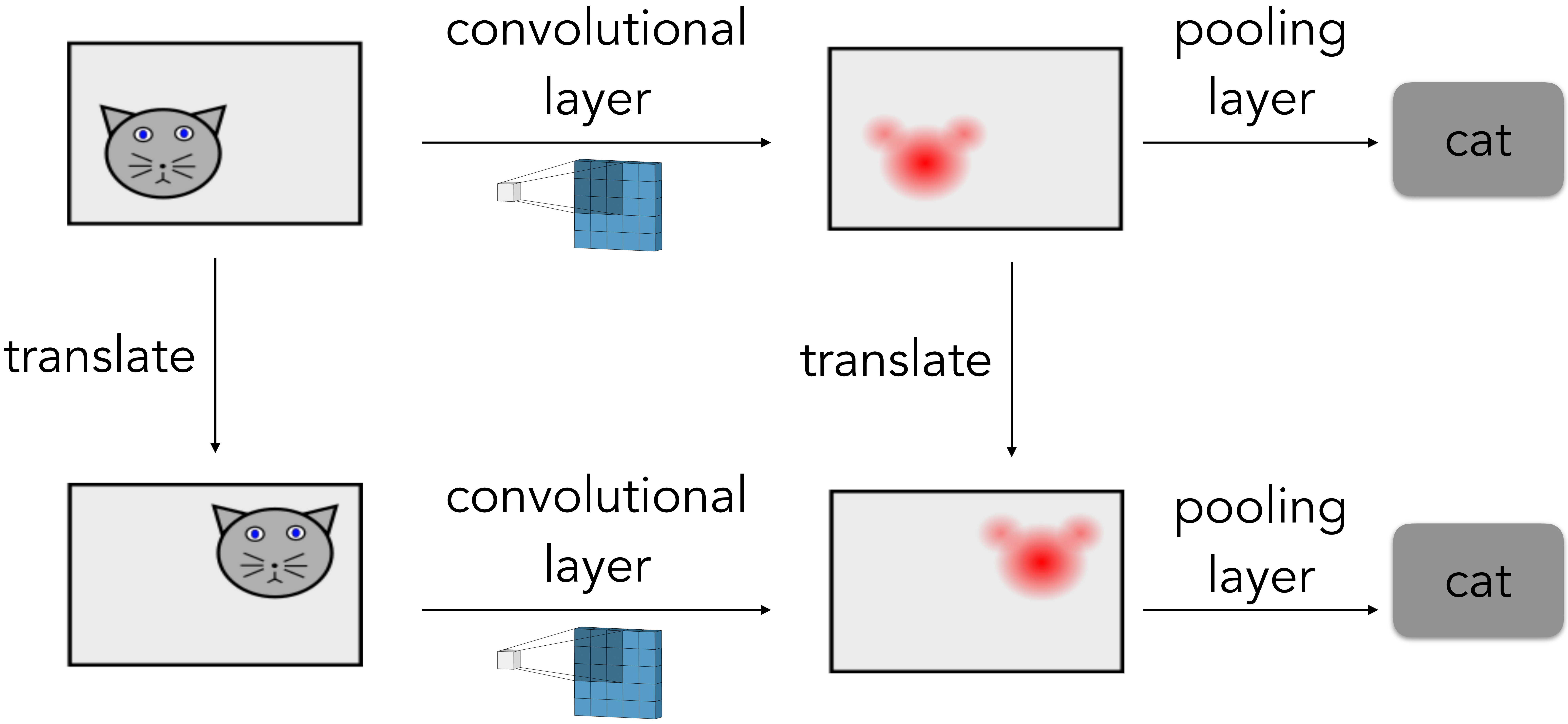
# Convolutions "make sense" for images



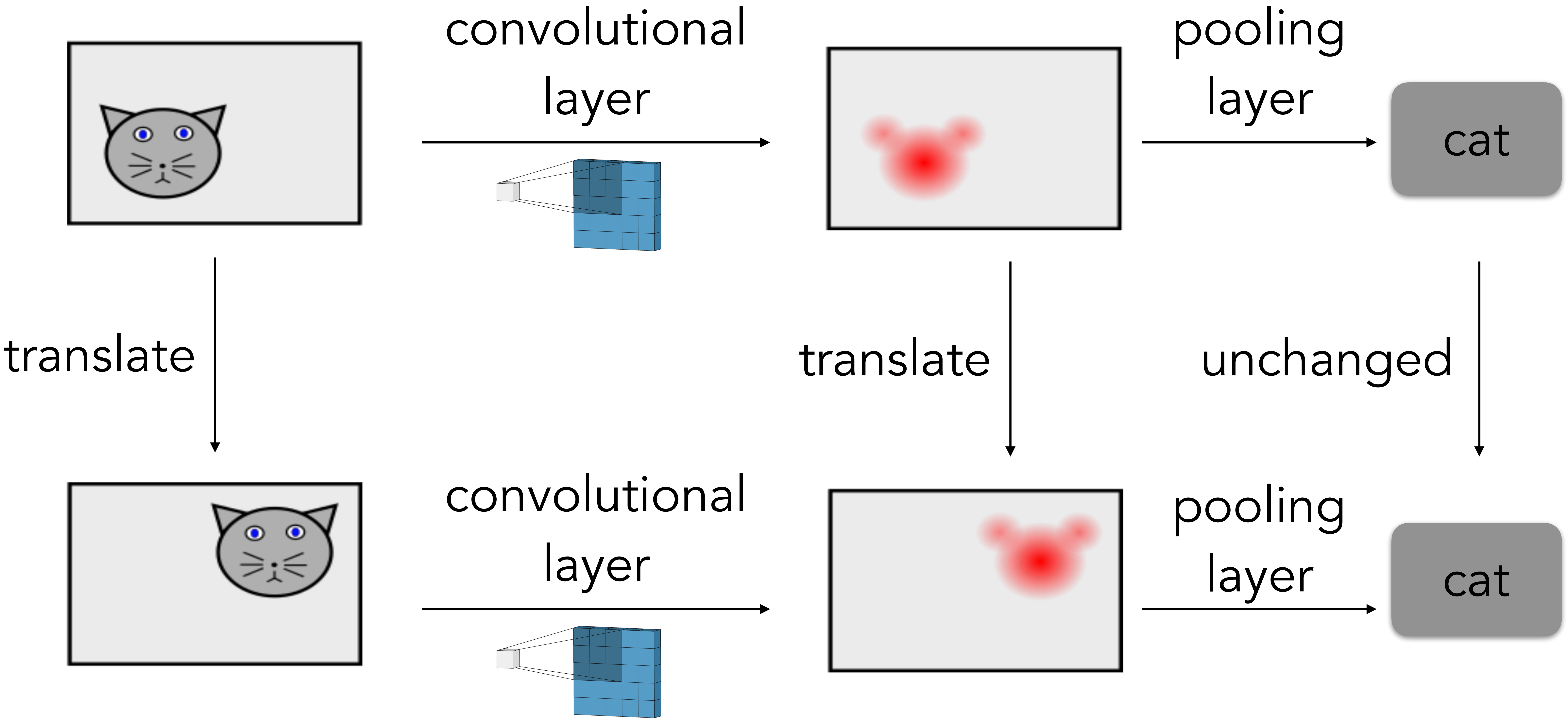
# Convolutions "make sense" for images



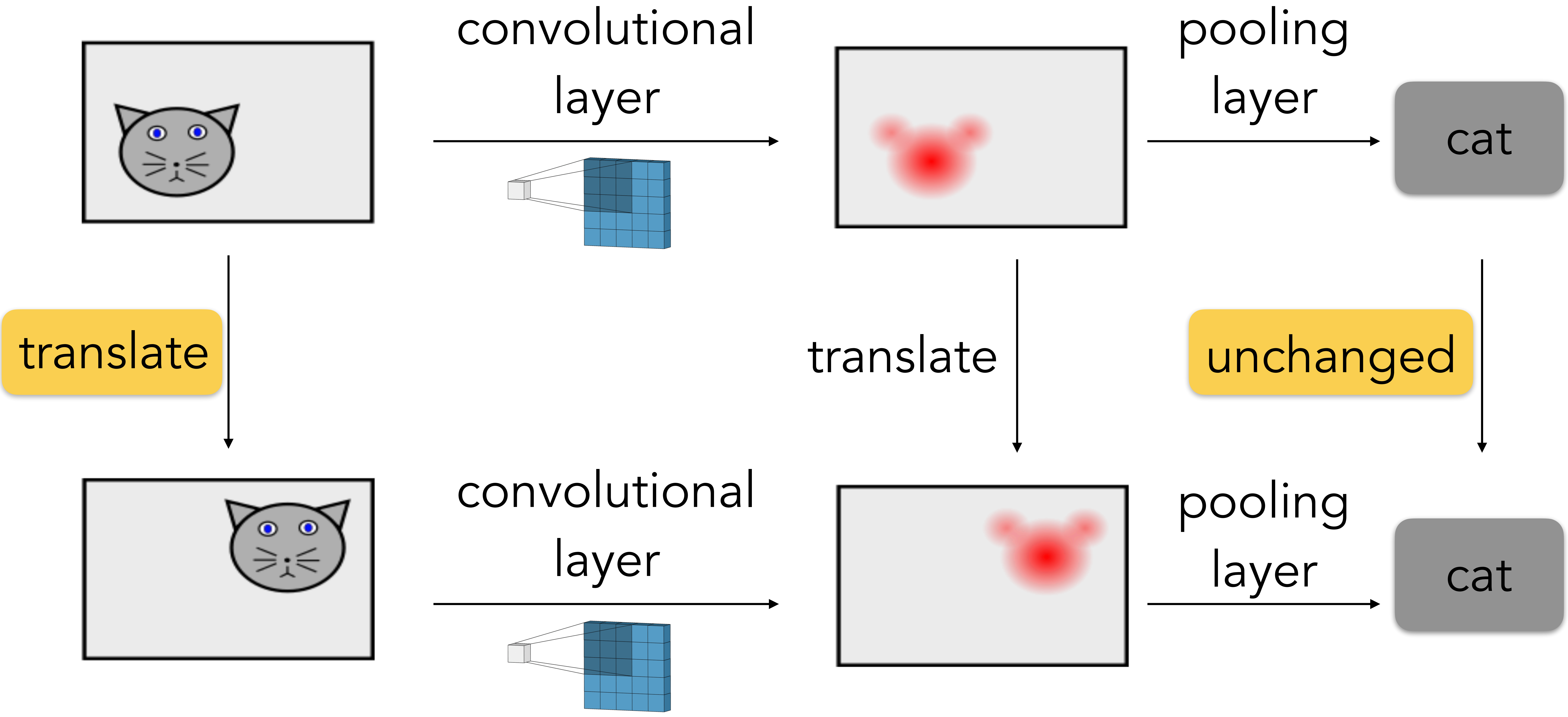
# Convolutions “make sense” for images



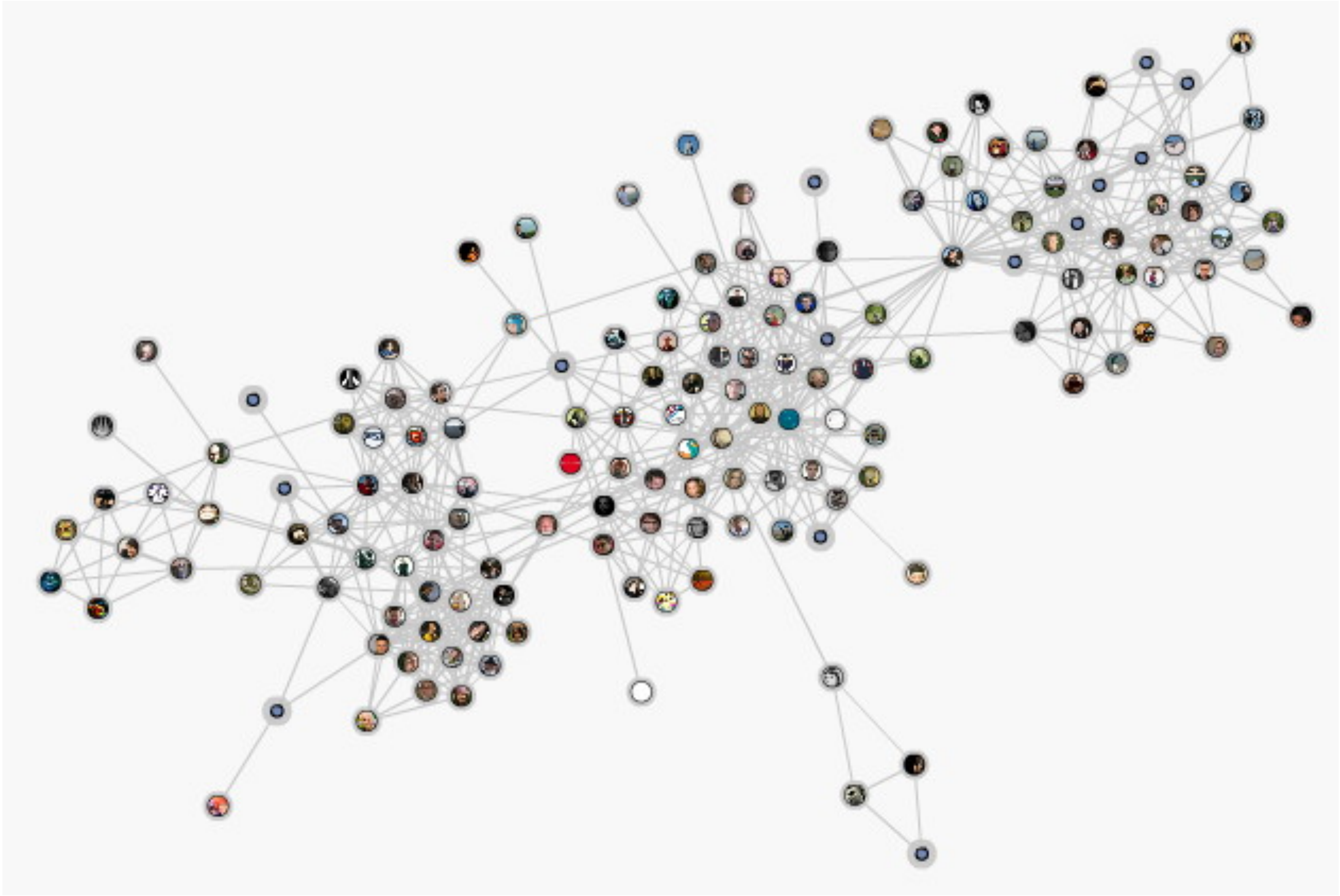
# Convolutions “make sense” for images



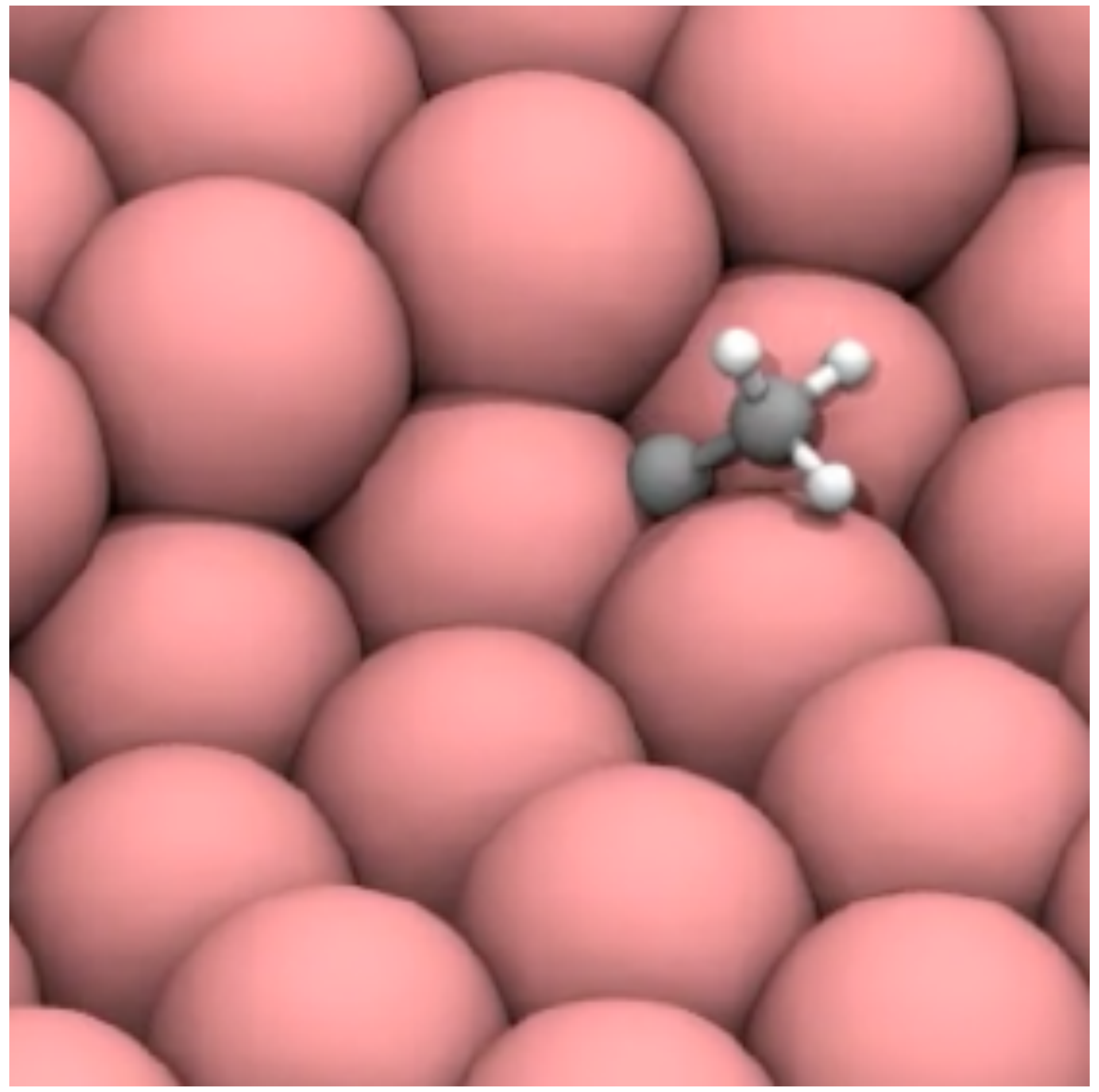
# Convolutions “make sense” for images



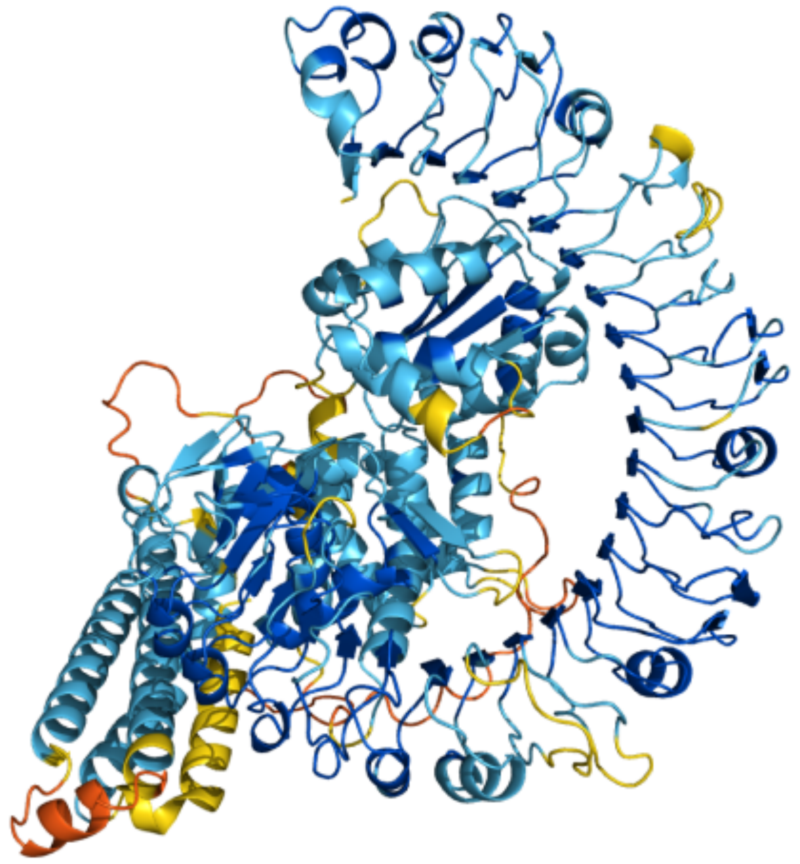
# But, not all data is images + text

# AI for Science: the next frontier

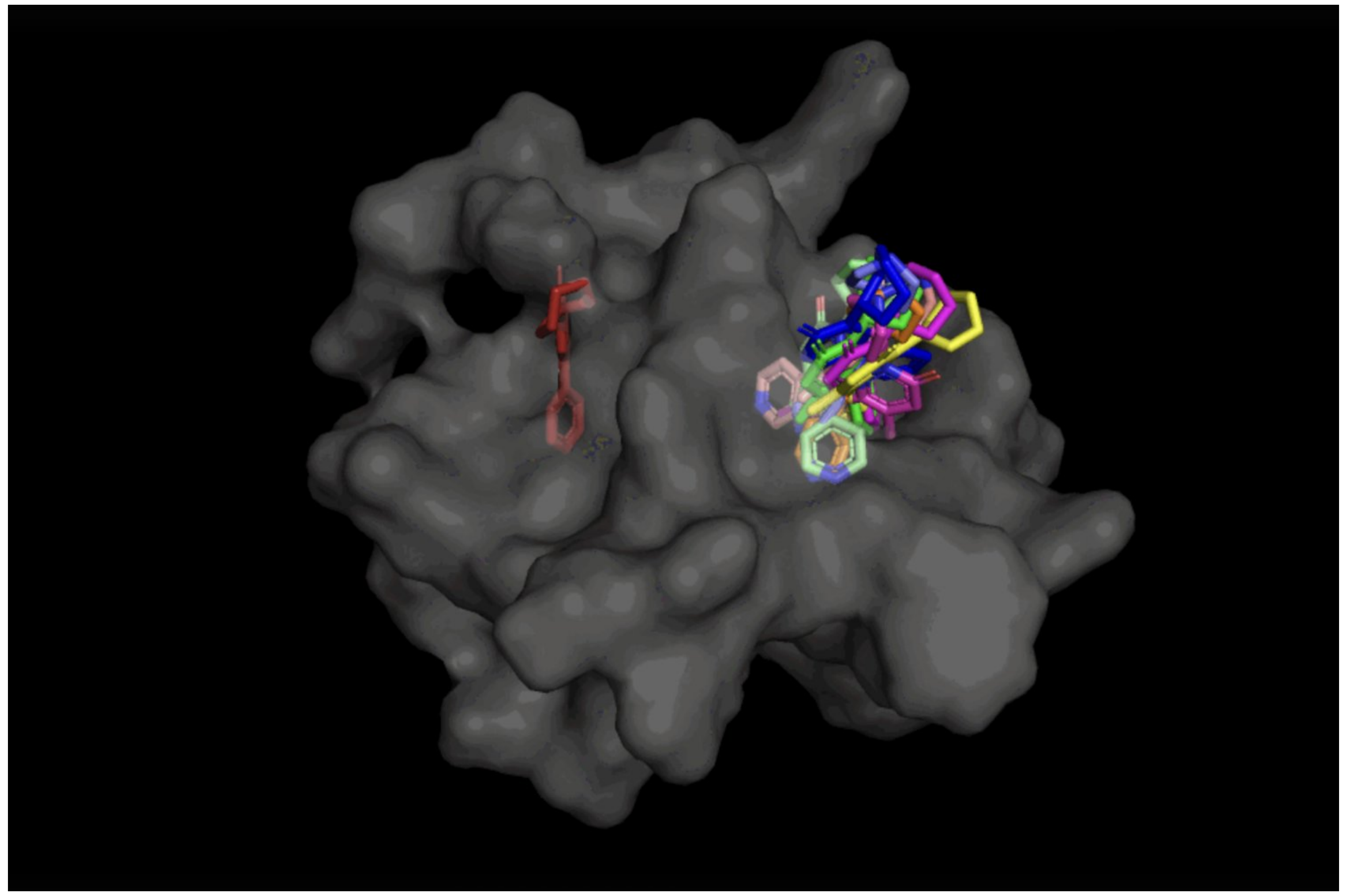


Materials



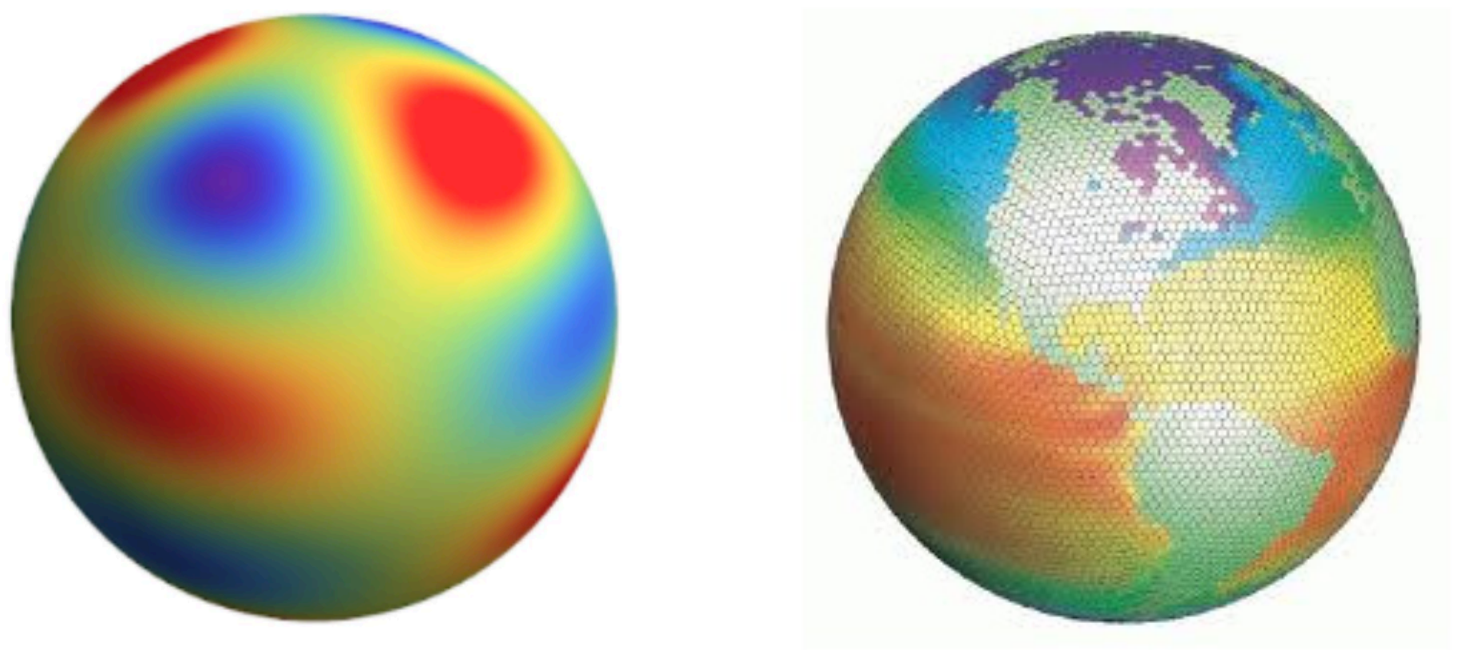
Protein folding

Climate

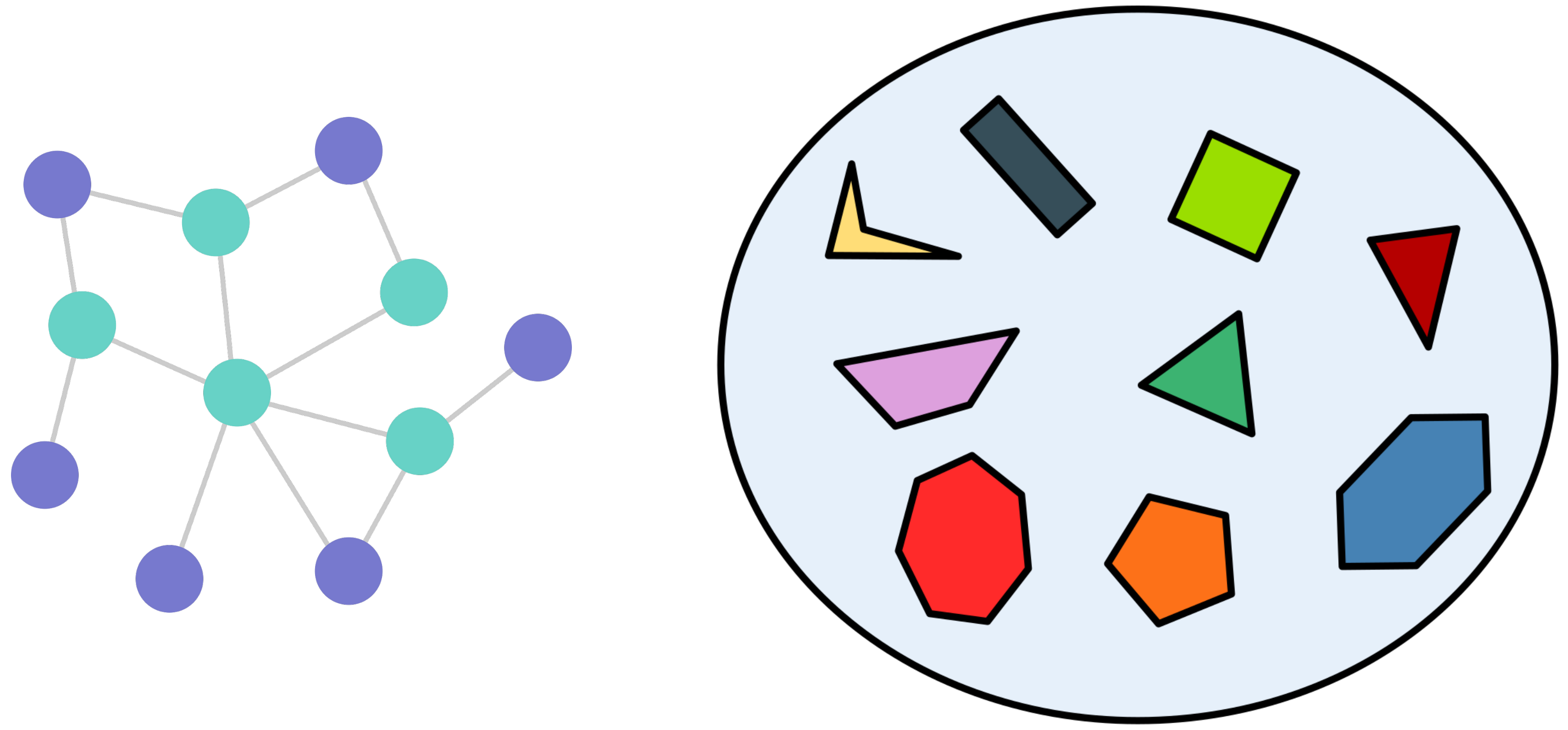


Drug discovery

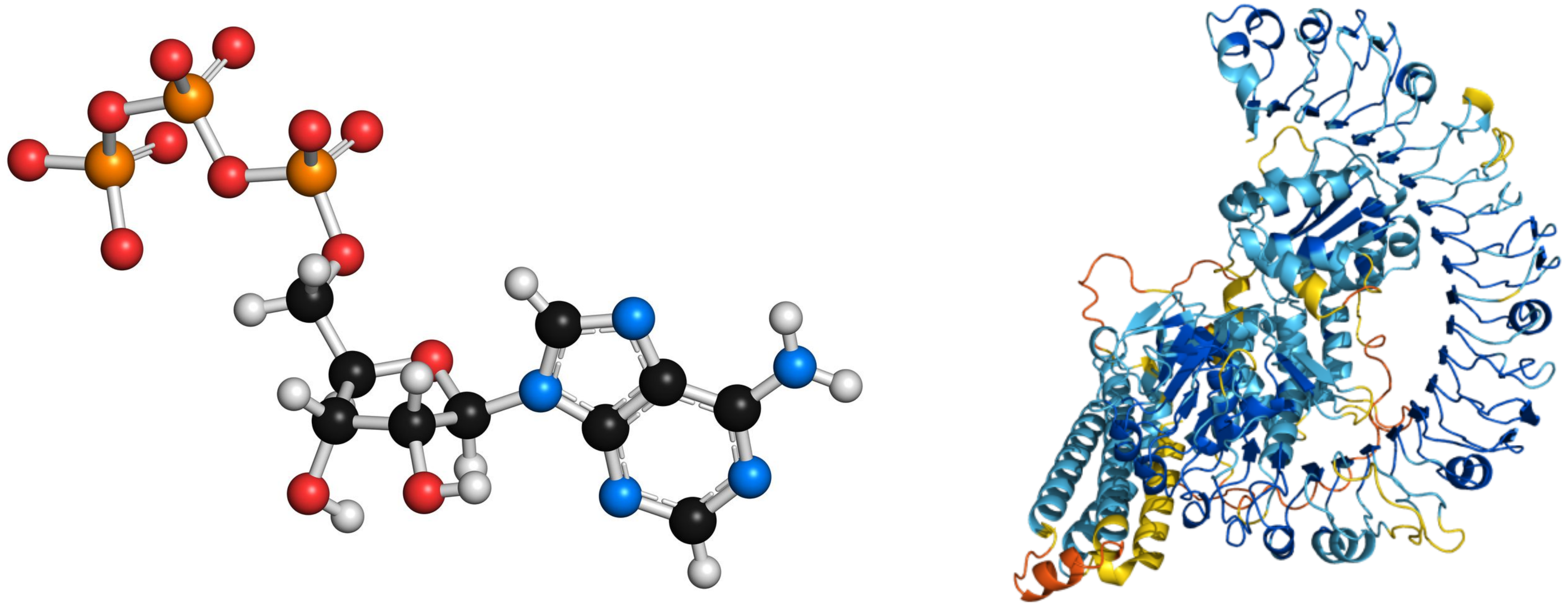
# New, geometric data types



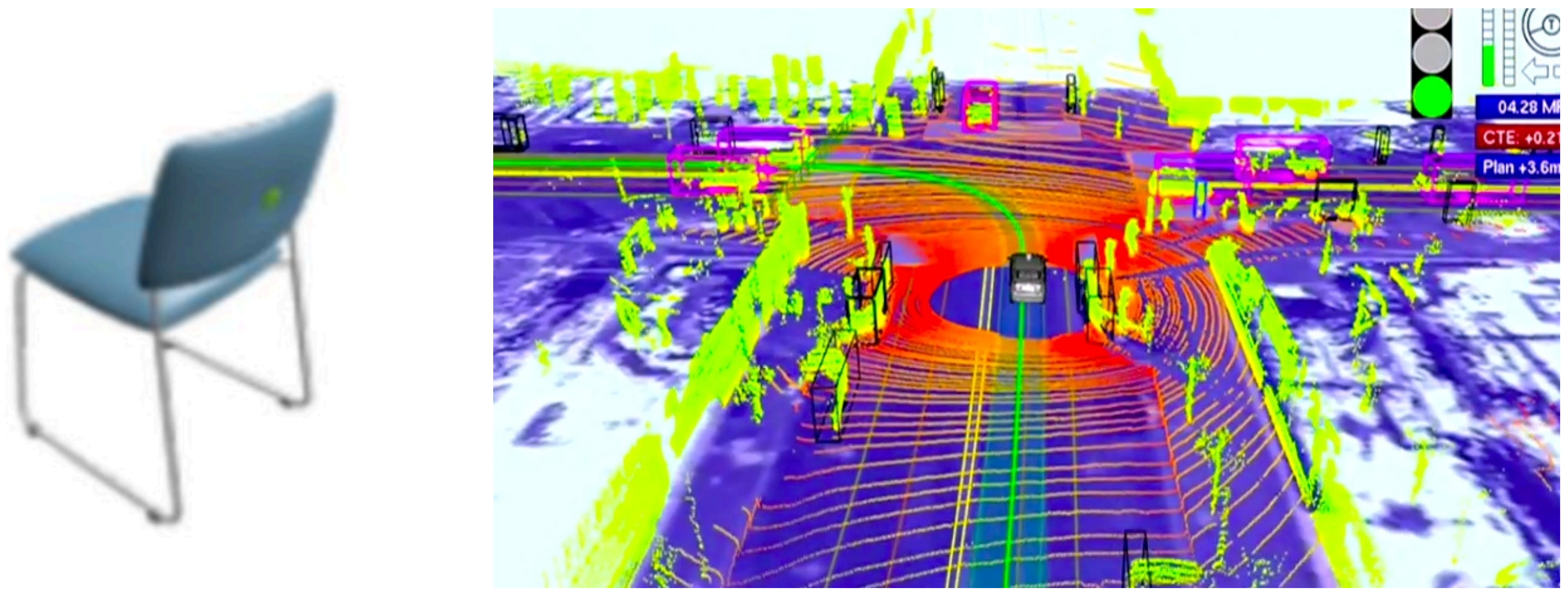
Spherical functions



Graphs and sets

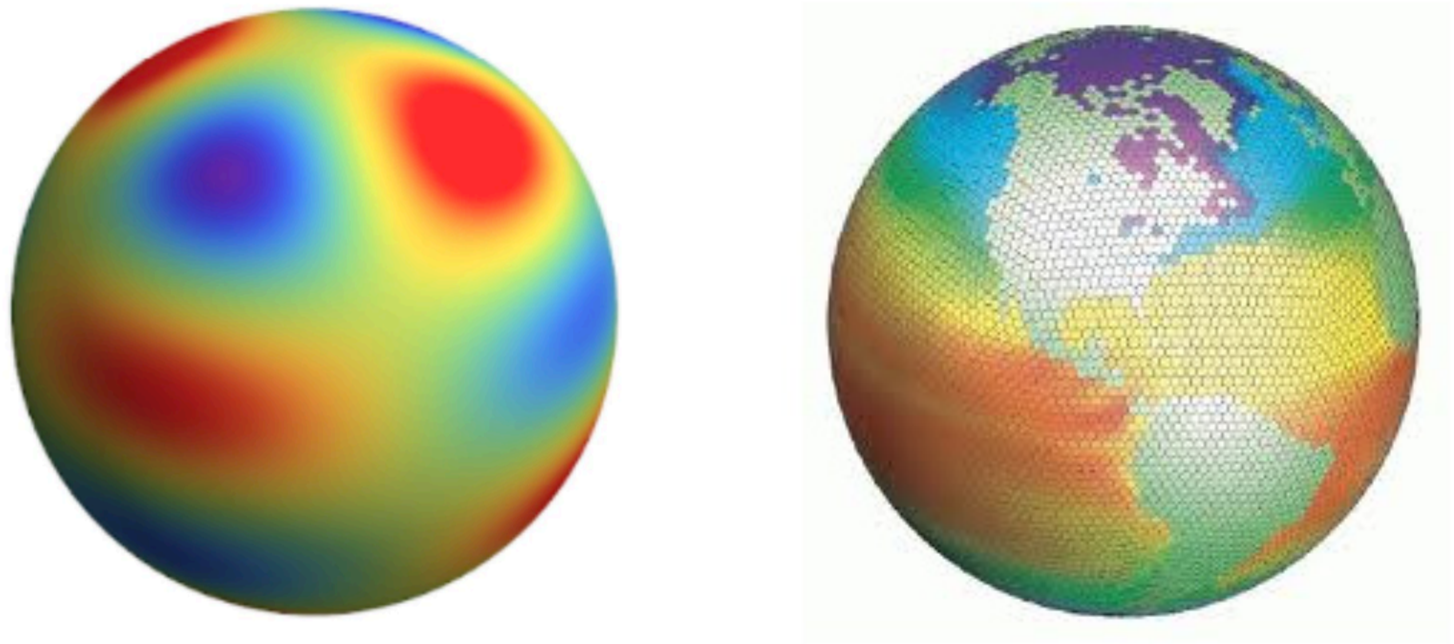


Small molecules and proteins

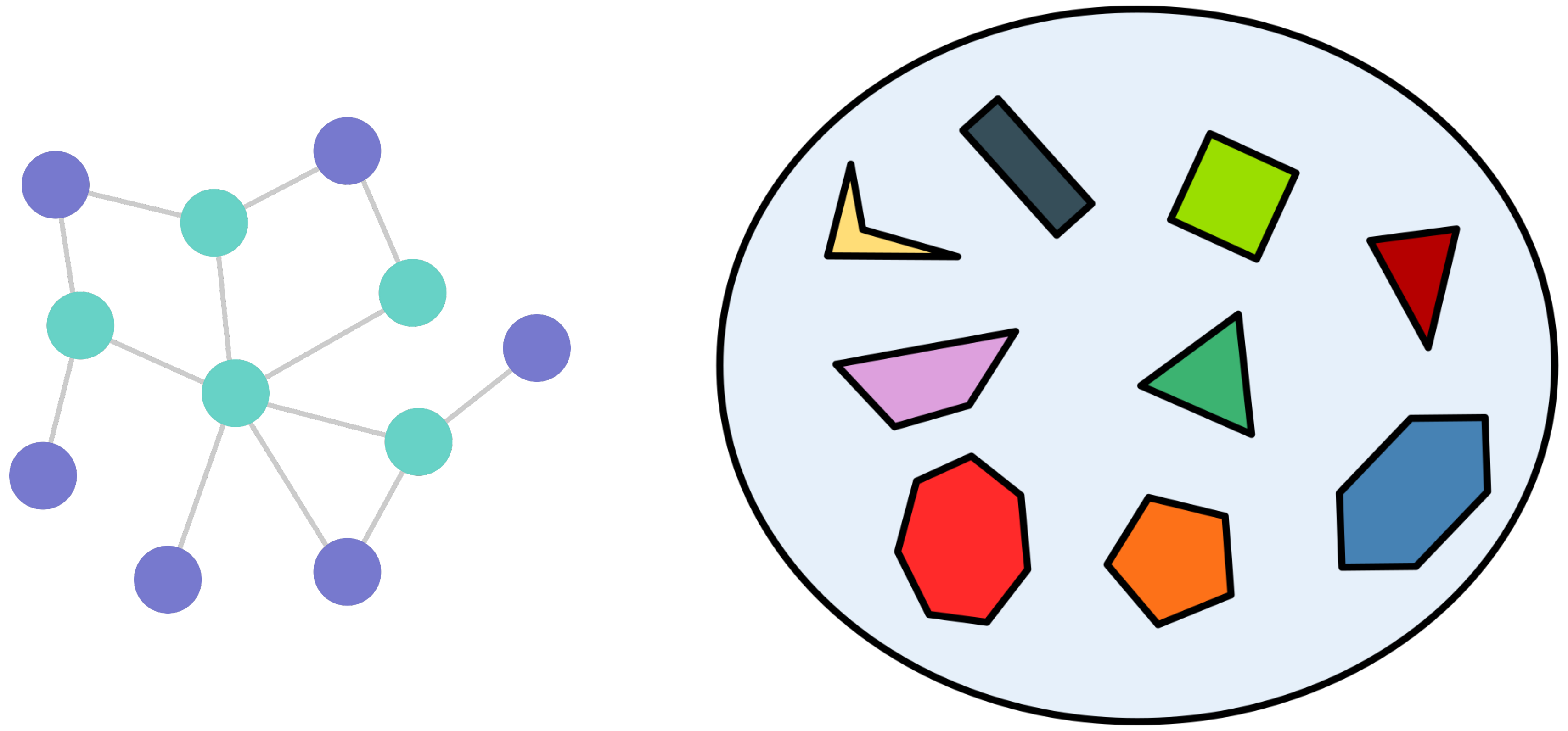


3D scans and objects

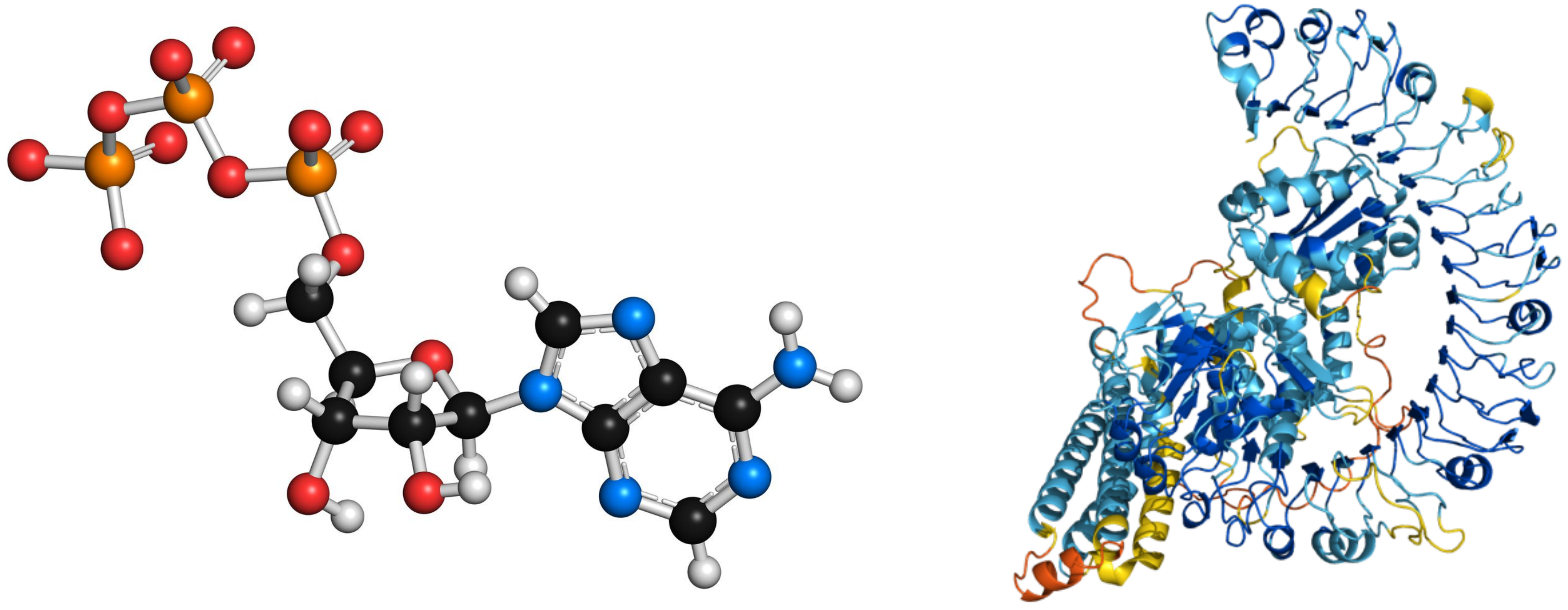
# Data may be scarce or expensive to collect



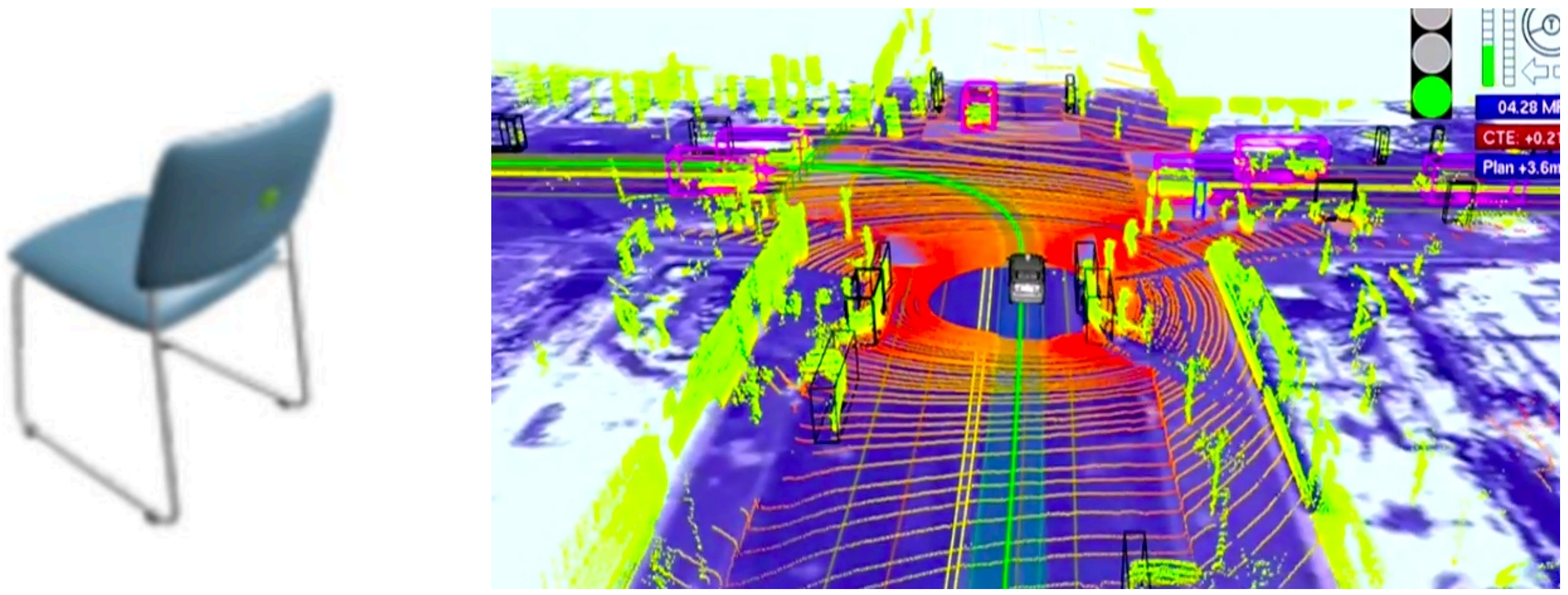
Spherical functions



Graphs and sets



Small molecules and proteins

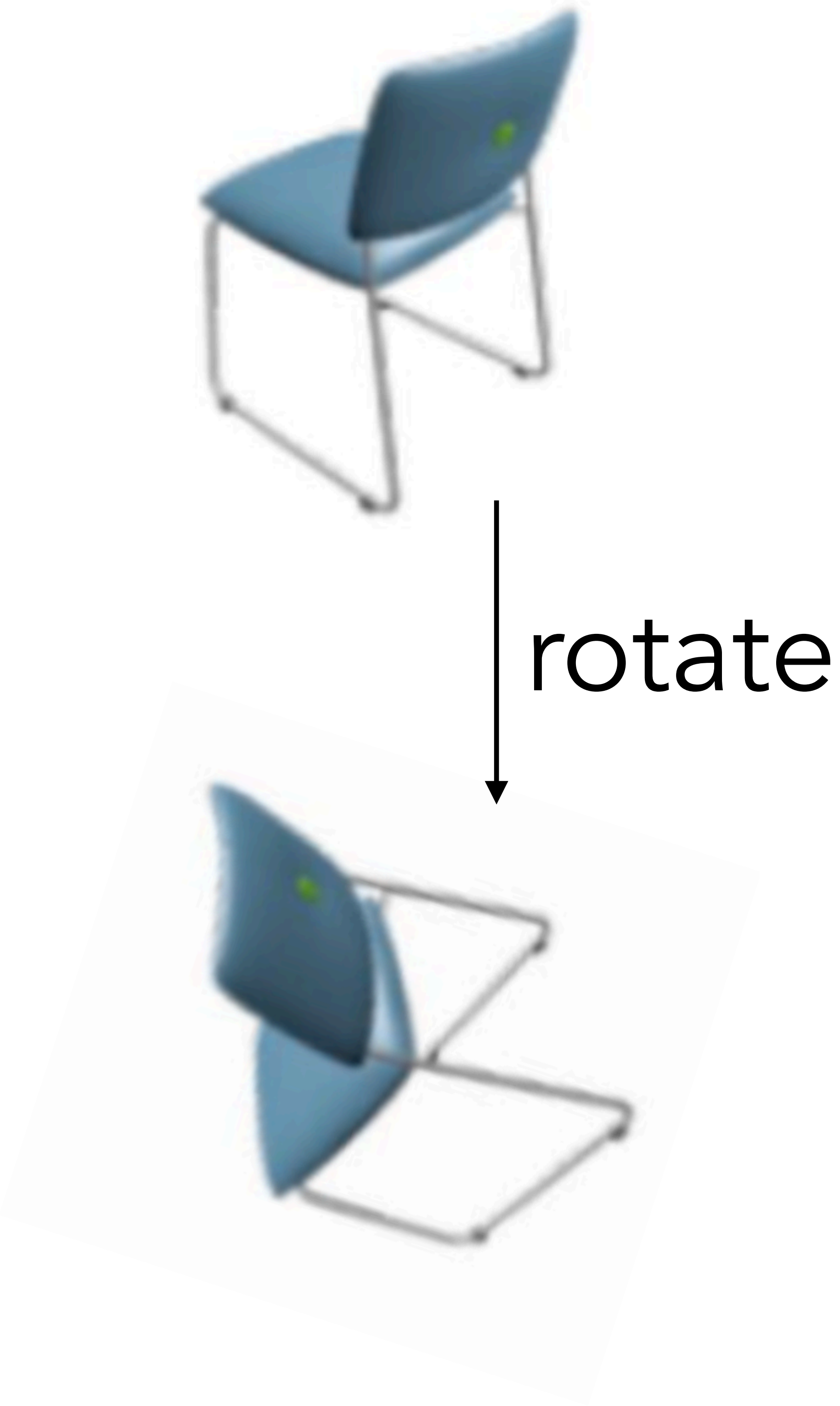


3D scans and objects

# Data contains symmetries!

"chair"

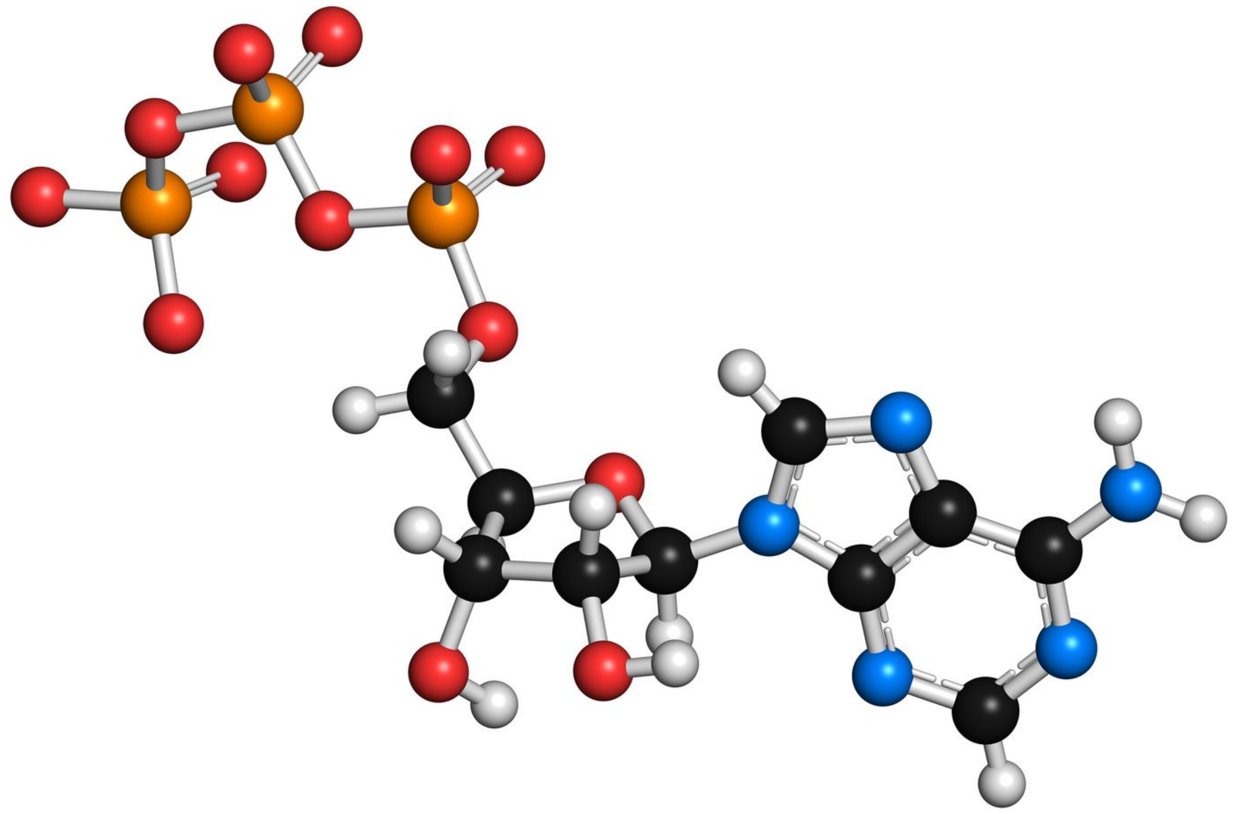
# Data contains symmetries!




“chair”

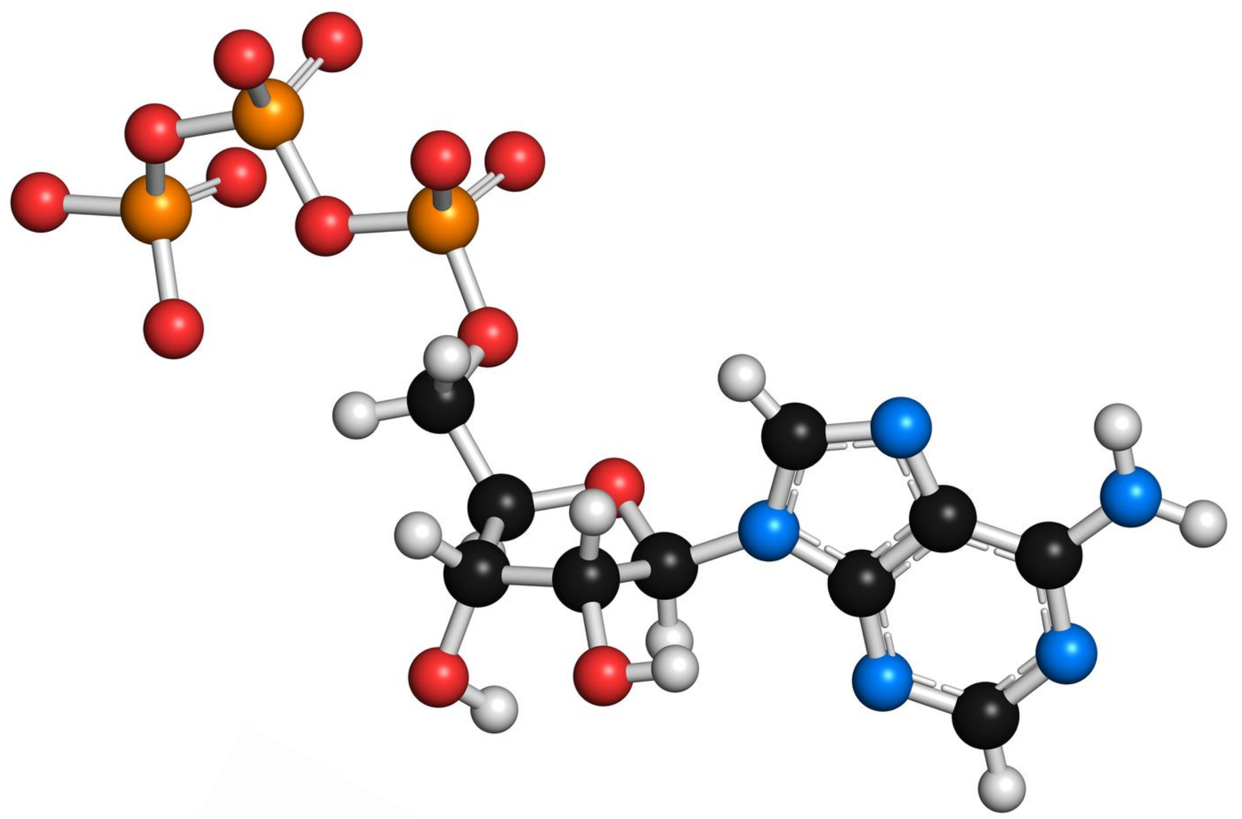
“chair”

# Data contains symmetries!



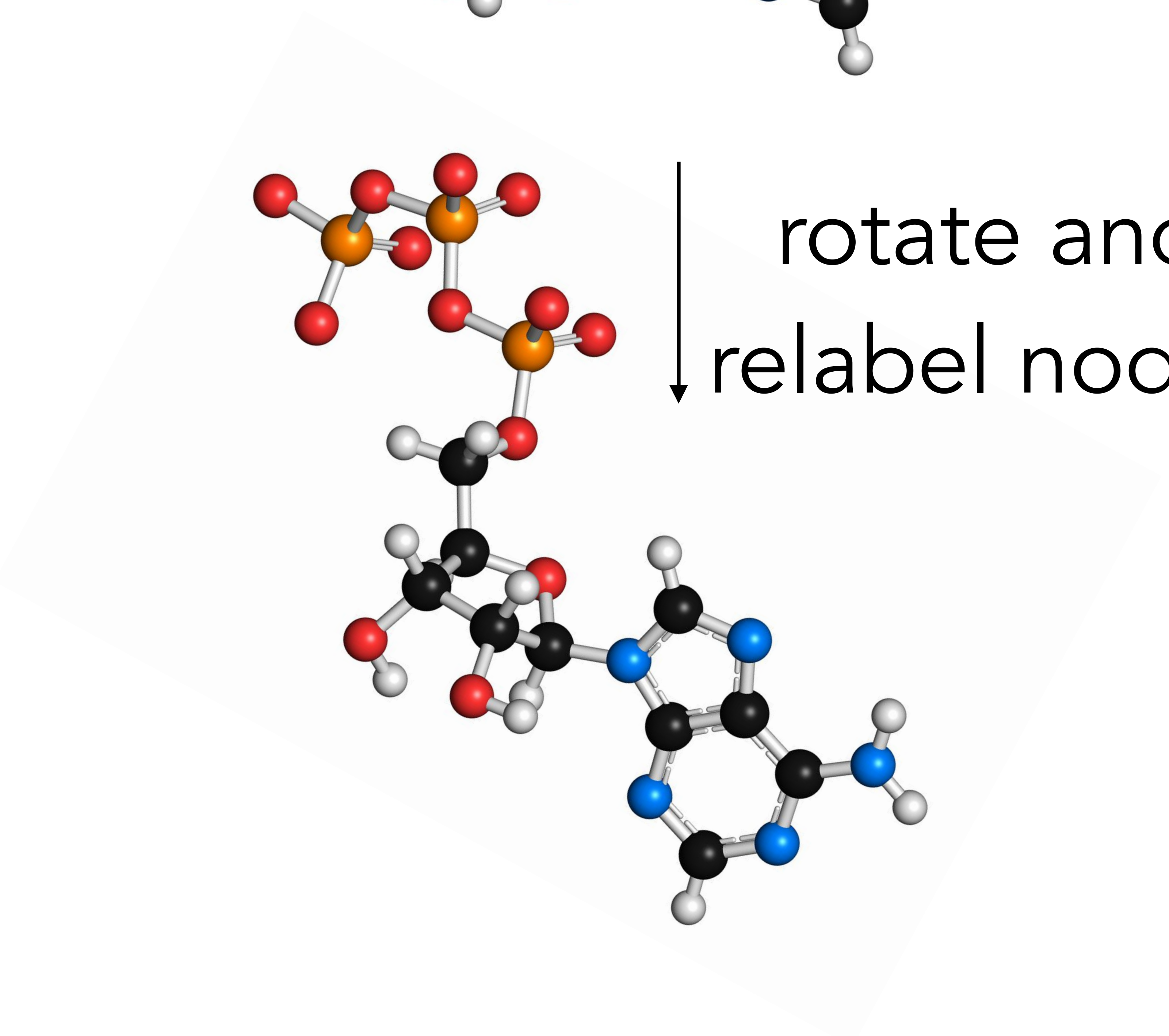
"good drug for disease X"

# Data contains symmetries!



“good drug for disease X”

rotate and relabel nodes

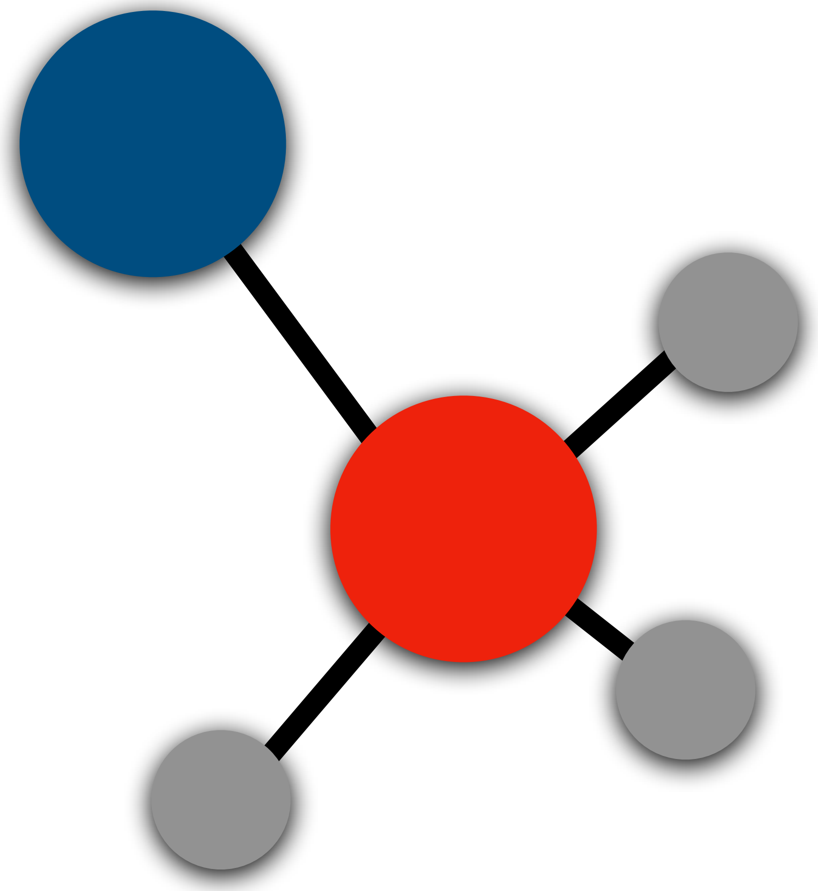


“good drug for disease X”

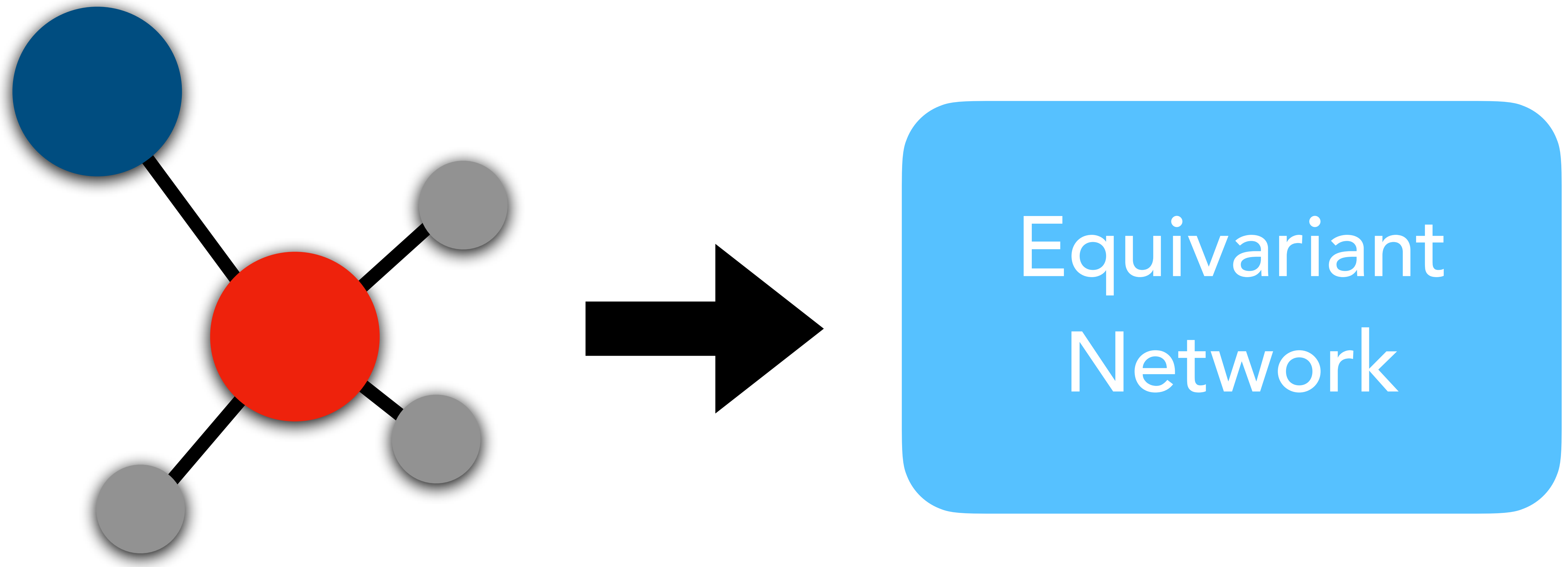
Let's build this into our machine learning pipeline!

Let’s build this into our network architecture!

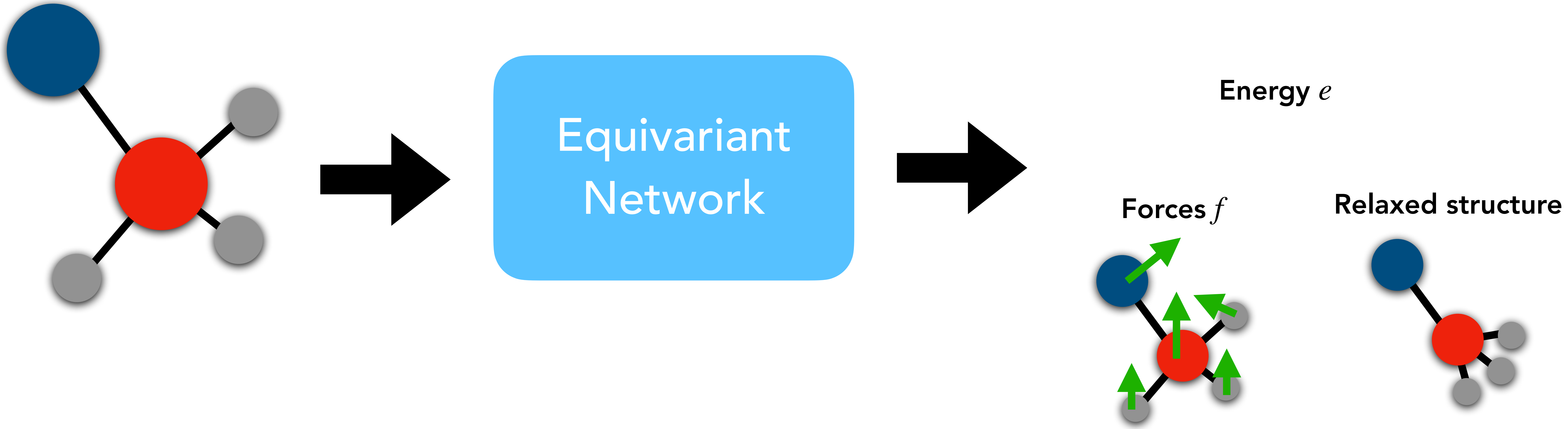
# Equivariance: $f(gx) = gf(x)$



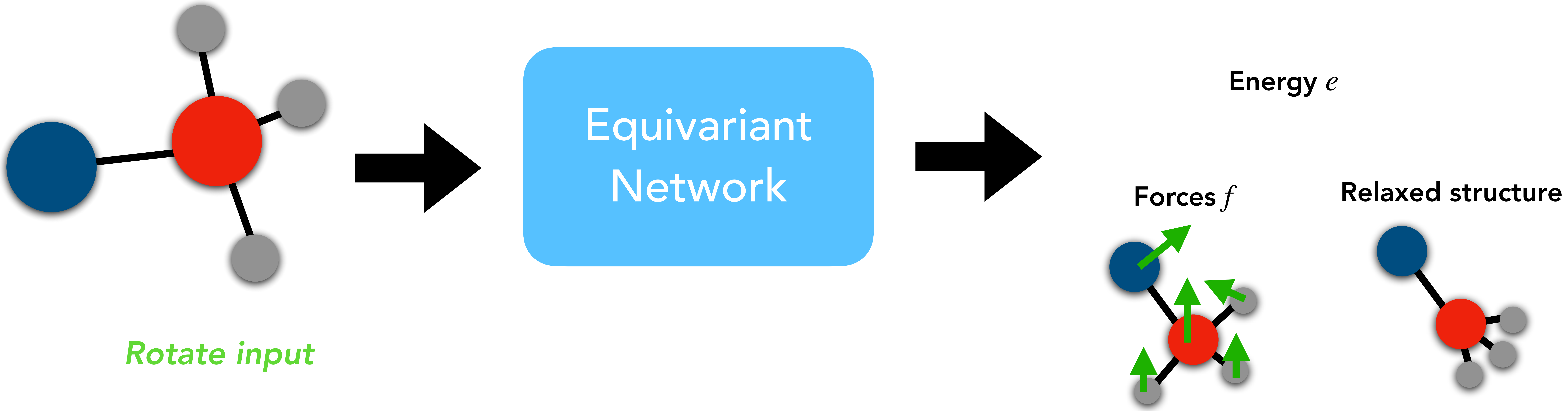
# Equivariance: $f(gx) = gf(x)$



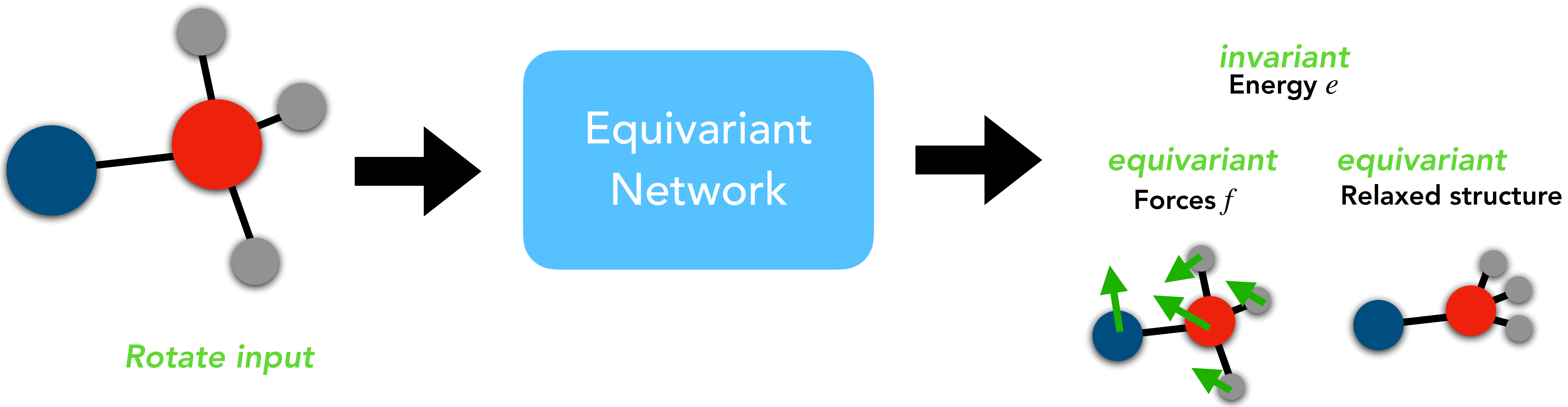
# Equivariance: $f(gx) = gf(x)$



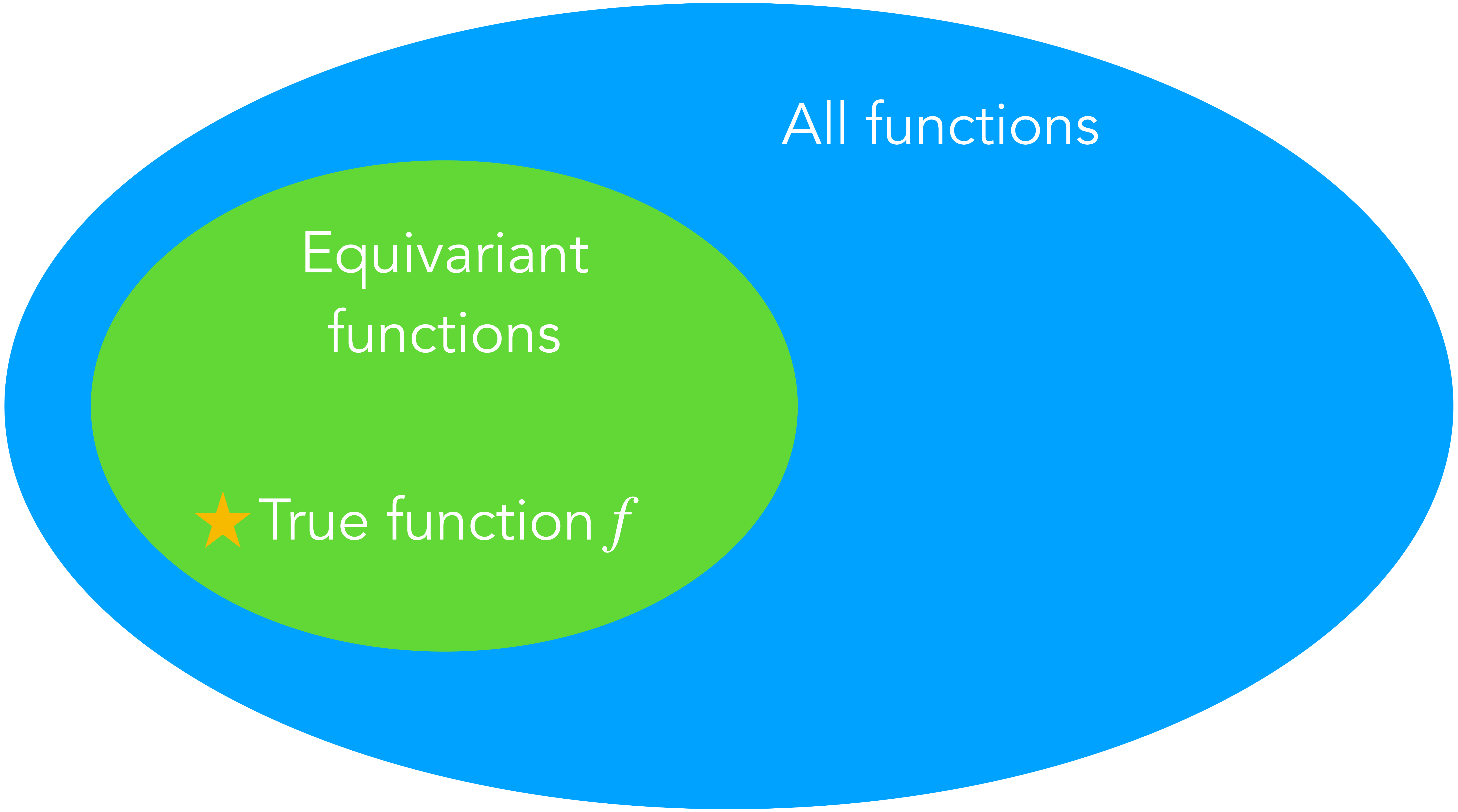
# Equivariance: $f(gx) = gf(x)$



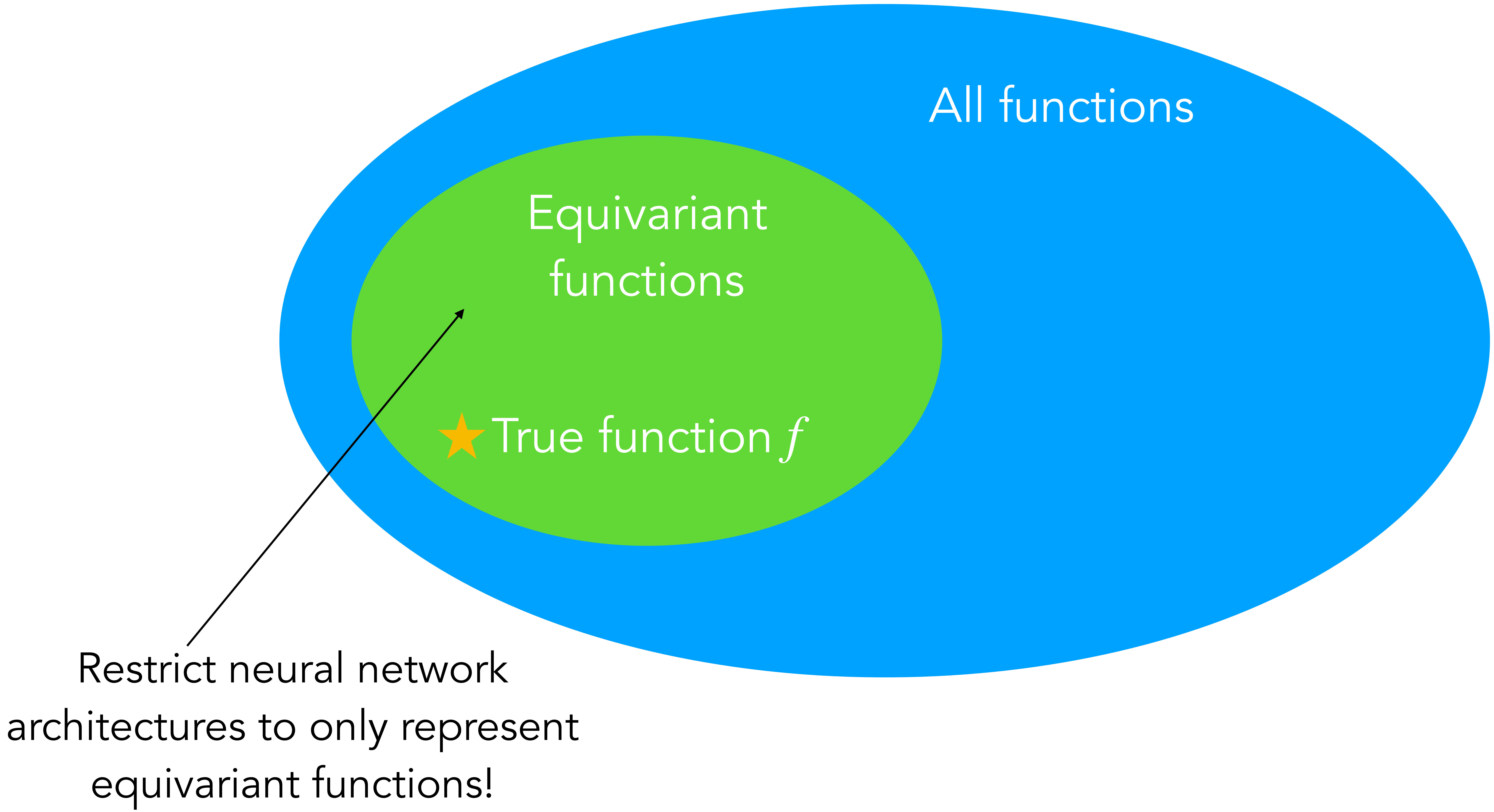
# Equivariance: $f(gx) = gf(x)$



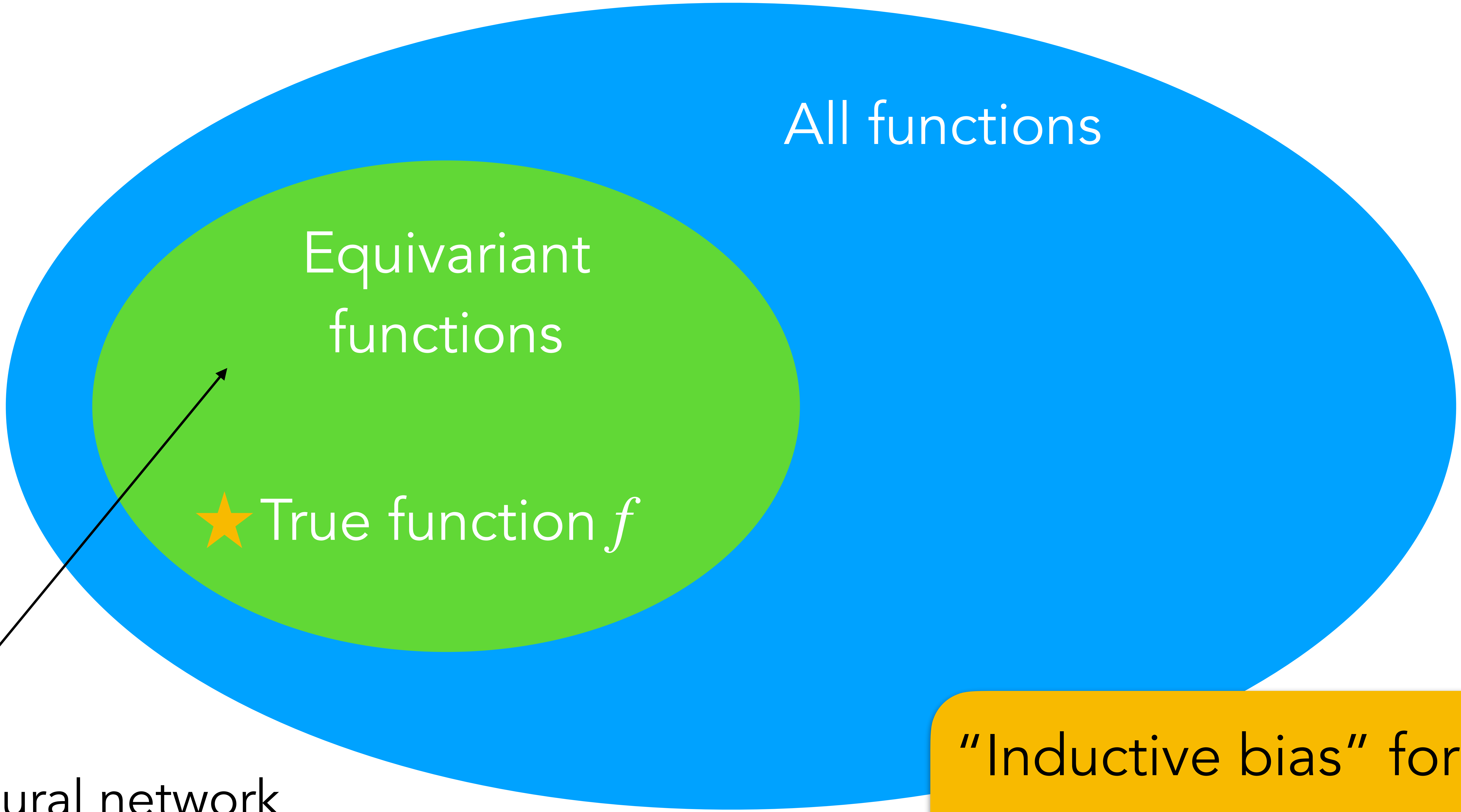
# Main idea of equivariant learning



# Main idea of equivariant learning



# Main idea of equivariant learning




Restrict neural network architectures to only represent equivariant functions!

"Inductive bias" for the appropriate symmetry structure, **known** to occur in your data

# Why encode these symmetries?

- Like CNNs, they encode properties that we know our data has
  - The network doesn't have to *learn* these invariances, so it will have better sample complexity! (**verified empirically + theoretically)**
  - It can generalize to unseen translations, rotations, etc
- Faster than data augmentation, which is especially intractable for things like permutations

# How do you encode these symmetries?

Early work on architectures: generalize convolution to work with new data types!

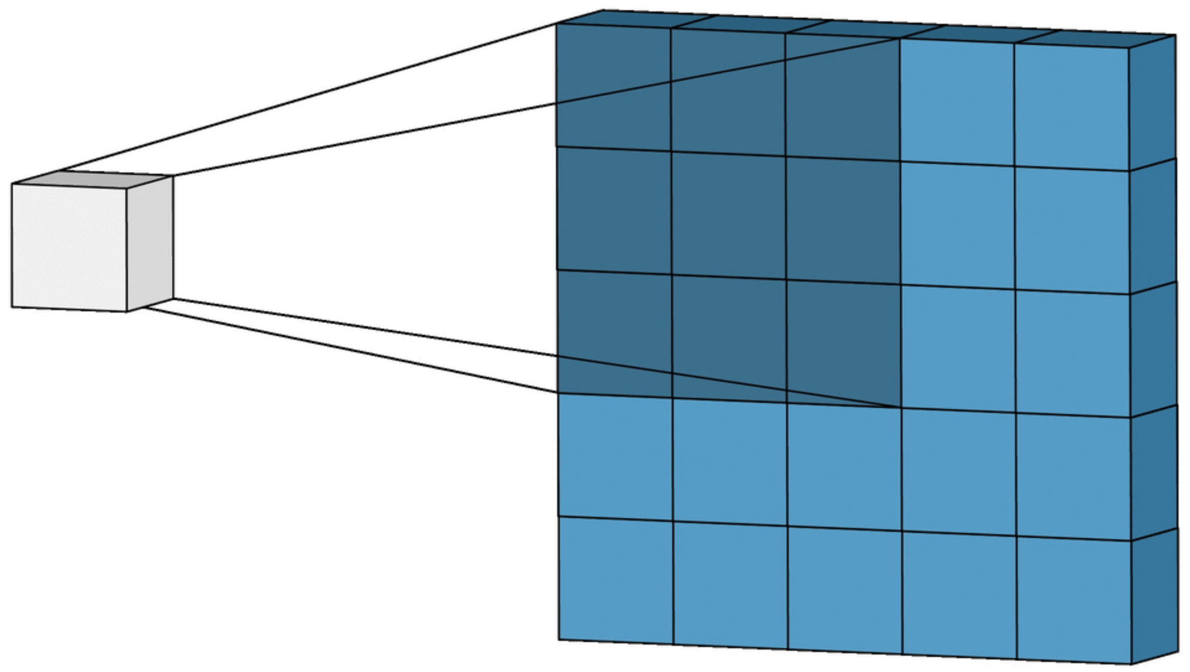
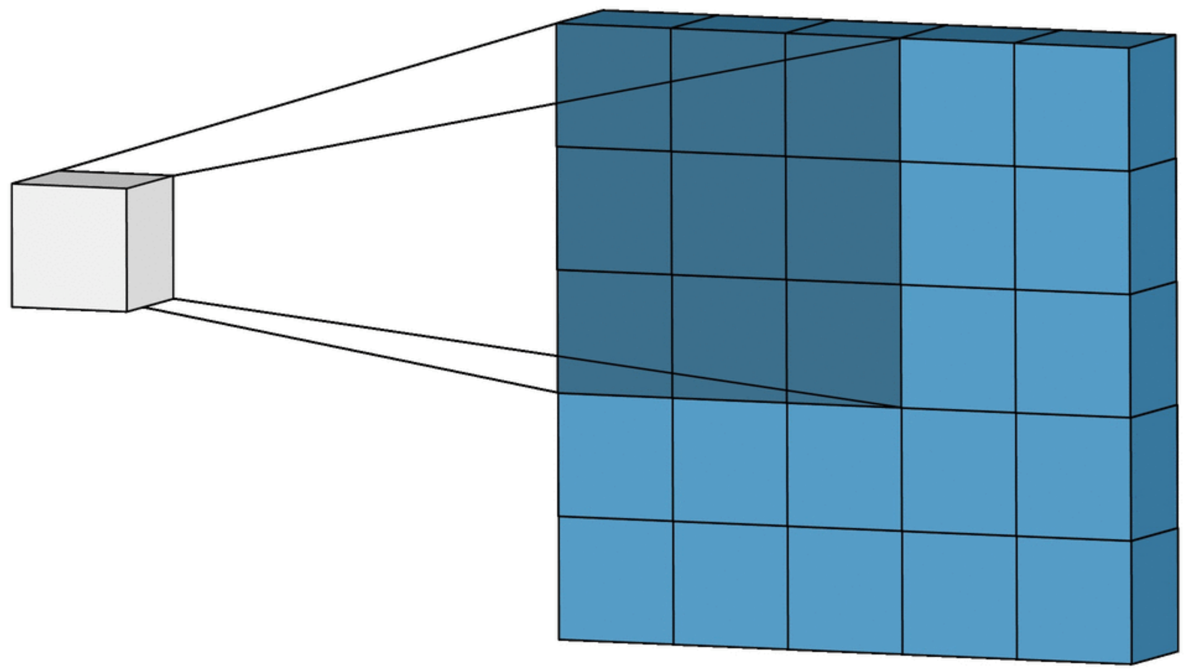


Image convolution: **translate** a filter all around the image

# How do you encode these symmetries?

Early work on architectures: generalize convolution to work with new data types!



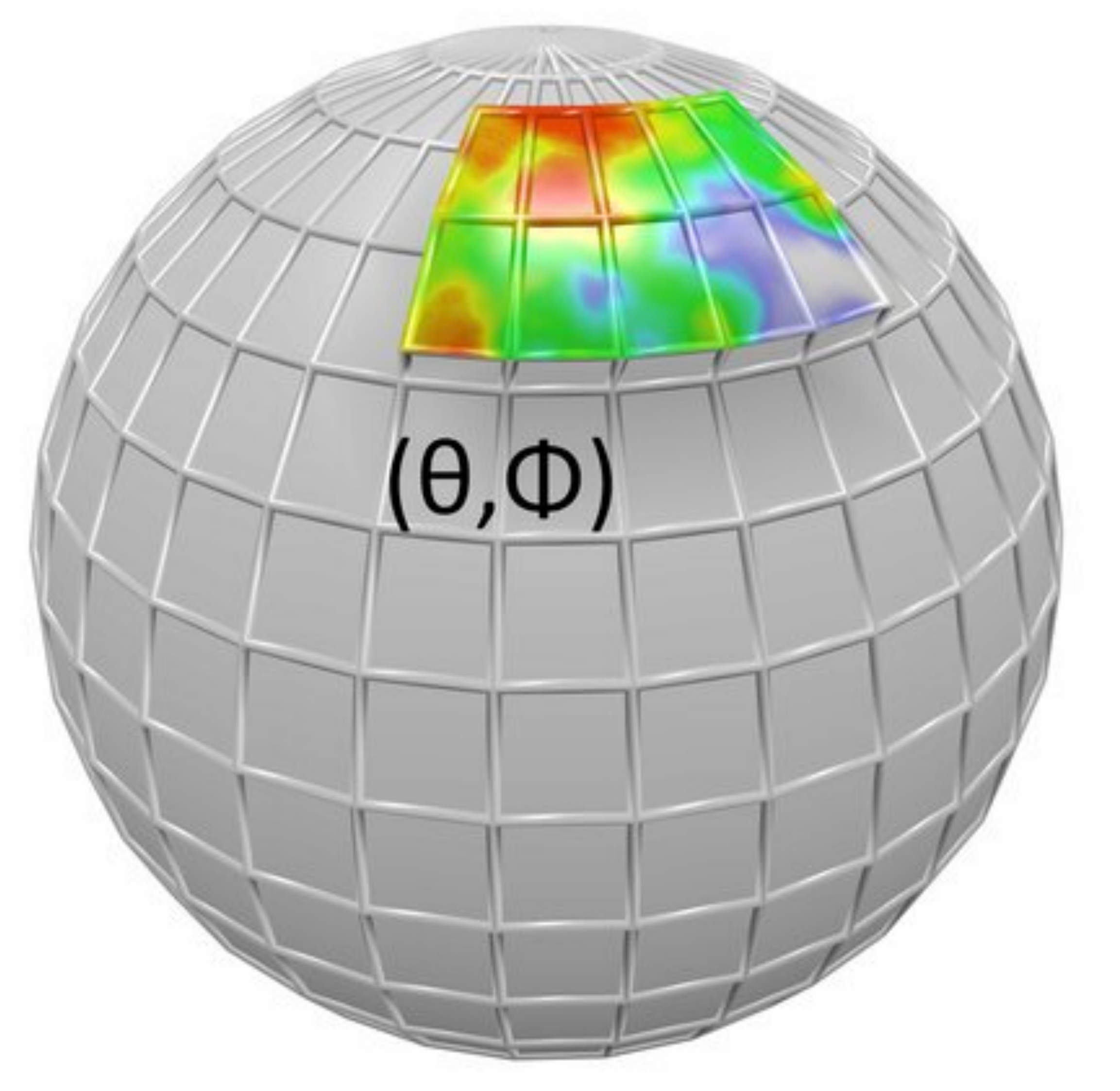
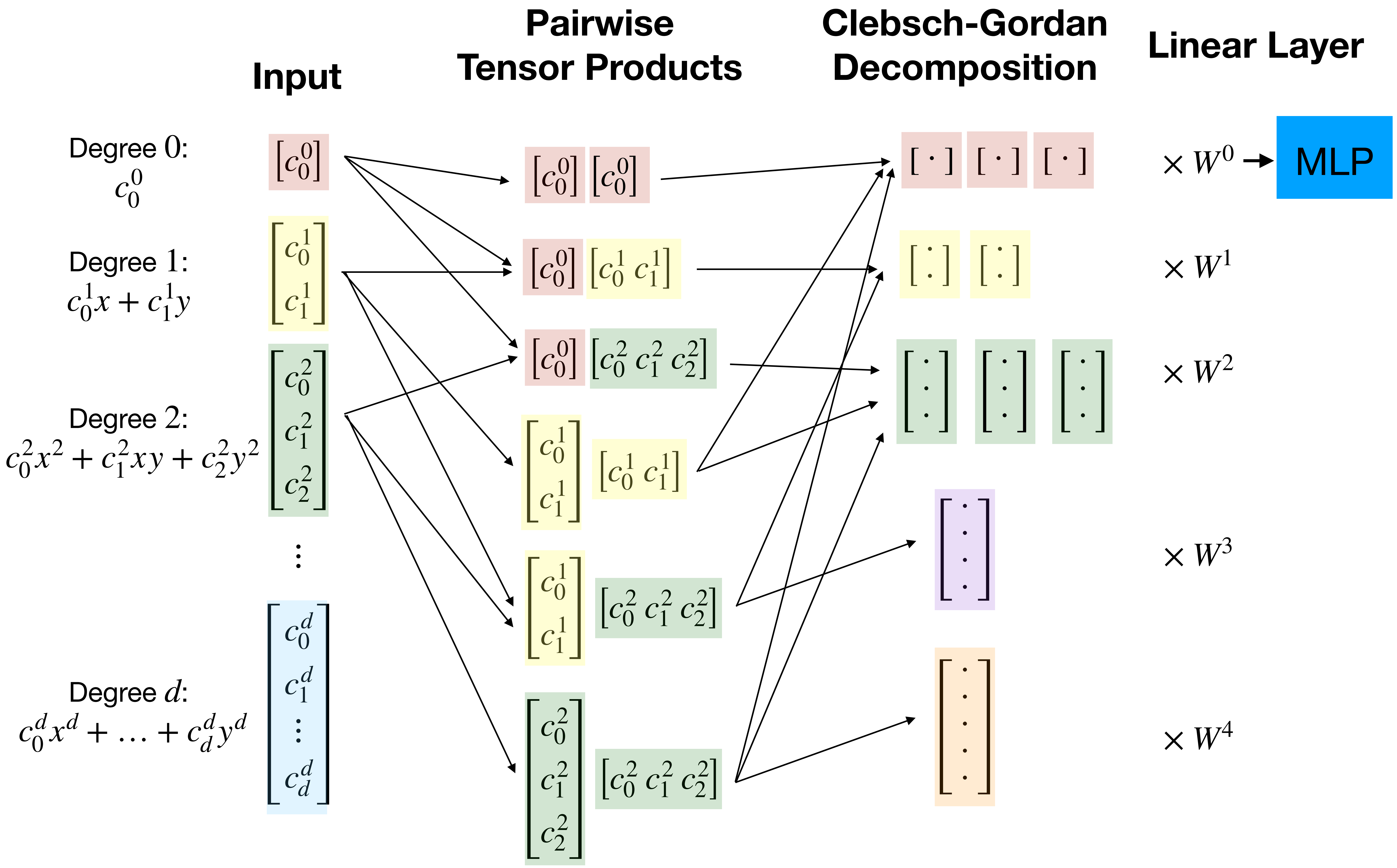


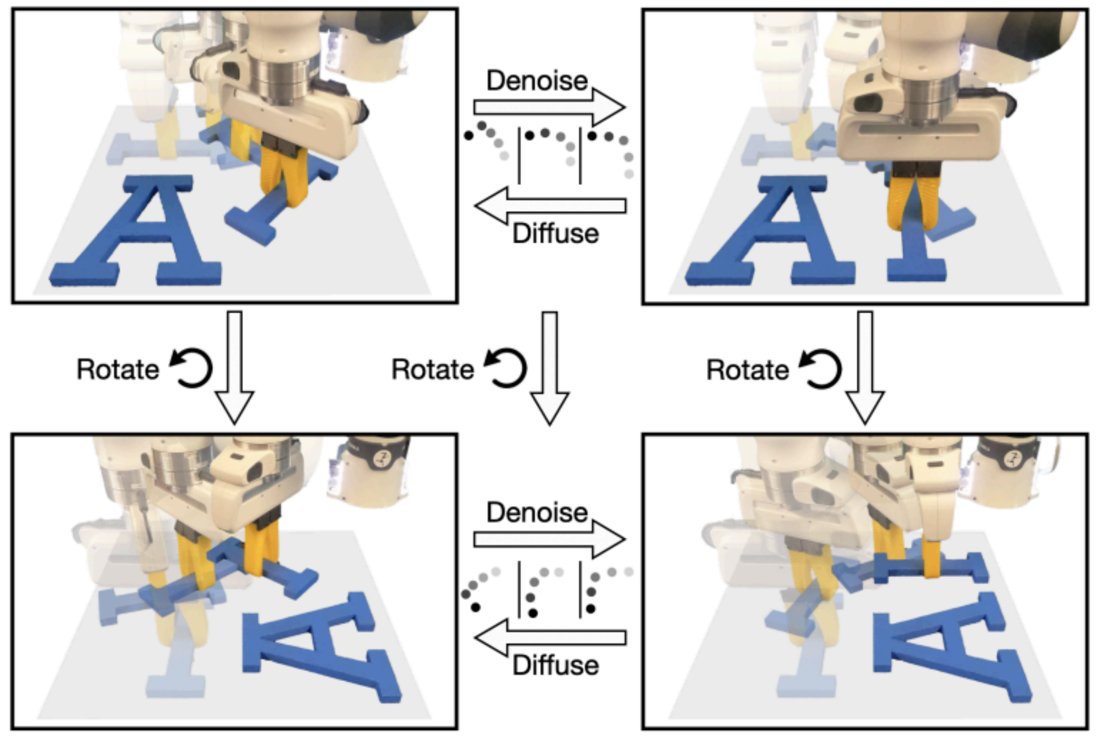

Image convolution: **translate** a filter all around the image

Spherical convolution: **rotate** a filter all around the spherical function

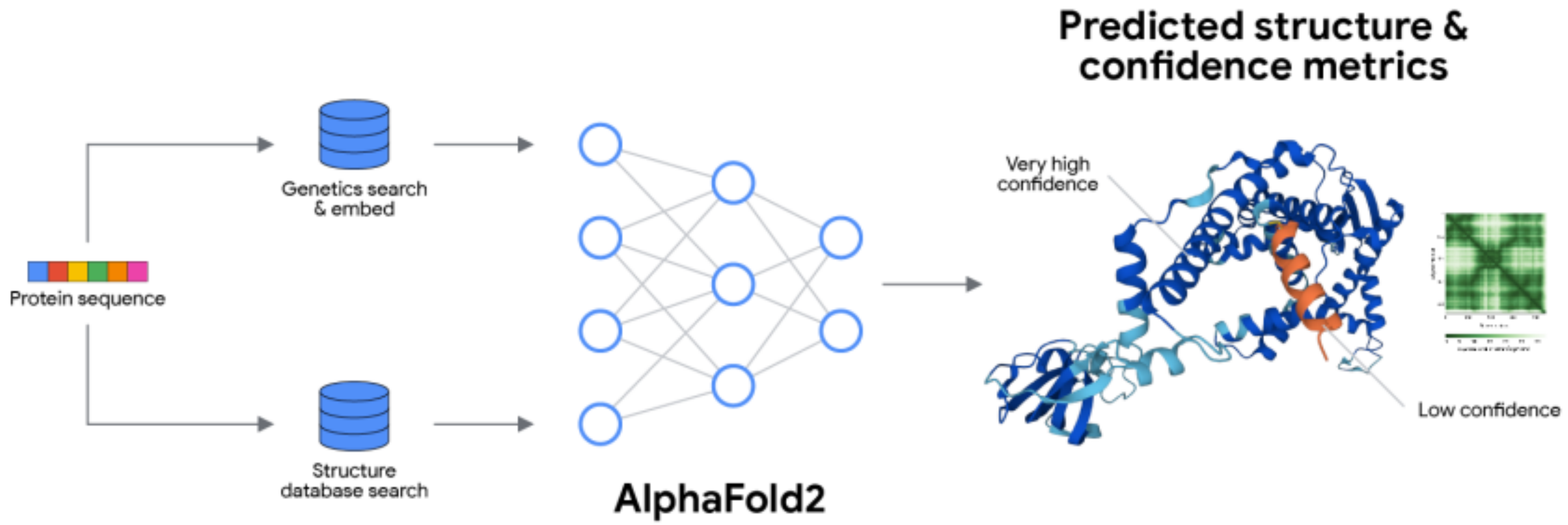
# Gets even more complicated!

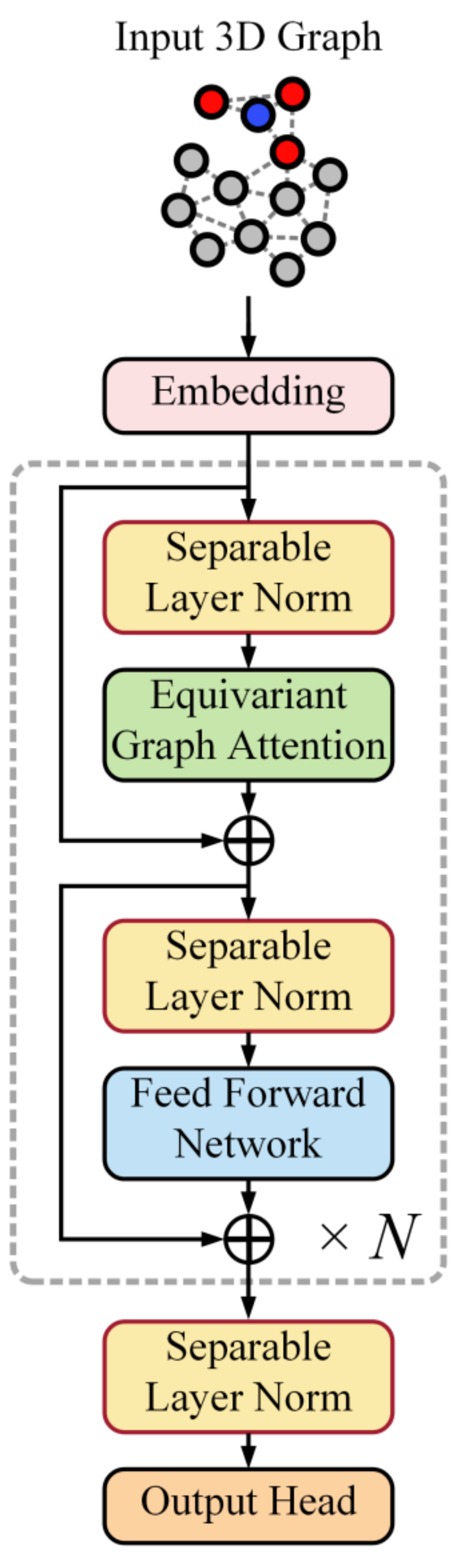


# Equivariant nets have been successful




Equivariant diffusion policy, Wang et al 2024




**(a) EquiformerV2 Architecture**

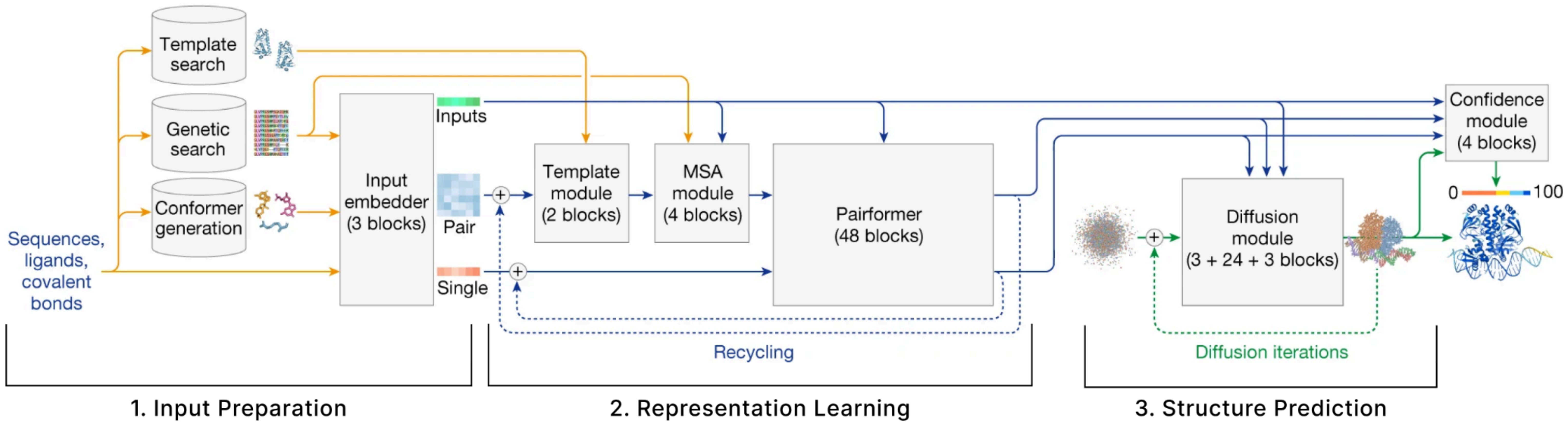
Especially in low-data regimes, and for big groups (e.g. permutations)

# The bitter lesson

"general methods that leverage computation are ultimately the most effective, and by a large margin"

"…methods that continue to scale with increased computation even as the available computation becomes very great"

Example: AlphaFold3




# The bitter lesson

"general methods that leverage computation are ultimately the most effective, and by a large margin"

"…methods that continue to scale with increased computation even as the available computation becomes very great"

"…researchers seek to leverage their human knowledge of the domain, but the only thing that matters in the long run is the leveraging of computation. **These two need not run counter to each other**, but in practice they tend to"

# Outline

1. Canonicalization

2. Positional Encodings

3. Tokenization

# What's "wrong" with equivariant nets?

- Architecture is more complicated than your average transformer
- Specialized engineering required for normalizing, optimization, GPU usage — still being actively developed!
    - Often, slower forward passes (e.g. Equiformer)
- No way of turning a pretrained black-box (closed source) architecture into an equivariant one
- Often, rigid constraint on input type

# Part 1: Canonicalization

# At a high level…

## Past approaches:

Build equivariance into the architecture

*or*

Data augmentation during training — expensive/less effective

# At a high level…

## Past approaches:

Build equivariance into the architecture

*or*

Data augmentation during training — expensive/less effective

## Canonicalization:

Build equivariance into the **data pre-processing**

# “Canonical”

**Canon:** “a body of principles, rules, standards, or norms”

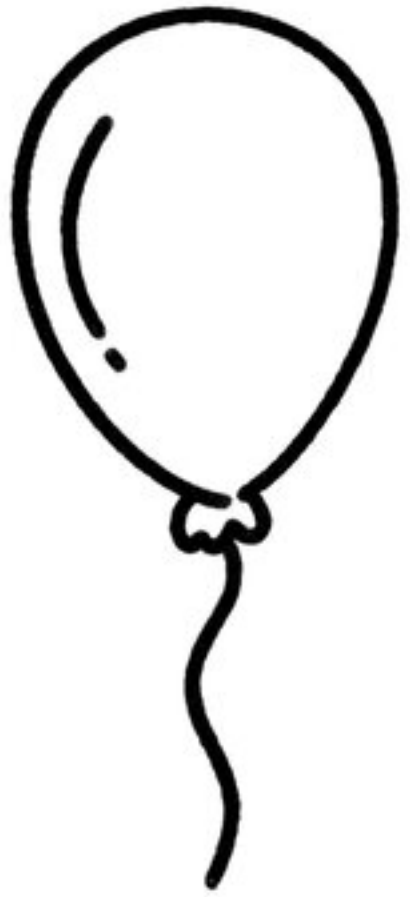
**Canonical:** “conforming to a general rule or acceptable procedure” or “reduced to the canonical form”


$$\begin{pmatrix} \boxed{\begin{matrix}\lambda_1 & 1 & \\ & \lambda_1 & 1 \\ & & \lambda_1\end{matrix}} & & & & \\ & \boxed{\begin{matrix}\lambda_2 & 1 \\ & \lambda_2\end{matrix}} & & & \\ & & \boxed{\lambda_3} & & \\ & & & \ddots & \\ & & & & \boxed{\begin{matrix}\lambda_n & 1 \\ & \lambda_n\end{matrix}} \end{pmatrix}$$

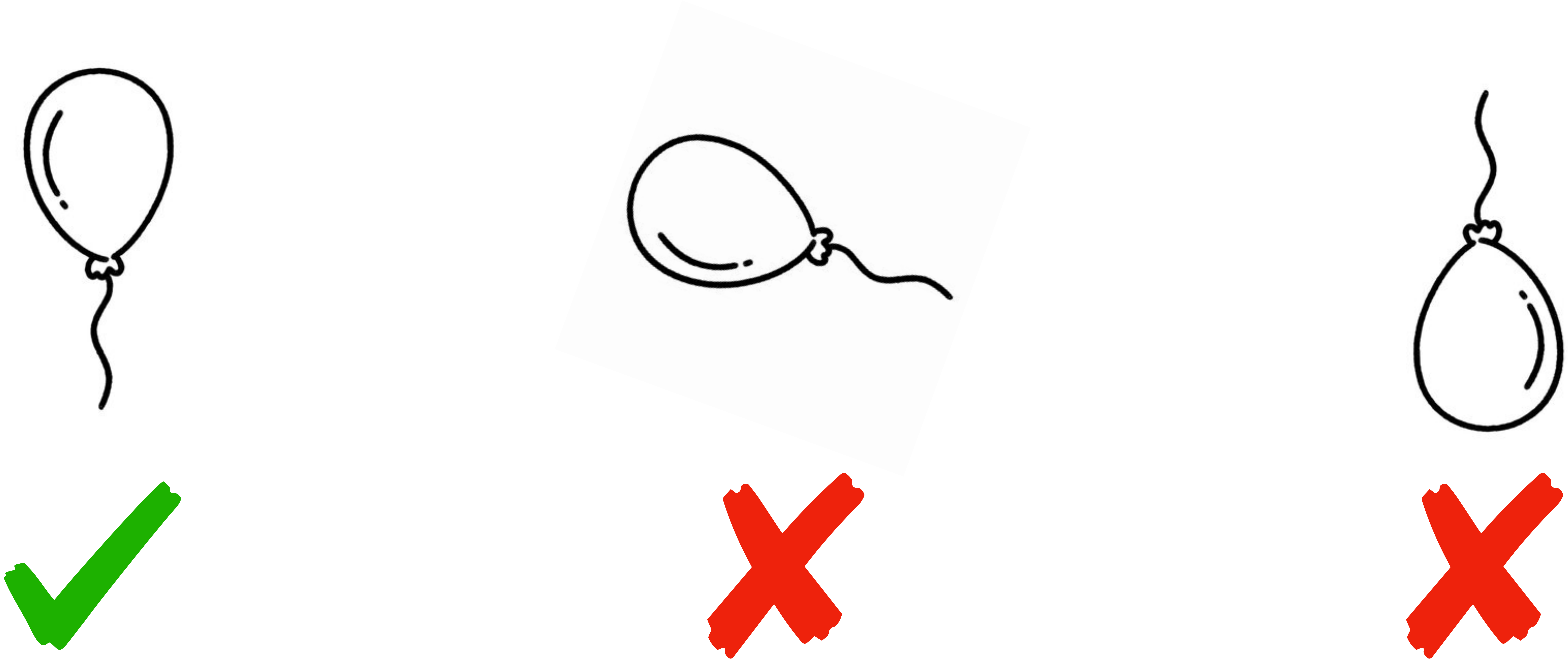
Sources: Wikipedia, Merriam-Webster

# “Canonical” in images

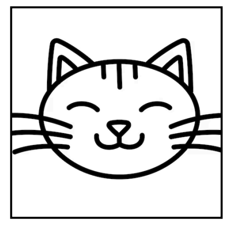
# “Canonical” in images



# “Canonical” in images

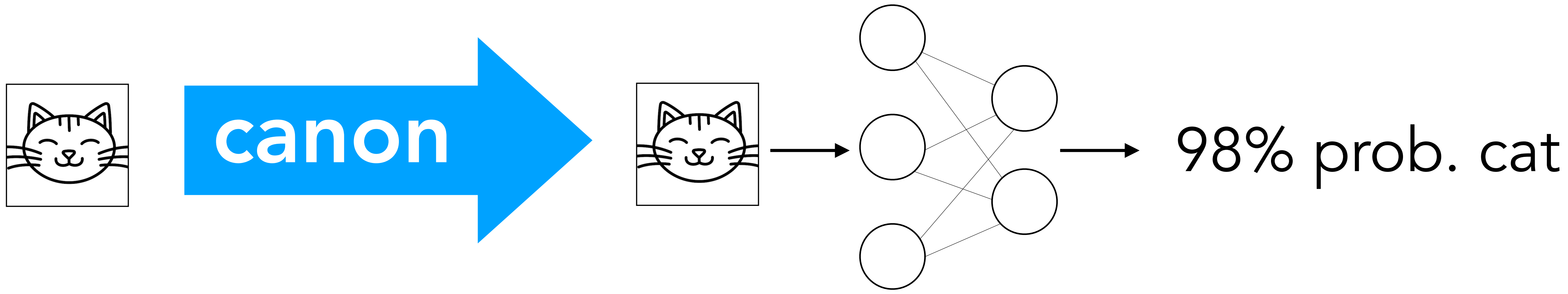


# Canonicalization → Equivariance

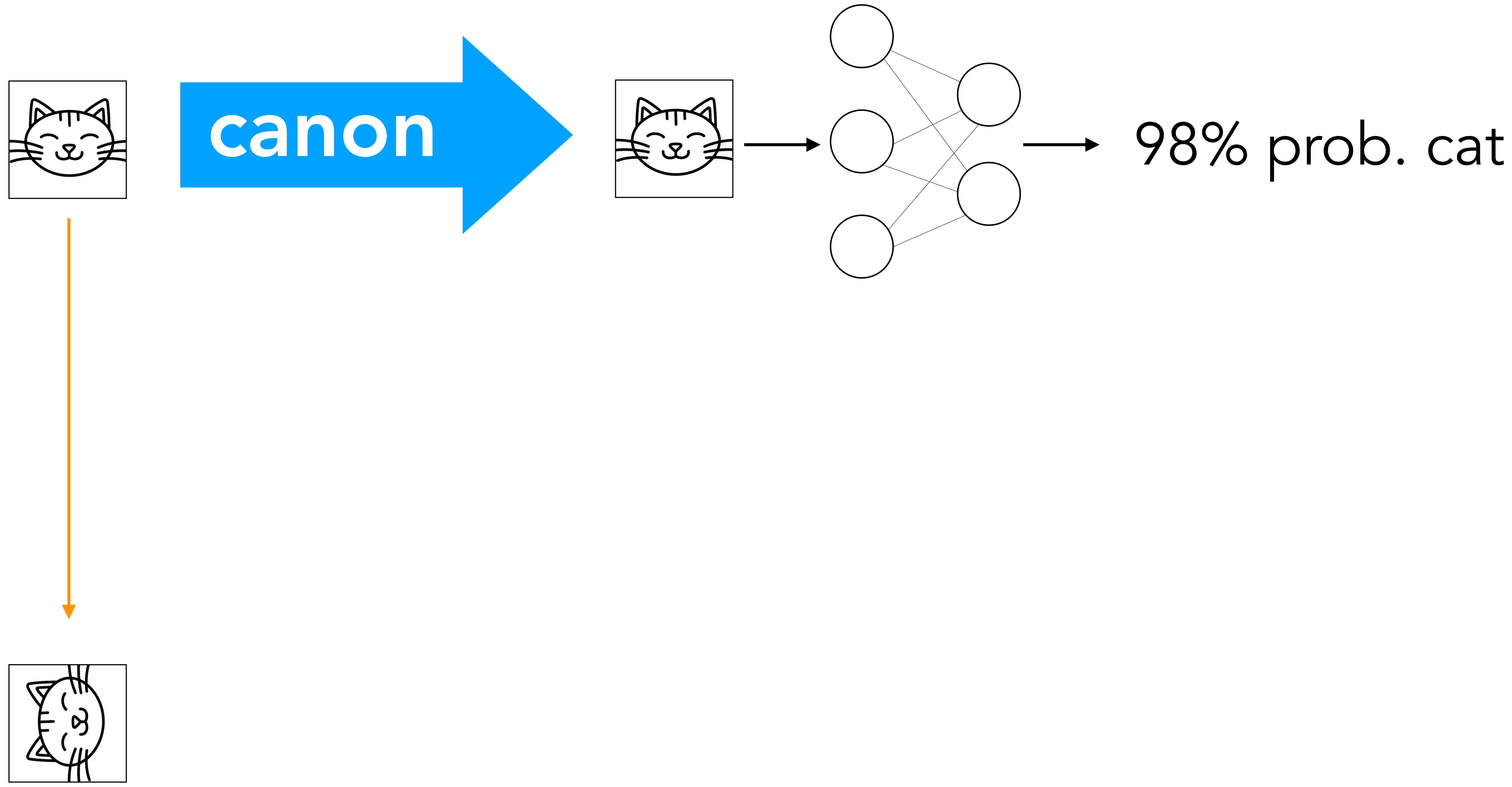


# Canonicalization → Equivariance

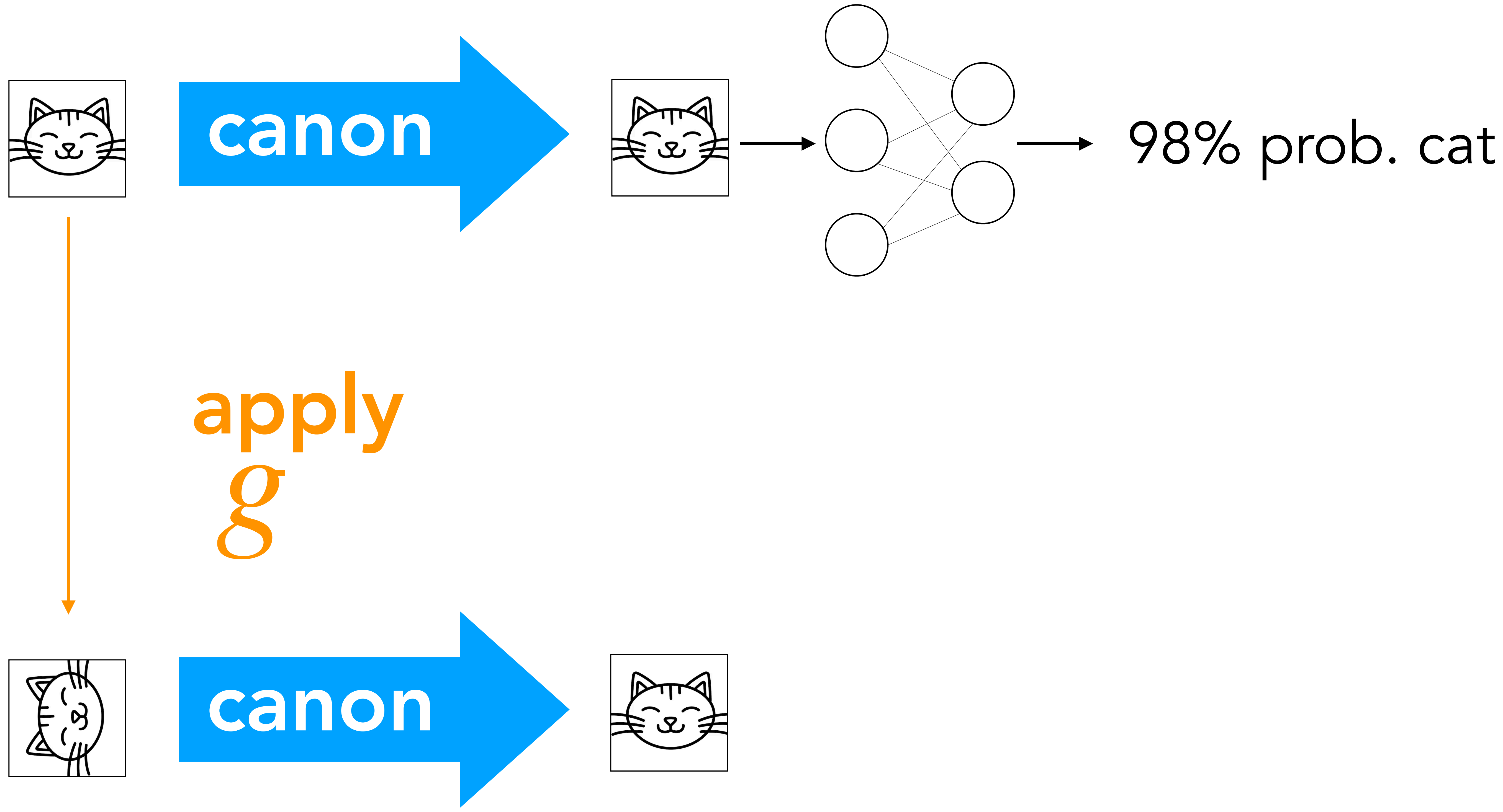
# Canonicalization → Equivariance



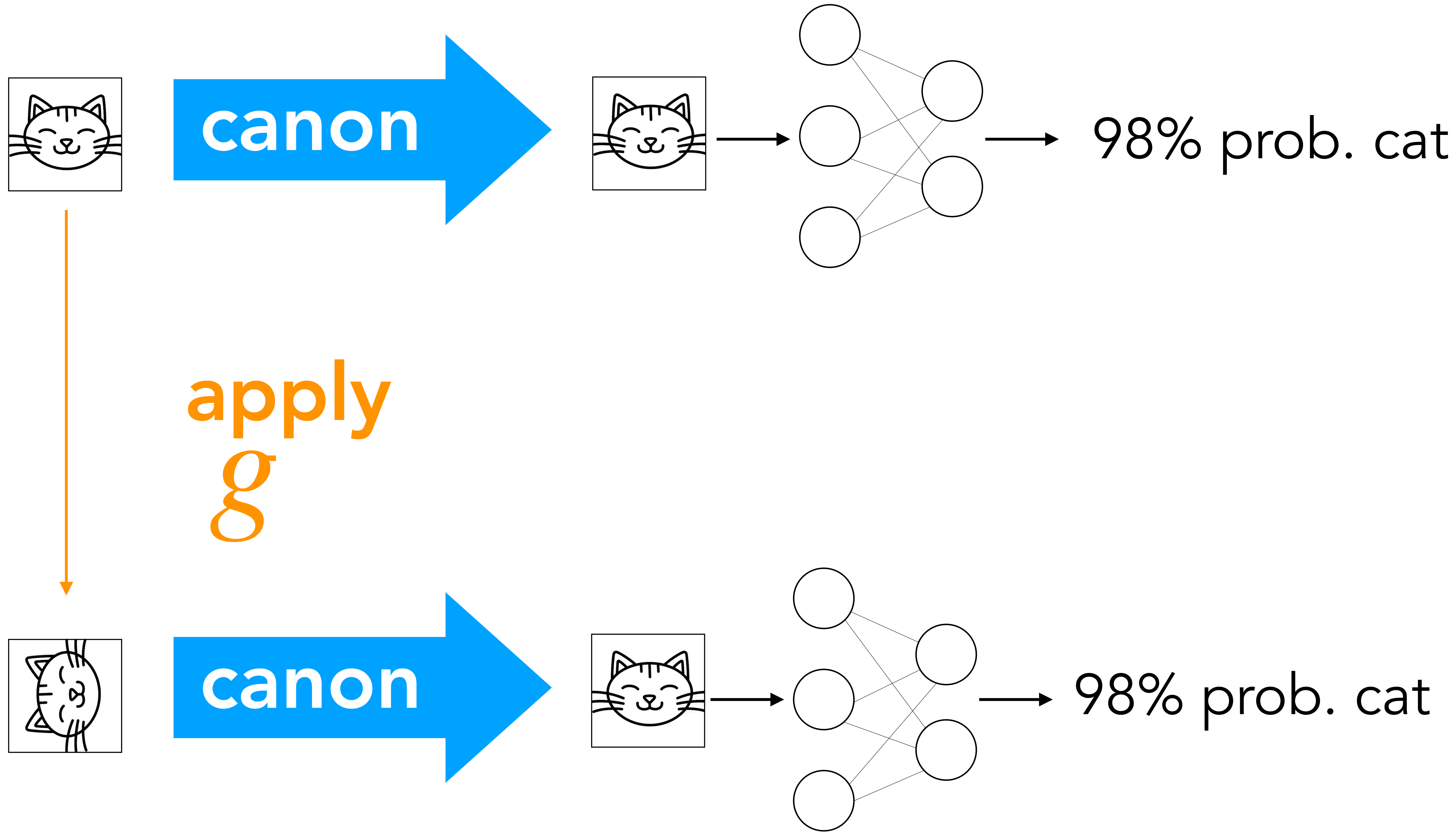
# Canonicalization → Equivariance



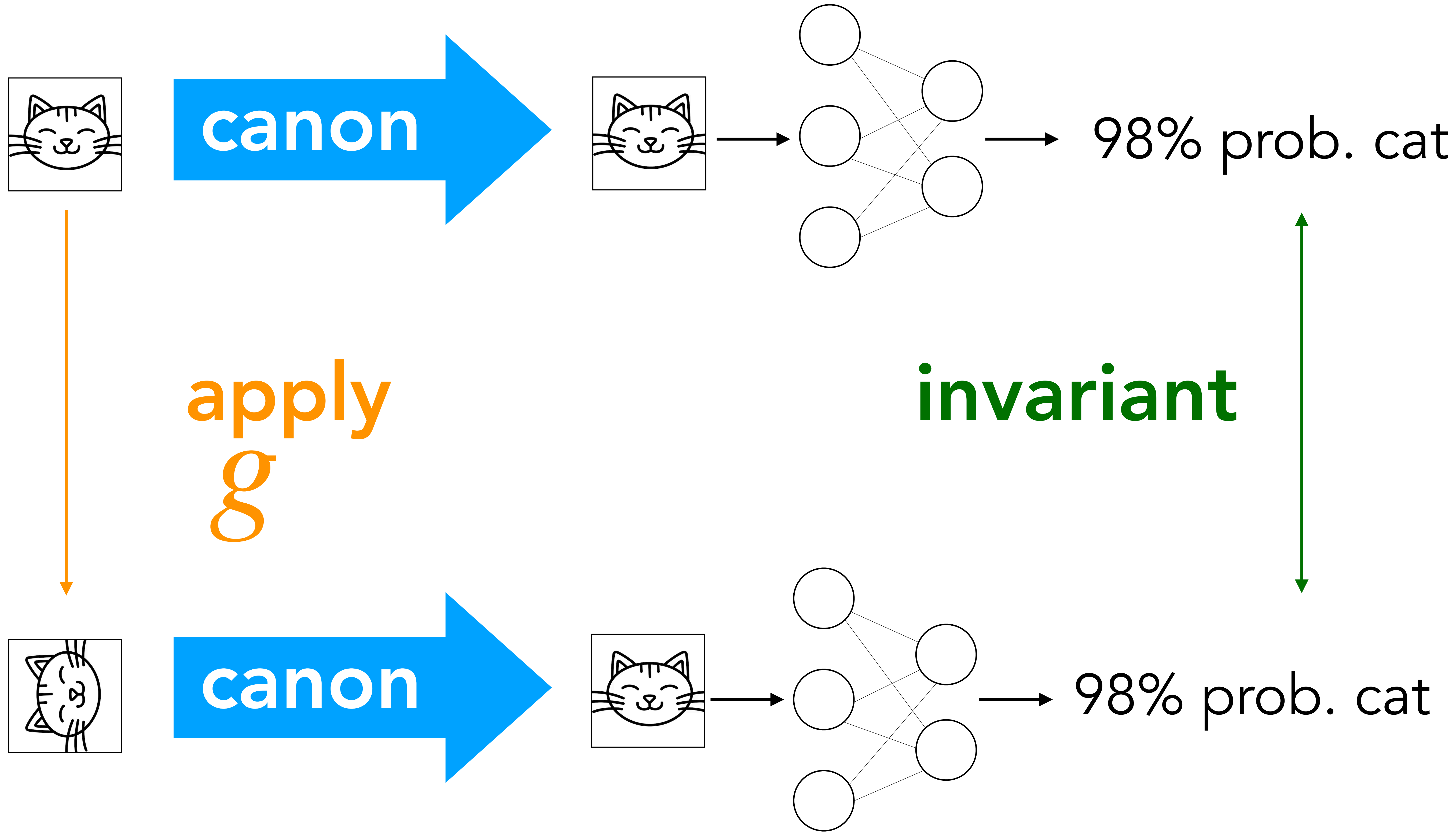
# Canonicalization → Equivariance



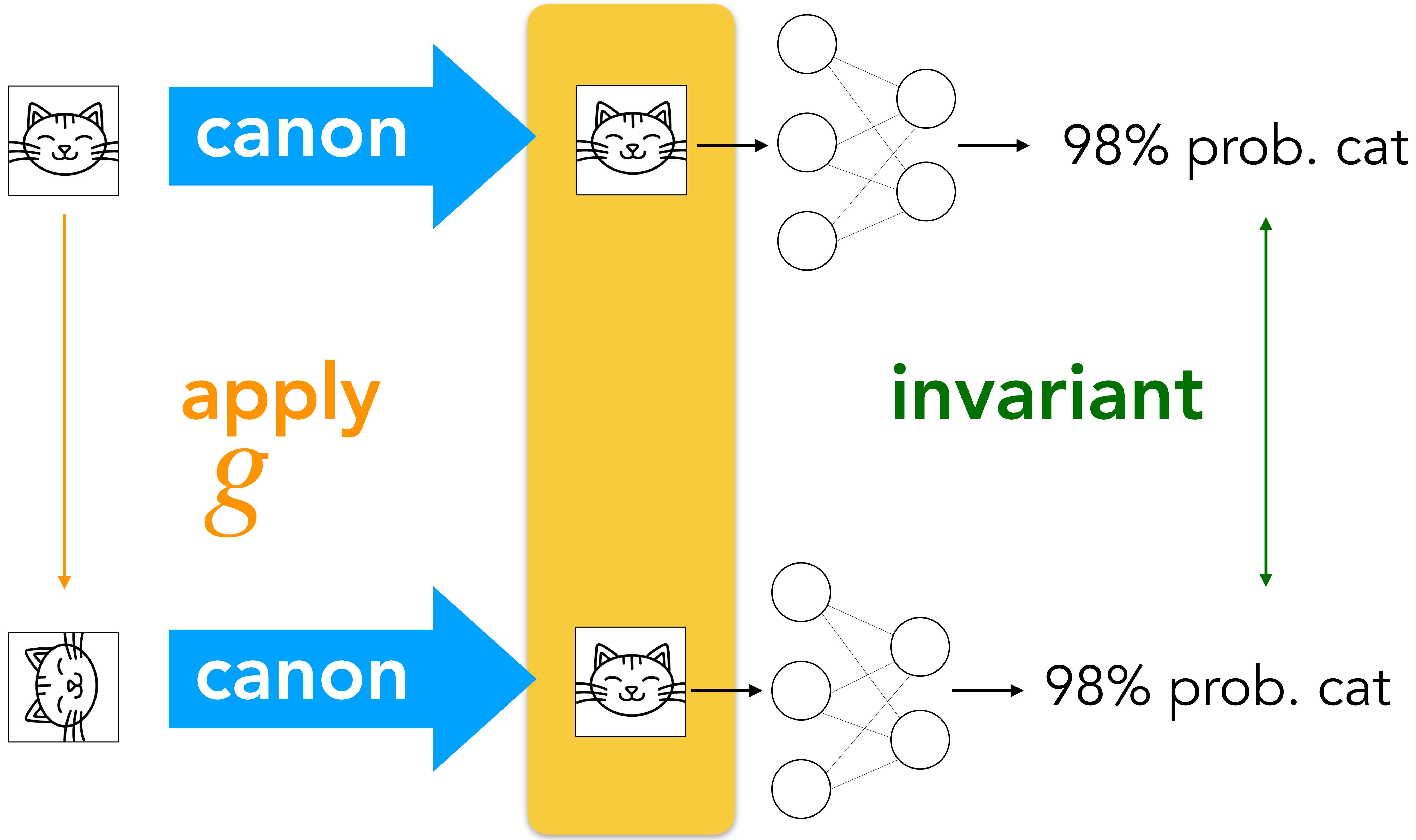
# Canonicalization → Equivariance



# Canonicalization → Equivariance



# Canonicalization → Equivariance



# Again, in summary:

## Past approaches:

Build equivariance into the architecture

*or*

Data augmentation during training — expensive/less effective

## Canonicalization:

Build equivariance into the **data pre-processing**

# Classical idea, but recent excitement in ML!

**Frame Averaging for Invariant and Equivariant Network Design**

Omri Puny[*1] Matan Atzmon[*1] Heli Ben-Hamu[*1] Ishan Misra[2]
Aditya Grover[2] Edward J. Smith[2] Yaron Lipman[2 1]
[1]Weizmann Institute of Science [2]Facebook AI Research

ICLR 2022

**SE(3) Equivariant Graph Neural Networks with Complete Local Frames**

Weitao Du[*†1] He Zhang[*†2] Yuanqi Du[†3] Qi Meng[4] Wei Chen[†1] Nanning Zheng[2] Bin Shao[4] Tie-Yan Liu[4]

ICML 2022

**Equivariant Adaptation of Large Pretrained Models**

Arnab Kumar Mondal[*†] Mila, McGill University, ServiceNow Research
Siba Smarak Panigrahi[*] Mila, McGill University
Sékou-Oumar Kaba Mila, McGill University
Sai Rajeswar ServiceNow Research
Siamak Ravanbakhsh Mila, McGill University

NeurIPS 2023

**Learning Probabilistic Symmetrization for Architecture Agnostic Equivariance**

Jinwoo Kim Tien Dat Nguyen Ayhan Suleymanzade
Hyeokjun An Seunghoon Hong
KAIST

NeurIPS 2023

**Smooth, exact rotational symmetrization for deep learning on point clouds**

Sergey N. Pozdnyakov and Michele Ceriotti
Laboratory of Computational Science and Modelling,
Institute of Materials, Ecole Polytechnique Fédérale de Lausanne,
Lausanne 1015, Switzerland
sergey.pozdnyakov@epfl.ch, michele.ceriotti@epfl.ch

NeurIPS 2023

**FAENet: Frame Averaging Equivariant GNN for Materials Modeling**

Alexandre Duval[*1 2] Victor Schmidt[*2] Alex Hernandez Garcia[2] Santiago Miret[3] Fragkiskos D. Malliaros[1]
Yoshua Bengio[2 4] David Rolnick[2 5]

ICML 2023

**Equivariance with Learned Canonicalization Functions**

Sékou-Oumar Kaba[*1 2] Arnab Kumar Mondal[*1 2] Yan Zhang[3] Yoshua Bengio[4 2] Siamak Ravanbakhsh[1 2]

ICML 2023

**A new perspective on building efficient and expressive 3D equivariant graph neural networks**

Weitao Du[1*] Yuanqi Du[2*] Limei Wang[3*] Dieqiao Feng[2] Guifeng Wang[4]
Shuiwang Ji[3] Carla P Gomes[2] Zhi-Ming Ma[1]

NeurIPS 2023

**A Canonicalization Perspective on Invariant and Equivariant Learning**

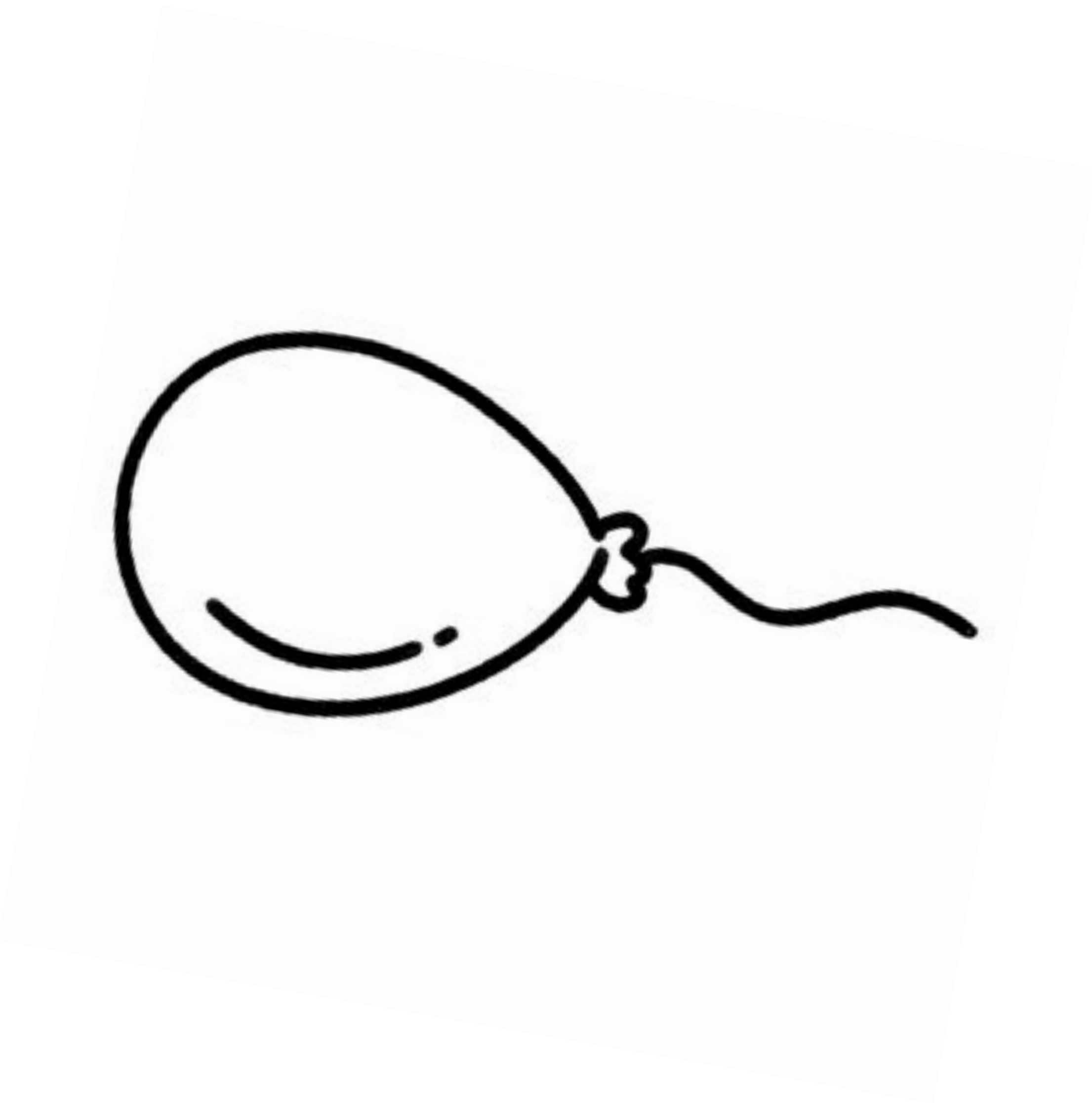
George Ma[*1] Yifei Wang[*2] Derek Lim[2] Stefanie Jegelka[3] Yisen Wang[4,5†]
[1] School of EECS, Peking University
[2] MIT CSAIL
[3] TUM CIT/MCML/MDSI & MIT EECS/CSAIL
[4] State Key Lab of General Artificial Intelligence,
School of Intelligence Science and Technology, Peking University
[5] Institute for Artificial Intelligence, Peking University

NeurIPS 2024

# Canonicalization vs Equivariant Architecture

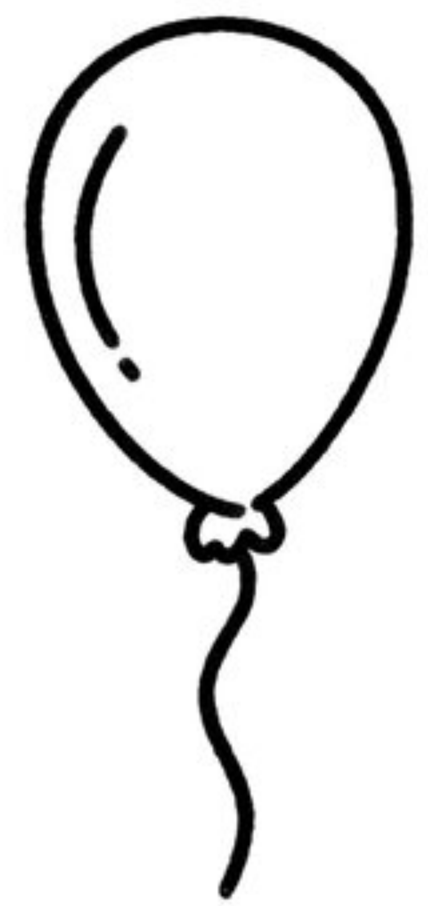
- Recall these aspects of equivariant architectures:
  - ~~Much more complicated than your average transformer~~ → **canonicalization is compatible with any downstream architecture**
  - ~~Specialized engineering required for normalizing, optimization, GPU usage~~ → **canonicalization is a preprocessing step, so maybe standard optimization practices will suffice?**
  - ~~No way of turning a pretrained black-box (closed source) architecture into an equivariant one~~ → **canonicalization is independent of the architecture weights; it only affects the input and the output**

# How to canonicalize?



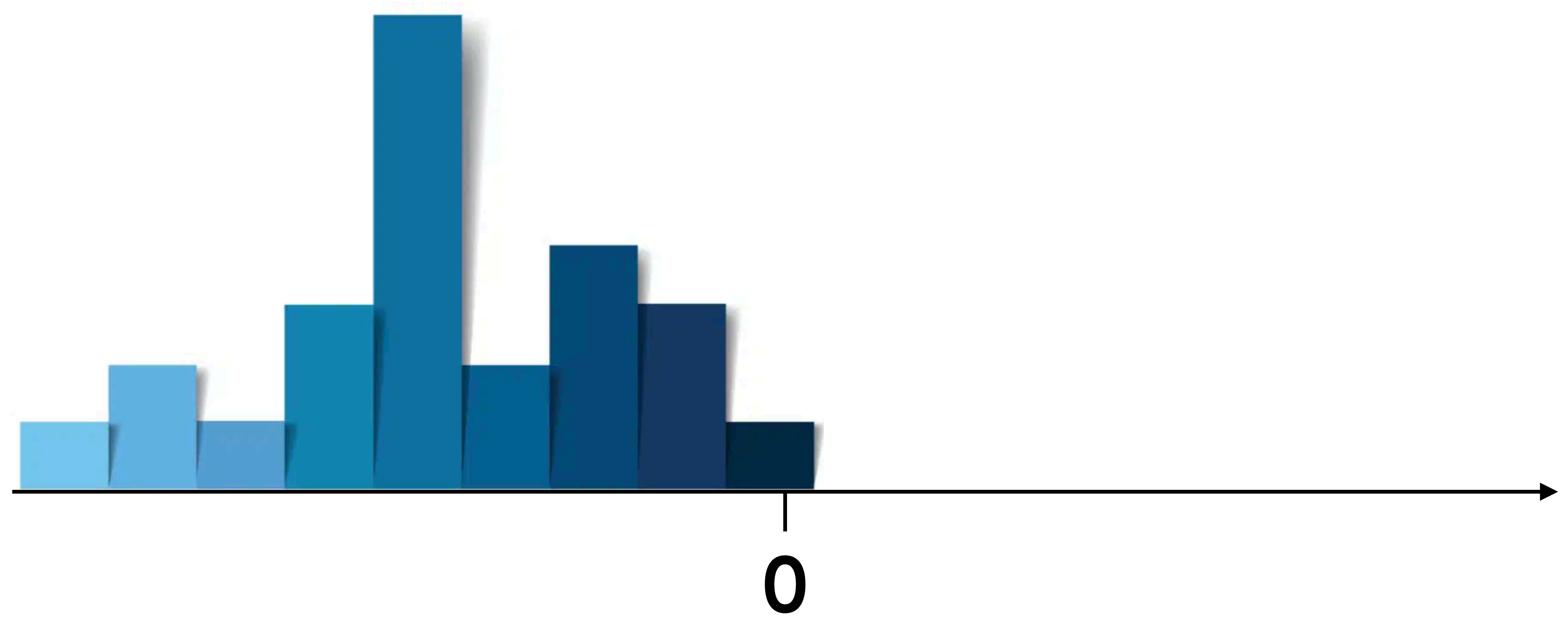
“Put the round inflated part at the top”

# How to canonicalize?



“Put the round inflated part at the top”

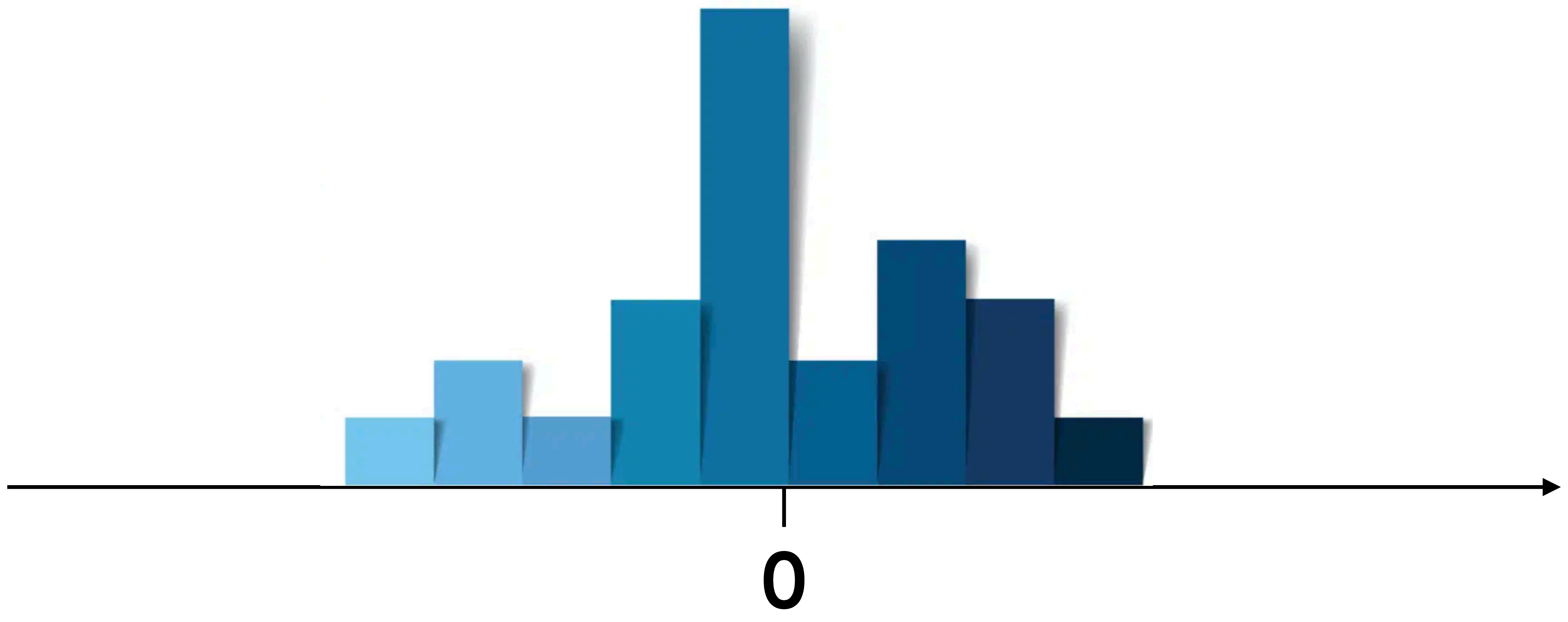
# How to canonicalize?




With respect to translations

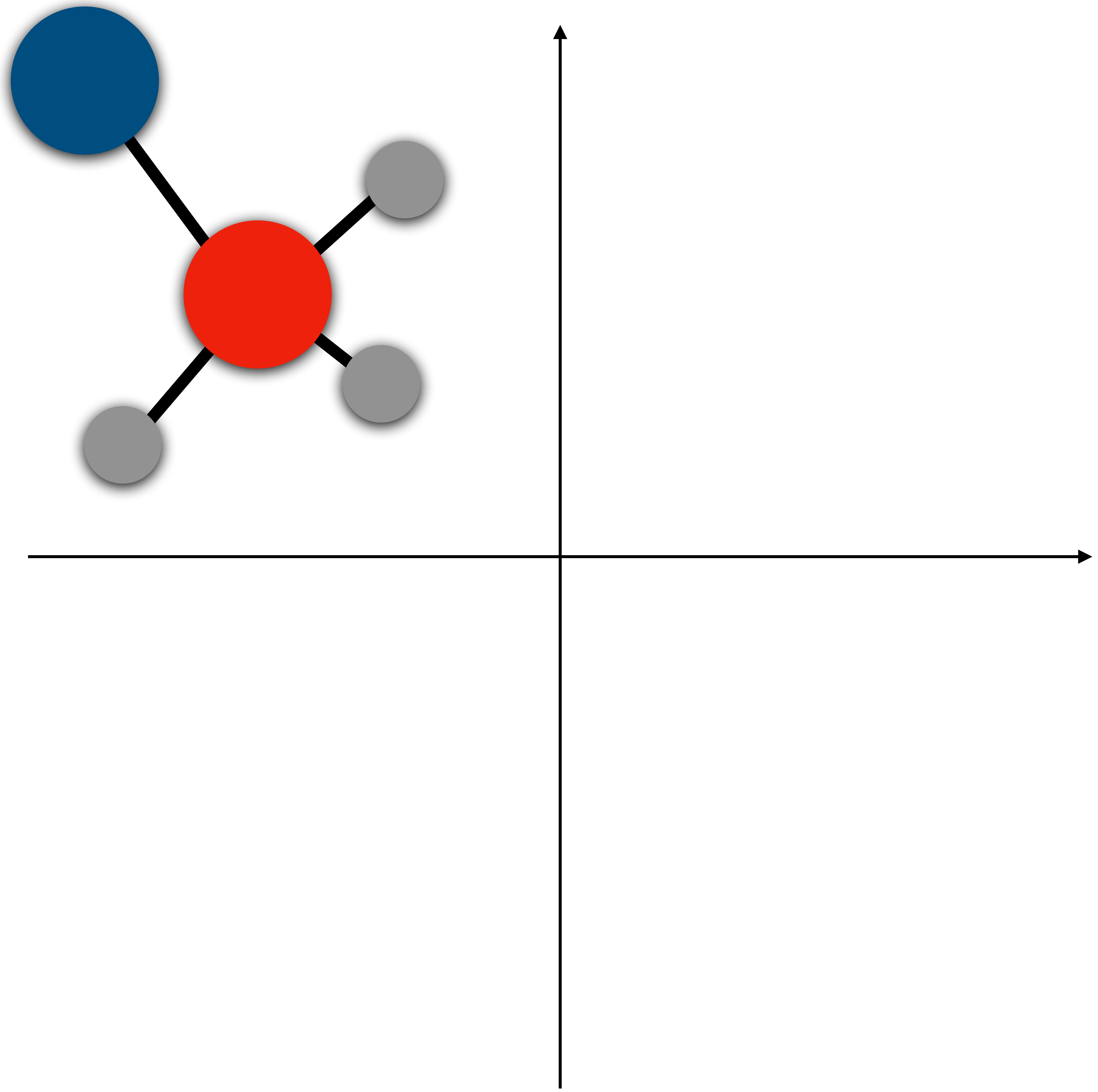
# How to canonicalize?

“Put the mean at $0$”



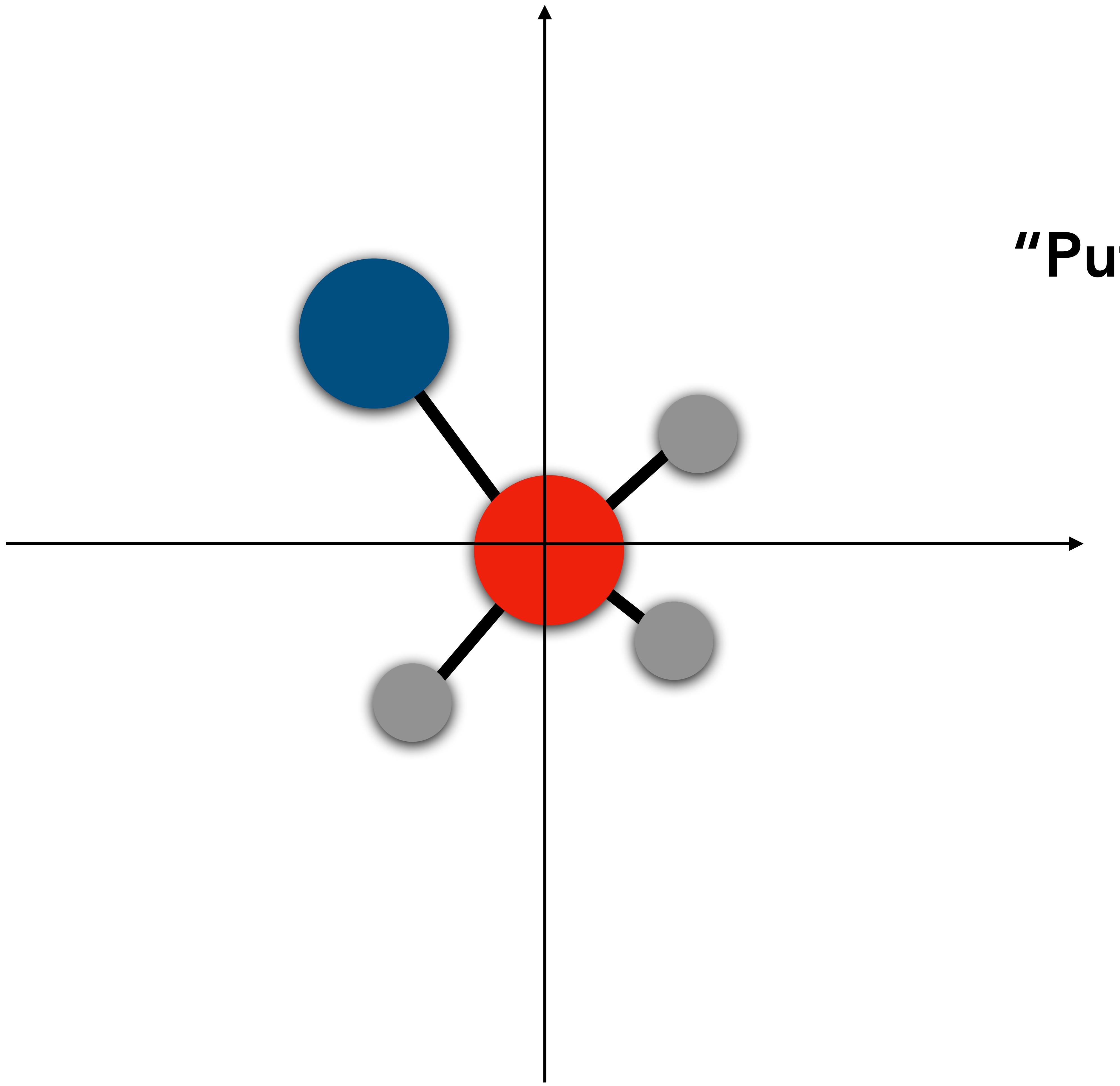

With respect to translations

# How to canonicalize?



With respect to translations

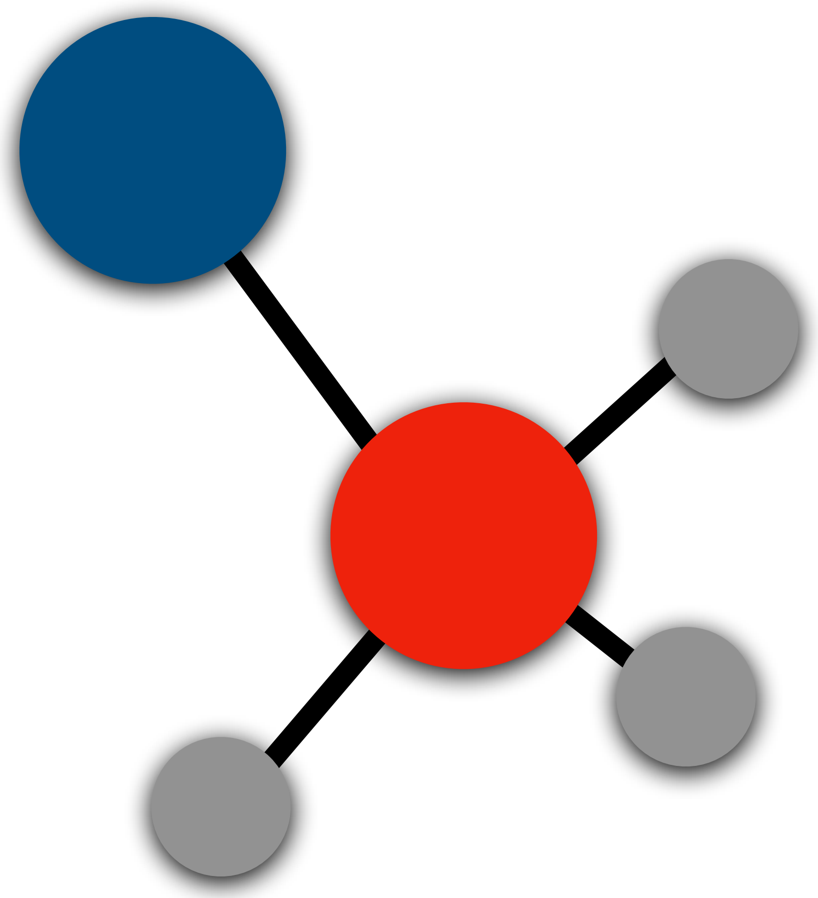
# How to canonicalize?



"Put the central atom at $(0,0)$"

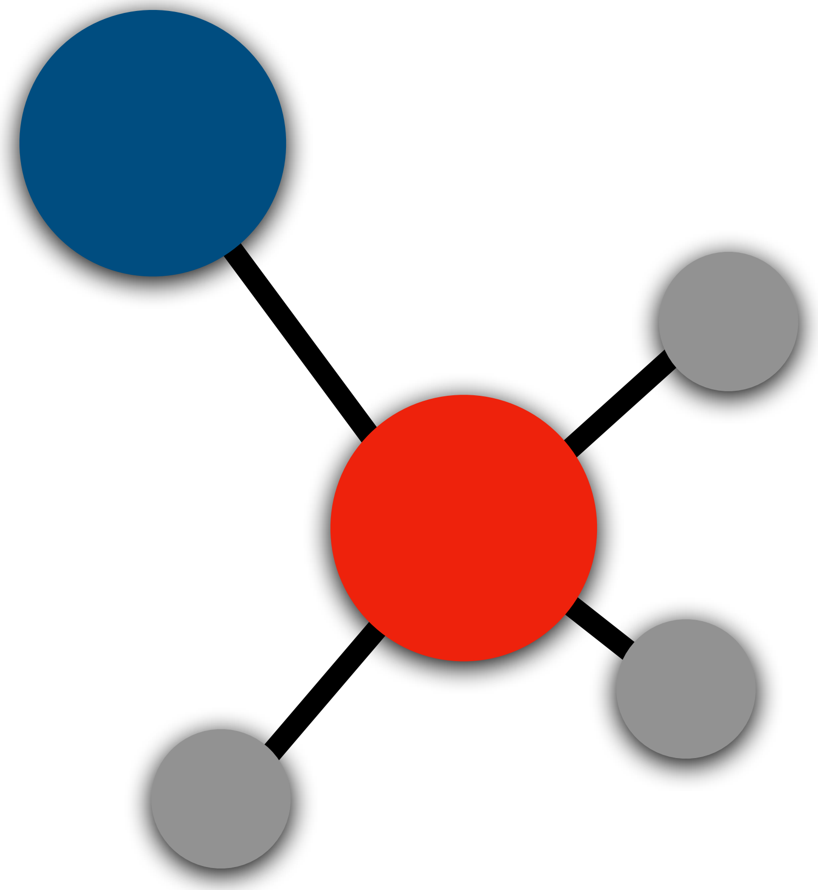
With respect to translations

# How to canonicalize?



With respect to <u>rotations</u>

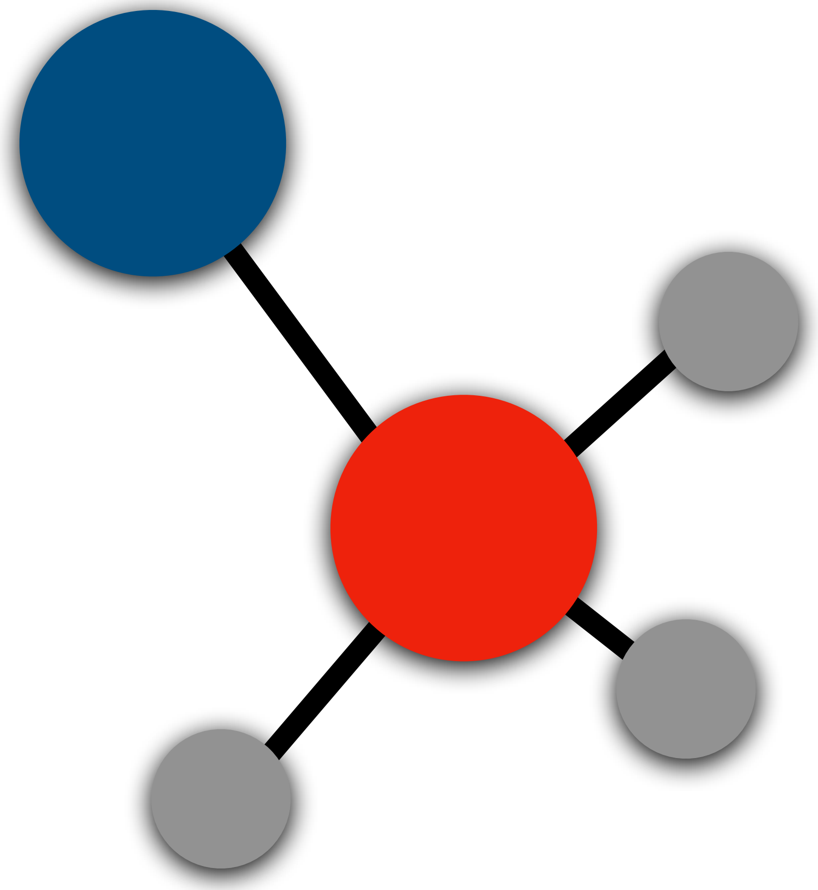
# How to canonicalize?



????

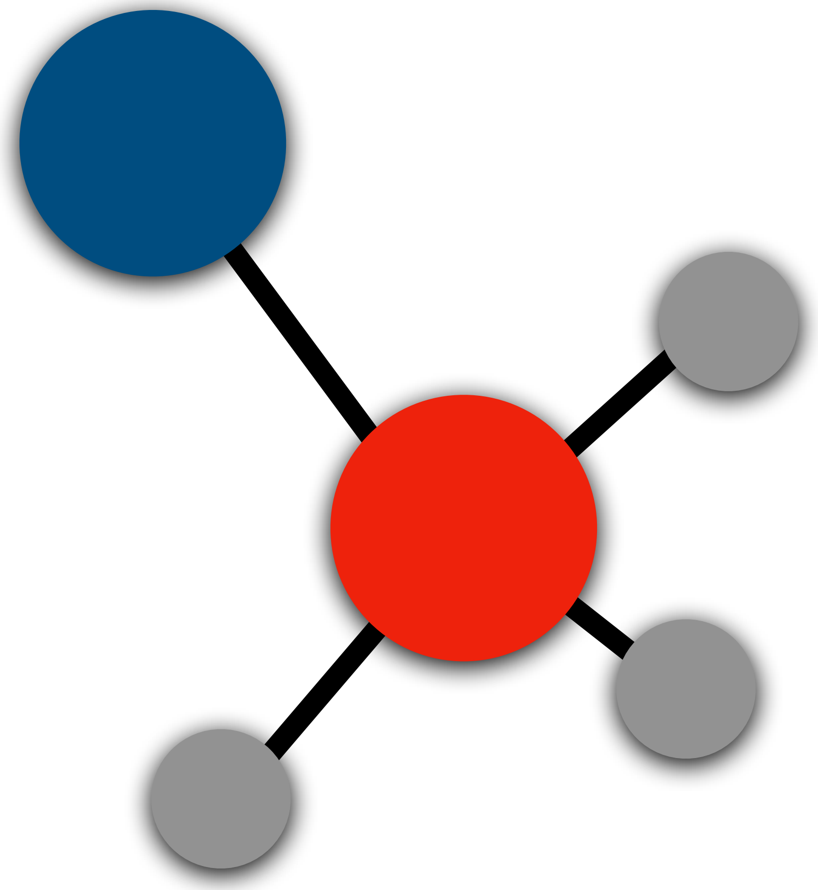
**With respect to <u>rotations</u>**

# How to canonicalize?



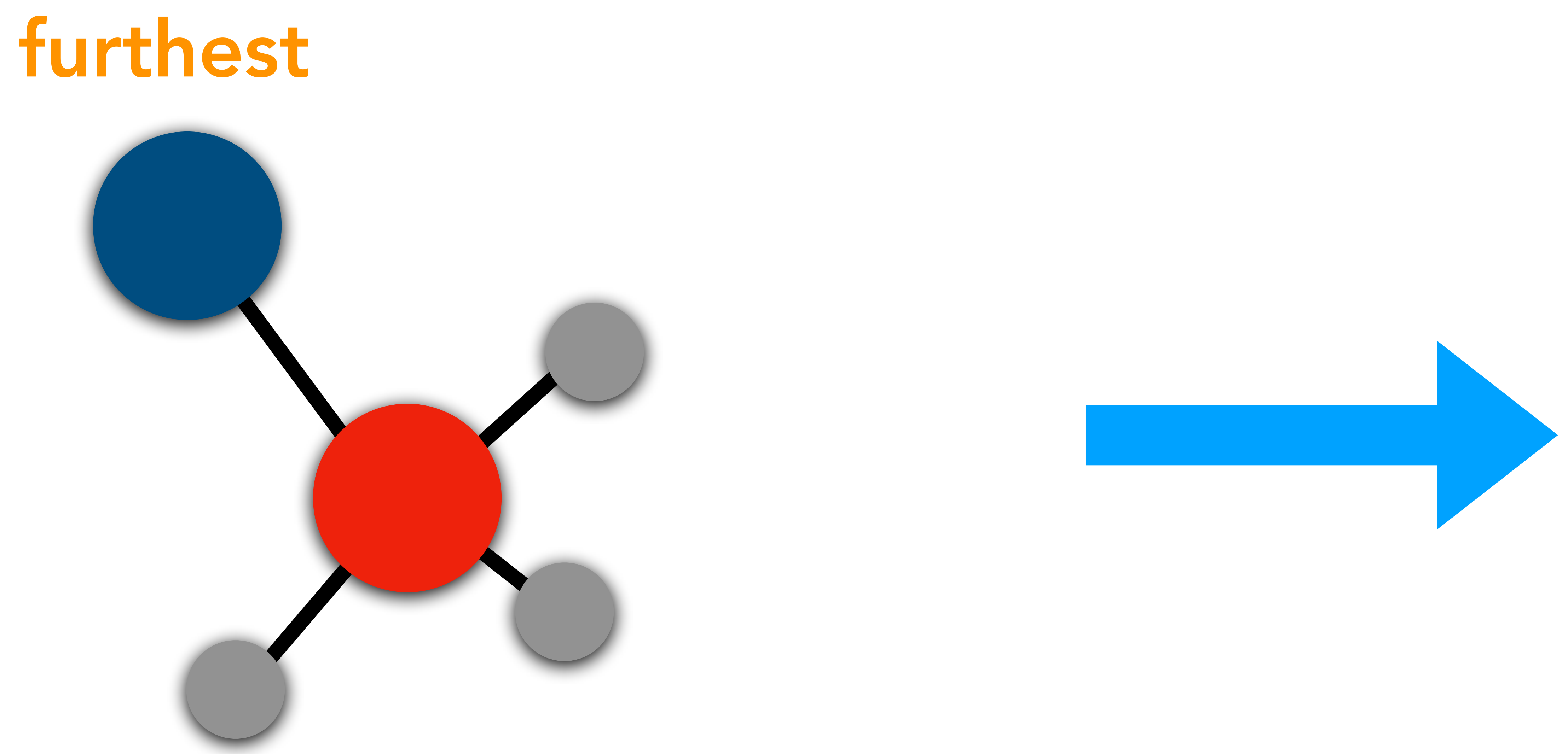
"Rotate so that the point furthest from the center is on the x-axis"

# How to canonicalize?



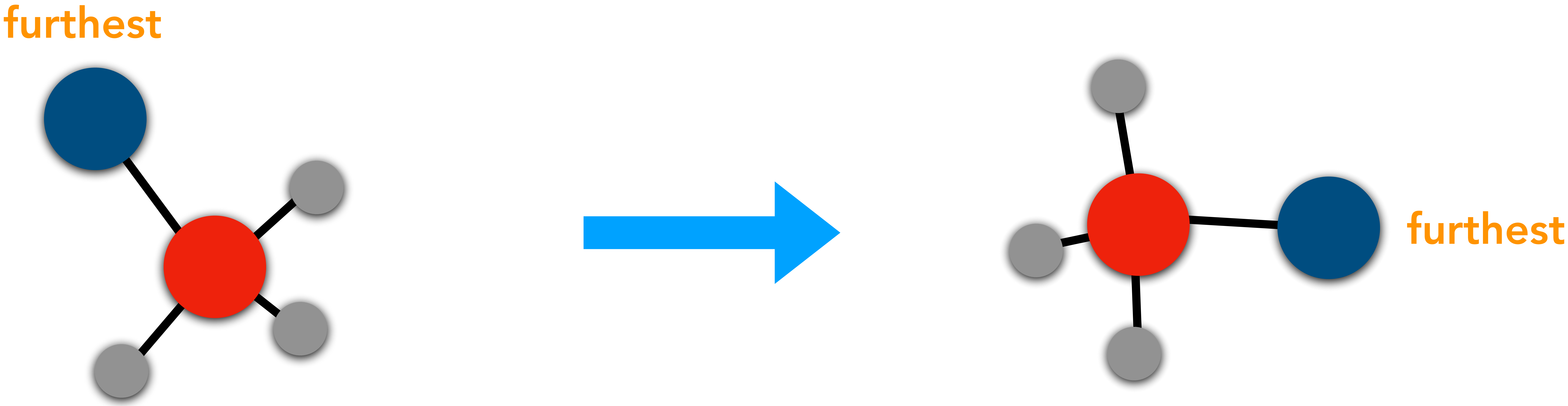

"Rotate so that the point furthest from the center is on the x-axis"

# How to canonicalize?



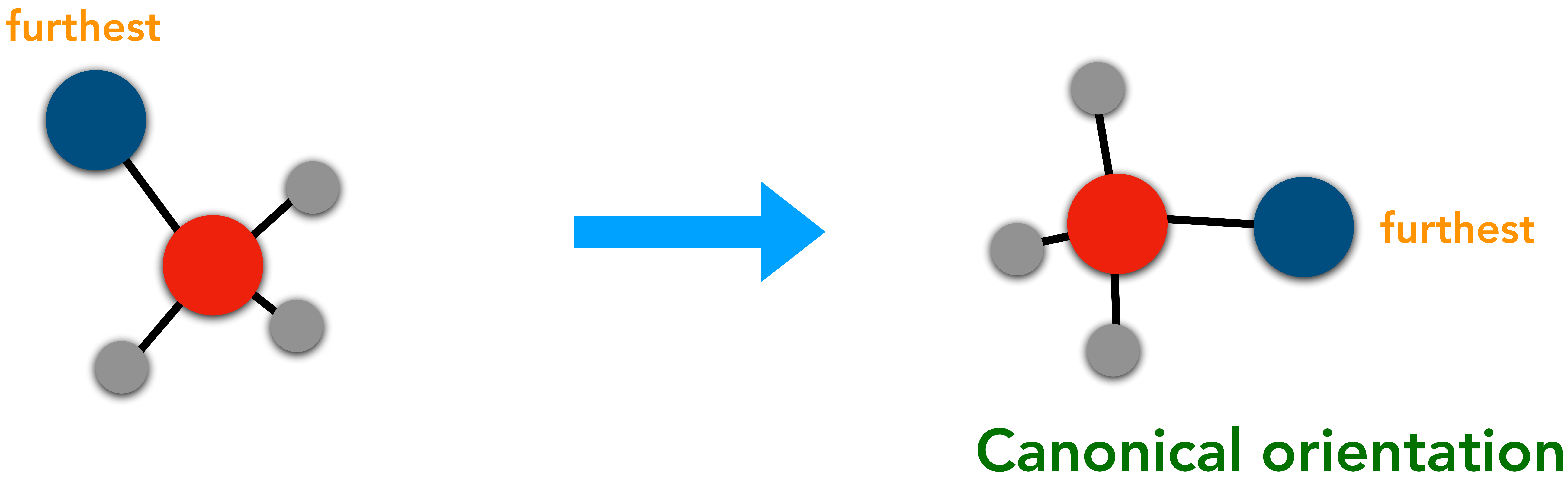

"Rotate so that the point furthest from the center is on the x-axis"

# How to canonicalize?



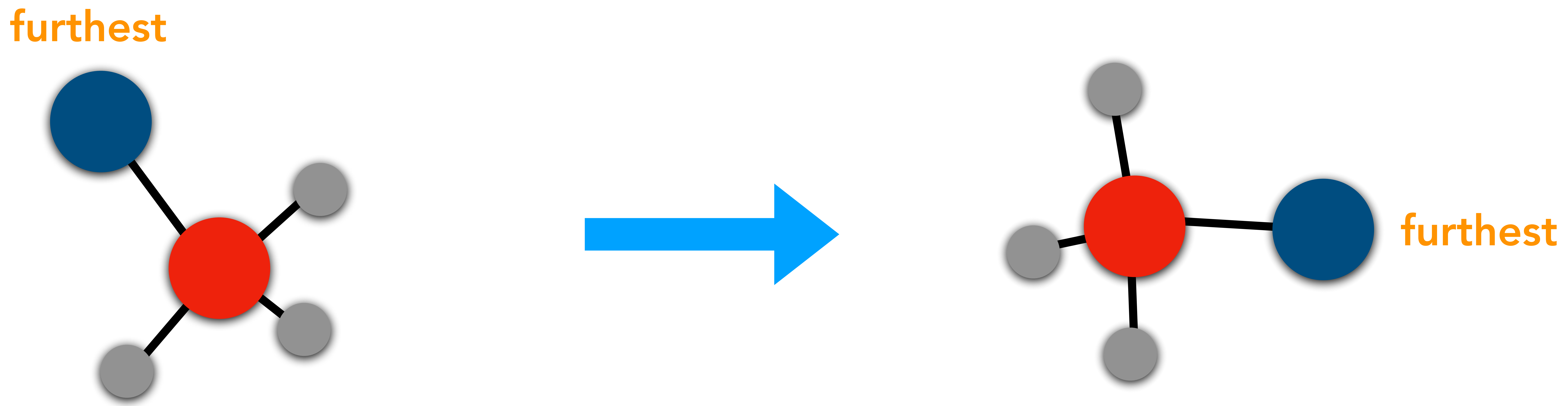

"Rotate so that the point furthest from the center is on the x-axis"

# How to canonicalize?




"Rotate so that the point furthest from the center is on the x-axis"

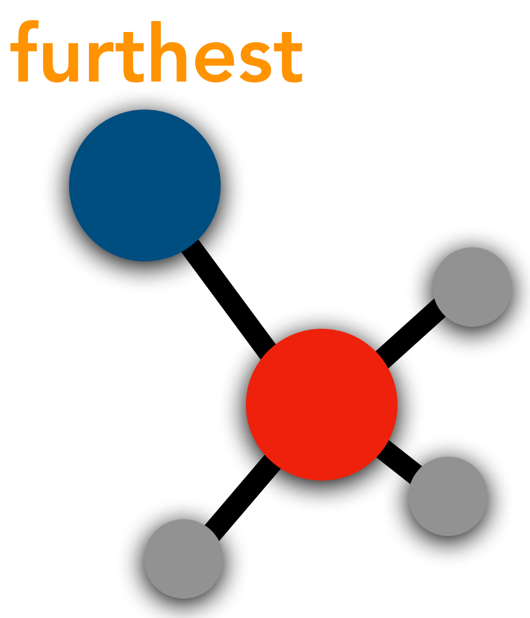
# How to canonicalize?



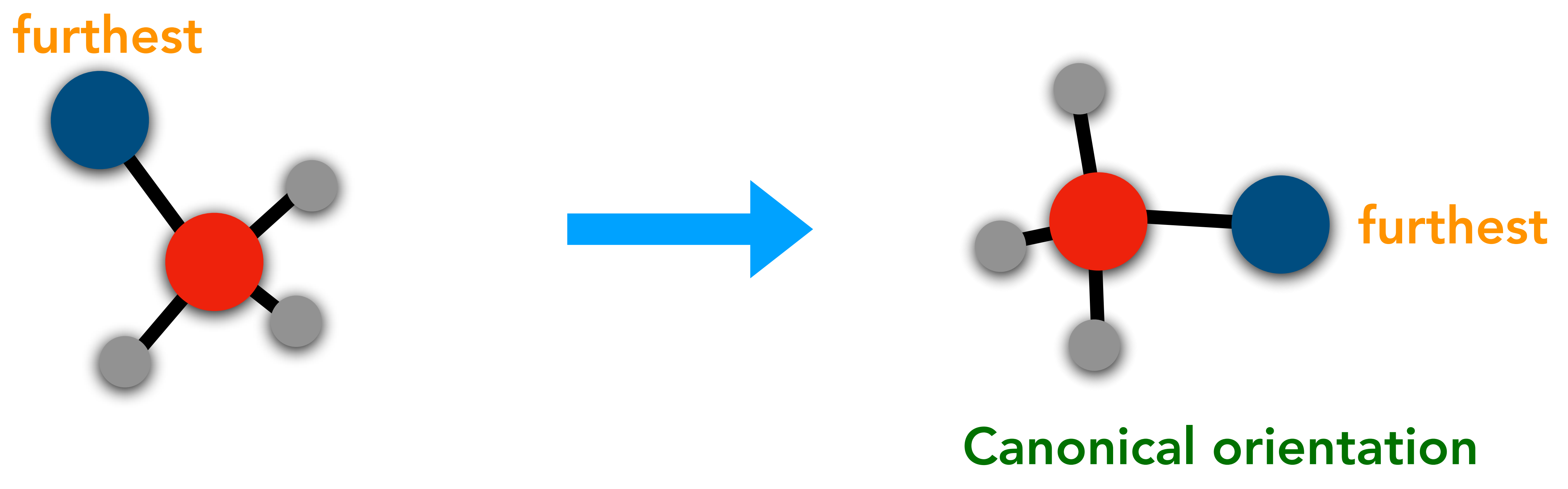

Canonical orientation

"Rotate so that the point furthest from the center is on the x-axis"

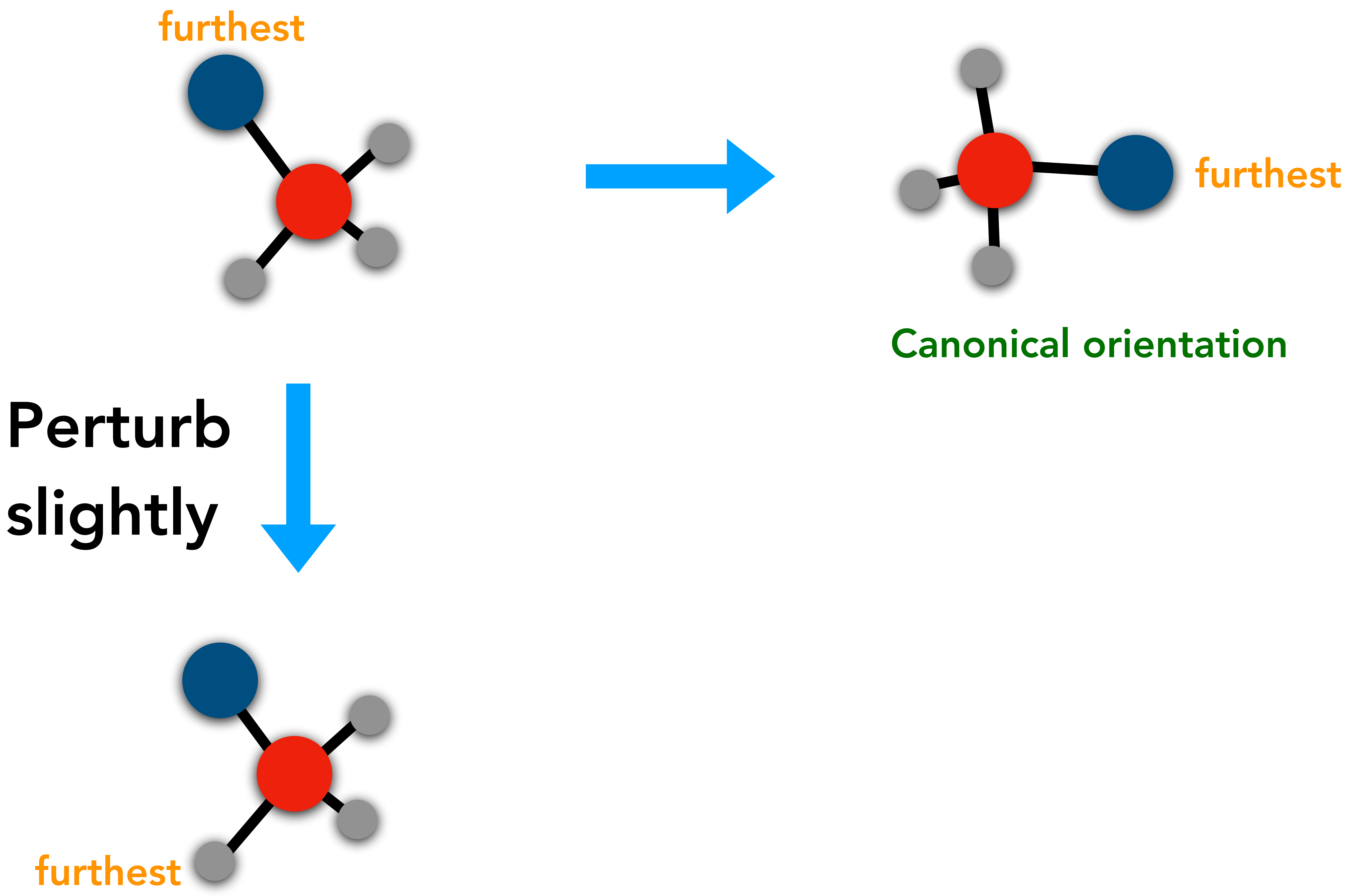
# How to canonicalize?



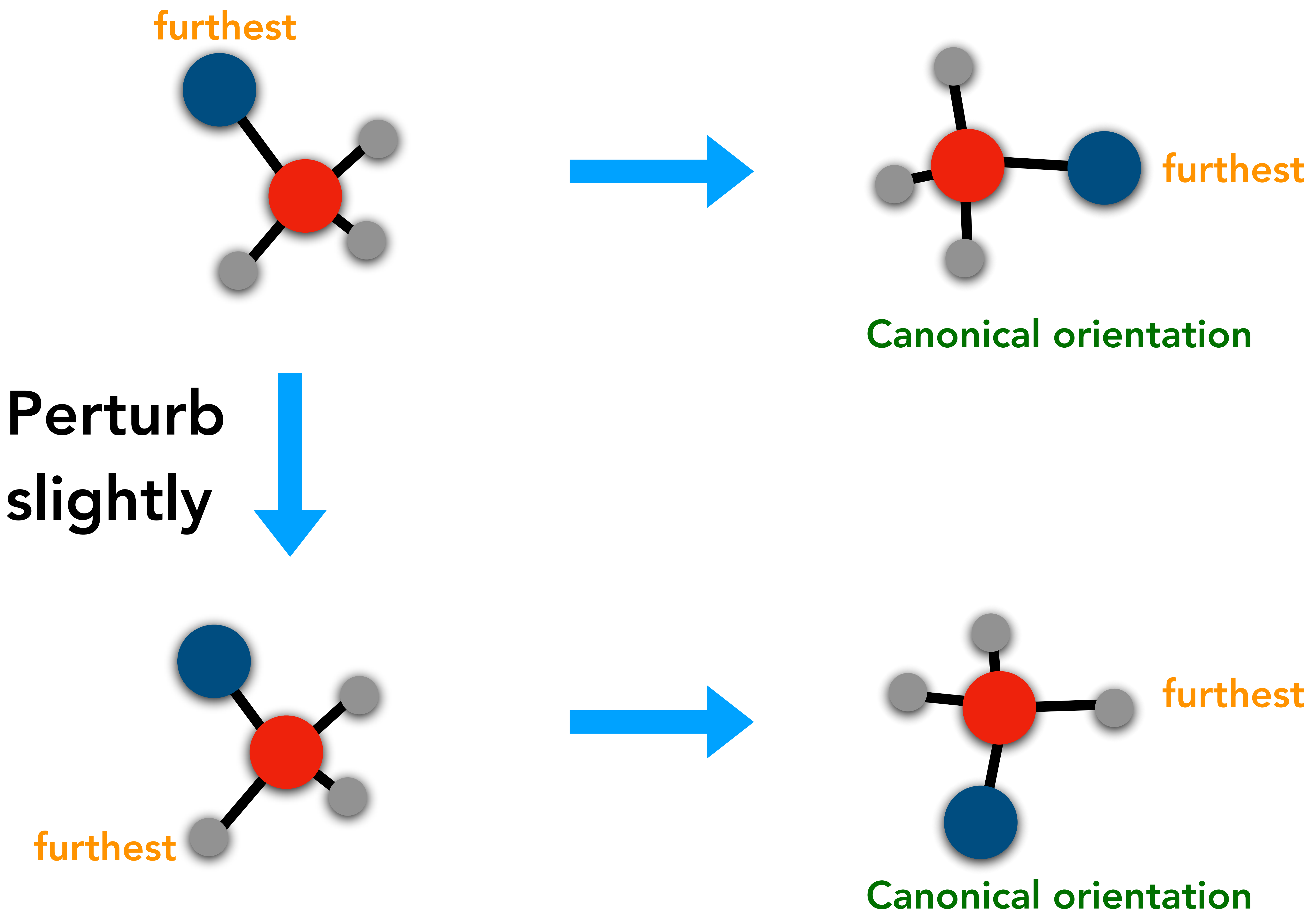
# How to canonicalize?



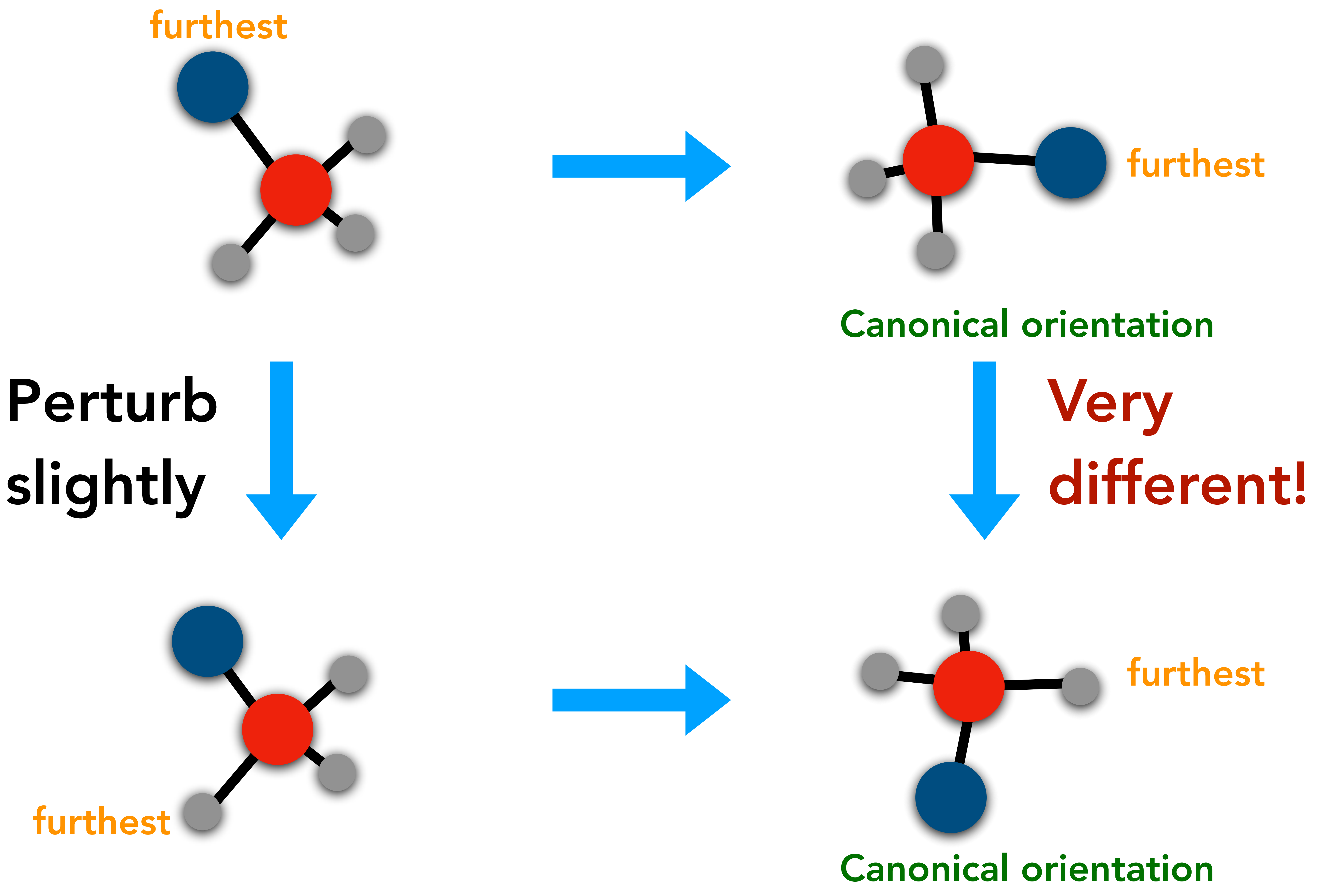
# How to canonicalize?



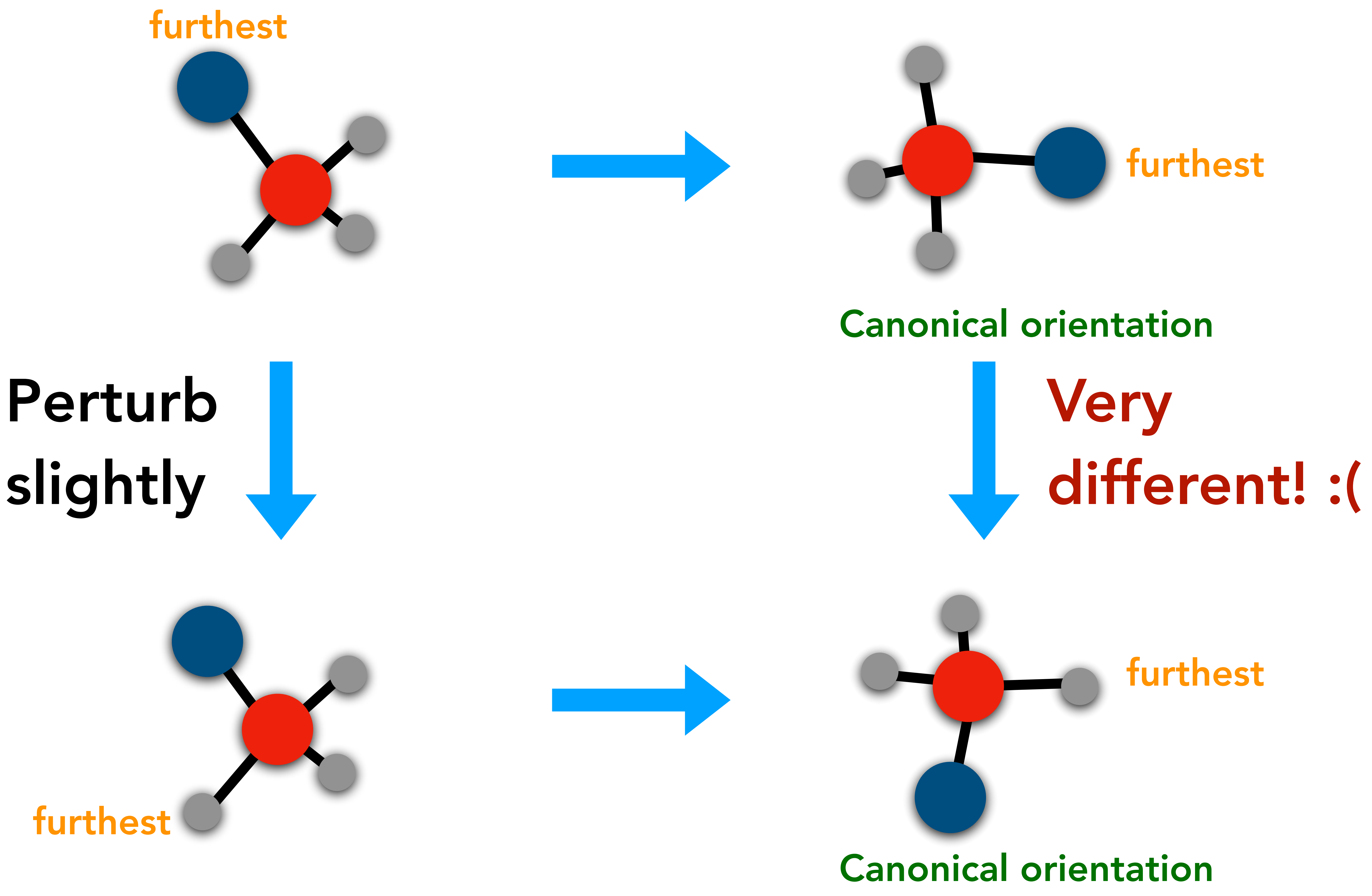
# How to canonicalize?



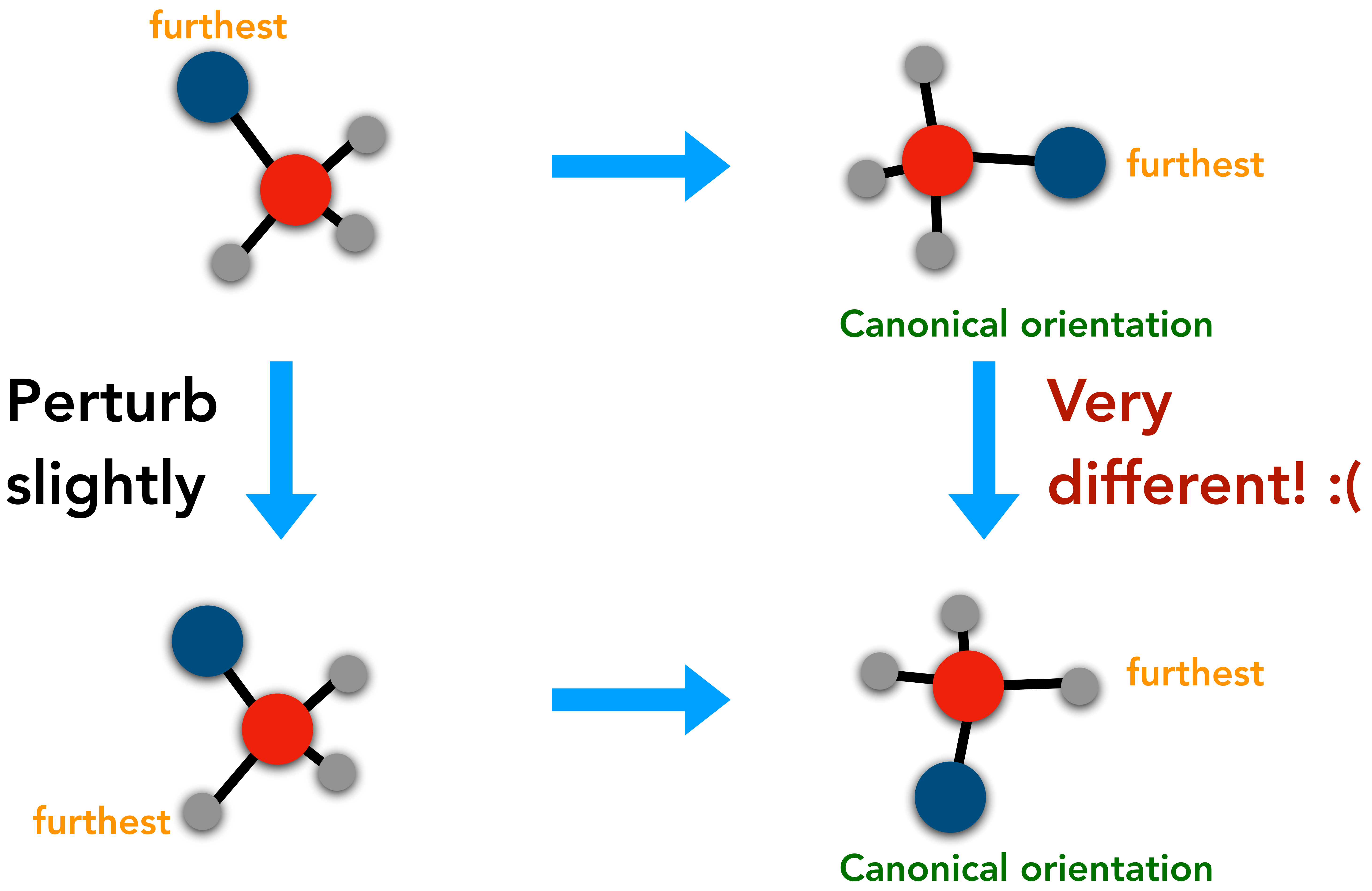
# How to canonicalize?



# How to canonicalize?

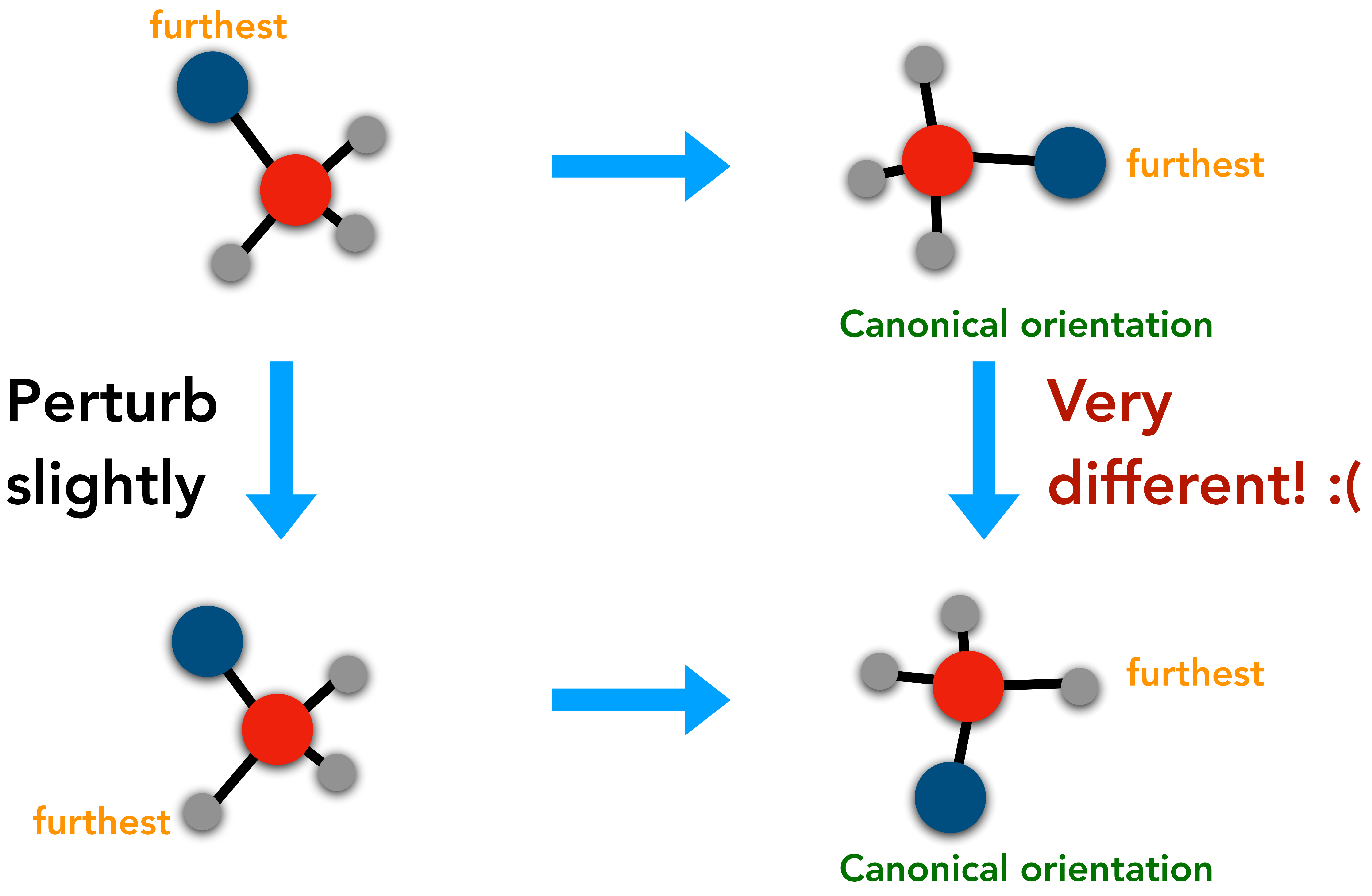


# How to canonicalize?



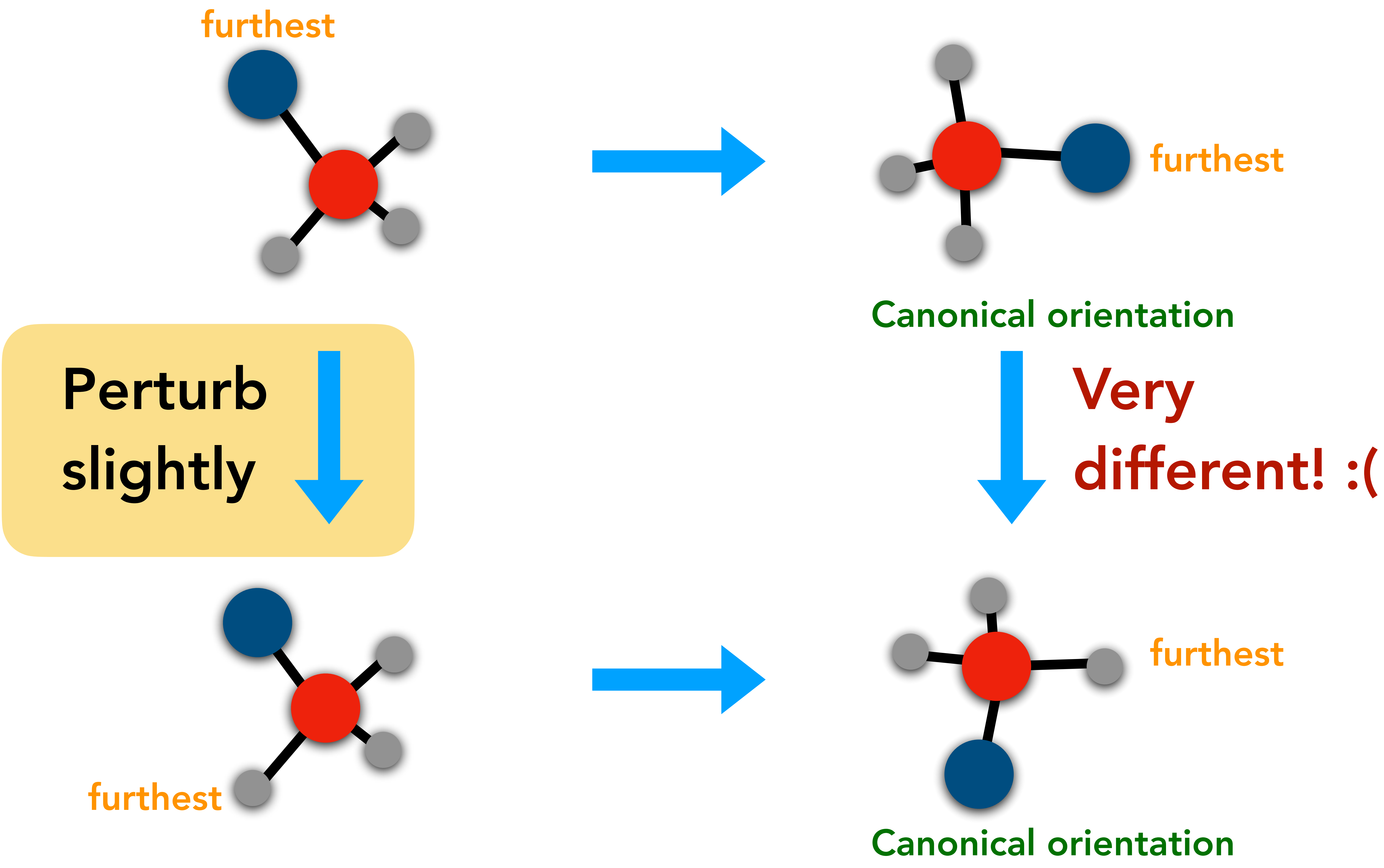

Continuity means: small change in input → small change in output. So, not continuous!

# How to canonicalize?



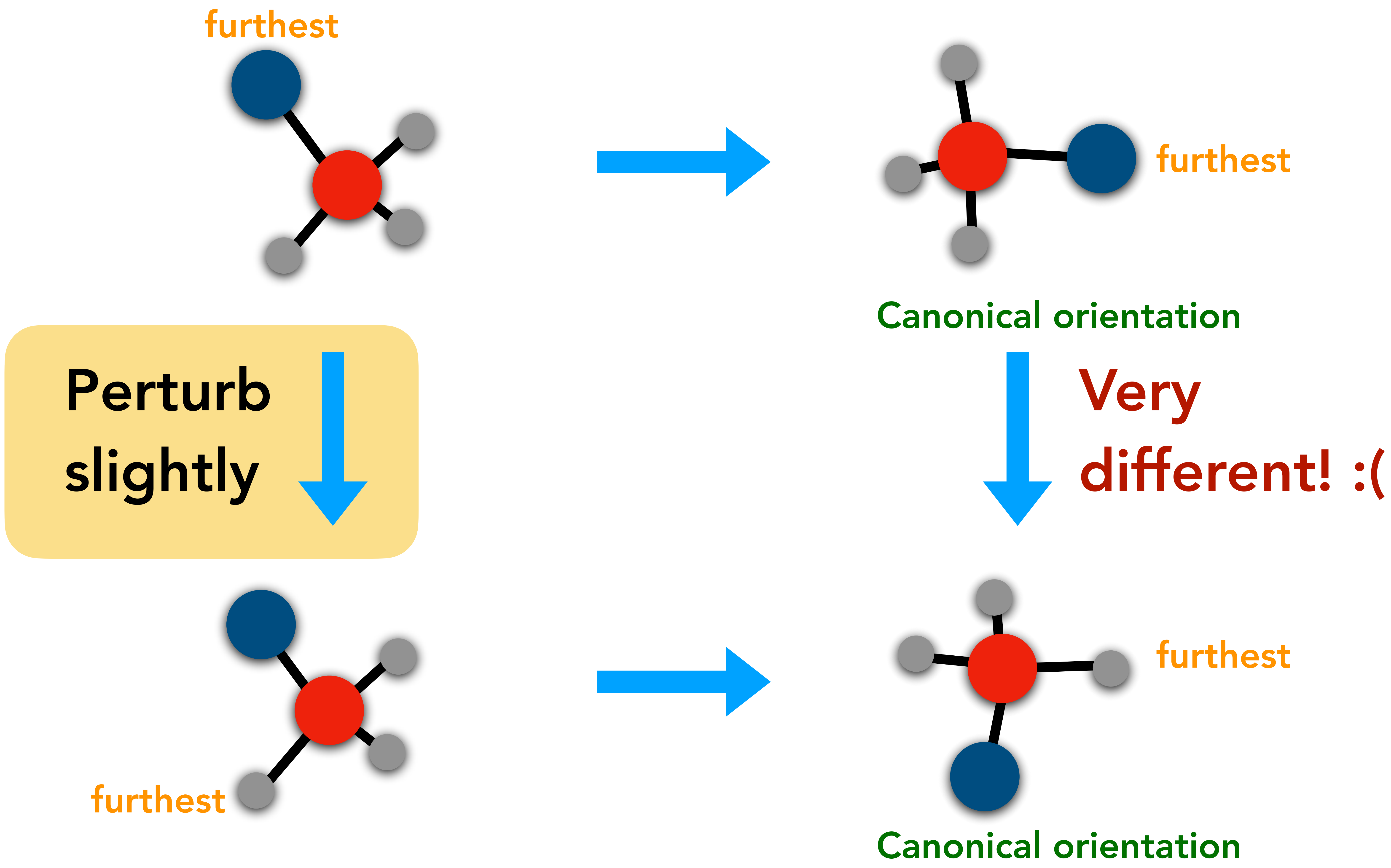

Continuity means: small change in input → small change in output. So, not continuous!

# How to canonicalize?



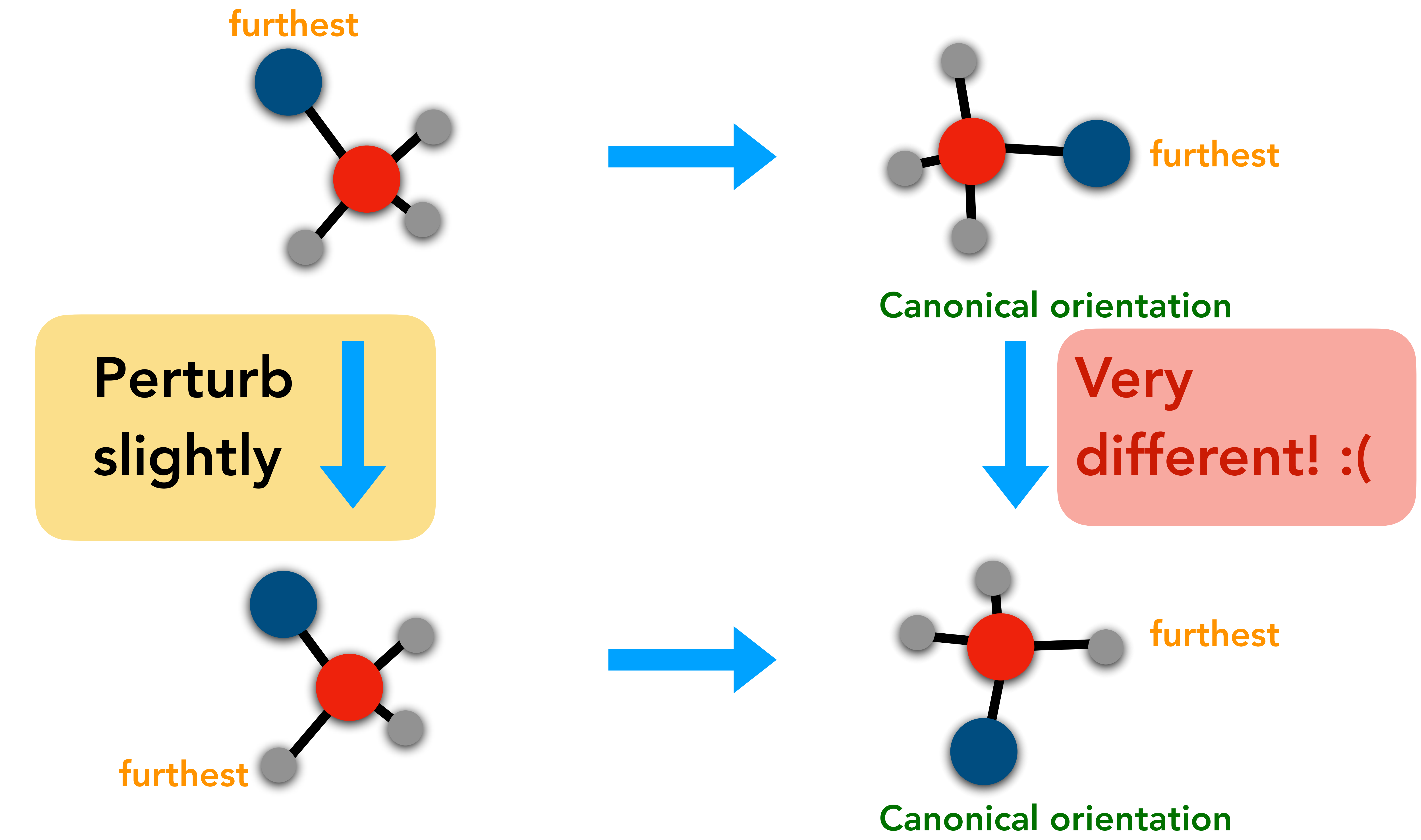

Continuity means: small change in input → small change in output. So, not continuous!

# How to canonicalize?




Continuity means: small change in input → small change in output. So, not continuous!

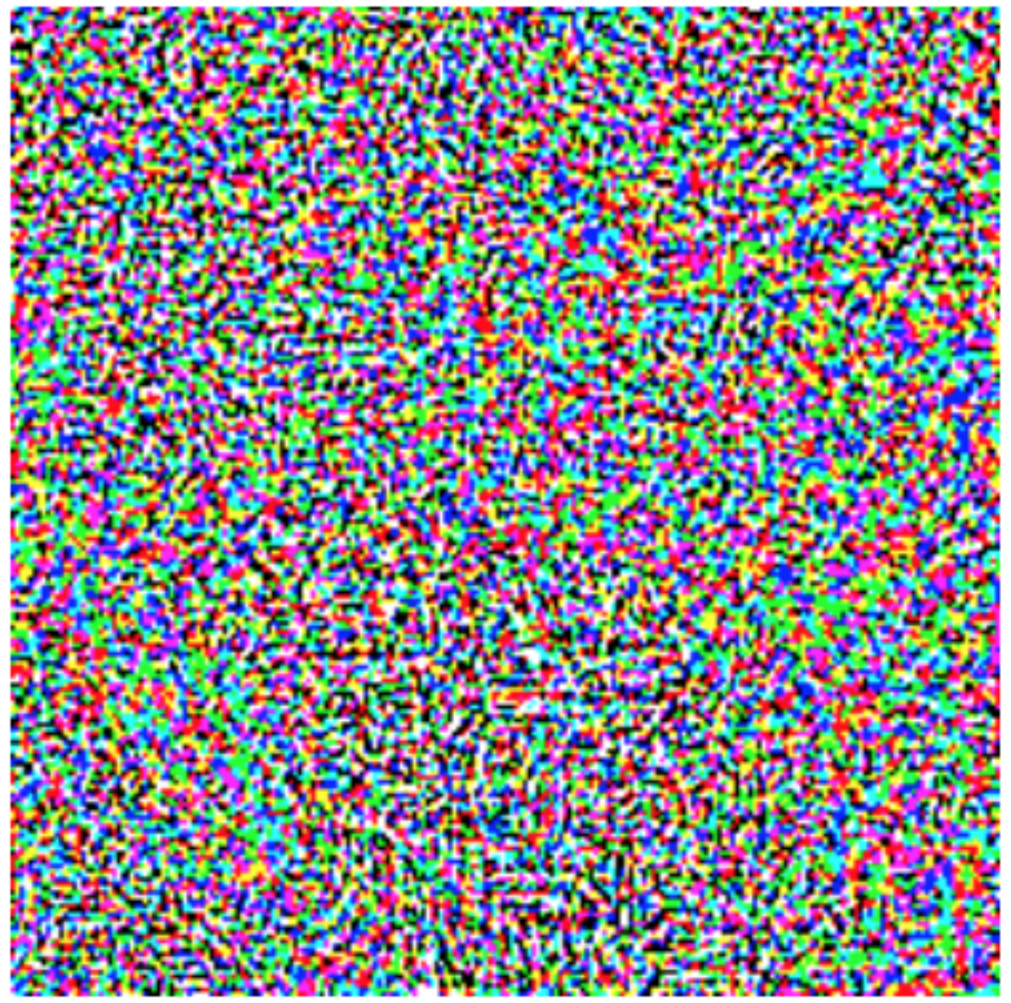
# How to canonicalize?




Continuity means: small change in input → small change in output. So, not continuous!

# Why is this bad?

$+ .007 \times$



$=$

"panda"
57.7% confidence

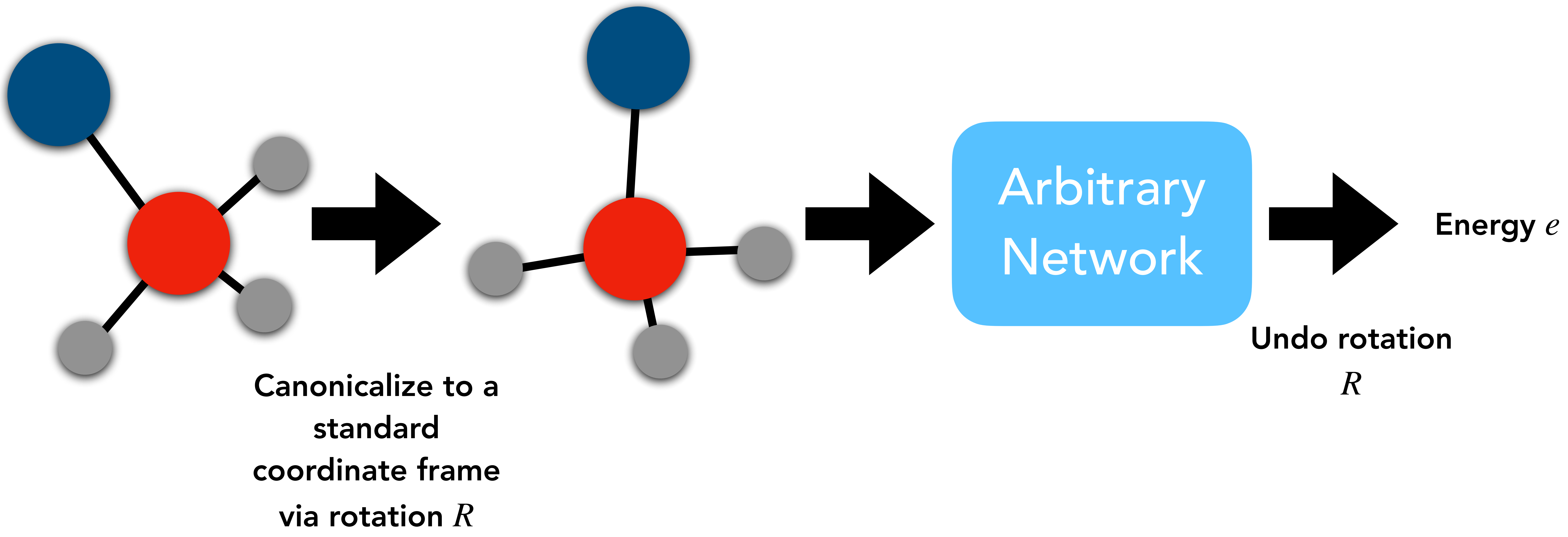
"nematode"
8.2% confidence

"gibbon"
99.3 % confidence

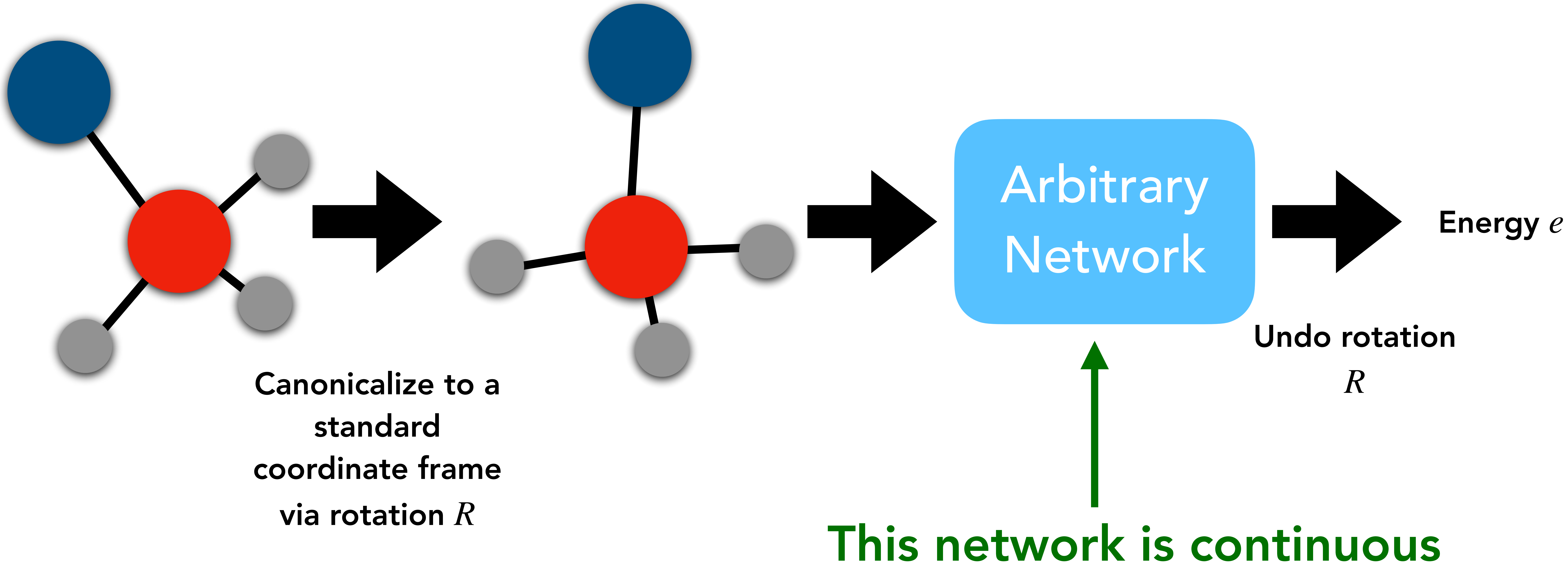
# How fundamental is this problem?

What if we were smarter?

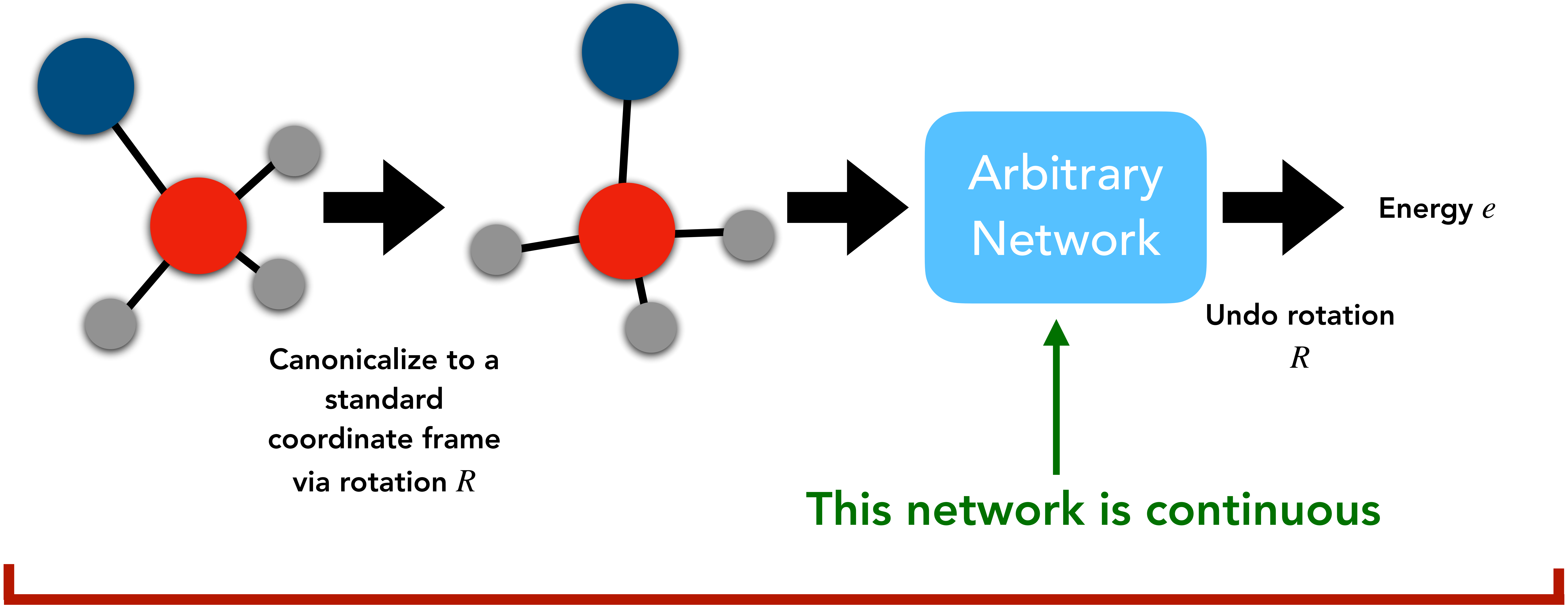
# The goal: end-to-end continuity



# The goal: end-to-end continuity

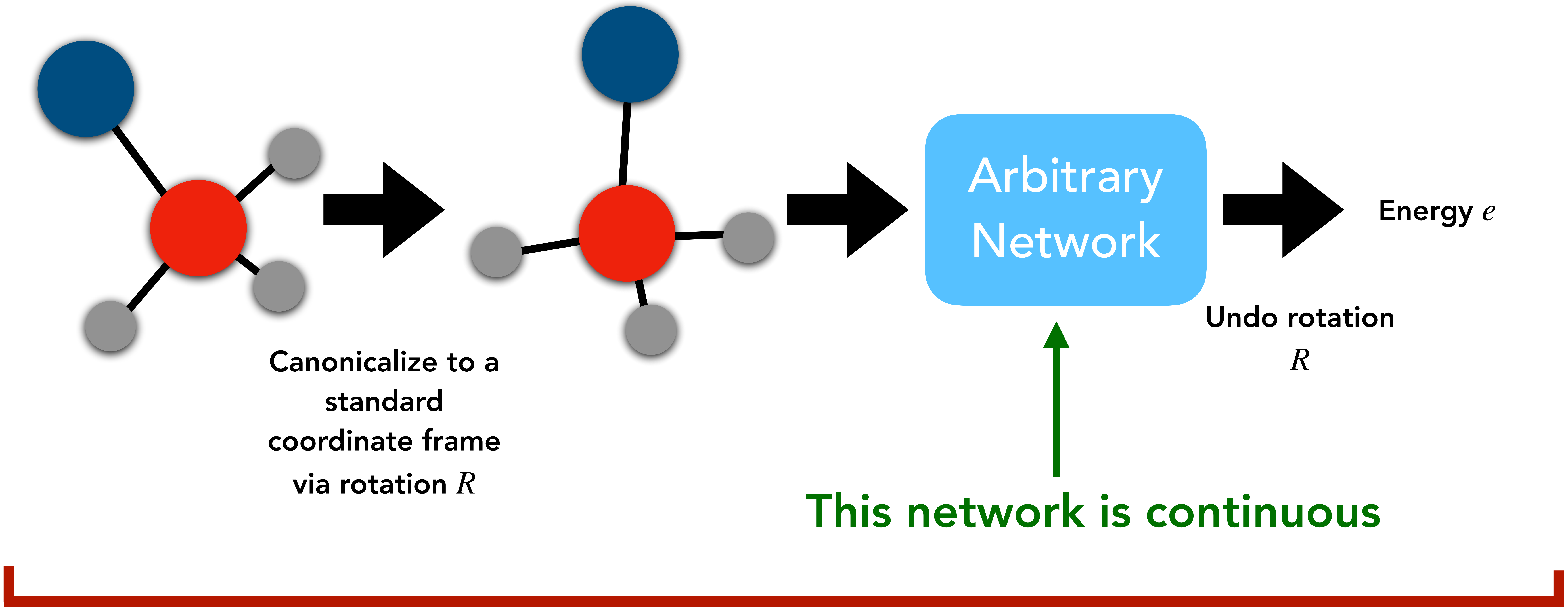


# The goal: end-to-end continuity




Is this end-to-end function continuous?

# The goal: end-to-end continuity




**Is this end-to-end function continuous?**

**Property: "continuity preservation"**

# Continuity is sometimes impossible!

- We show that canonicalization preserves end-to-end continuity iff the canonicalization mapping itself is continuous

- For certain symmetries, there is **no** continuous canonicalization:

    - Permutations on sets with features dimension $d \geq 2$

    - Rotations of ordered point clouds of $\geq 3$ points

Dym, Lawrence, and Siegel, ICML 2024

# How do we get back continuity?

It turns out: there's a tradeoff between computation time and continuity

# How do we get back continuity?

Canonicalization

**Discontinuous**

**Cheap**

# How do we get back continuity?

Canonicalization

Full data augmentation

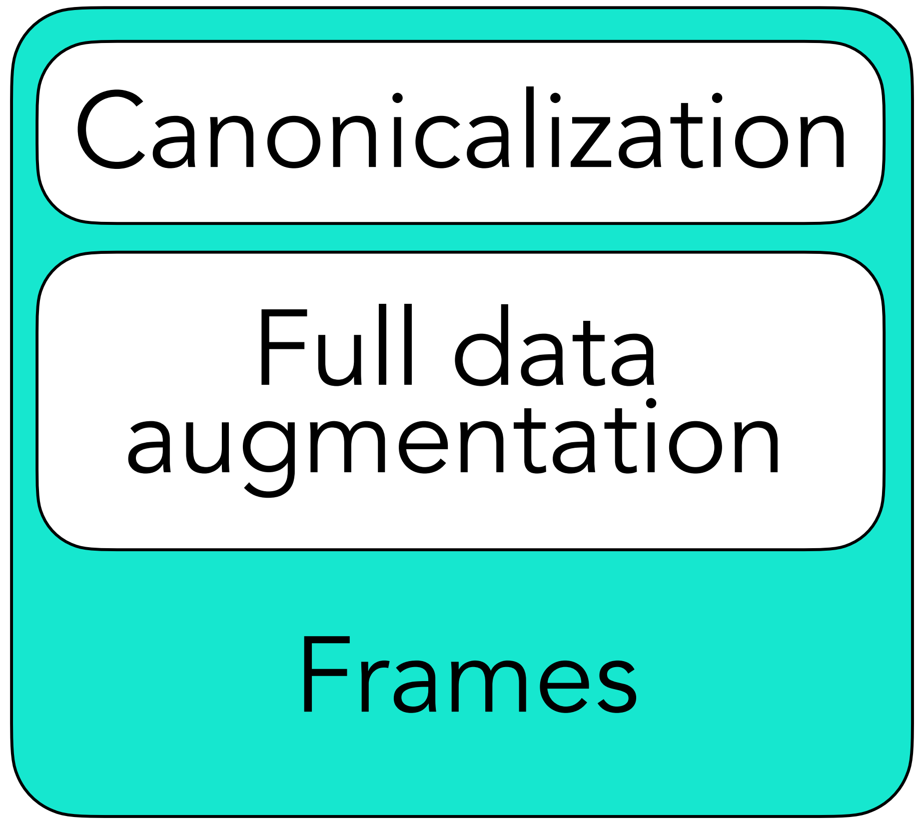
**Discontinuous**

**Continuous**

**Cheap**

**Expensive**

# How do we get back continuity?




Canonicalization | Frames | Full data augmentation

**Discontinuous** | **Continuity depends on size!** | **Continuous**

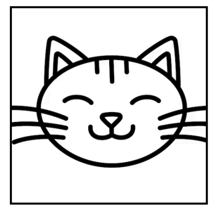
**Cheap** | **Cost depends on size!** | **Expensive**

# Frame-averaging

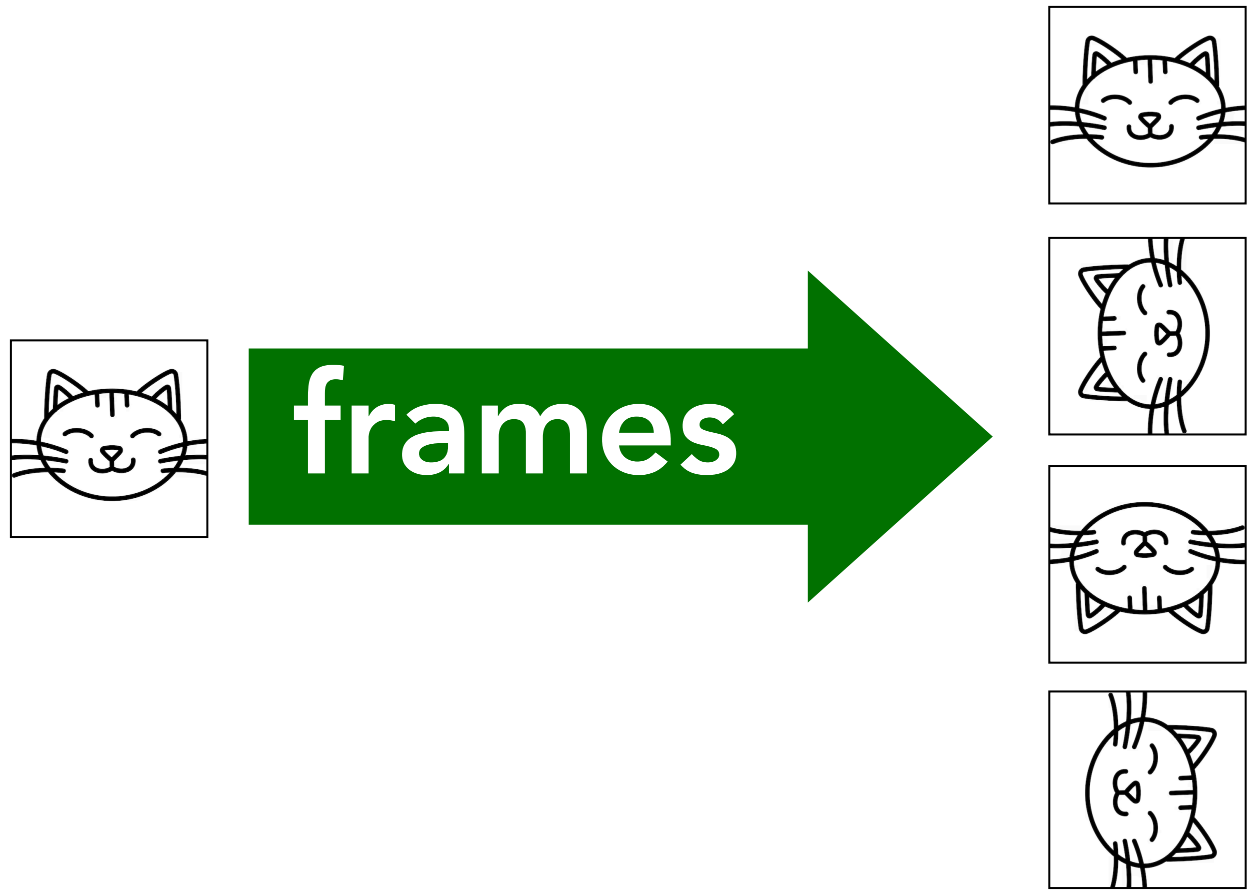
$$f(x) \mapsto \sum_{g \in \mathscr{F}(x)} f(g^{-1}x)$$

Subset of $G$, hopefully not all of $G$

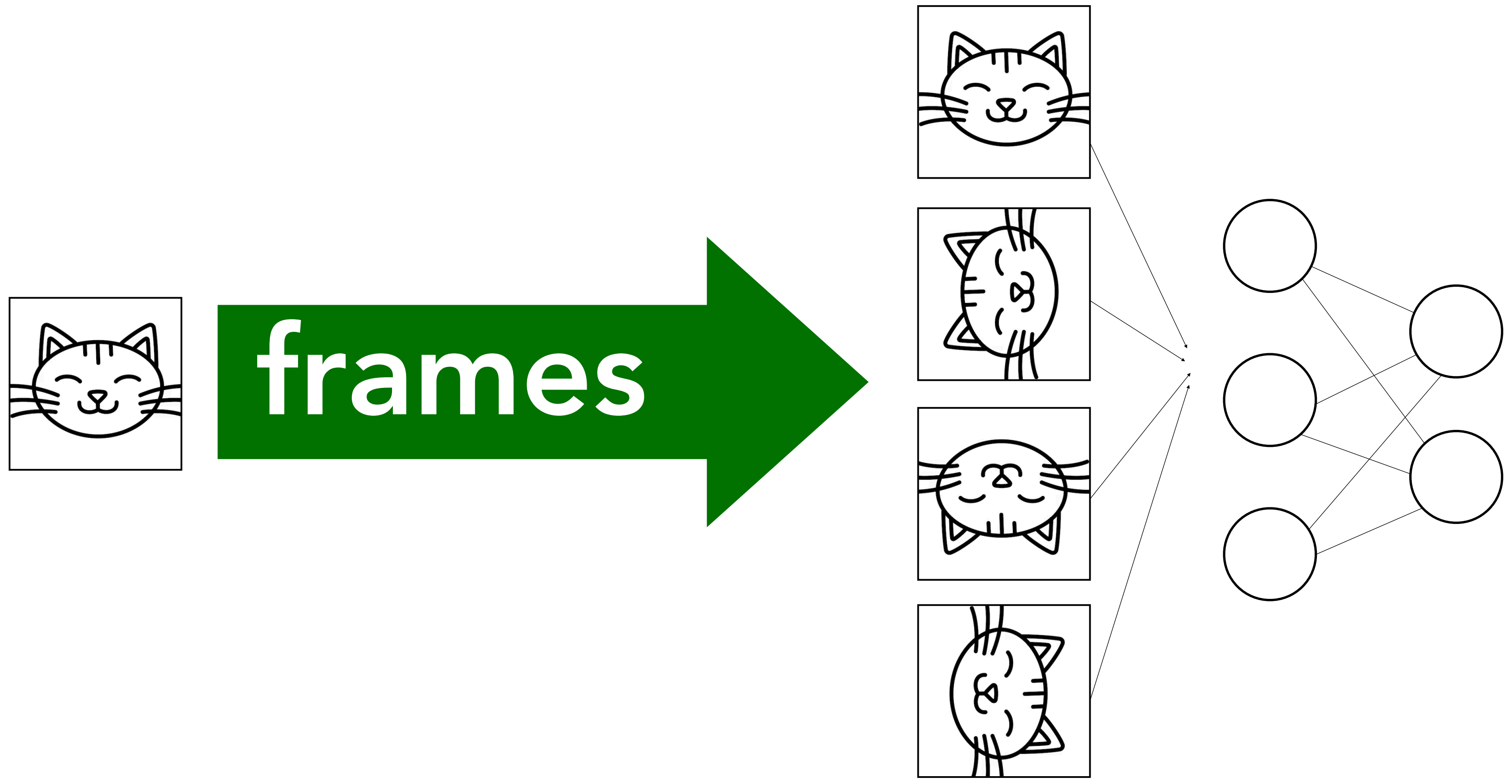
# Frame Averaging Illustration



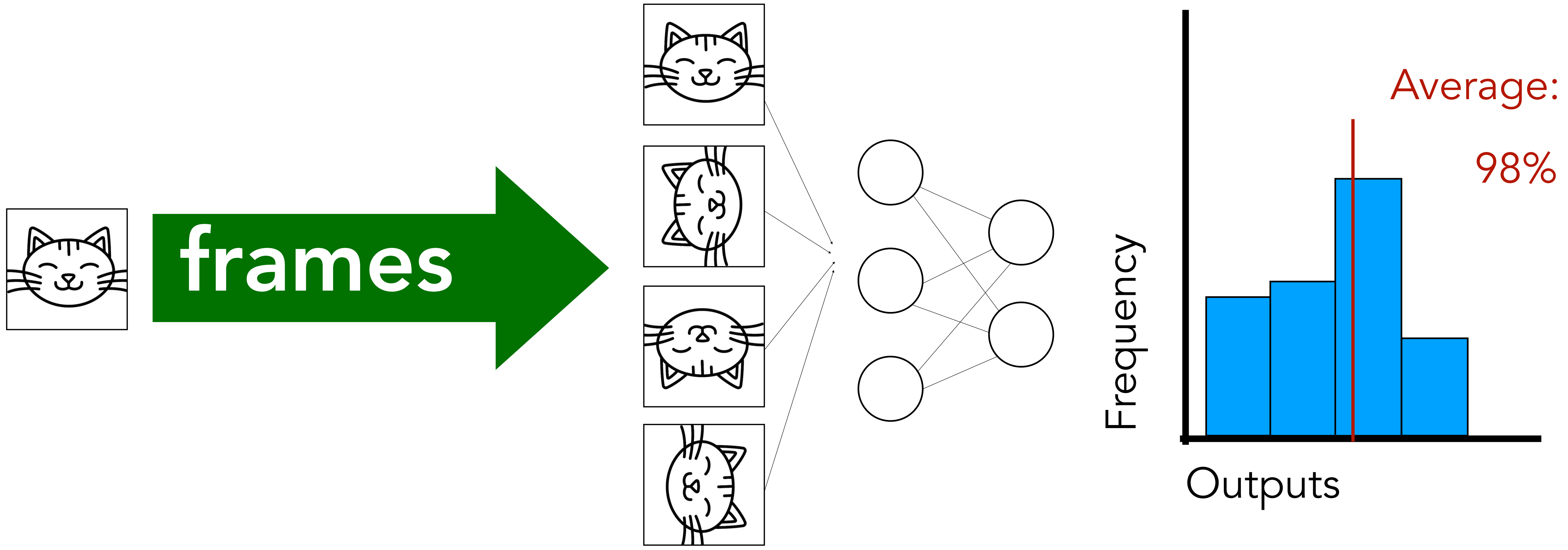
# Frame Averaging Illustration



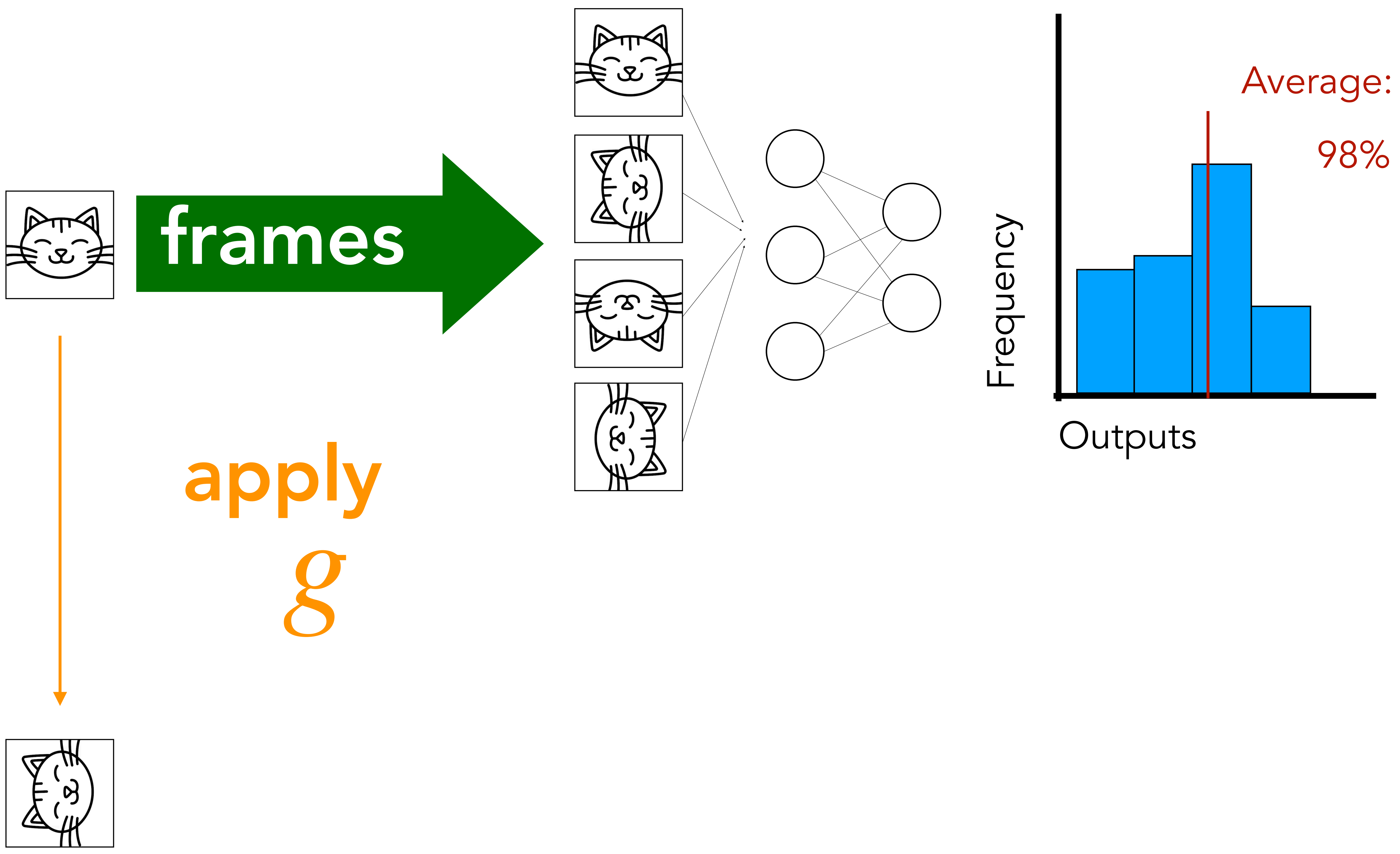
# Frame Averaging Illustration



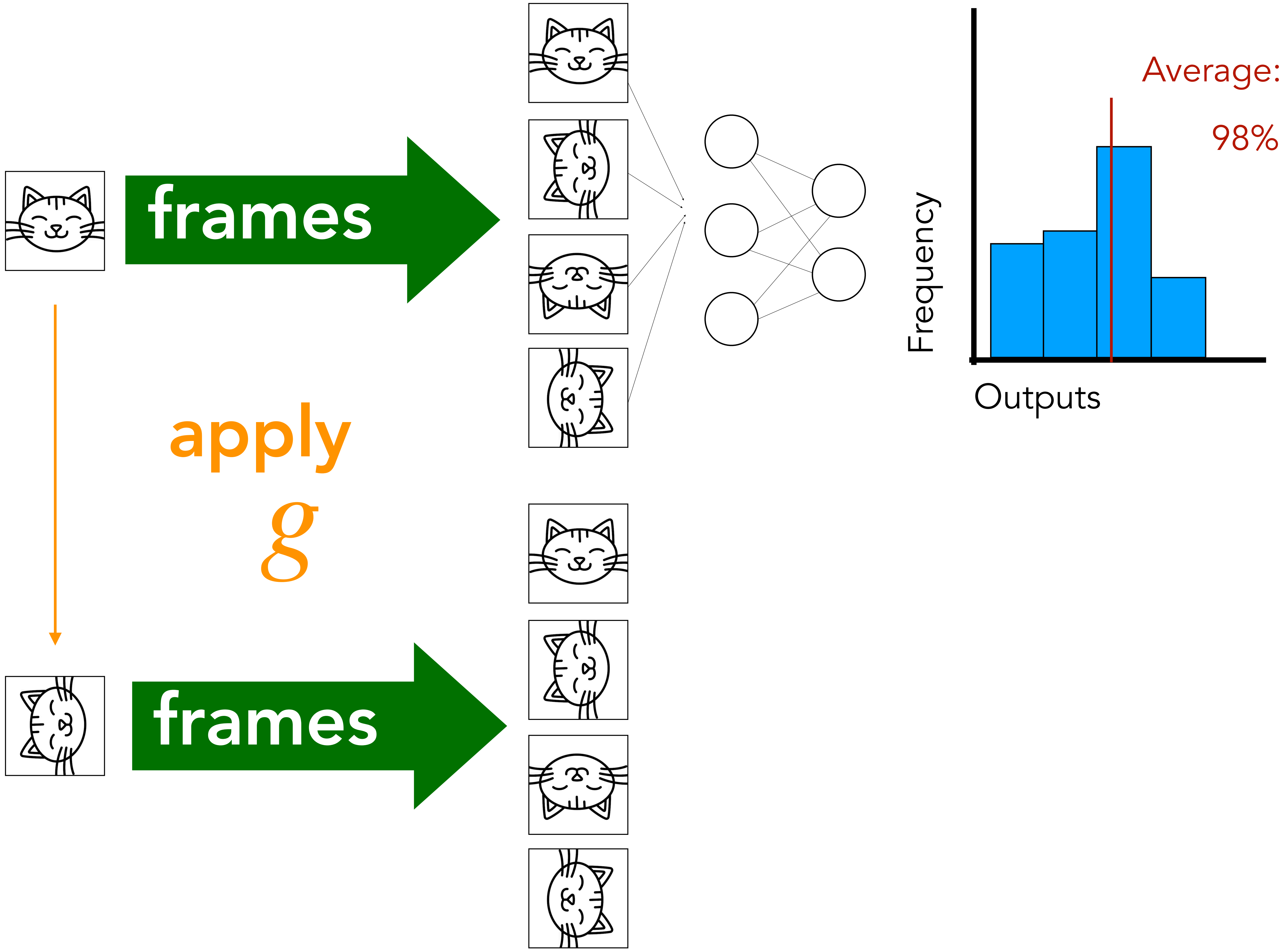
# Frame Averaging Illustration



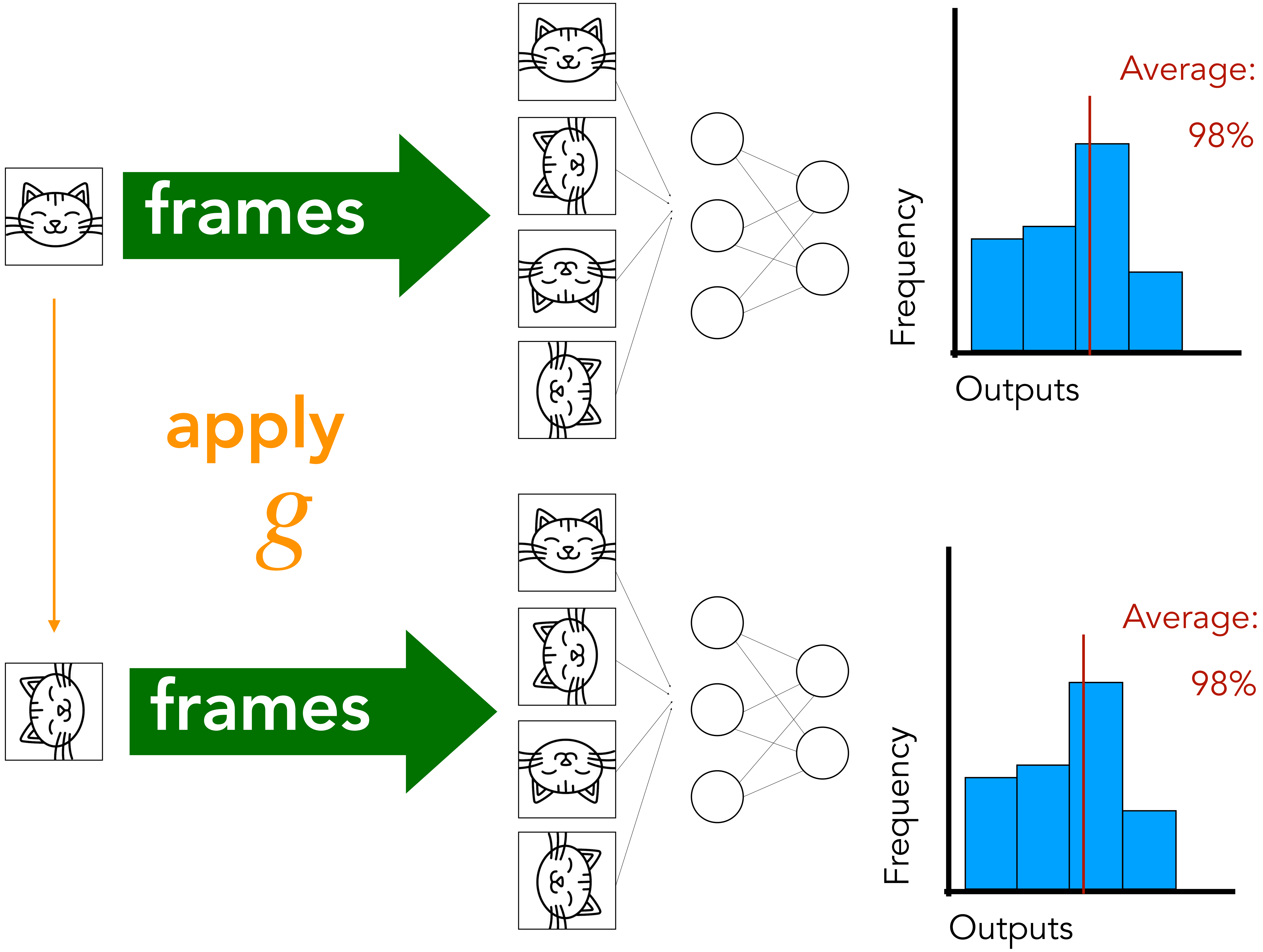
# Frame Averaging Illustration



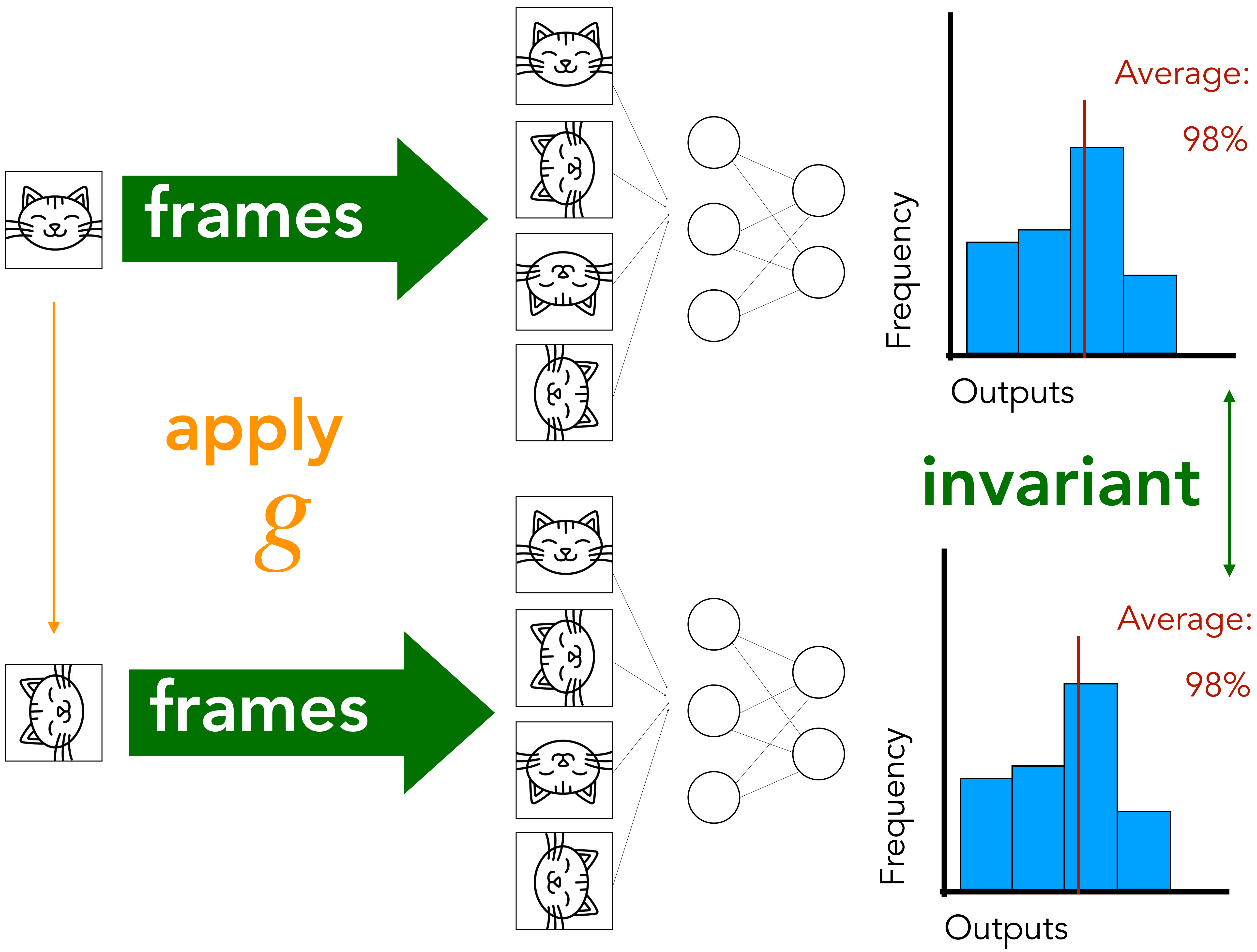
# Frame Averaging Illustration



# Frame Averaging Illustration



# Frame Averaging Illustration



# How do we get back continuity?

Canonicalization

Group Averaging

Frames

Canonicalization

Frames

Full data augmentation

Discontinuous

Continuity depends on size!

Continuous

Cheap

Cost depends on size!

Expensive

# How do we get back continuity?

**Randomized canonicalization!**

Canonicalization

Group Averaging

Frames

**Randomized Canonicalization**

Canonicalization

Frames

Full data augmentation

**Discontinuous**

**Continuity depends on size!**

**Continuous**

**Cheap**

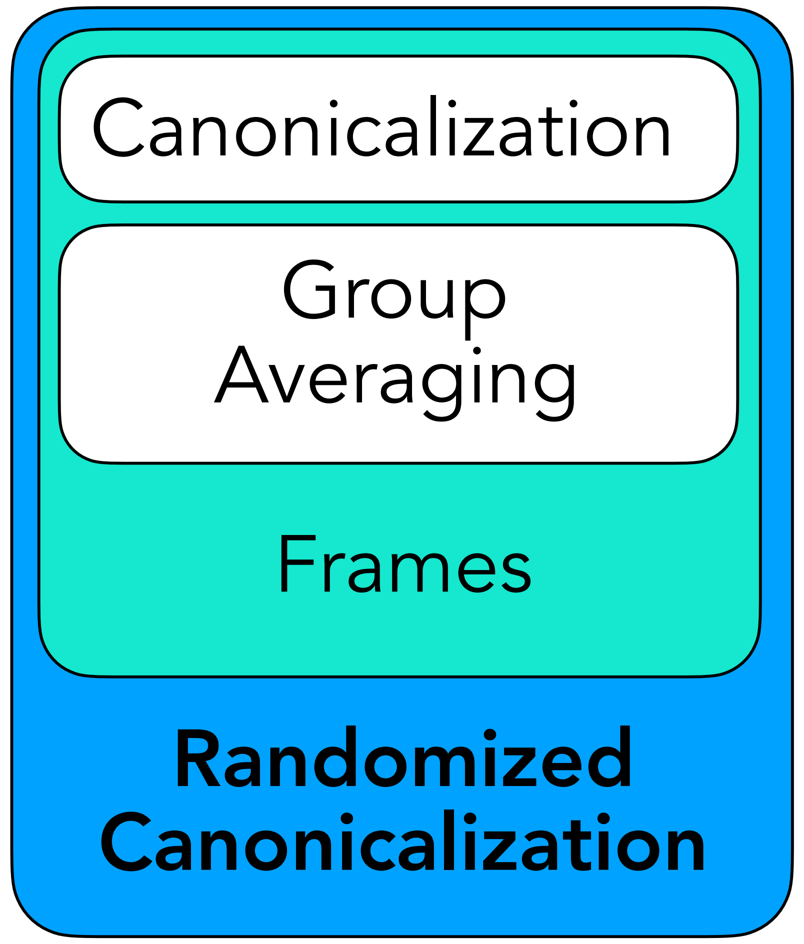
**Cost depends on size!**

**Expensive**

# How do we get back continuity?

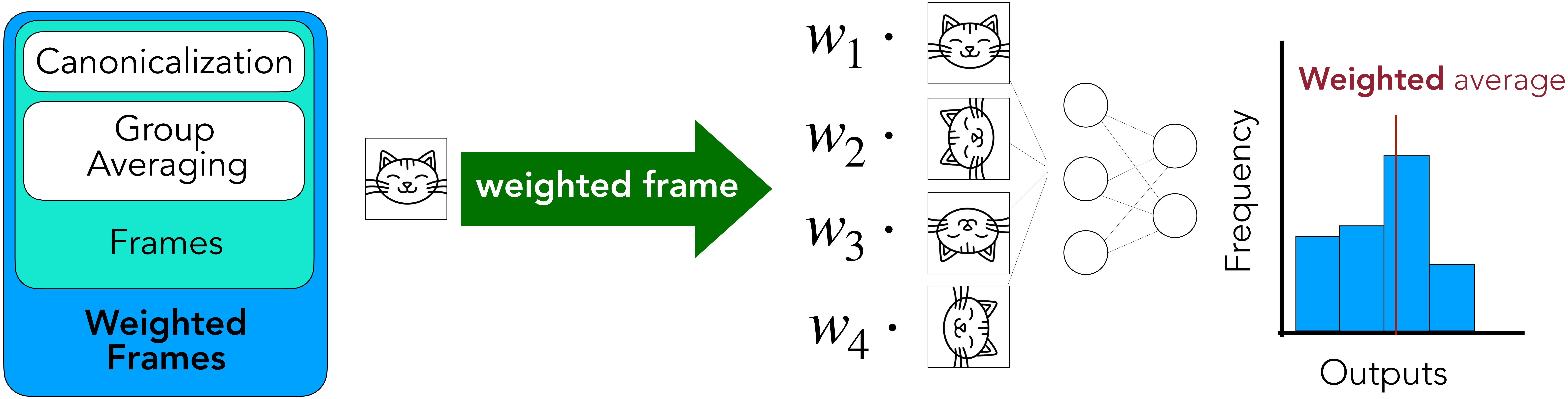
**Randomized canonicalization!**

By randomizing, preserve continuity **and** remains cheap/ flexible!

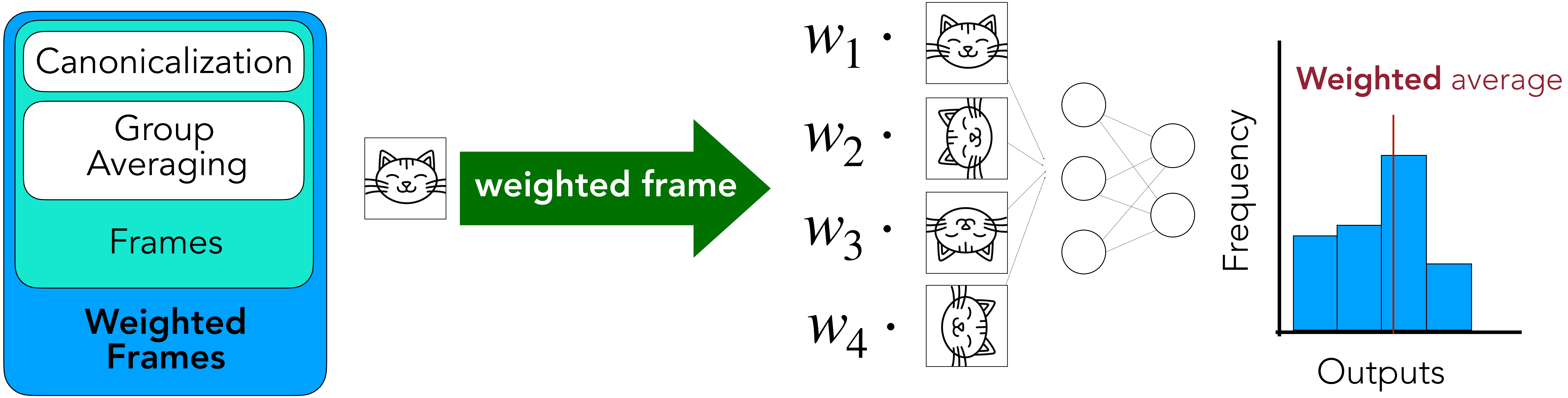



| Canonicalization | Frames | Full data augmentation |
| --- | --- | --- |
| **Discontinuous** | **Continuity depends on size!** | **Continuous** |
| **Cheap** | **Cost depends on size!** | **Expensive** |

# How do we get back continuity?



# How do we get back continuity?




**Continuity-preserving weighted frames can be much smaller than continuity-preserving unweighted frames! For instance, $n!$ vs $n^{O(d)}$ for $S_n$ acting on $\mathbb{R}^{d \times n}$**

# Empirical Results

| Invariance Method | Test Accuracy (%) |
|---|---|
| No Invariance | 25.5 |
| Discontinuous Canonicalization | 85.6 |
| Robust Frames (Sec. 4.1) | 75.5 / 85.6 / 87.1 / 88.4 |
| Robust Frames (Sec. 4.2) | 74.2 / 85.9 / 87.6 / 88.7 |
| Reynolds Operator | 21.0 / 22.4 / 22.6 / 22.6 |

*Table 2.* Comparison between permutation canonicalization and various frames. The right hand column shows 1/5/10/25 samples drawn during testing for the weighted frames.

Toy permutation task: better than Reynolds

| Pairwise Error Metric | $x_1$, $x_2$ near a singularity $b$ | $x_1$, $x_2$ near a generic point $g$ |
|---|---|---|
| $\frac{\|\|C(x_1)-C(x_2)\|\|}{\|\|C(x_1)\|\|}$ | 1.1088 | 1.7035e-5 |
| $\frac{\|\|f(C(x_1))-f(C(x_2))\|\|}{\|\|f(C(x_1))\|\|}$ | 0.0406 | 0.0009 |

*Table 3.* Average distance between pairs of points, near a singular point cloud and near a random point cloud.

Trained point cloud network has discontinuities

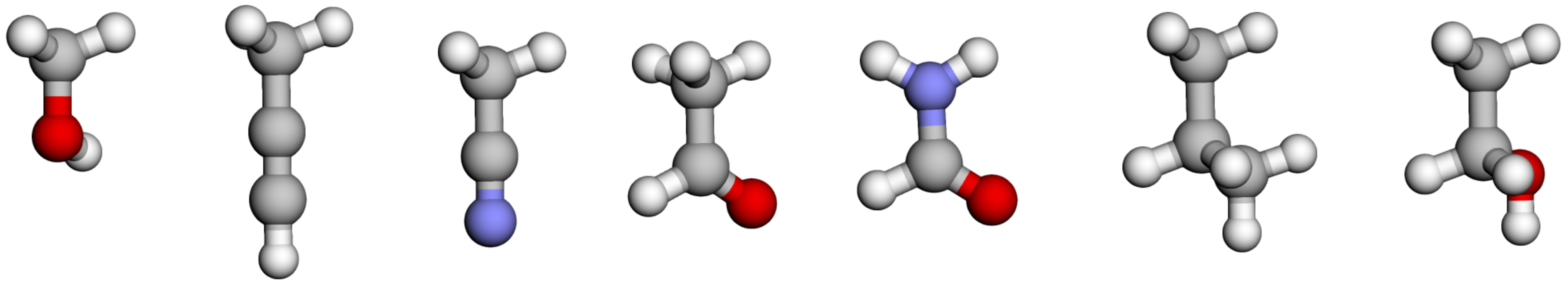
Table 3: Results for $S_n$ equivariant node classification on PATTERN. We report test accuracy at the best validation accuracy, along with the standard deviation for GA and Ours where predictions are stochastic. The results for GNN baselines are from [27].

| method | pretrain. | Accuracy ↑ |
|---|---|---|
| GCN [48], 16 layers | - | 85.614 |
| GAT [99], 16 layers | - | 78.271 |
| GatedGCN [11], 16 layers | - | 85.568 |
| GIN [103], 16 layers | - | 85.387 |
| RingGNN [16], 2 layers | - | 86.245 |
| RingGNN [16], 8 layers | - | diverged |
| PPGN [58], 3 layers | - | 85.661 |
| PPGN [58], 8 layers | - | diverged |
| ViT-GA, 1-sample | - | 76.956 ± 0.033 |
| ViT-GA, 10-sample | - | 83.220 ± 0.057 |
| ViT-GA, 1-sample | ImageNet-21k | 81.933 ± 0.075 |
| ViT-GA, 10-sample | ImageNet-21k | 84.641 ± 0.020 |
| ViT-FA | - | 71.377 |
| ViT-FA | ImageNet-21k | 80.015 |
| ViT-Canonical. | - | 85.825 |
| ViT-Canonical. | ImageNet-21k | 86.534 |
| ViT-PS (Ours), 1-sample | - | 85.868 ± 0.017 |
| ViT-PS (Ours), 10-sample | - | 85.989 ± 0.011 |
| ViT-PS (Ours), 1-sample | ImageNet-21k | 86.573 ± 0.030 |
| ViT-PS (Ours), 10-sample | ImageNet-21k | **86.650 ± 0.010** |

“Learning Probabilistic Symmetrization for Architecture Agnostic Equivariance” by Kim et al, NeurIPS 2023

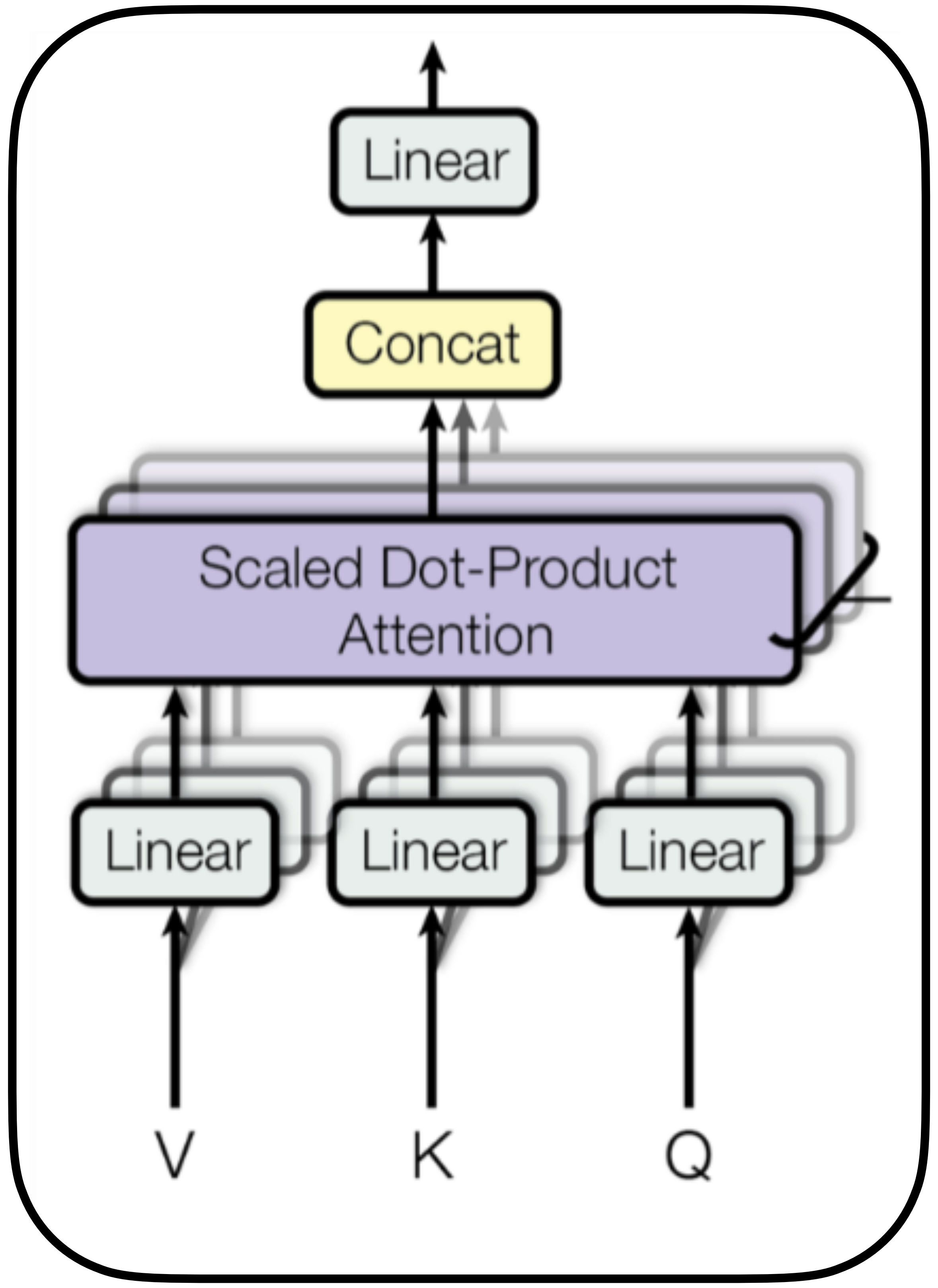
# Lots of exciting directions in canonicalization:

- Efficient, randomized canonicalization
- Empirical exploration of continuity problem — how important in practice?
- Permutation canonicalization for language models (e.g. in-context learning, as well as language models for scientific data)
- Statistical tests: is your dataset already canonicalized (like balloons)? Are language datasets like this? What should you do if it is & what does it tell us about the nature of equivariance?



# Part 2: Positional Encodings

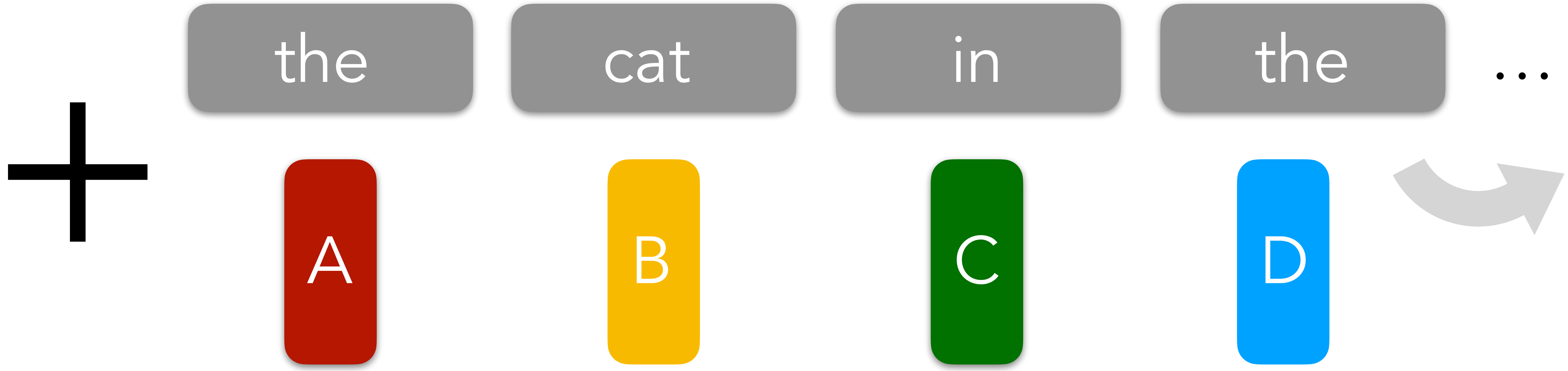
# Reminder: positional encodings

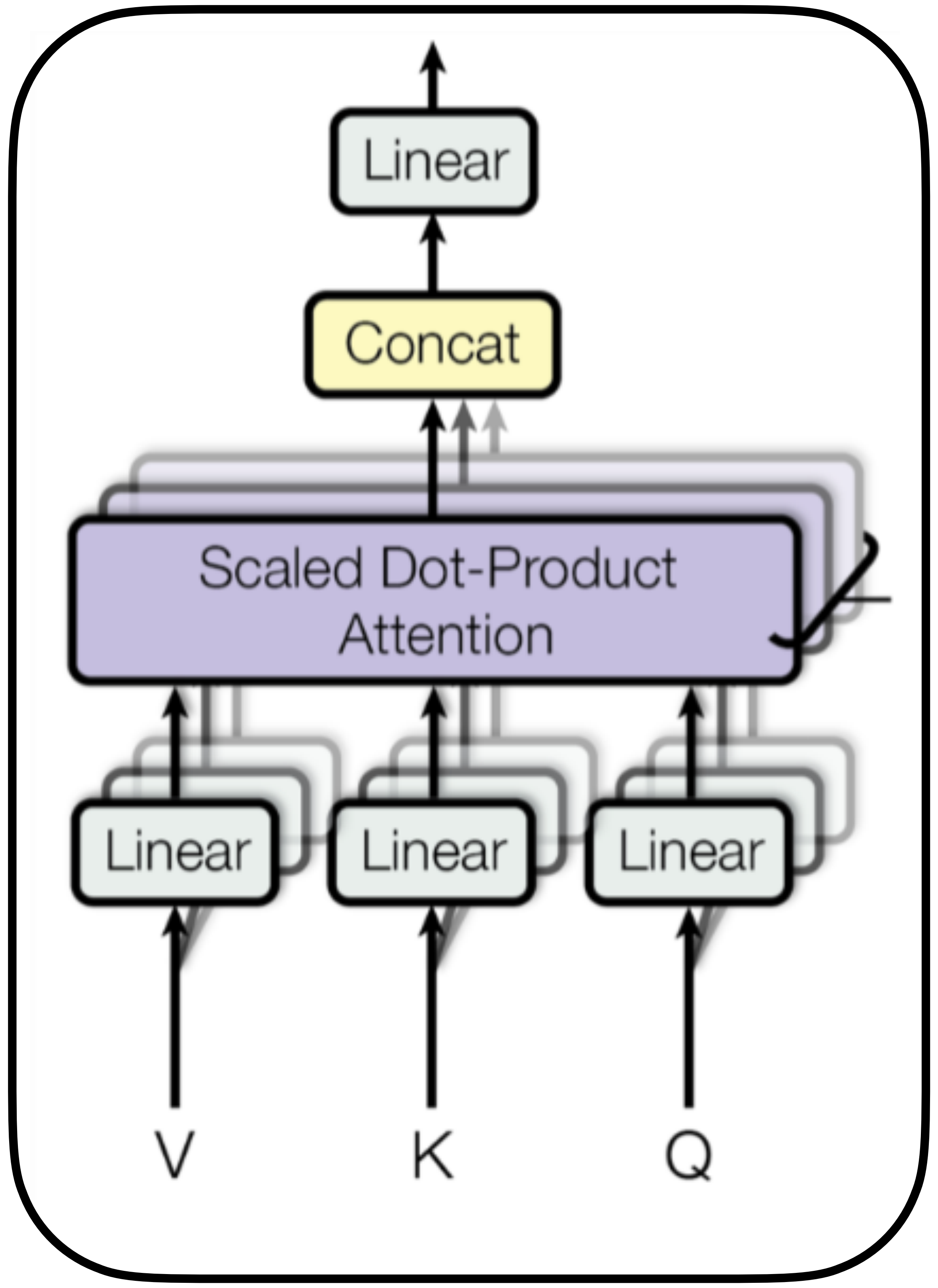




Vanilla transformer is permutation invariant: would predict the same next word for both of these

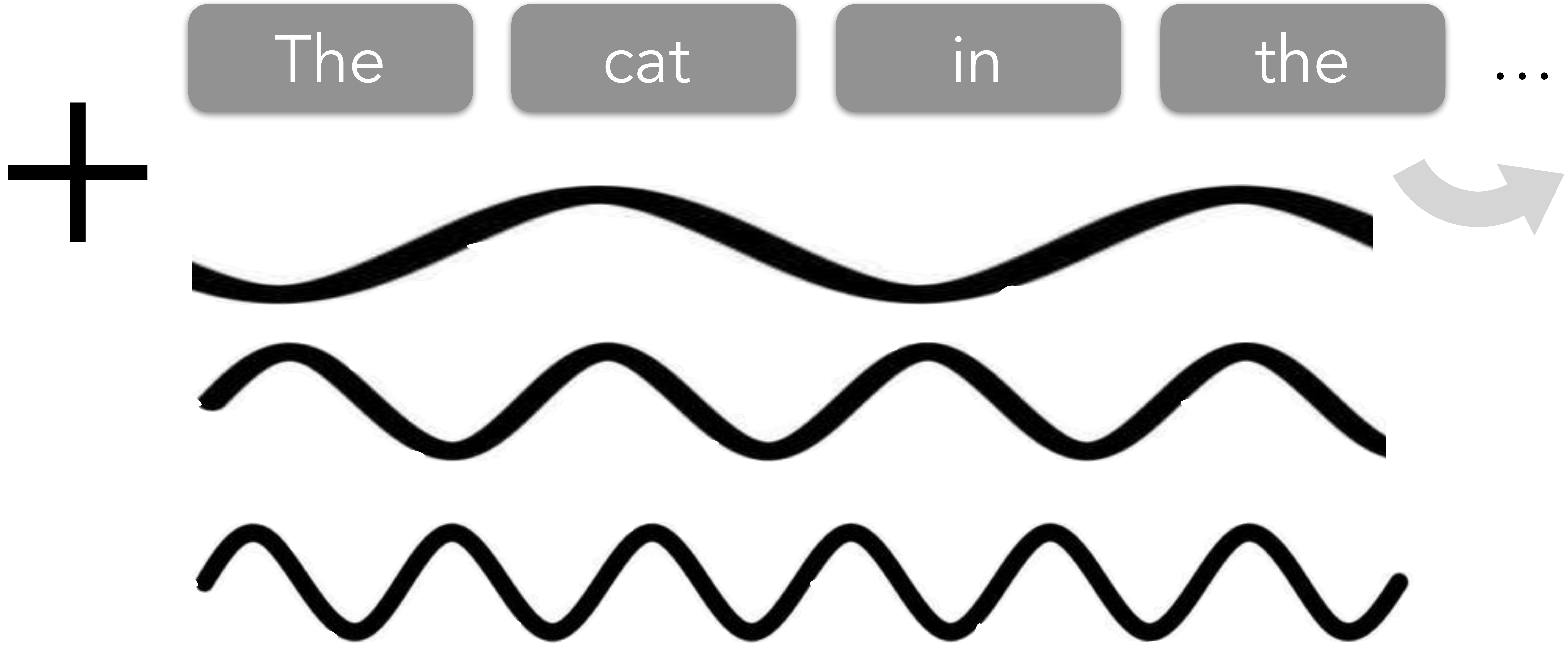
Solution: append a unique vector to each **position**

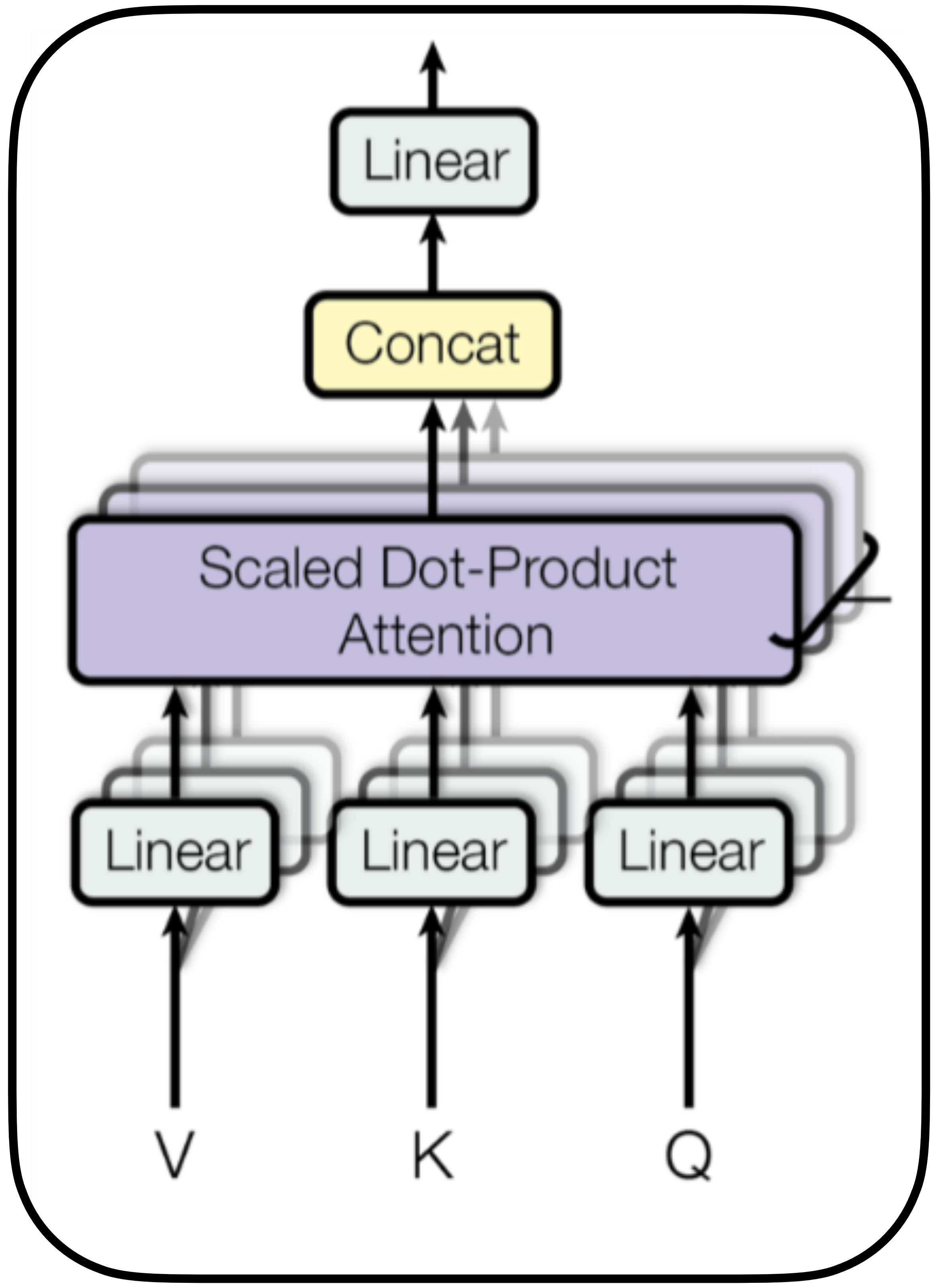
# Reminder: positional encodings



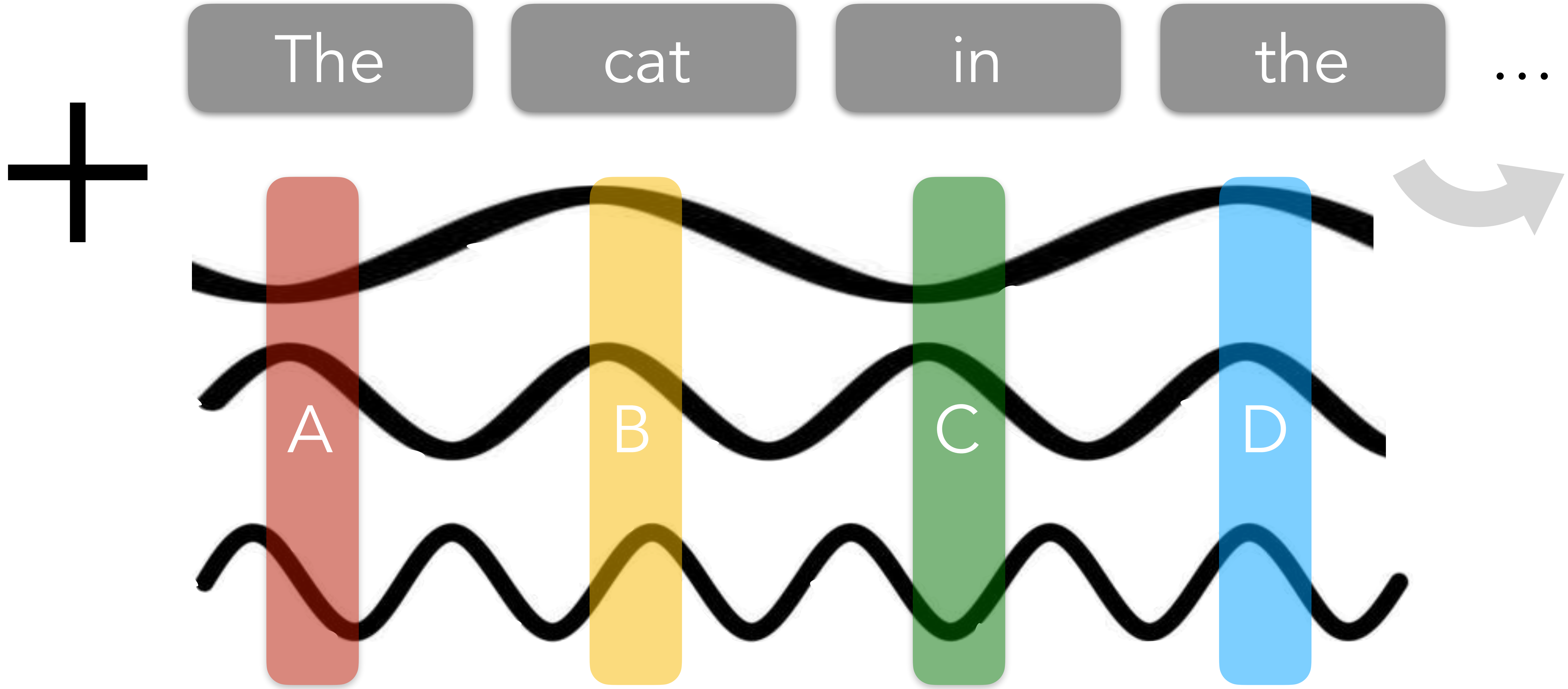



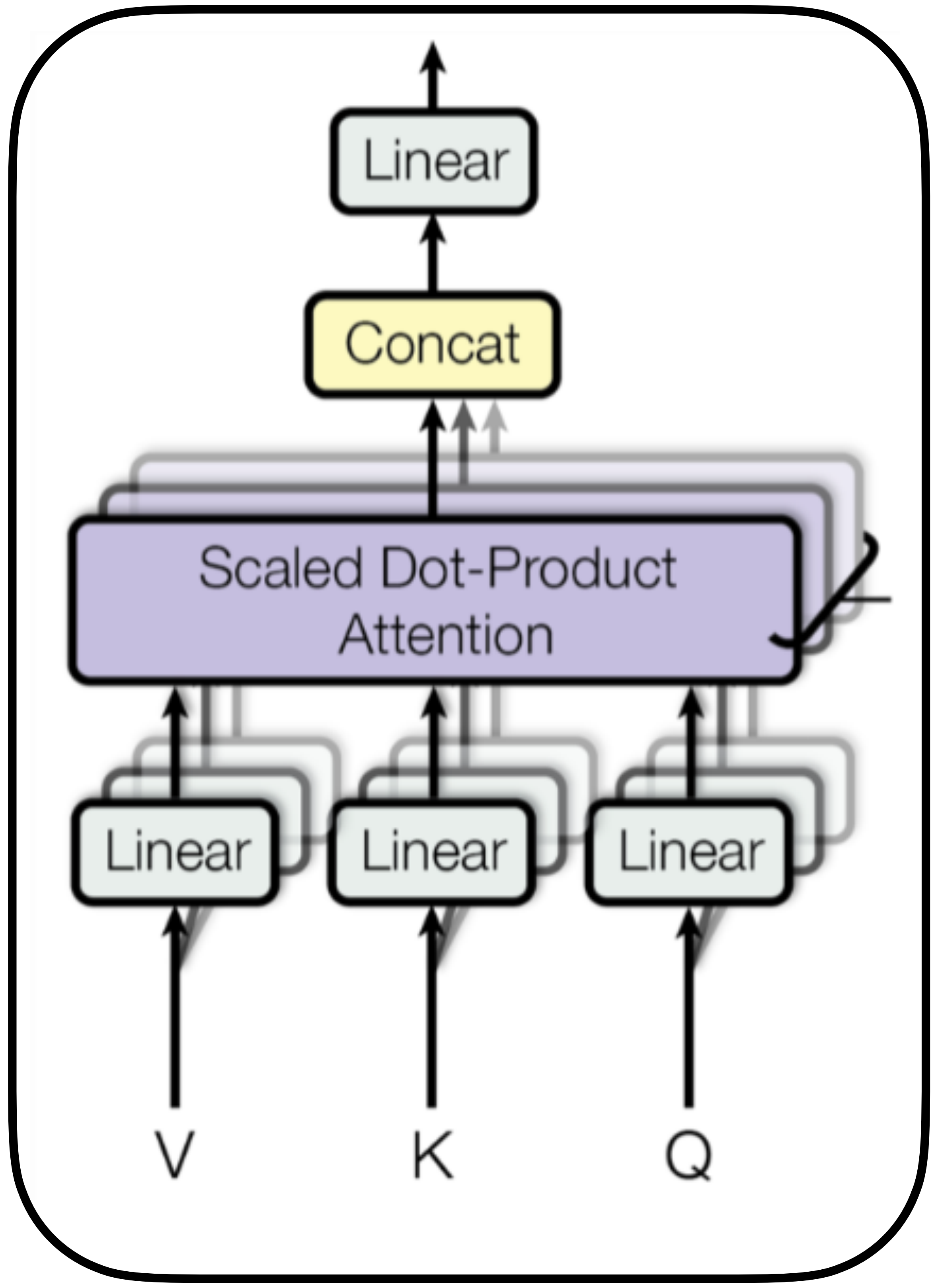
# Reminder: positional encodings



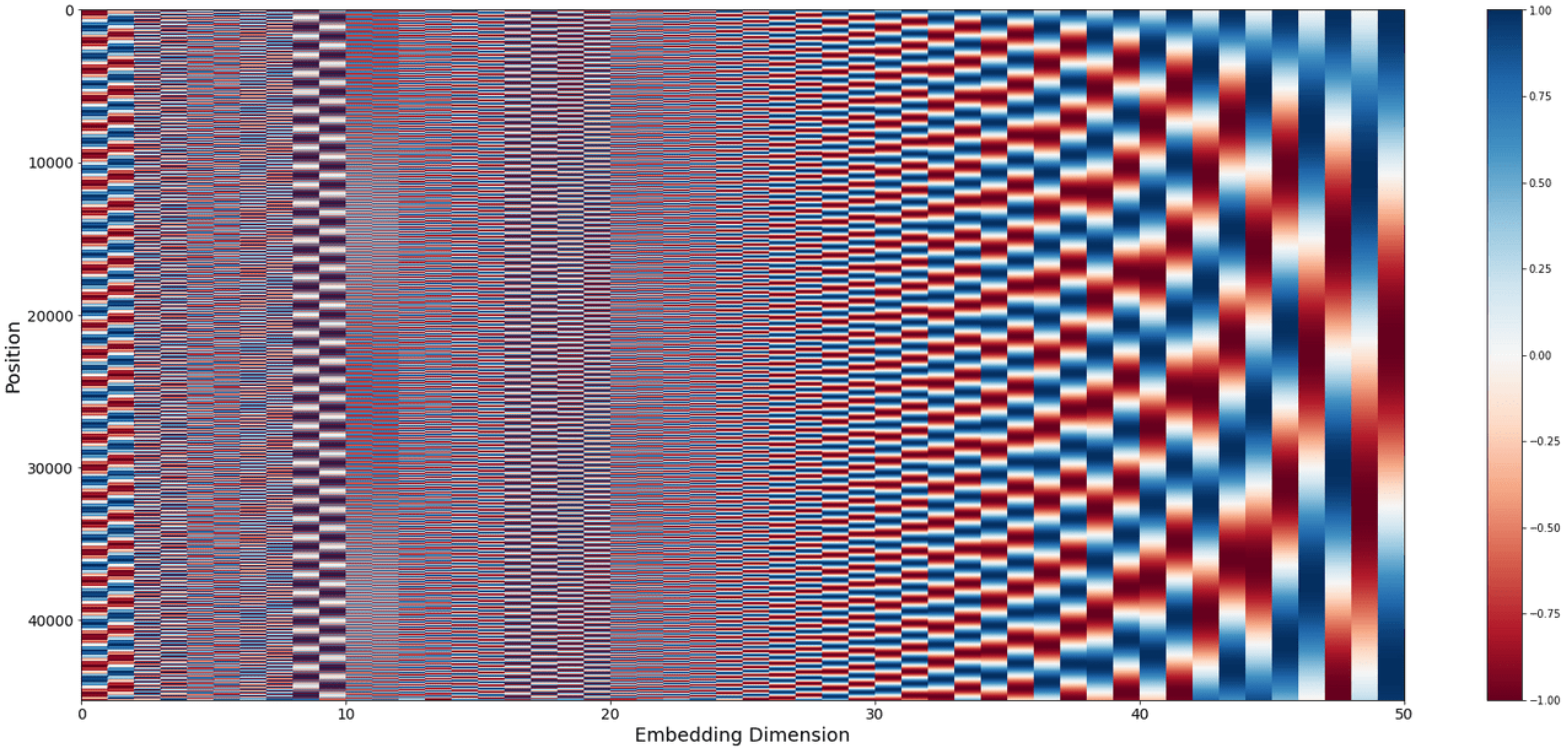



# Reminder: positional encodings

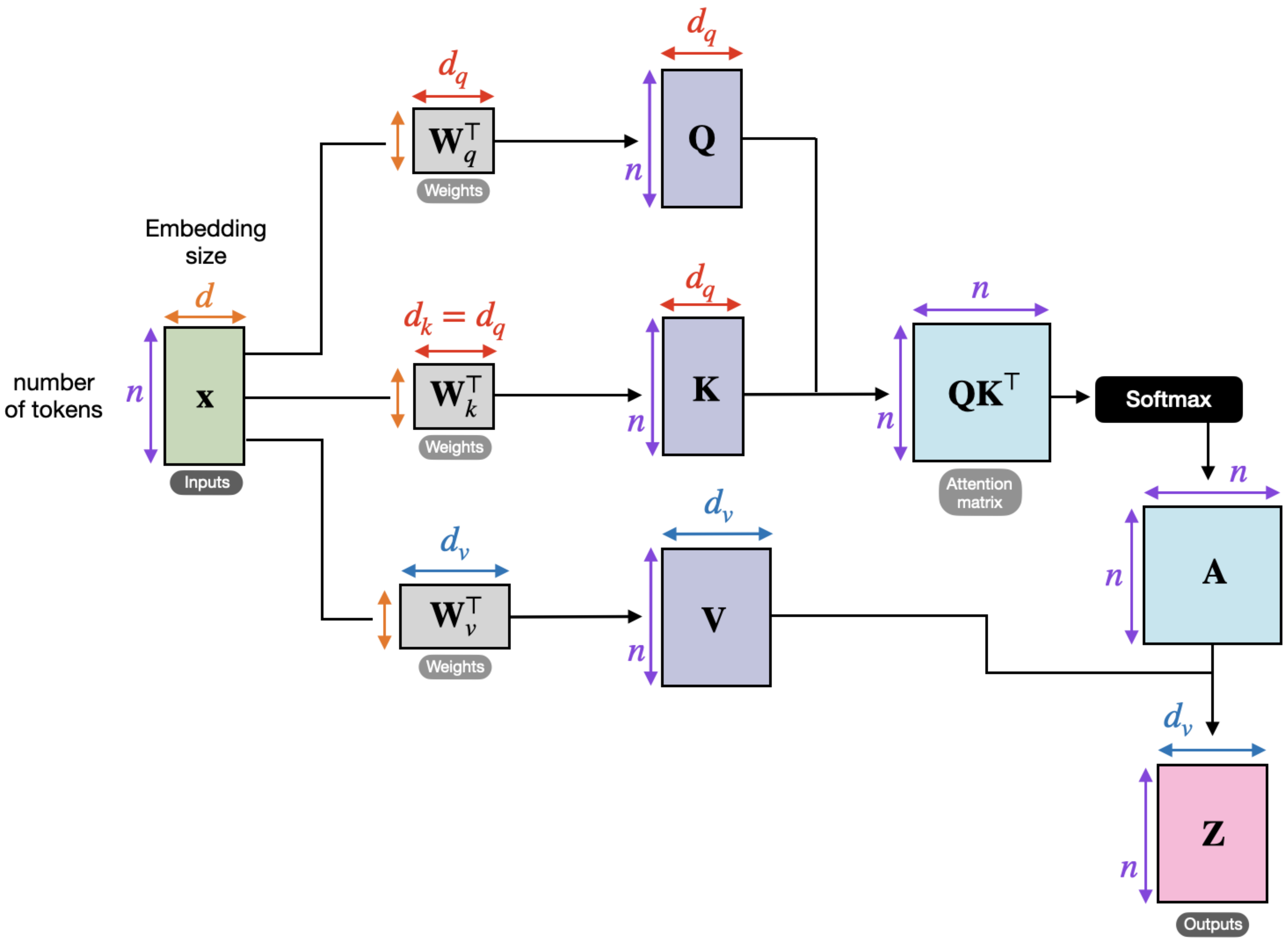


# Reminder: positional encodings

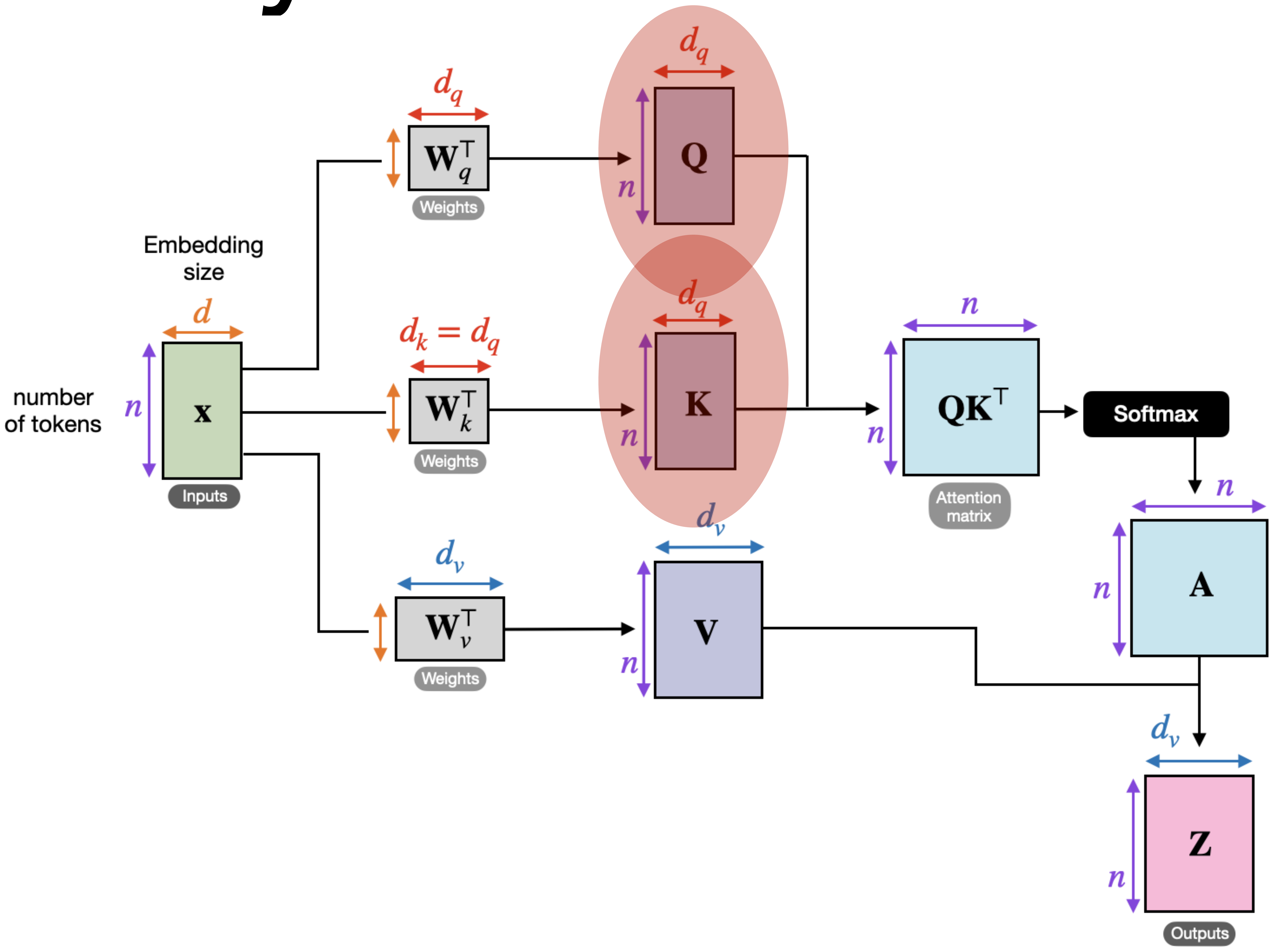


# Rotary Positional Encodings



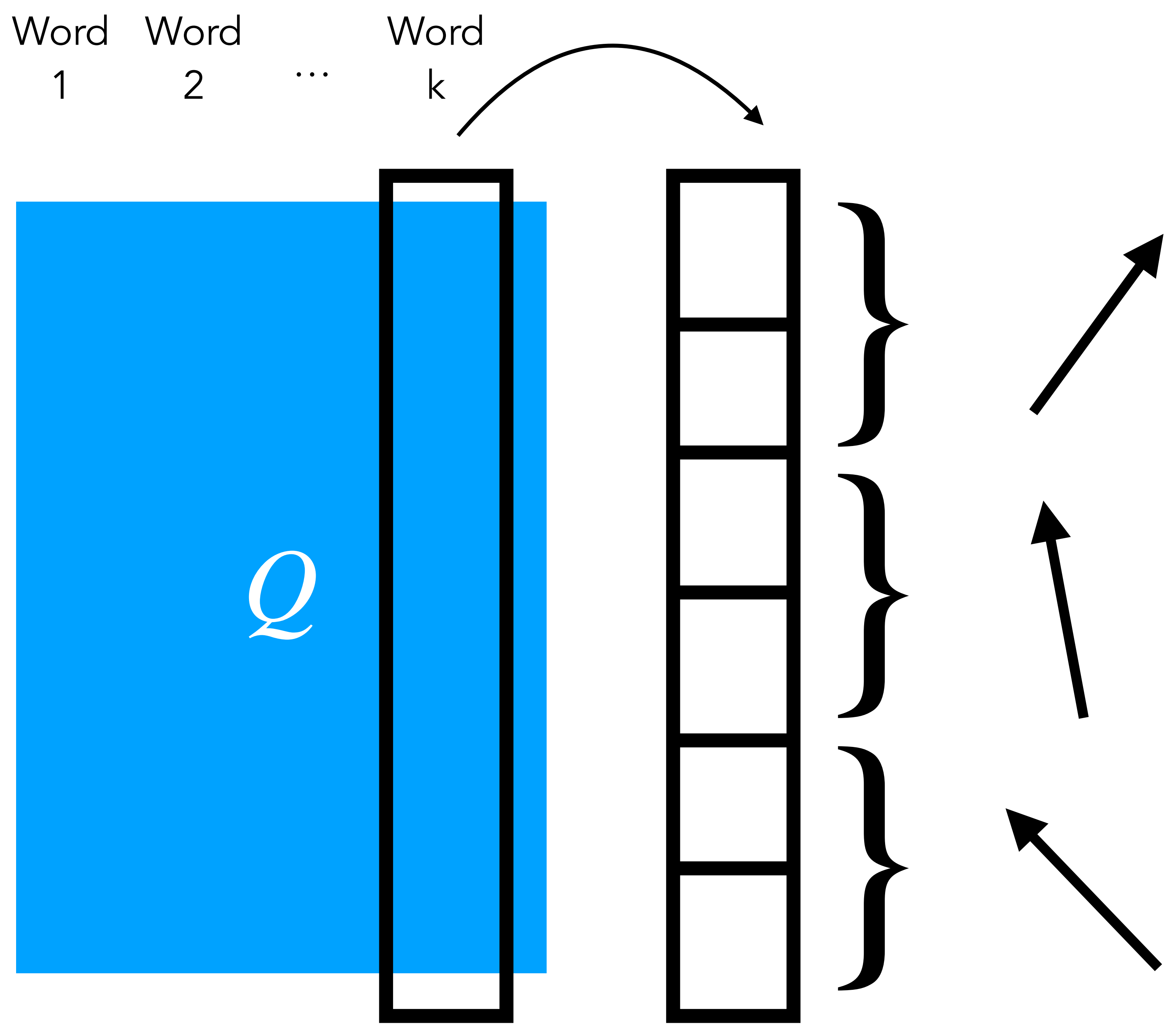

Sebastian Raschka

# Rotary Positional Encodings

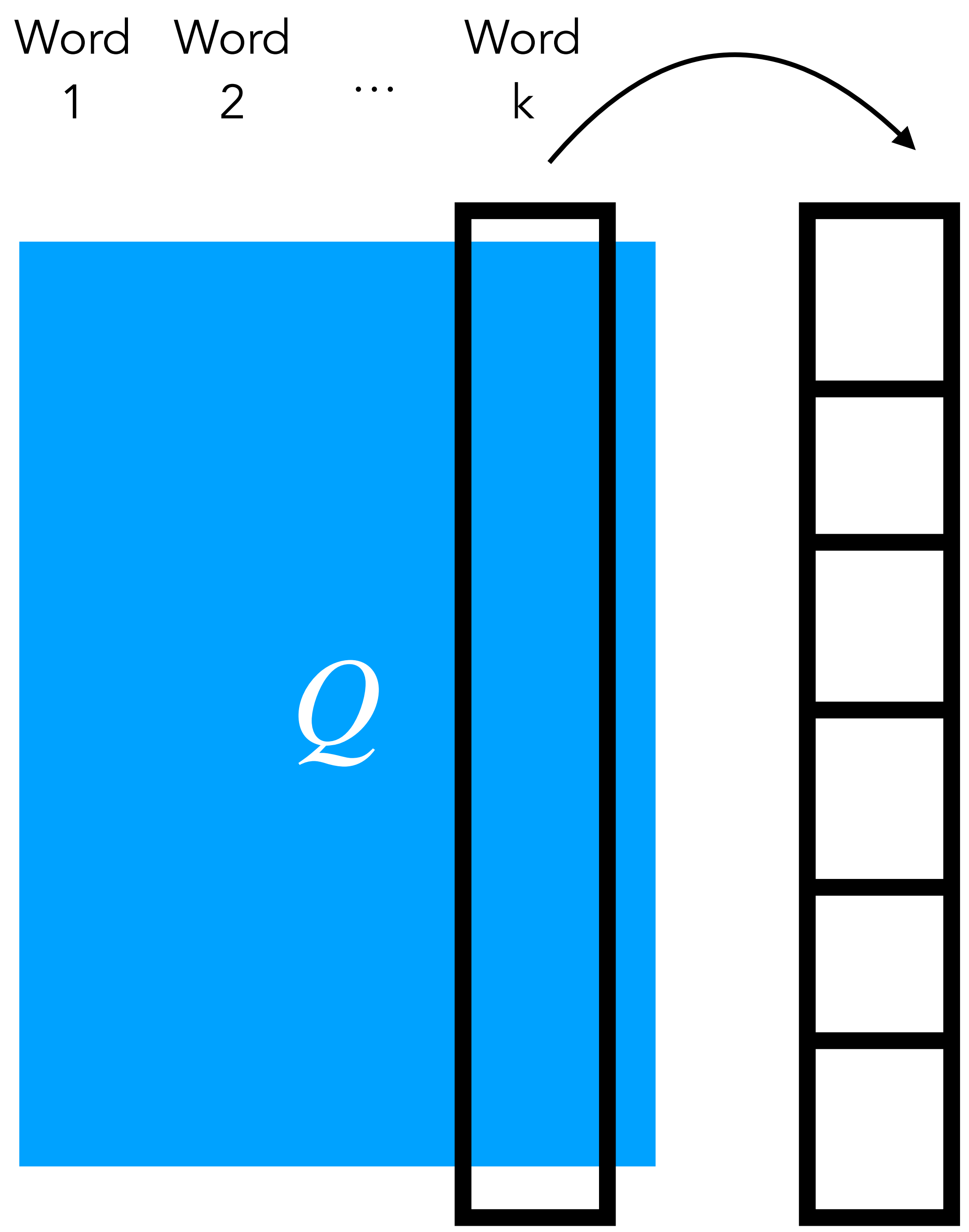



Sebastian Raschka

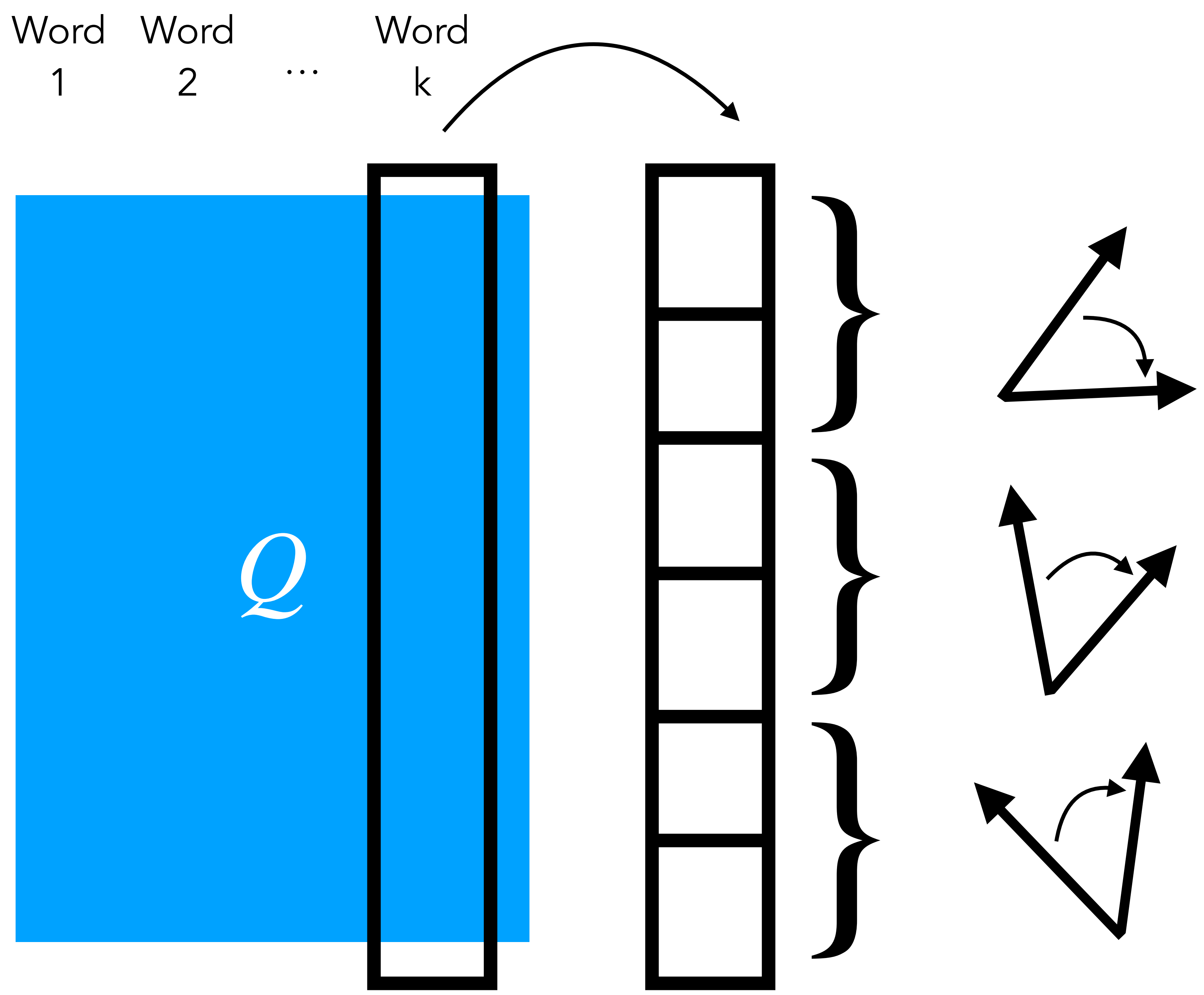
# Rotary Positional Encodings



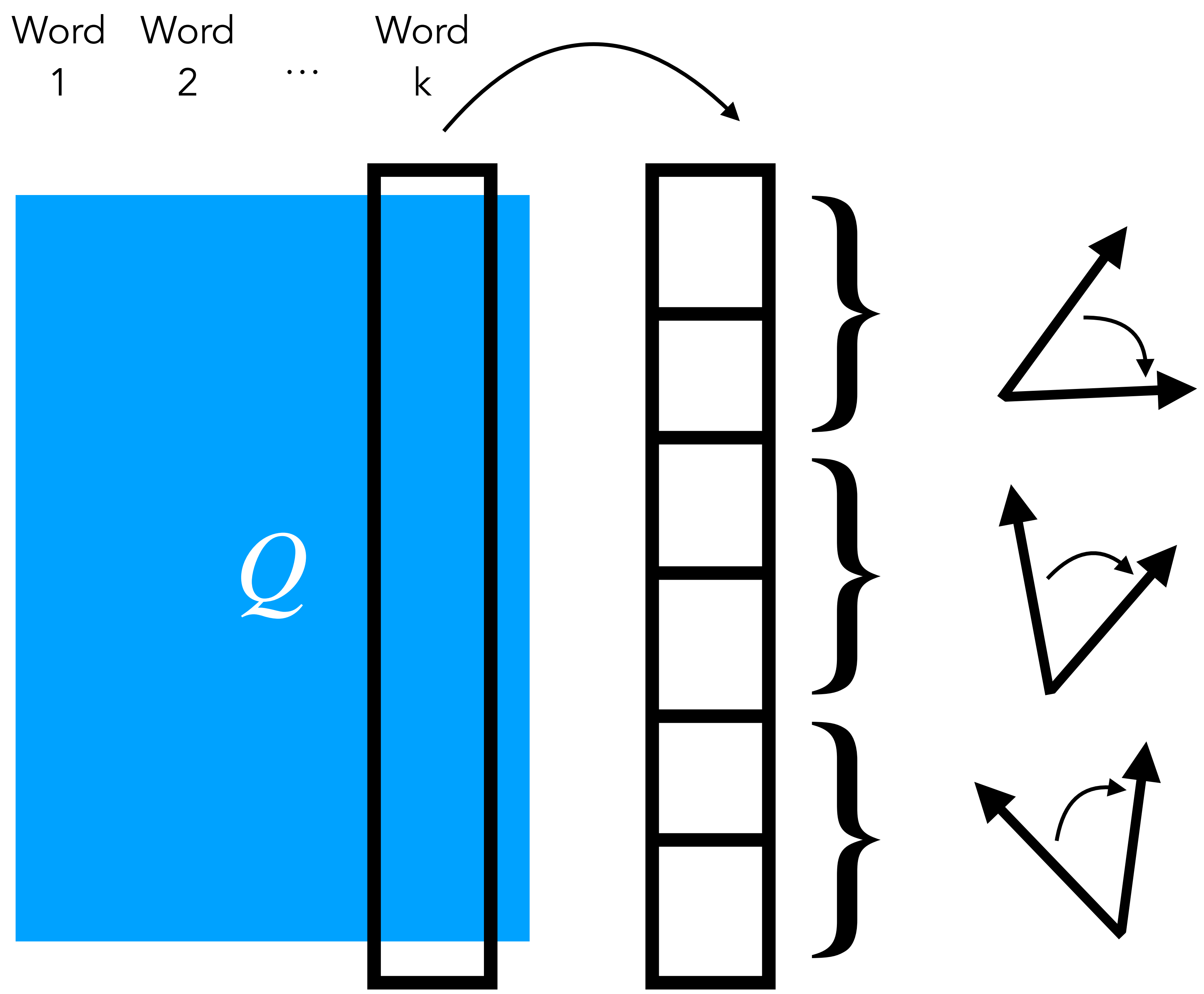
# Rotary Positional Encodings



# Rotary Positional Encodings



# Rotary Positional Encodings

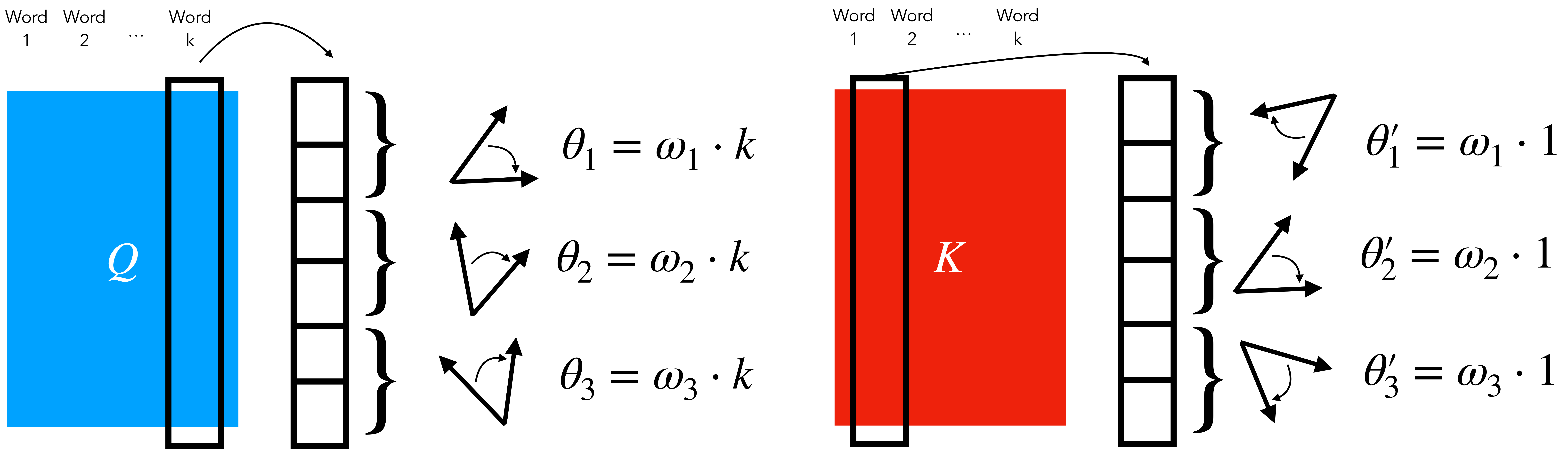



$$\theta_1 = \omega_1 \cdot k$$

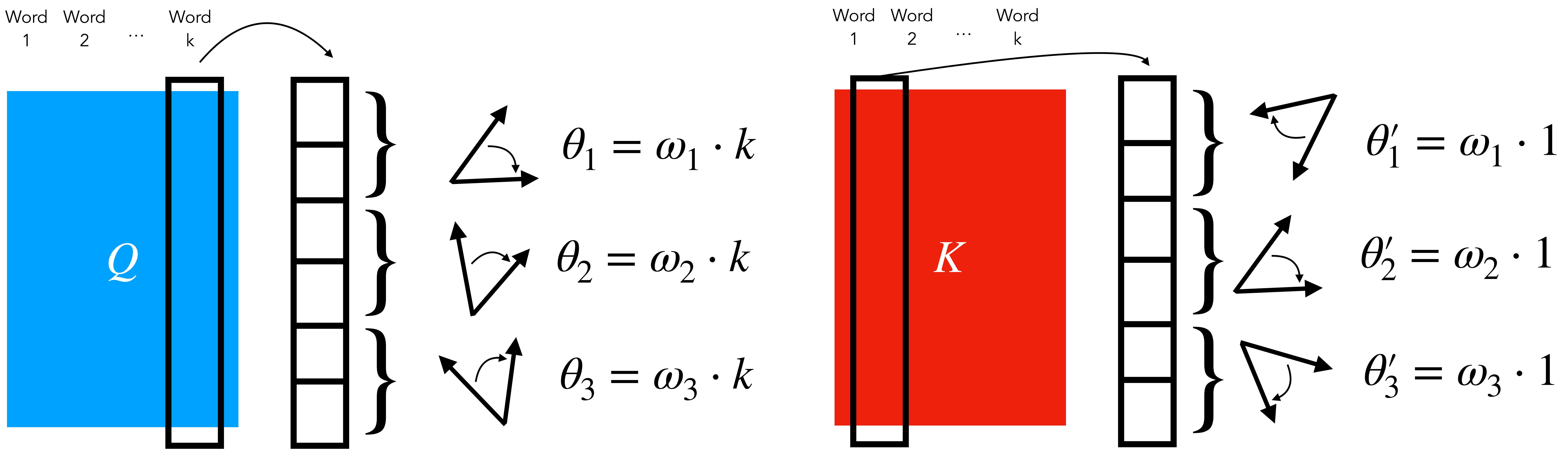
$$\theta_2 = \omega_2 \cdot k$$

$$\theta_3 = \omega_3 \cdot k$$

# Rotary Positional Encodings

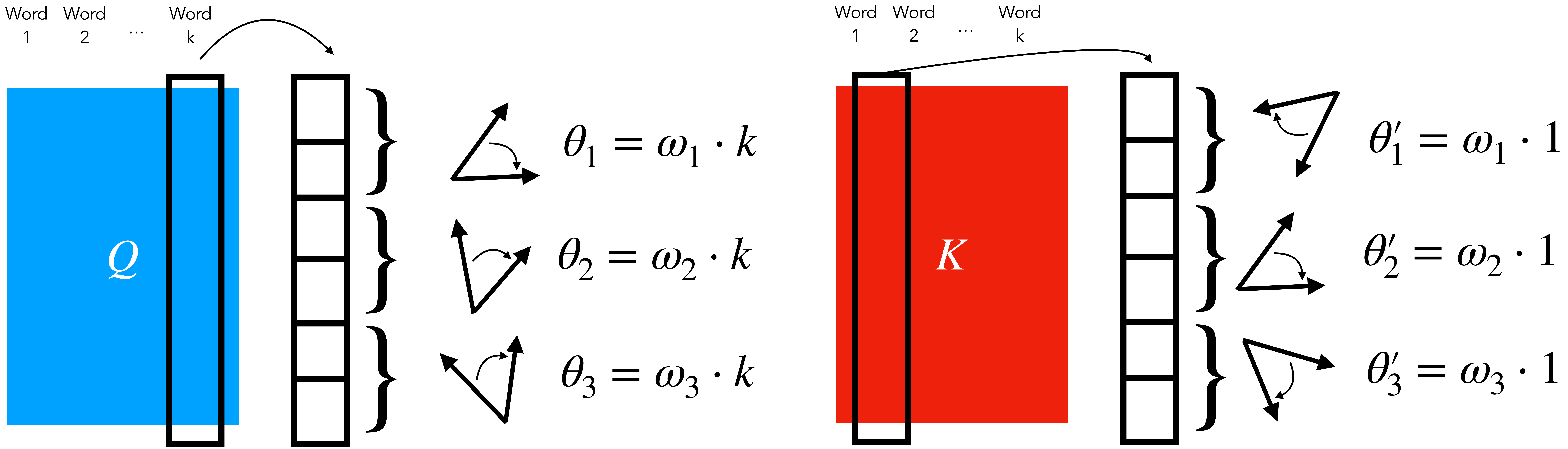


# Rotary Positional Encodings




→ Attention weights = inner products only depend on **relative** position between words, not absolute position

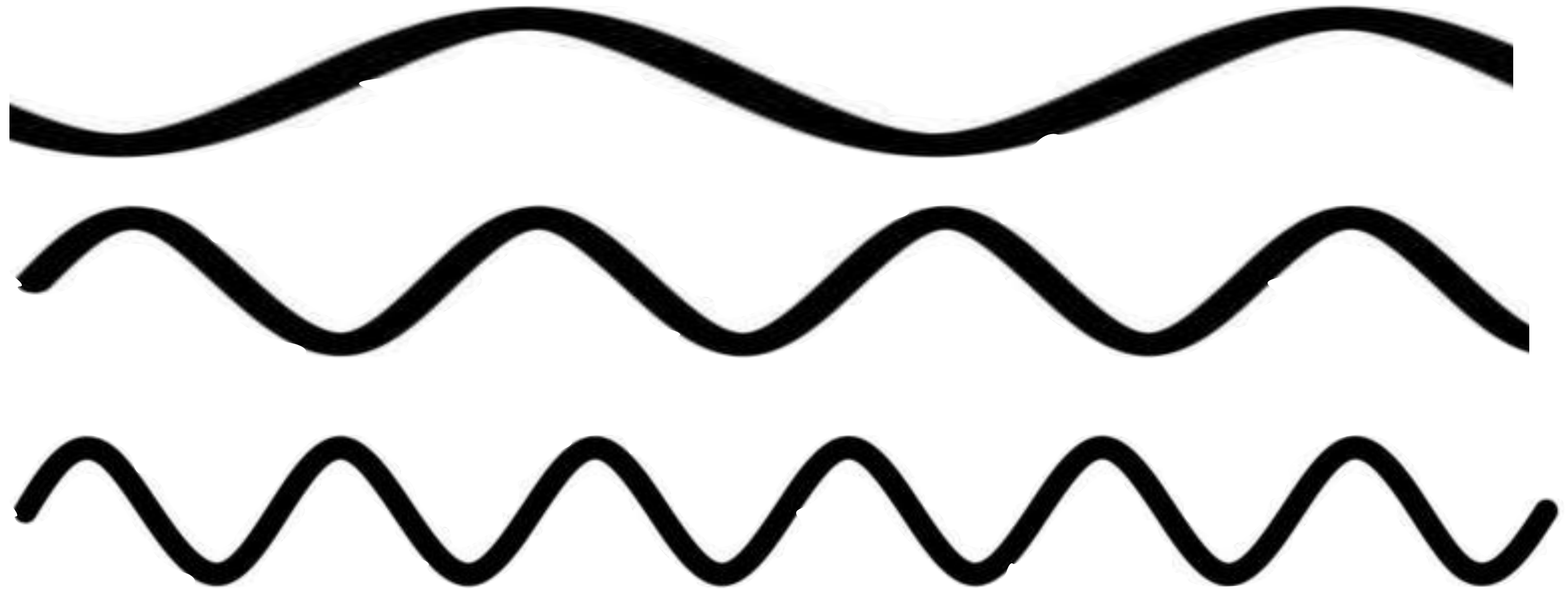
# Rotary Positional Encodings



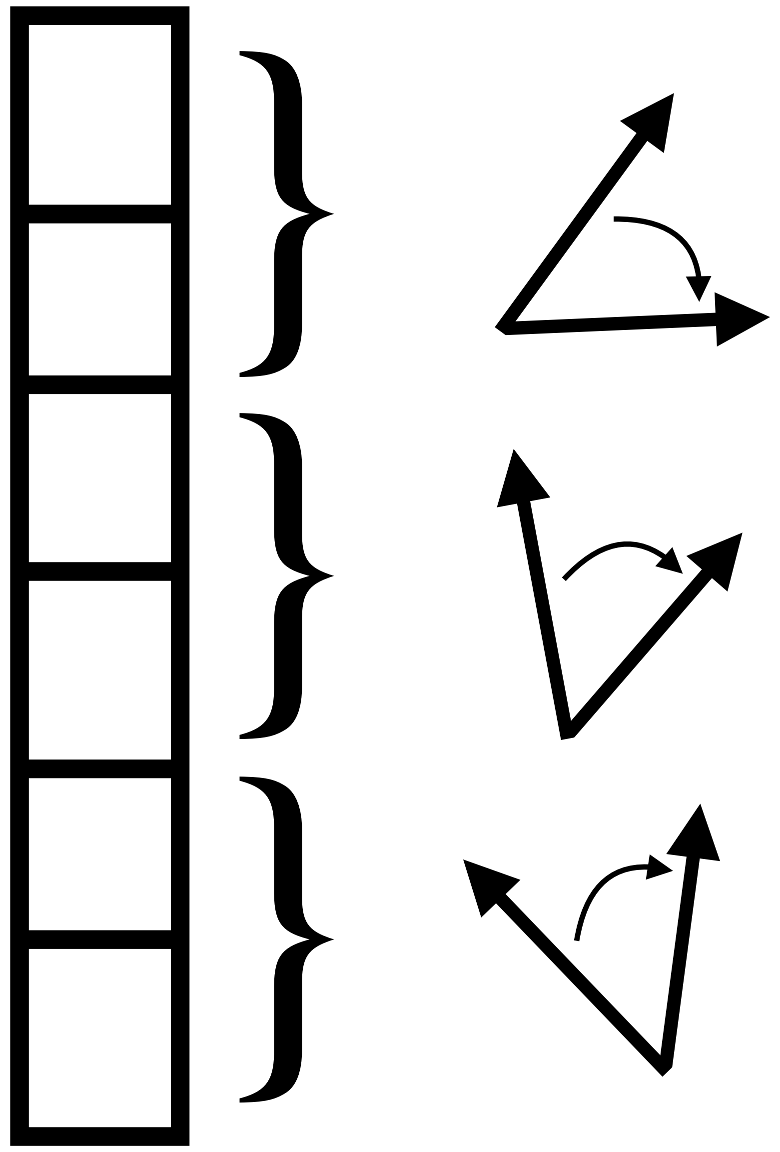

→ Attention weights = inner products only depend on **relative** position between words, not absolute position

→ Length generalization of transformers! (Train on short text, test on long text)

# Group symmetry view on PEs



Sinusoidal positional encodings



$$\theta_1 = \omega_1 \cdot k$$

$$\theta_2 = \omega_2 \cdot k$$

$$\theta_3 = \omega_3 \cdot k$$

Rotary positional encodings

**Both of these: group representations of the group of cyclic translations (equivalently, 2D rotations)**

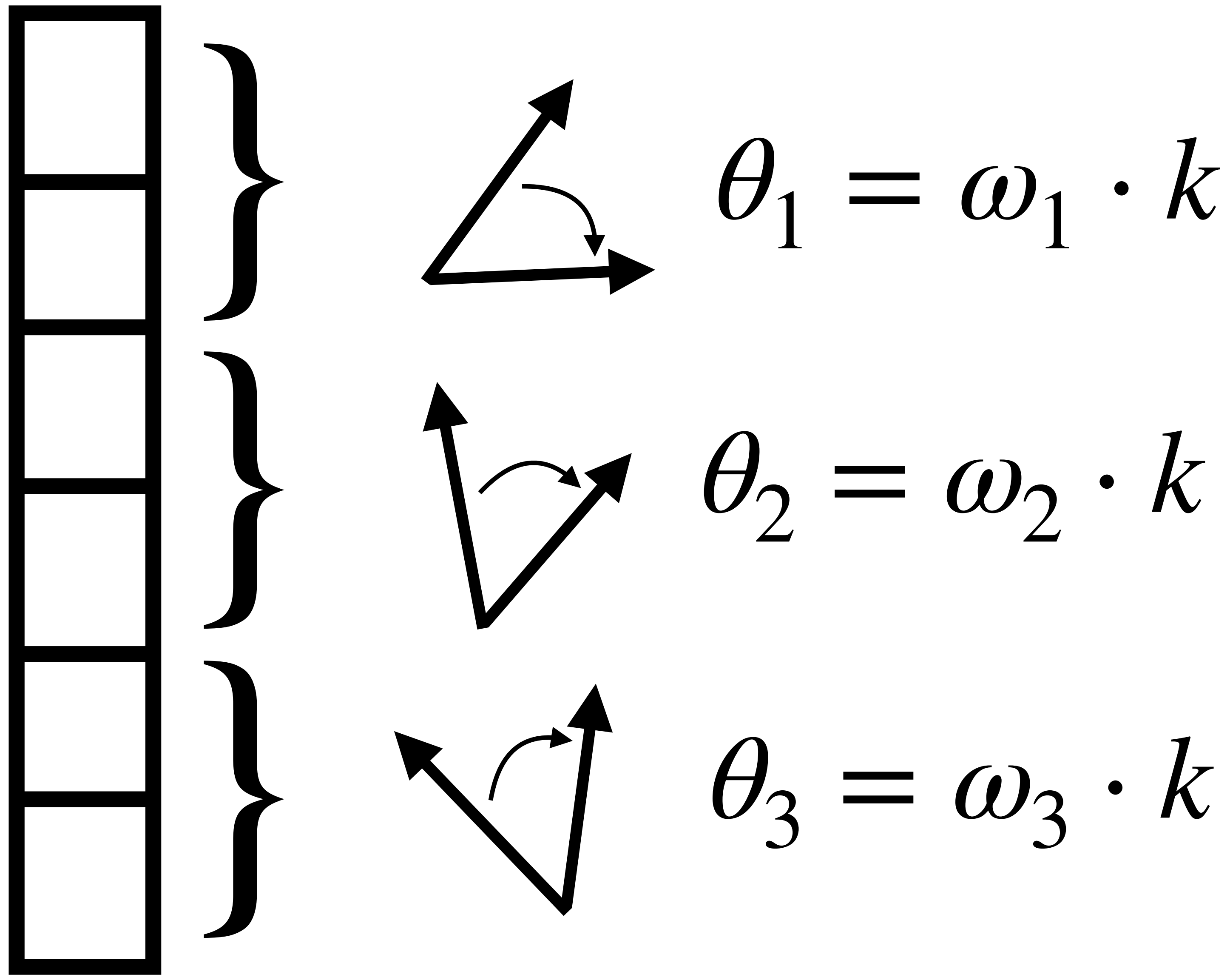
Lim*, Lawrence*, Huang, Thiede at TAG, ICML 2023

# Defining properties of irreps = useful properties of PEs

**Composition under attention:**

Symmetry $g$ acts on object via matrix $\rho(g)$

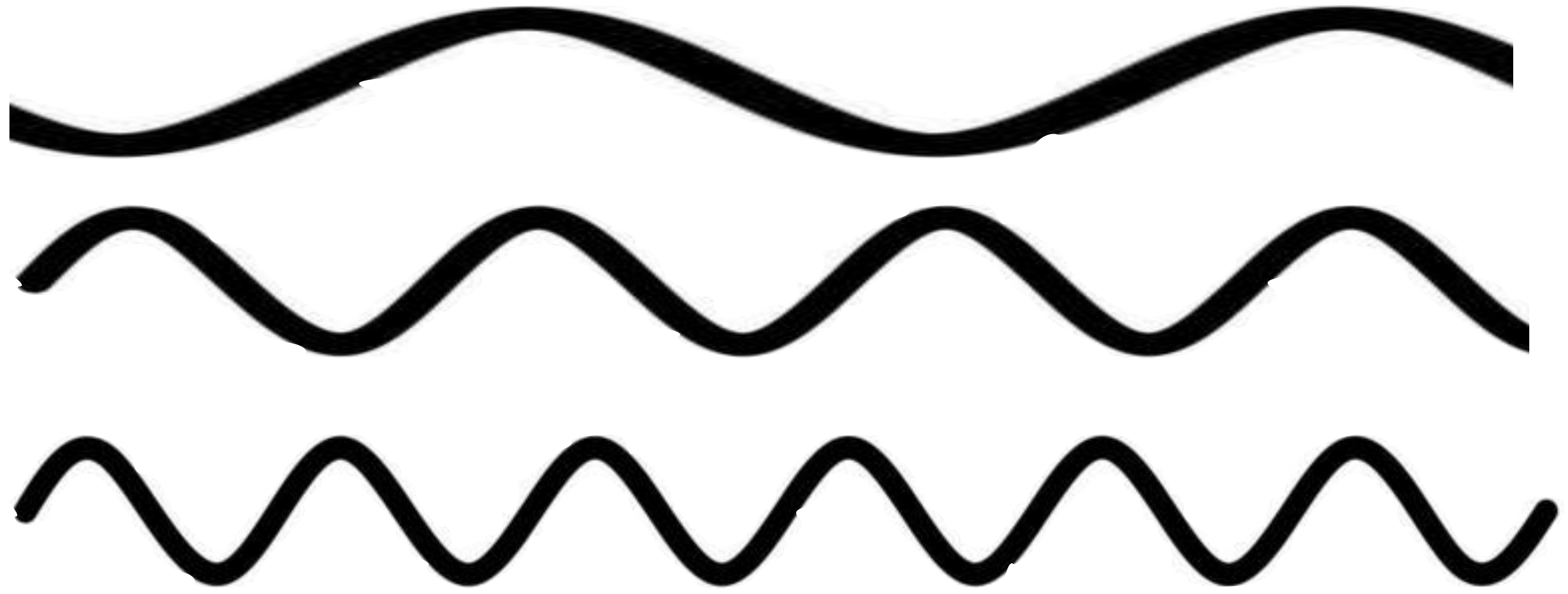
$$\rho(g_1)\rho(g_2) = \rho(g_1 g_2)$$




Rotary positional encodings

Lim*, Lawrence*, Huang, Thiede at TAG, ICML 2023

# Group symmetry view on PEs



Sinusoidal positional encodings

**Notion of varying scale, or hierarchy:**

Common for irreps in many groups

Related to: forming a functional basis

Lim*, Lawrence*, Huang, Thiede at TAG, ICML 2023

# Positional encodings as group representations

*Table 1.* Examples of positional encodings, interpreted as group representations. $Y_\ell^m$ denotes spherical harmonics, $v_i$ the $i$-th eigenvector of the graph Laplacian, $J(r)$ a radial function, and $R^{2\times 2}(\theta)$ is the $2 \times 2$ rotation matrix by $\theta$.

| Data Type | Group | Encoding | Ref. |
|---|---|---|---|
| Text | $T$ | $(x) \mapsto \{(\cos(\alpha x), \sin(\alpha x))\}_\alpha$ | Vaswani et al. (2017) |
| Image | $T \times T$ | $(x, y) \mapsto \{(\cos(\alpha_1 x + \alpha_2 y), \sin(\alpha_1 x + \alpha_2 y))\}_{\alpha_1, \alpha_2}$ | Dosovitskiy et al. (2021) |
| Molecule | $SO(3)$ | $(r, \theta, \phi) \mapsto \{Y_\ell^m(\theta, \phi) J(r)\}_{\ell, m}$ | Thomas et al. (2018) |
| Graph | $S_{\lvert\mathcal{X}\rvert}$ | $(x) \mapsto \{v_i(x)\}_i$ | Lim et al. (2023) |
| Any (learned embedding) | $S_{\lvert\mathcal{X}\rvert}$ | $x \mapsto \texttt{one\_hot}(x)$ | Gehring et al. (2017) |
| Text (spherical embedding) | $SO(2)^{n/2}$ | $(m) \mapsto \{\bigoplus R^{2\times 2}(m\alpha)\}_\alpha$ | Su et al. (2021) |
| $\mathcal{X}$, homogeneous space | $G$ | $x \mapsto \{\rho_\lambda(x^G)\}_\lambda$ | Ours |

Lim*, Lawrence*, Huang, Thiede at TAG, ICML 2023

# Positional encodings as group representations

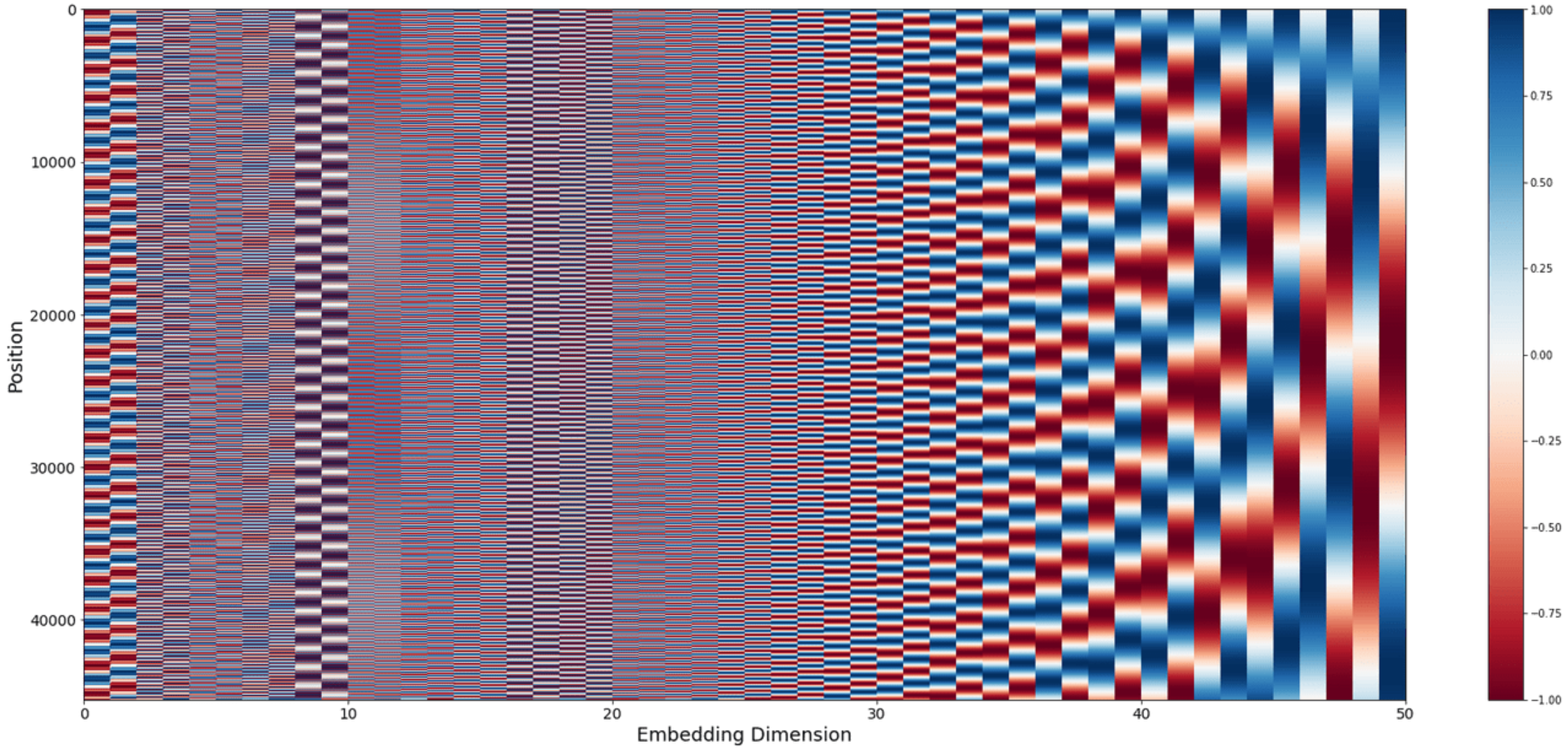
*Table 1.* Examples of positional encodings, interpreted as group representations. $Y_\ell^m$ denotes spherical harmonics, $v_i$ the $i$-th eigenvector of the graph Laplacian, $J(r)$ a radial function, and $R^{2\times 2}(\theta)$ is the $2 \times 2$ rotation matrix by $\theta$.

| Data Type | Group | Encoding | Ref. |
|---|---|---|---|
| Text | $T$ | $(x) \mapsto \{(\cos(\alpha x), \sin(\alpha x))\}_\alpha$ | Vaswani et al. (2017) |
| Image | $T \times T$ | $(x, y) \mapsto \{(\cos(\alpha_1 x + \alpha_2 y), \sin(\alpha_1 x + \alpha_2 y))\}_{\alpha_1,\alpha_2}$ | Dosovitskiy et al. (2021) |
| Molecule | $SO(3)$ | $(r, \theta, \phi) \mapsto \{Y_\ell^m(\theta, \phi)J(r)\}_{\ell,m}$ | Thomas et al. (2018) |
| Graph | $S_{\lvert\mathcal{X}\rvert}$ | $(x) \mapsto \{v_i(x)\}_i$ | Lim et al. (2023) |
| Any (learned embedding) | $S_{\lvert\mathcal{X}\rvert}$ | $x \mapsto \texttt{one\_hot}(x)$ | Gehring et al. (2017) |
| Text (spherical embedding) | $SO(2)^{n/2}$ | $(m) \mapsto \{\bigoplus R^{2\times 2}(m\alpha)\}_\alpha$ | Su et al. (2021) |
| $\mathcal{X}$, homogeneous space | $G$ | $x \mapsto \{\rho_\lambda(x^G)\}_\lambda$ | Ours |

Also: prescription for how to design positional encodings for new data

Lim*, Lawrence*, Huang, Thiede at TAG, ICML 2023

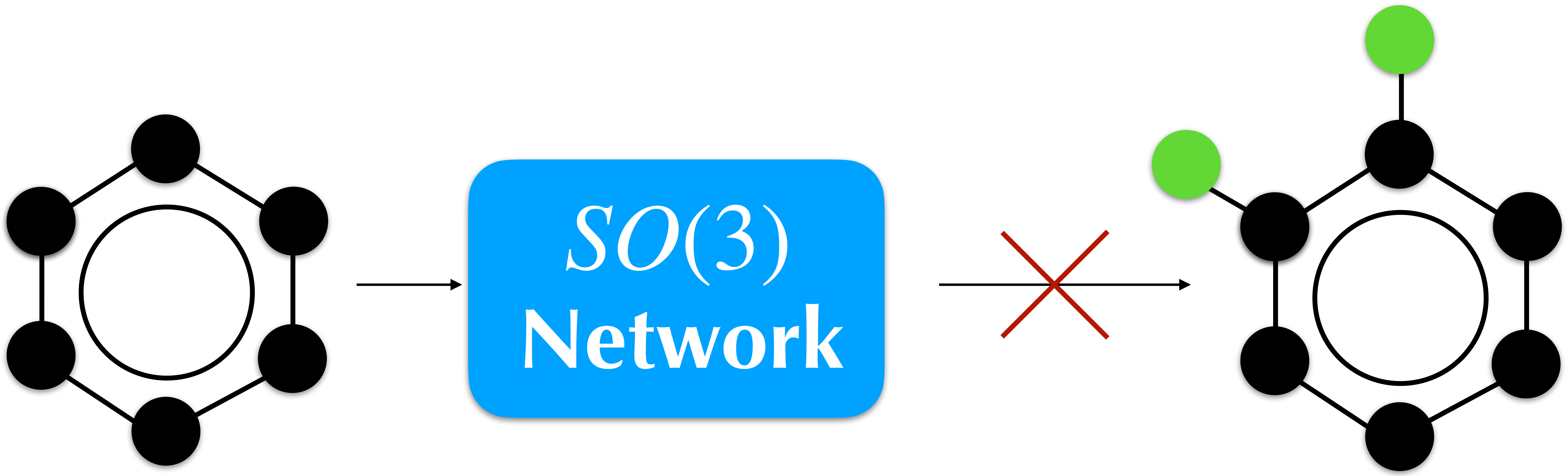
# Positional encodings break permutation invariance



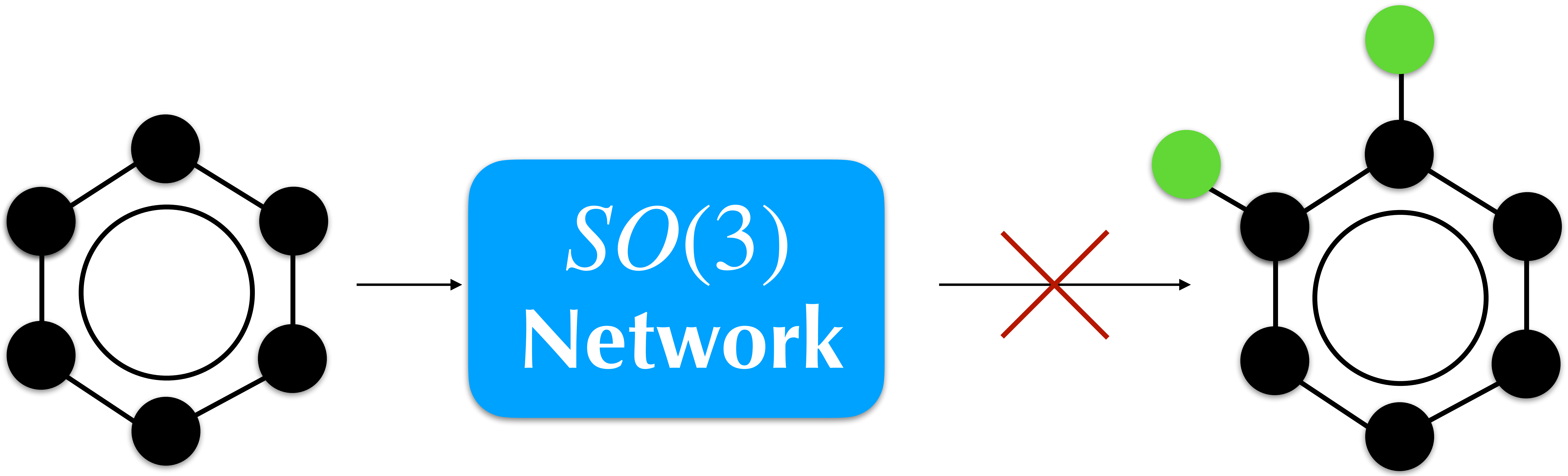
# Symmetry-breaking positional encodings

Lawrence*, Portilheiro*, Zhang, Kaba at ICLR 2025

# Equivariant functions can't break symmetries



# Equivariant functions can't break symmetries




$$x = gx \rightarrow f(x) = f(gx) = gf(x)$$

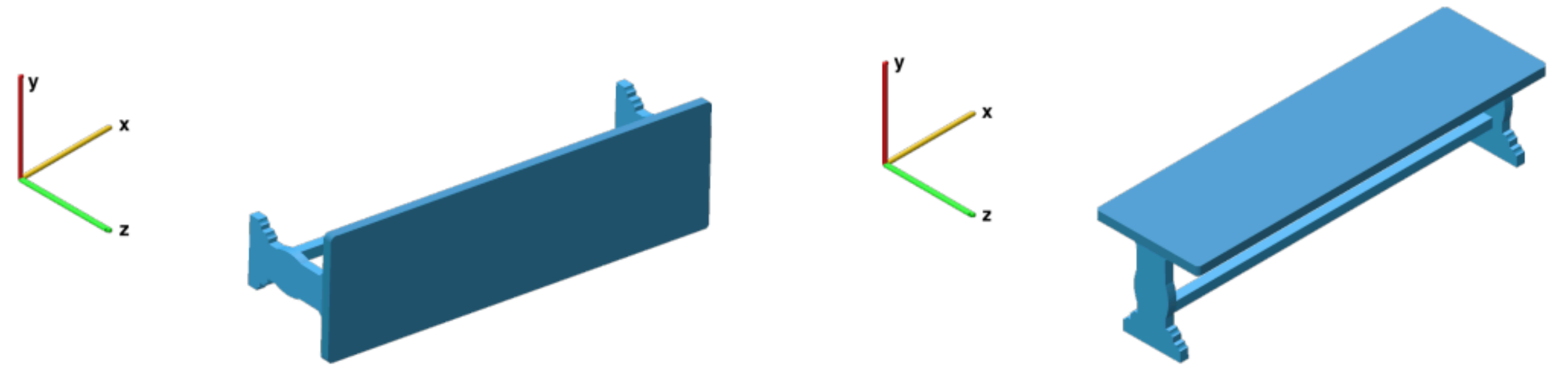
# This is a problem in many applications

ORIENT ANYTHING

**Christopher Scarvelis** *
MIT CSAIL
Cambridge, MA
scarv@mit.edu

**David Benhaim**
Backflip AI
San Francisco, CA
david@backflip.ai

**Paul Zhang**
Backflip AI
San Francisco, CA
paul.zhang@backflip.ai



(a) Rotated shape (b) Shape in canonical orientation

Learn a "standard" orientation for 3D models

**SymILO: A Symmetry-Aware Learning Framework for Integer Linear Optimization**

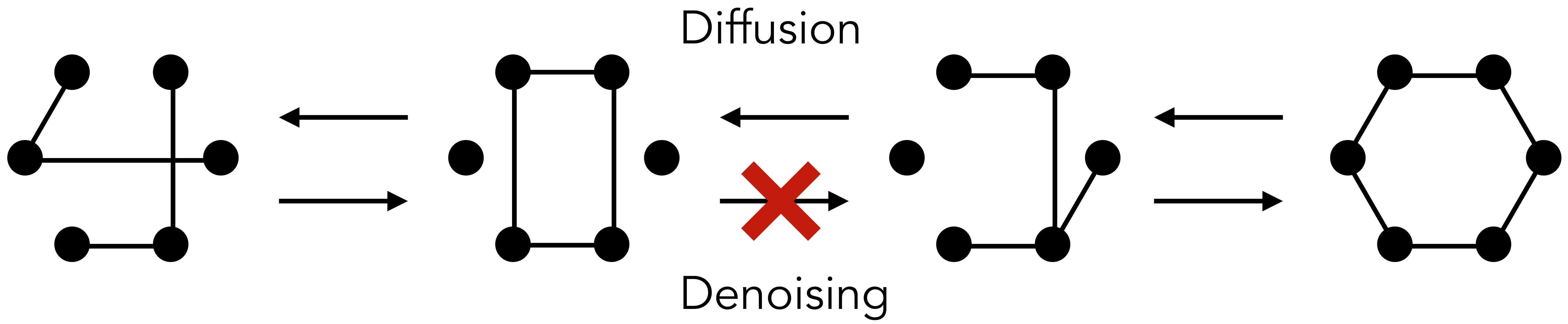
Qian Chen[1,2], Tianjian Zhang[1,2], Linxin Yang[2,3], Qingyu Han[2], Akang Wang[2,3,*], Ruoyu Sun[2,3], Xiaodong Luo[2,3], and Tsung-Hui Chang[1,2]

$$\begin{aligned} \min_{\boldsymbol{x}} \ & \boldsymbol{c}^\top \boldsymbol{x} \\ \text{s.t. } & \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b} \\ & \boldsymbol{x} \in \mathbb{Z}^n, \end{aligned}$$

Solving integer linear programs

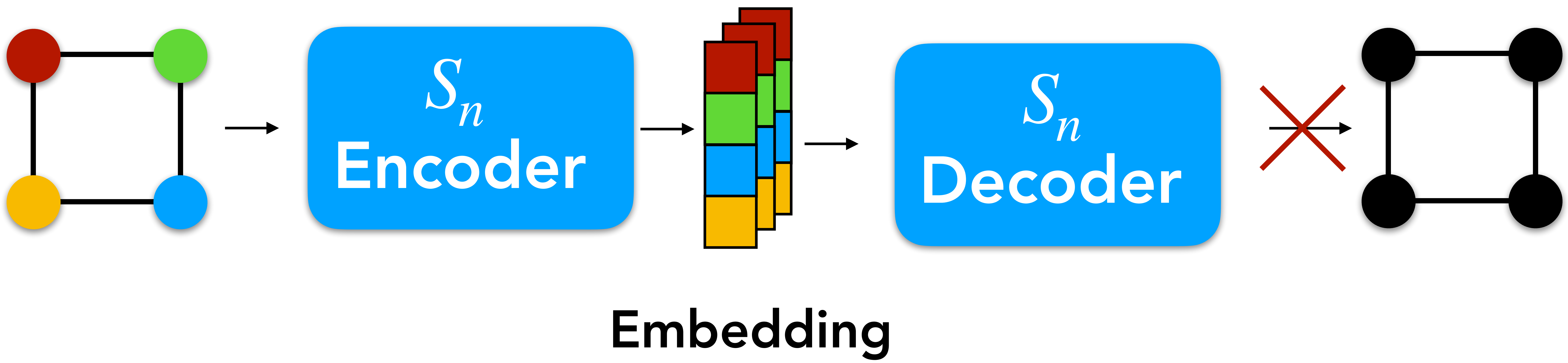
# It can also arise in generative modeling

**Problem:** If noising process introduces symmetries, they cannot be denoised

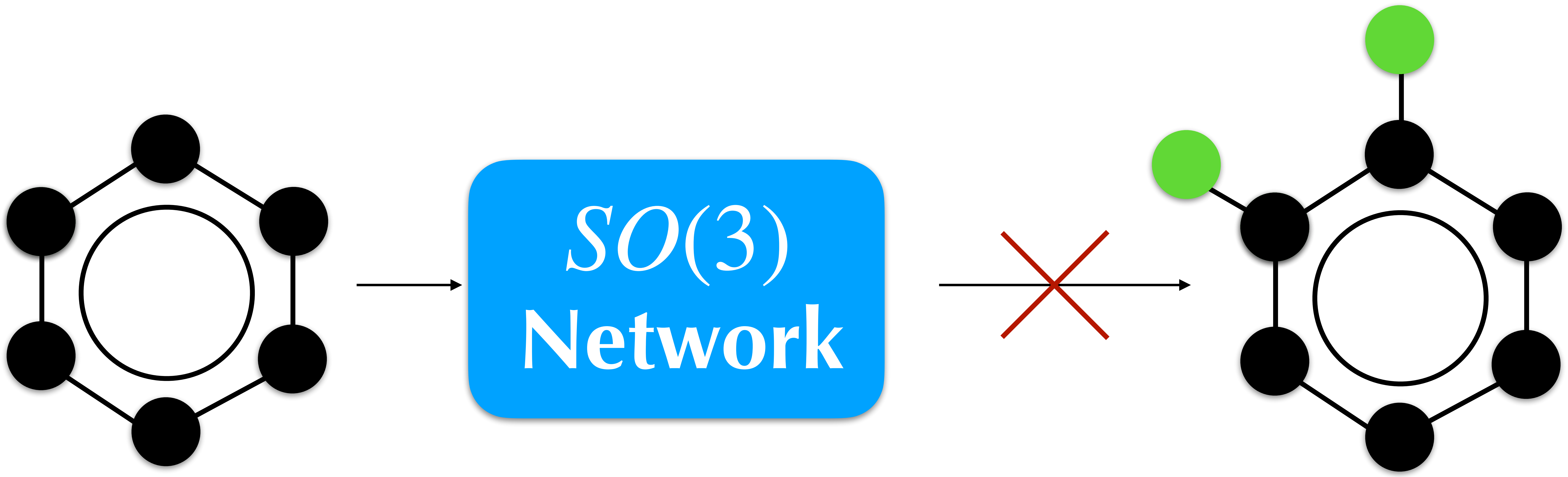


# It can also arise in generative modeling

## Graph Autoencoder

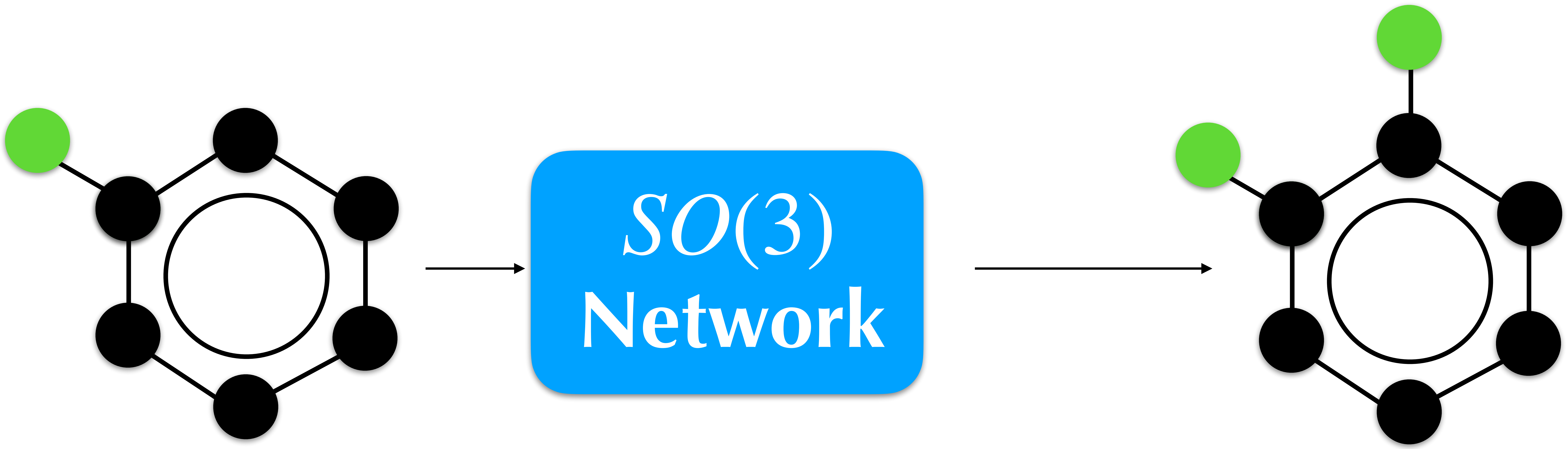


# Equivariant functions can't break symmetries

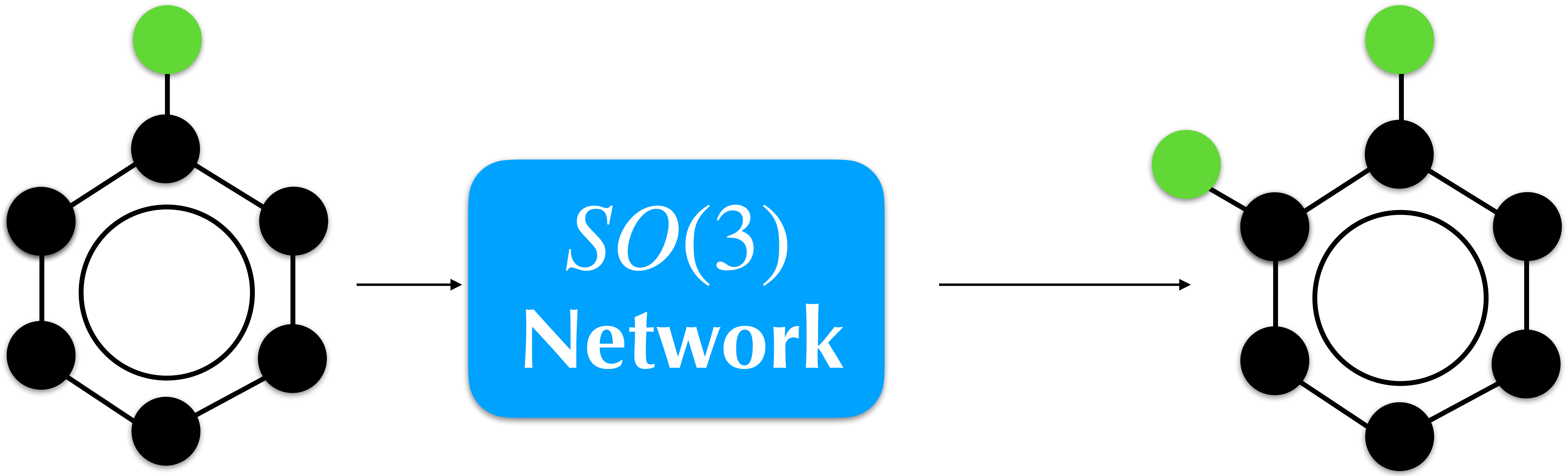



$$x = gx \rightarrow f(x) = f(gx) = gf(x)$$

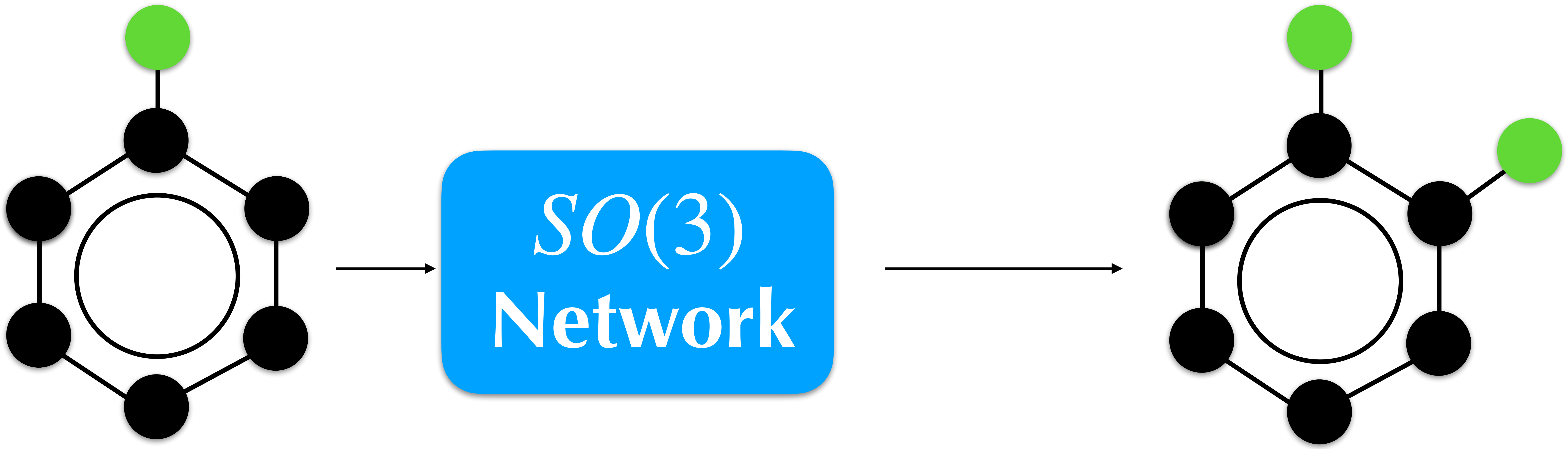
# But, want equivariance when possible!



# But, want equivariance when possible!



# But, want equivariance when possible!



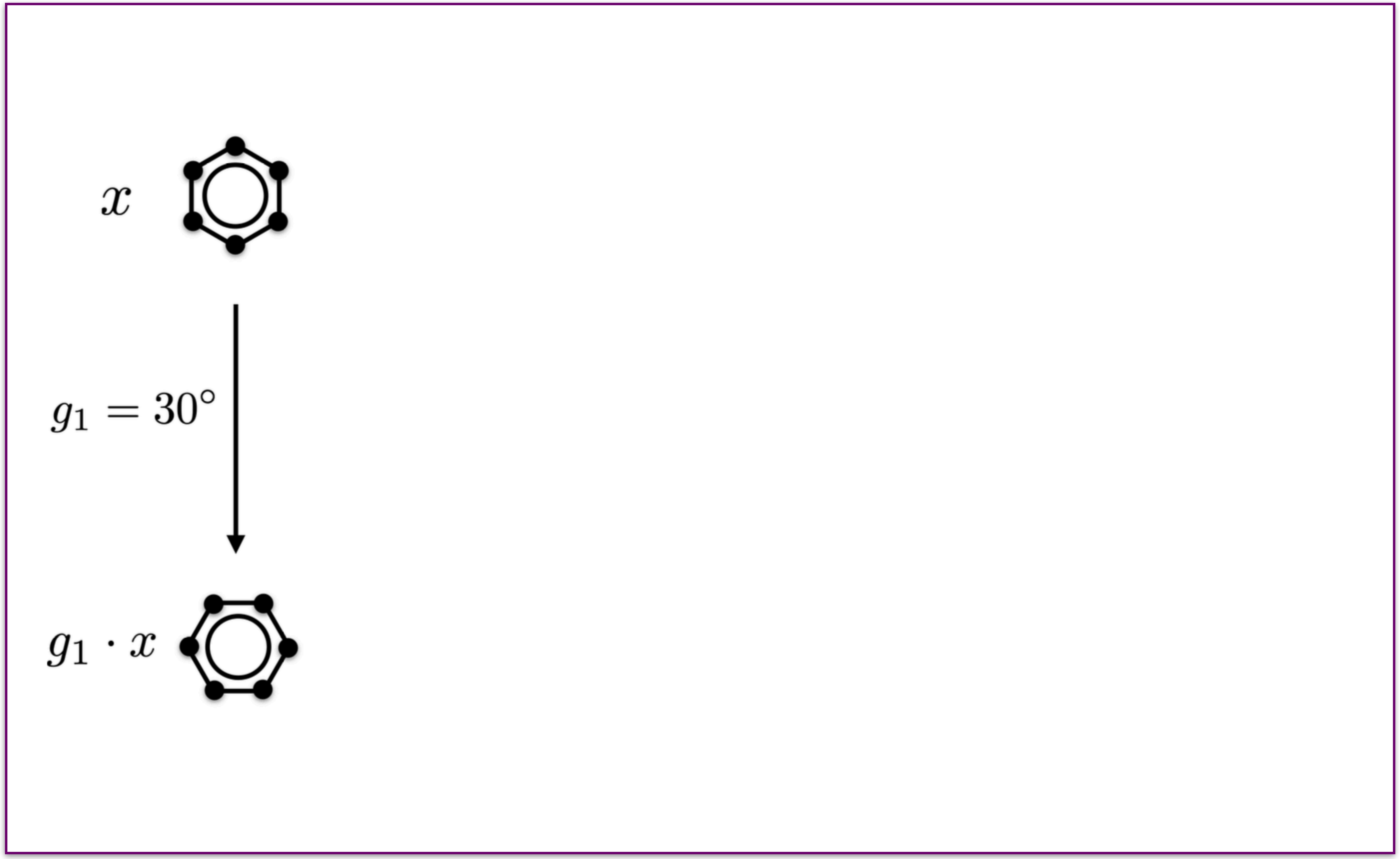
How do we even formulate this class of functions?

# One solution: probabilistic

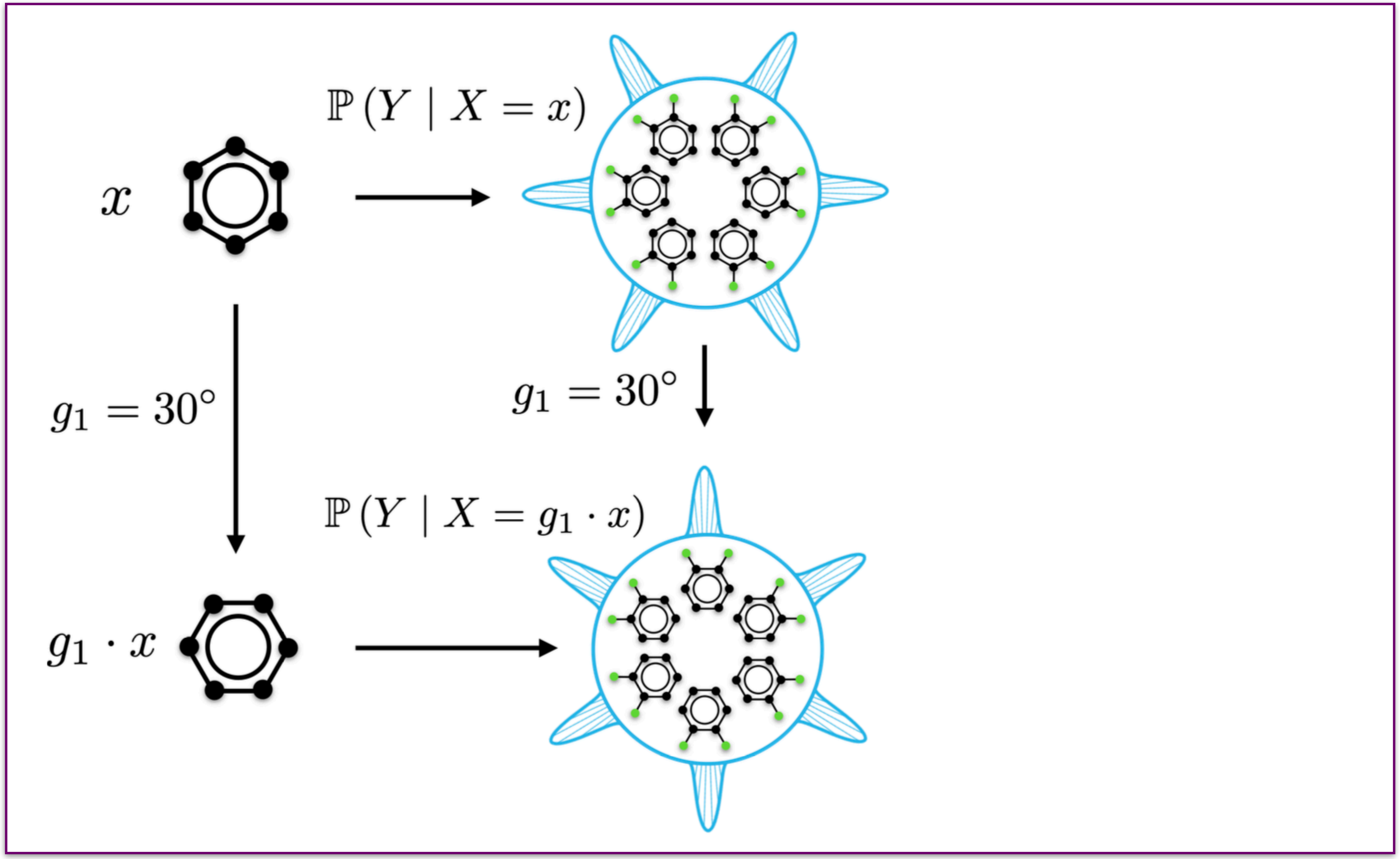
Learn an equivariant *distribution* $f : X \to \mathscr{P}(Y)$, such that individual samples from $f(x)$ can break the symmetry of $x$

*Extension of SBS perspective of outputting a set!*

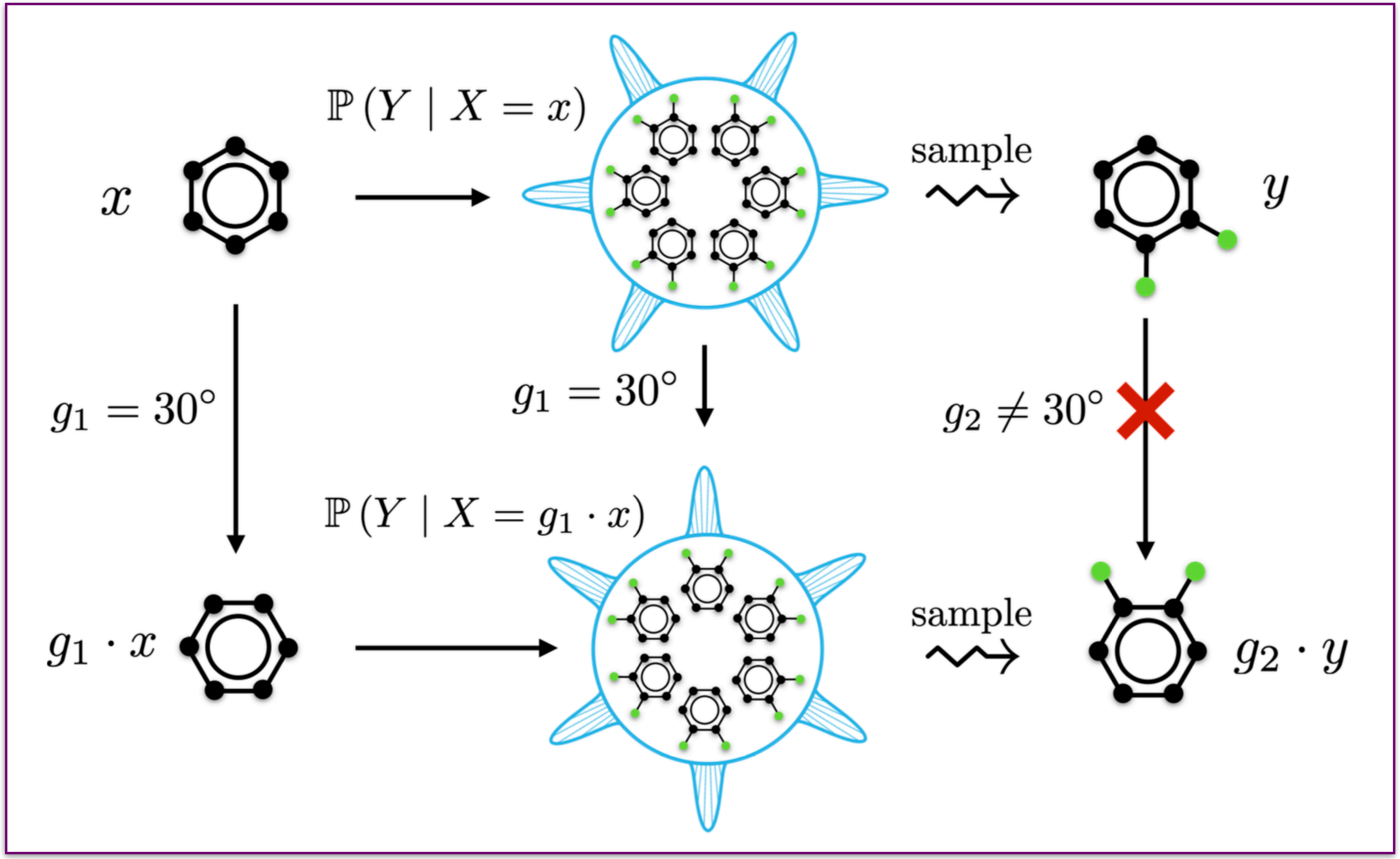
# One solution: probabilistic



# One solution: probabilistic



# One solution: probabilistic



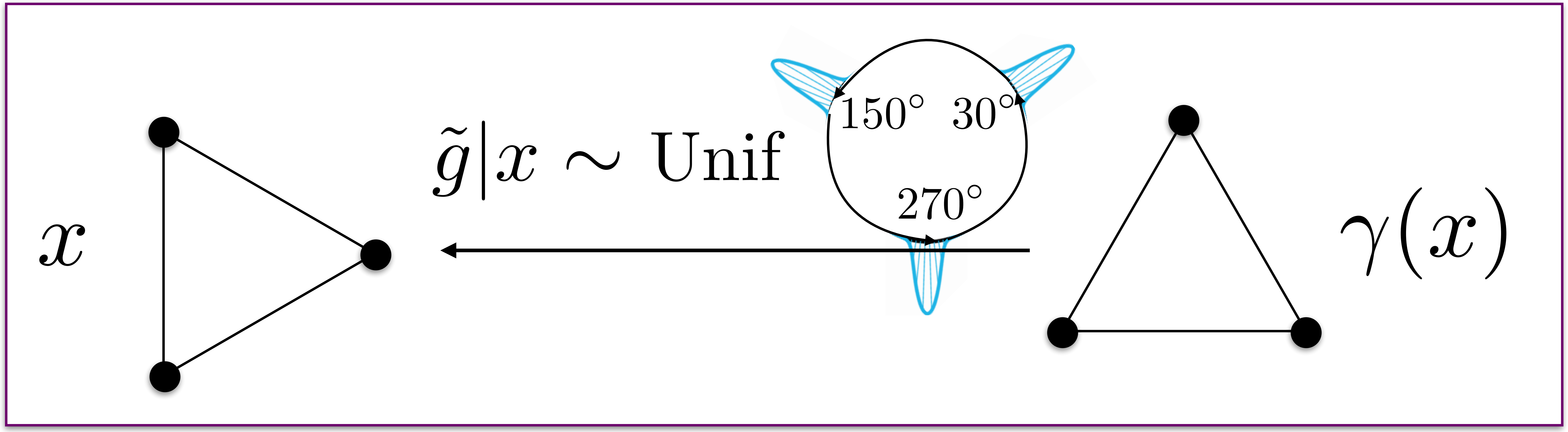
How do you learn equivariant **distributions?**

# Noise outsourcing

Turn a regular neural network $\phi : \mathscr{X} \times [0,1] \rightarrow \mathscr{Y}$ into a network that outputs distributions, $\phi' : \mathscr{X} \rightarrow \mathscr{P}(\mathscr{Y})$, by sampling noise $\epsilon$

# Quick definition: $\tilde{g}$

**Conditions**: Let $\gamma(X)$ be a canonicalization, and $\tilde{g}$ be a probabilistic inverse of $\gamma(X)$.



# Solution: symmetry-breaking input

**Practical corollary:** $Y \mid X$ is equivariant iff, for some $f : X \times G \times (0,1) \to Y$ jointly equivariant in $X$ and $g$,

$$Y \overset{a.s.}{=} f(X, \tilde{g}, \epsilon)$$

Symmetry-breaking positional encoding

"Arbitrary randomness"

# Solution: symmetry-breaking input

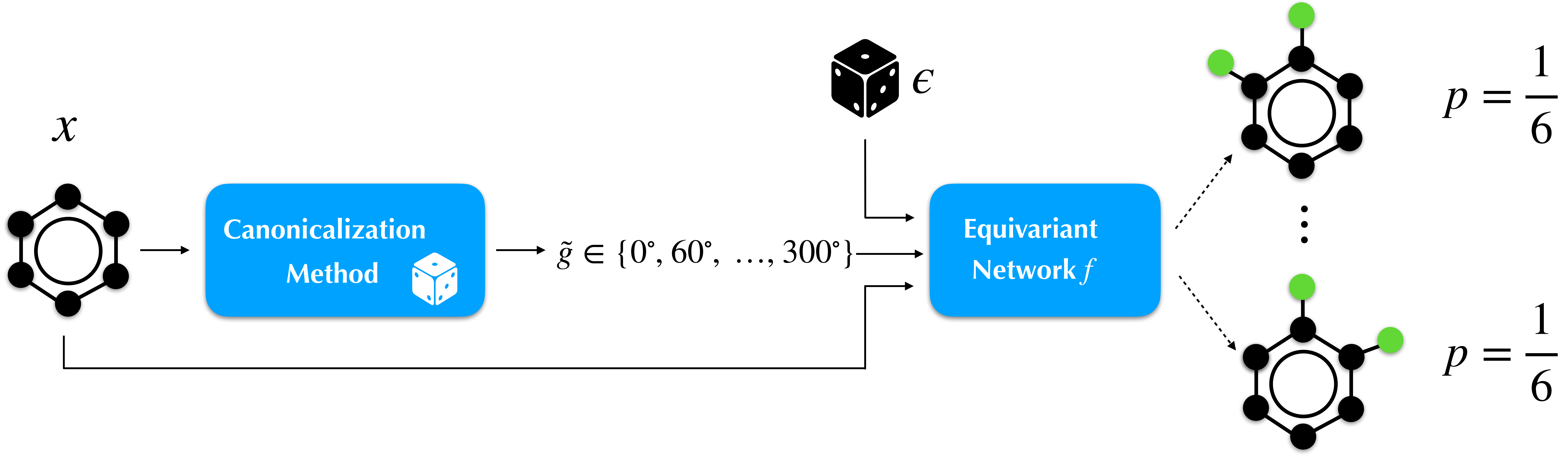
**Practical corollary:** $Y \mid X$ is equivariant iff, for some $f : X \times G \times (0,1) \to Y$ jointly equivariant in $X$ and $g$,

$$Y \overset{a.s.}{=} f(X, \tilde{g}, \epsilon)$$

**Symmetry-breaking positional encoding** **"Arbitrary randomness"**

**Generalization to "noise injection":** can let $\tilde{g}$ more generally be a random variable with no self-symmetries and $\tilde{g} \mid X \sim h\tilde{g} \mid hX$. Important for problems where it's hard to canonicalize!

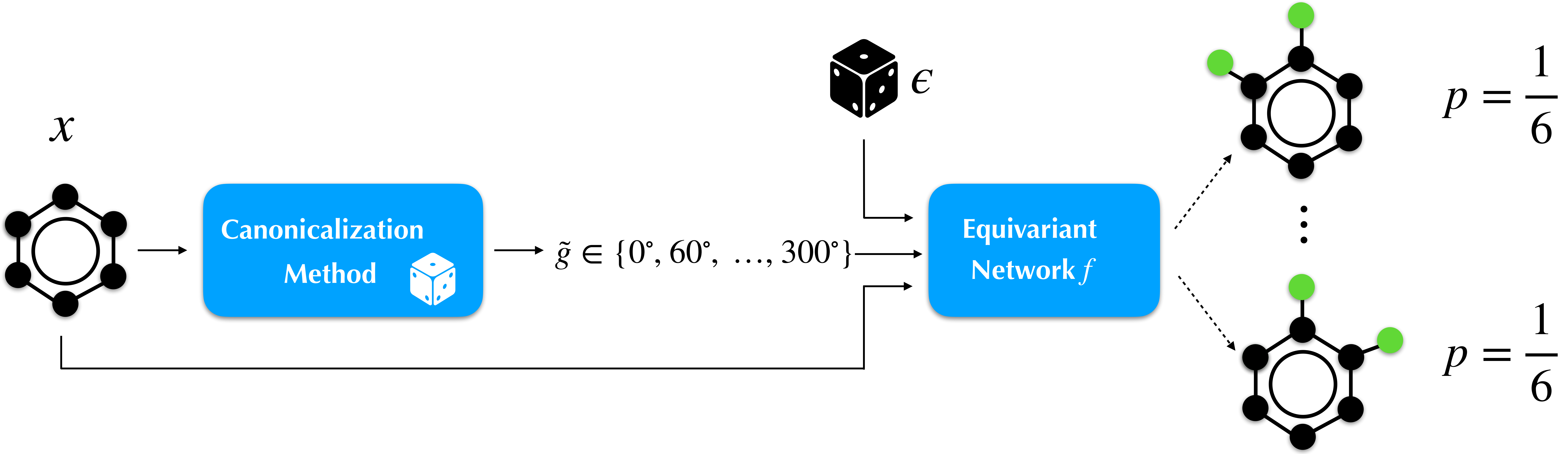
# A more convenient implementation




**To pass $g$ into a neural network, fix or learn a vector $v$ and pass $gv$ as input**

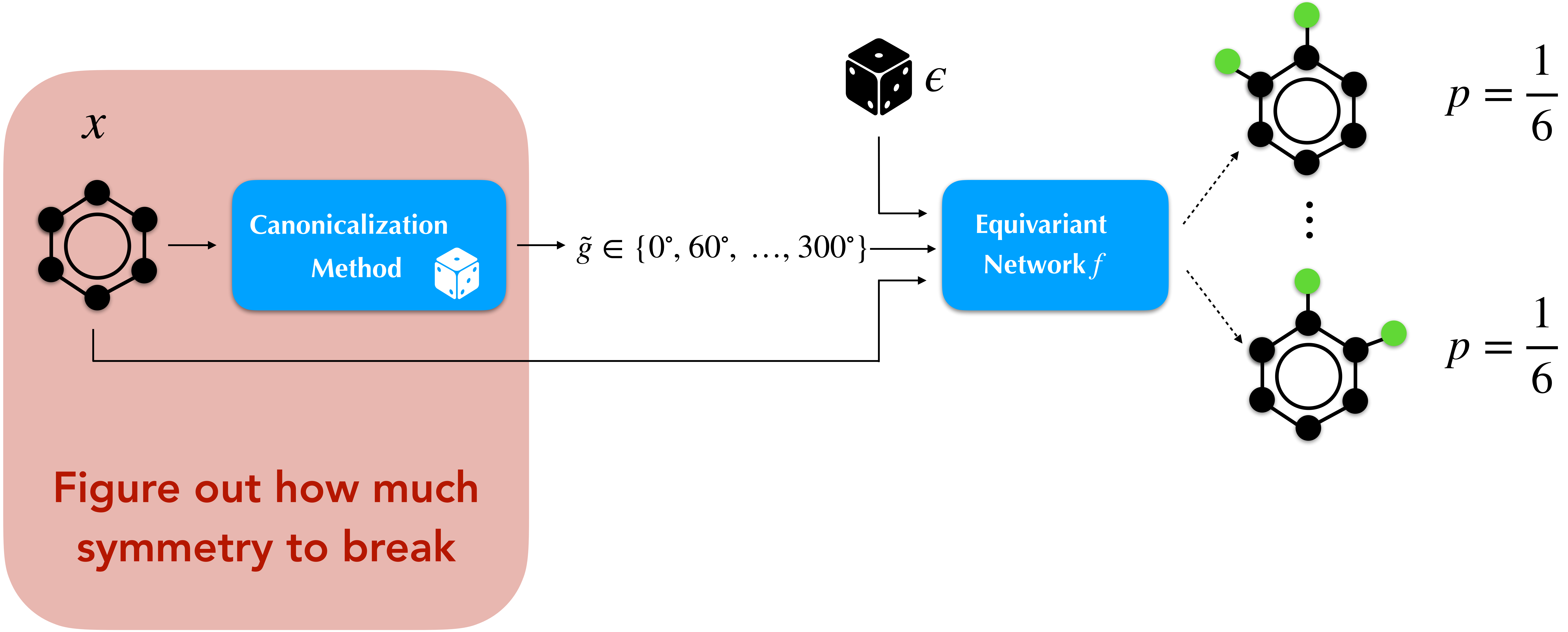
*"symmetry-breaking positional encoding"*

# A more convenient implementation



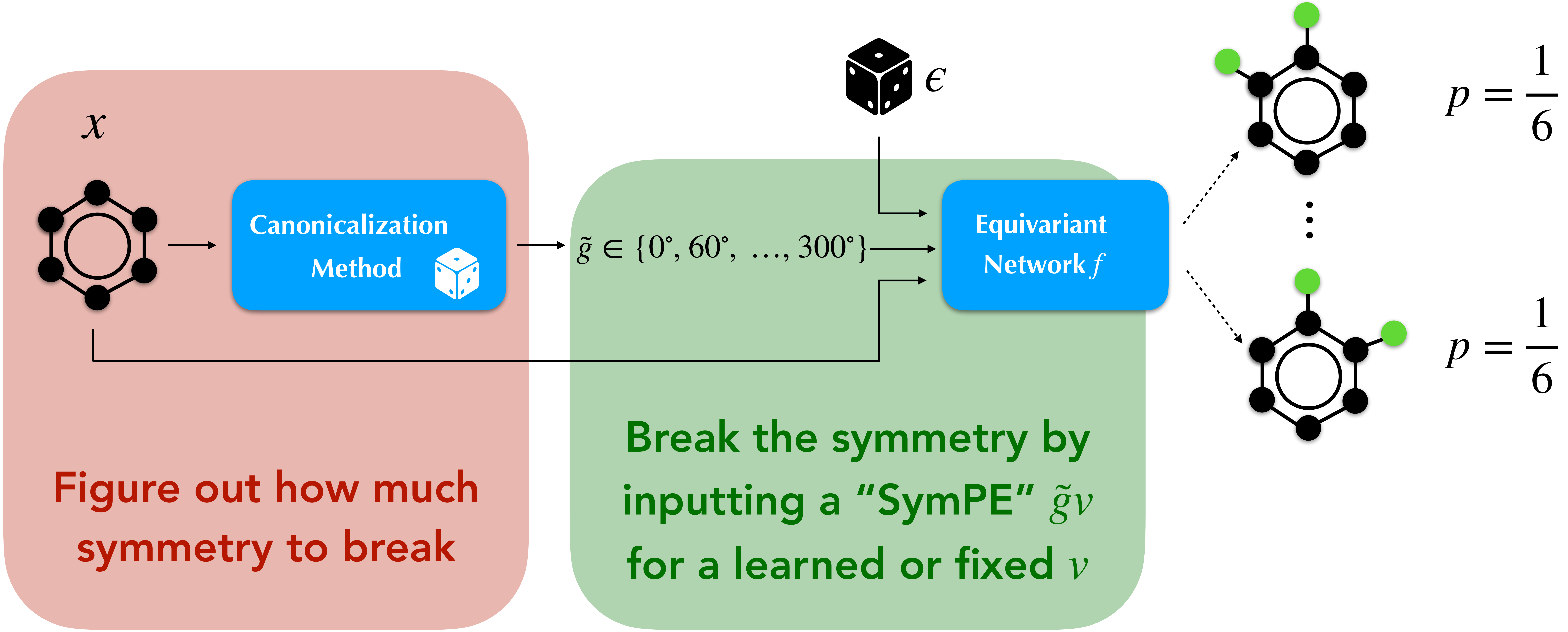

*SymPE = symmetry-breaking positional encoding*

# A more convenient implementation




*SymPE = symmetry-breaking positional encoding*

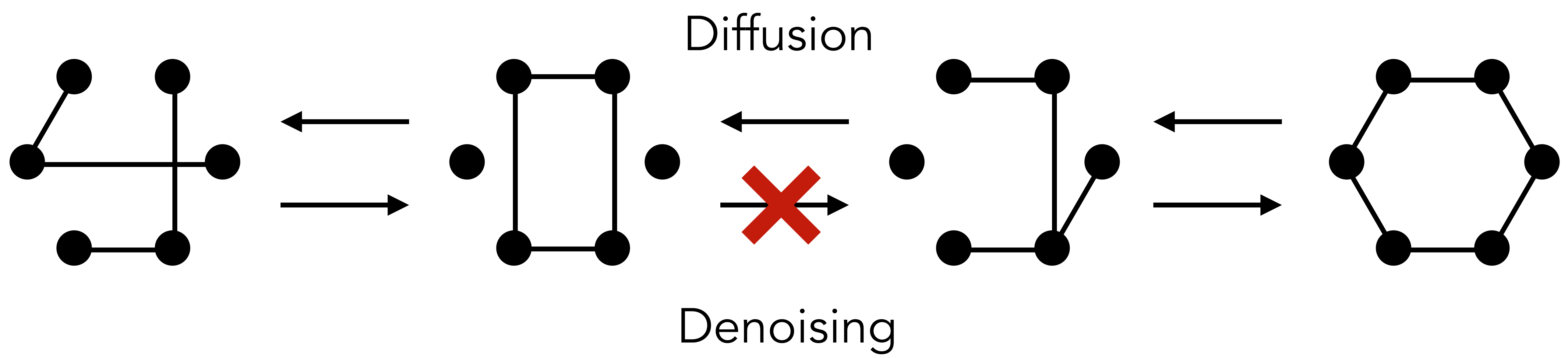
# A more convenient implementation




*SymPE = symmetry-breaking positional encoding*

# Experiment: Graph generation with diffusion model

**Problem:** Noising process is likely to introduce symmetries that cannot be denoised




**Experiment:** Digress discrete diffusion with graph transformer. Use graph network (IGN) to sample $\tilde{g}$

| Method | NLL |
|---|---|
| DiGress | 129.7 |
| DiGress + noise | 126.5 |
| DiGress + SymPE | 30.3 |

# Part 3: Tokenization

# How do LLMs process data?

**Data:** comes in an ordered *sequence*, e.g. paragraphs of text. How do you turn this into a learning task?

**Autoregressive learning task:** try to predict the next word, one at a time

| | | | | |
|---|---|---|---|---|
| The | cat | in | the | hat |

# How do LLMs process data?

**Data:** comes in an ordered *sequence*, e.g. paragraphs of text. How do you turn this into a learning task?

**Autoregressive learning task:** try to predict the next word, one at a time

The

# How do LLMs process data?

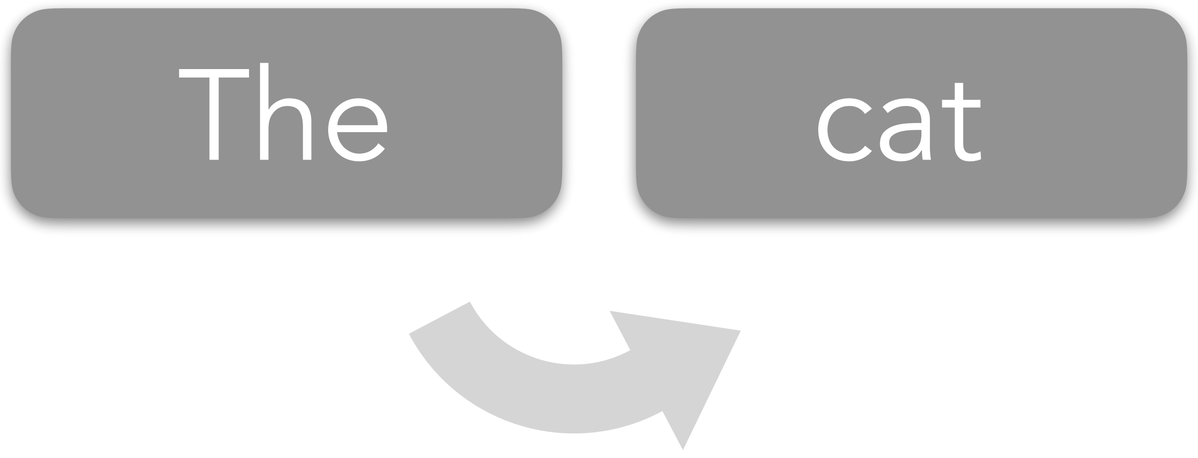
**Data:** comes in an ordered *sequence*, e.g. paragraphs of text. How do you turn this into a learning task?

**Autoregressive learning task:** try to predict the next word, one at a time

# How do LLMs process data?

**Data:** comes in an ordered *sequence*, e.g. paragraphs of text. How do you turn this into a learning task?

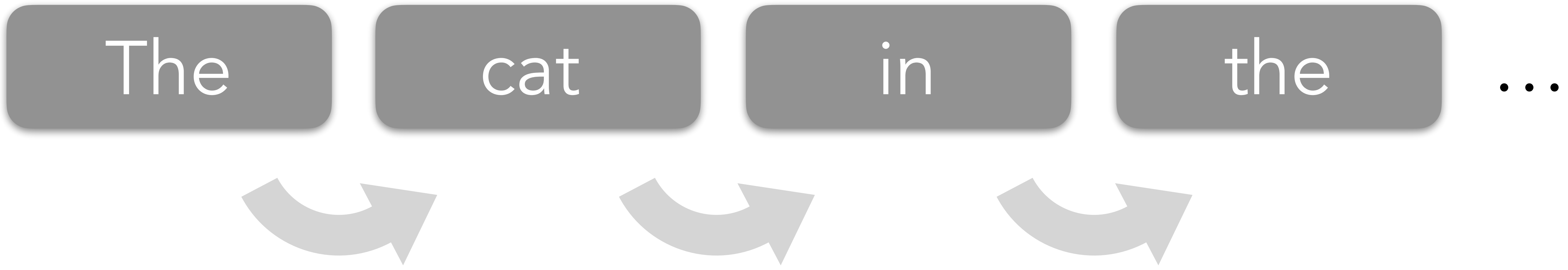
**Autoregressive learning task:** try to predict the next word, one at a time



# How do LLMs process data?

**Data:** comes in an ordered *sequence*, e.g. paragraphs of text. How do you turn this into a learning task?

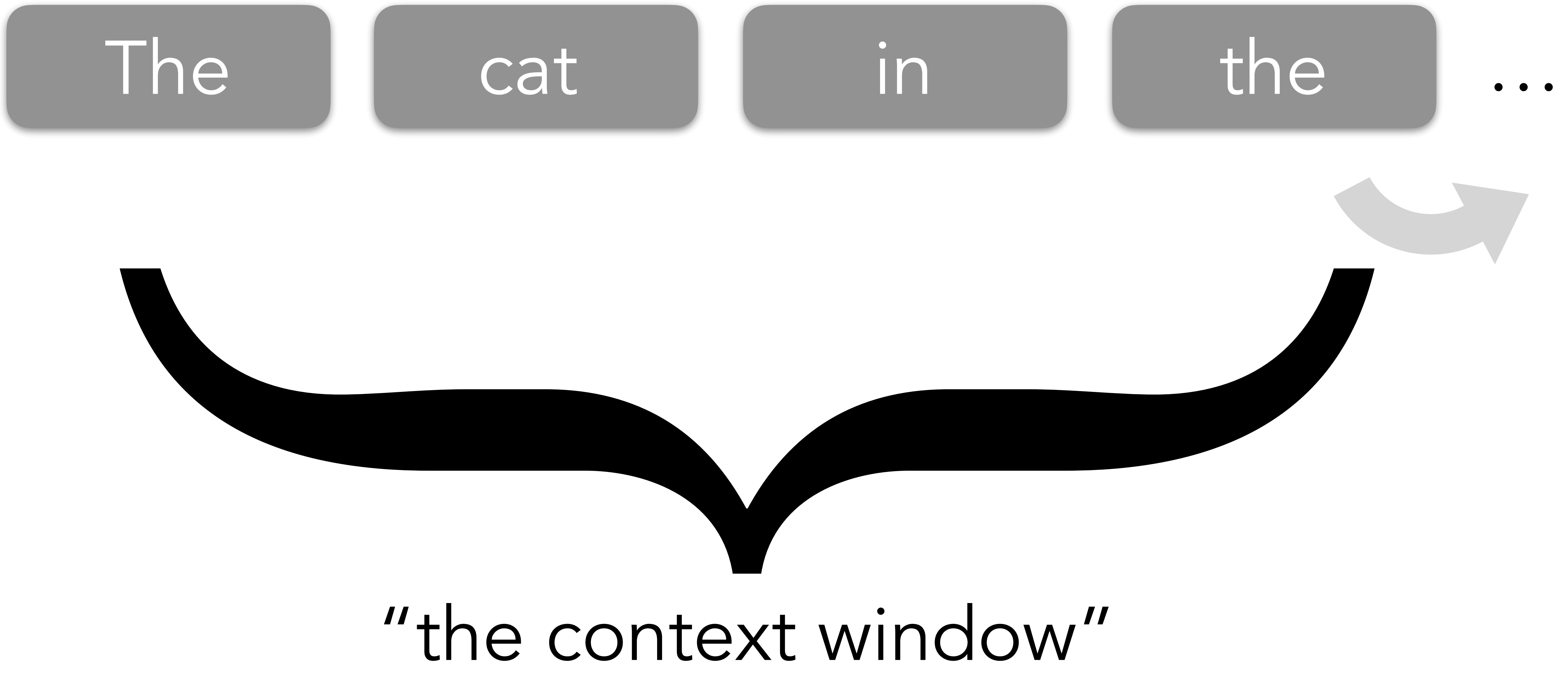
**Autoregressive learning task:** try to predict the next word, one at a time



# How do LLMs process data?

**Data:** comes in an ordered *sequence*, e.g. paragraphs of text. How do you turn this into a learning task?

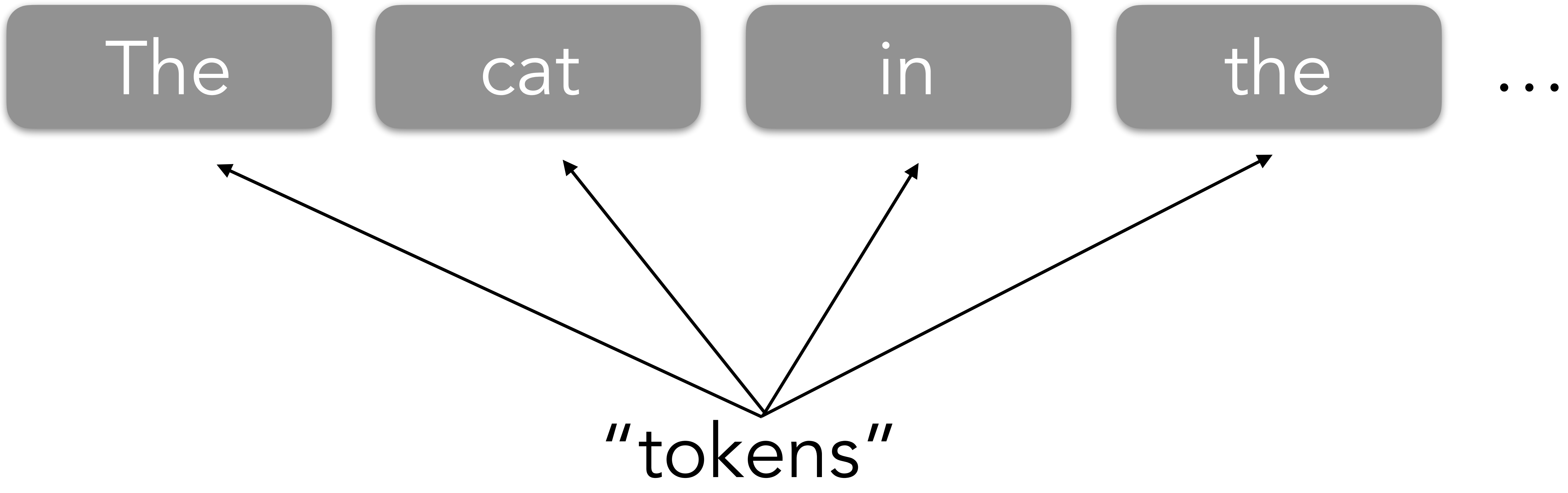
**Autoregressive learning task:** try to predict the next word, one at a time



# How do LLMs process data?

**Data:** comes in an ordered *sequence*, e.g. paragraphs of text. How do you turn this into a learning task?

**Autoregressive learning task:** try to predict the next word, one at a time



# How do LLMs process data, **concretely**?

The cat in the …

# How do LLMs process data, **concretely**?

The | cat | in | the | …

$$\begin{bmatrix} 12.1 \\ 42.0 \\ 0.8 \\ \vdots \\ 2.3 \end{bmatrix} \quad \begin{bmatrix} 0.2 \\ 0.7 \\ 0.6 \\ \vdots \\ 8.2 \end{bmatrix} \quad \begin{bmatrix} 2.5 \\ 0.1 \\ 0.2 \\ \vdots \\ 10.9 \end{bmatrix} \quad \begin{bmatrix} 0.2 \\ -1.9 \\ 0.4 \\ \vdots \\ 7.4 \end{bmatrix}$$

Token embeddings

Learn a **vector embedding** for every word/token in a vocabulary (size 30-100k for natural language)

# How do LLMs process data, **concretely**?

The | cat | in | the | …

$$\begin{bmatrix} 12.1 \\ 42.0 \\ 0.8 \\ \vdots \\ 2.3 \end{bmatrix} \quad \begin{bmatrix} 0.2 \\ 0.7 \\ 0.6 \\ \vdots \\ 8.2 \end{bmatrix} \quad \begin{bmatrix} 2.5 \\ 0.1 \\ 0.2 \\ \vdots \\ 10.9 \end{bmatrix} \quad \begin{bmatrix} 0.2 \\ -1.9 \\ 0.4 \\ \vdots \\ 7.4 \end{bmatrix}$$

Token embeddings

**Neural Net** {
*input:* **embeddings** of all the tokens in the context window
→
*output:* a **distribution over all possible words**

# How do LLMs process data, **concretely**?

The | cat | in | the | …

$$\begin{bmatrix} 12.1 \\ 42.0 \\ 0.8 \\ \vdots \\ 2.3 \end{bmatrix} \quad \begin{bmatrix} 0.2 \\ 0.7 \\ 0.6 \\ \vdots \\ 8.2 \end{bmatrix} \quad \begin{bmatrix} 2.5 \\ 0.1 \\ 0.2 \\ \vdots \\ 10.9 \end{bmatrix} \quad \begin{bmatrix} 0.2 \\ -1.9 \\ 0.4 \\ \vdots \\ 7.4 \end{bmatrix}$$

Token embeddings

Net →

Hat: 54%

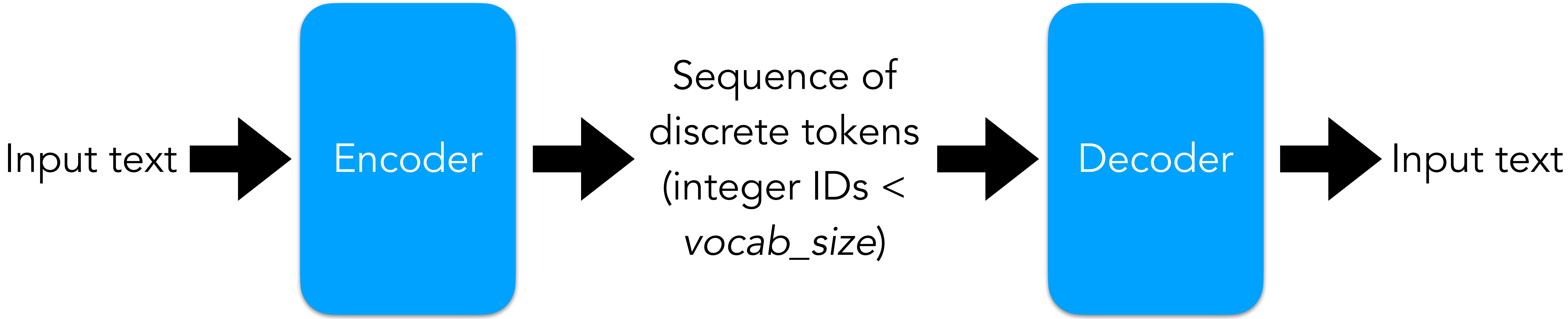
Chair: 19%

Window: 9%

Yard: 18%

**Neural Net**
- *input:* **embeddings** of all the tokens in the context window
- →
- *output:* a **distribution over all possible words**

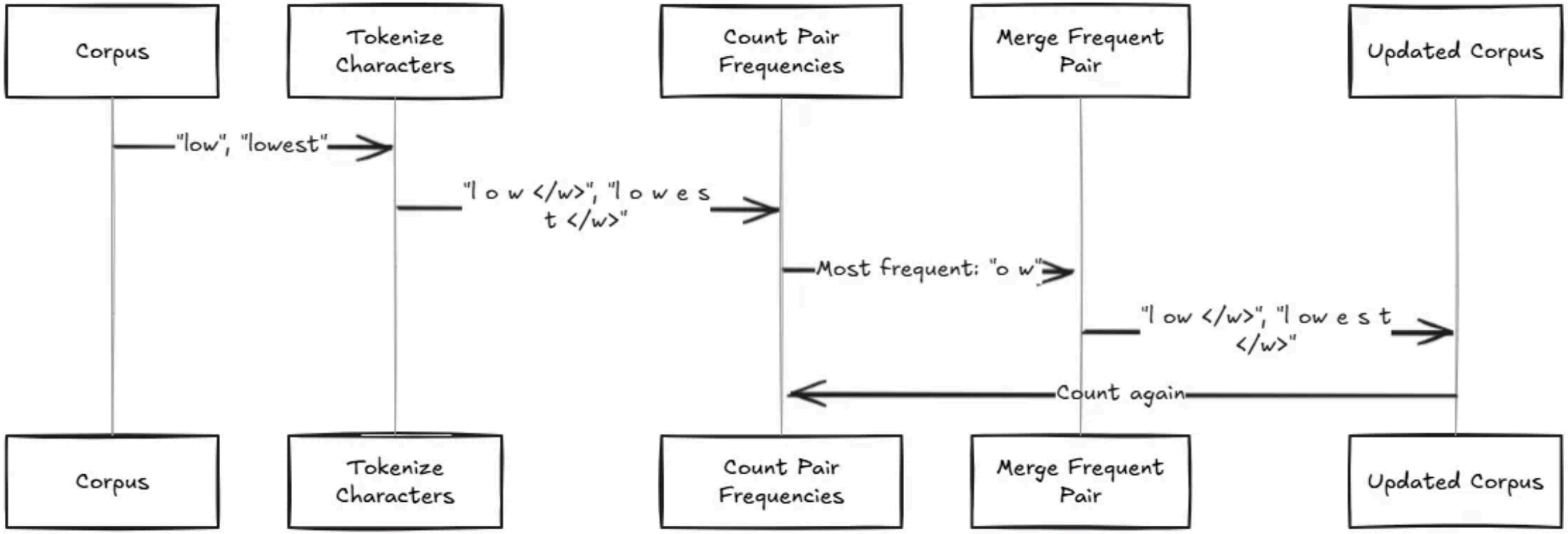
# Tokenization looks a lot like compression…




https://vizuara.substack.com/p/understanding-byte-pair-encoding

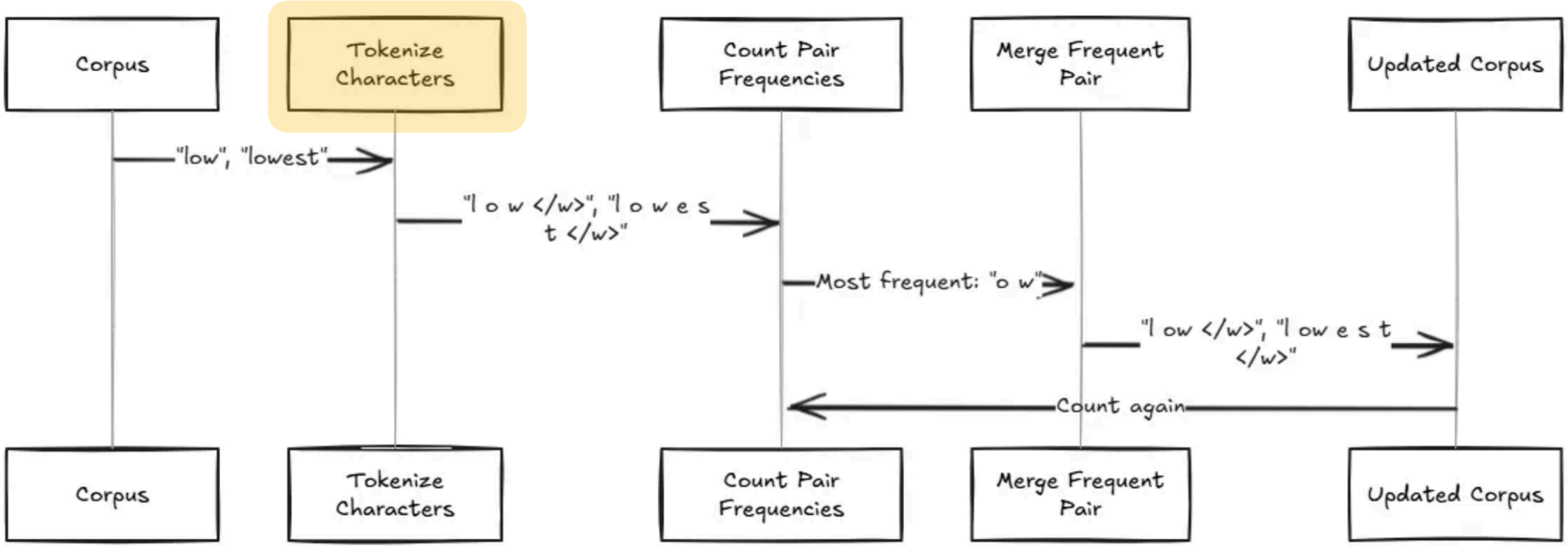
# Tokenization looks a lot like compression…

Byte pair encoding:



# Tokenization looks a lot like compression...

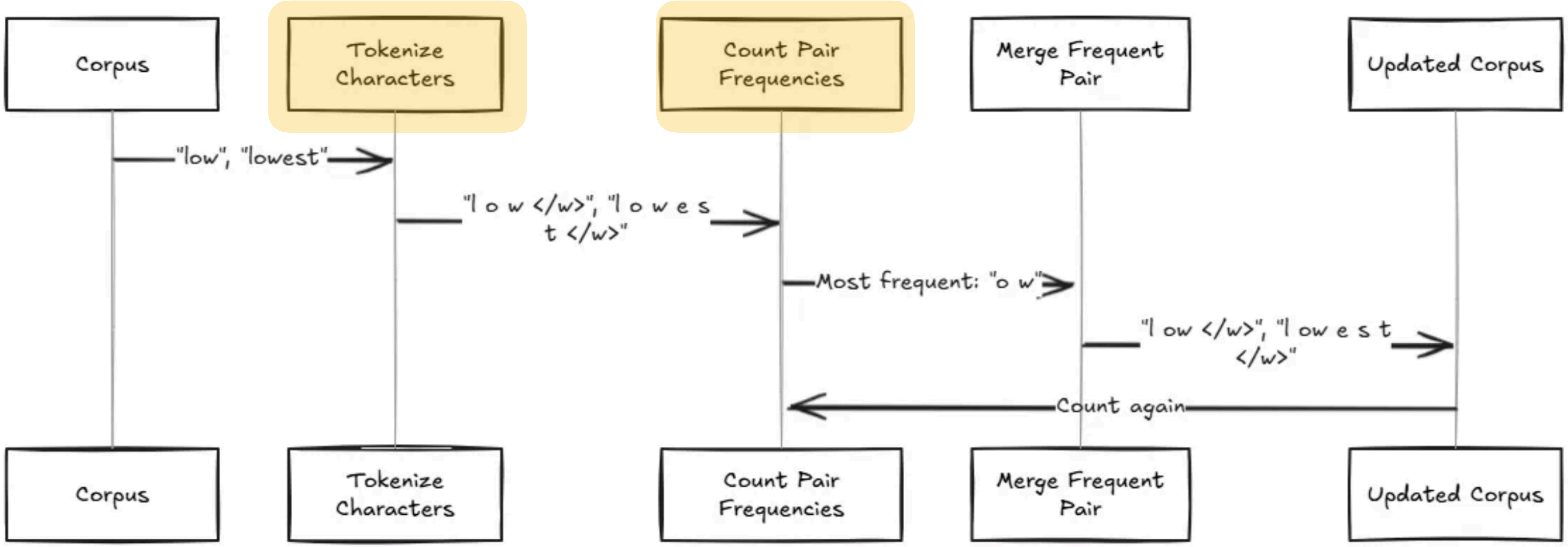
Byte pair encoding:




https://vizuara.substack.com/p/understanding-byte-pair-encoding

# Tokenization looks a lot like compression…

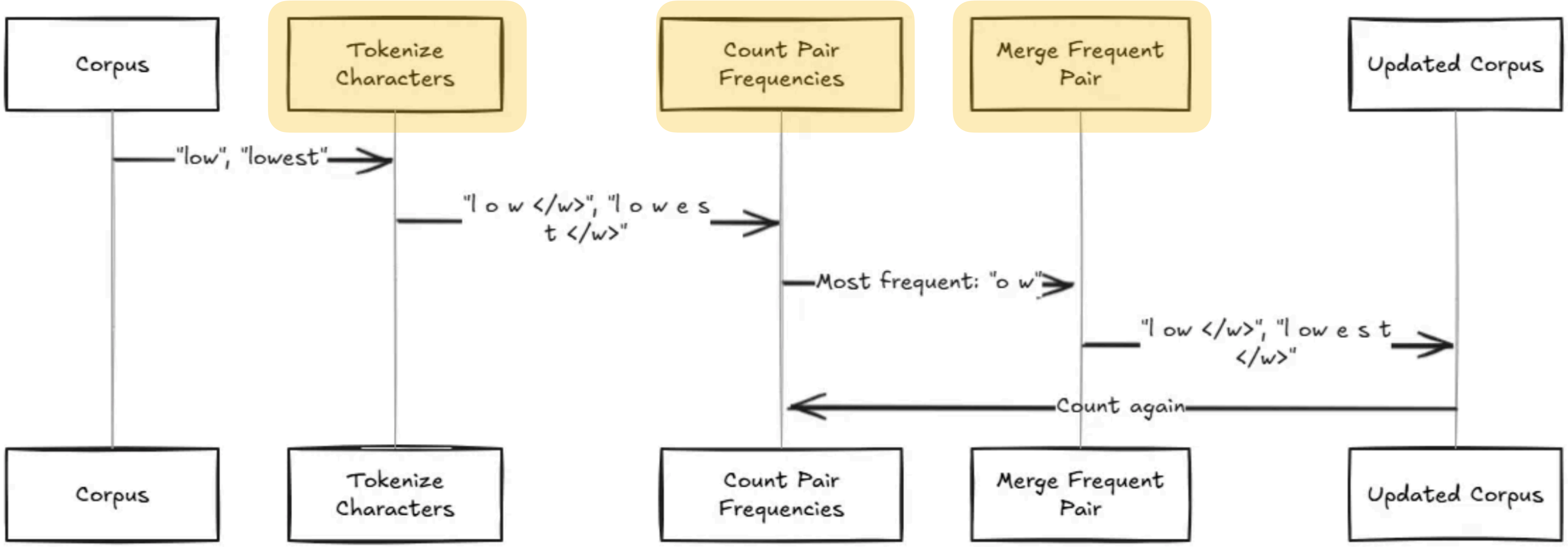
Byte pair encoding:




https://vizuara.substack.com/p/understanding-byte-pair-encoding

# Tokenization looks a lot like compression...

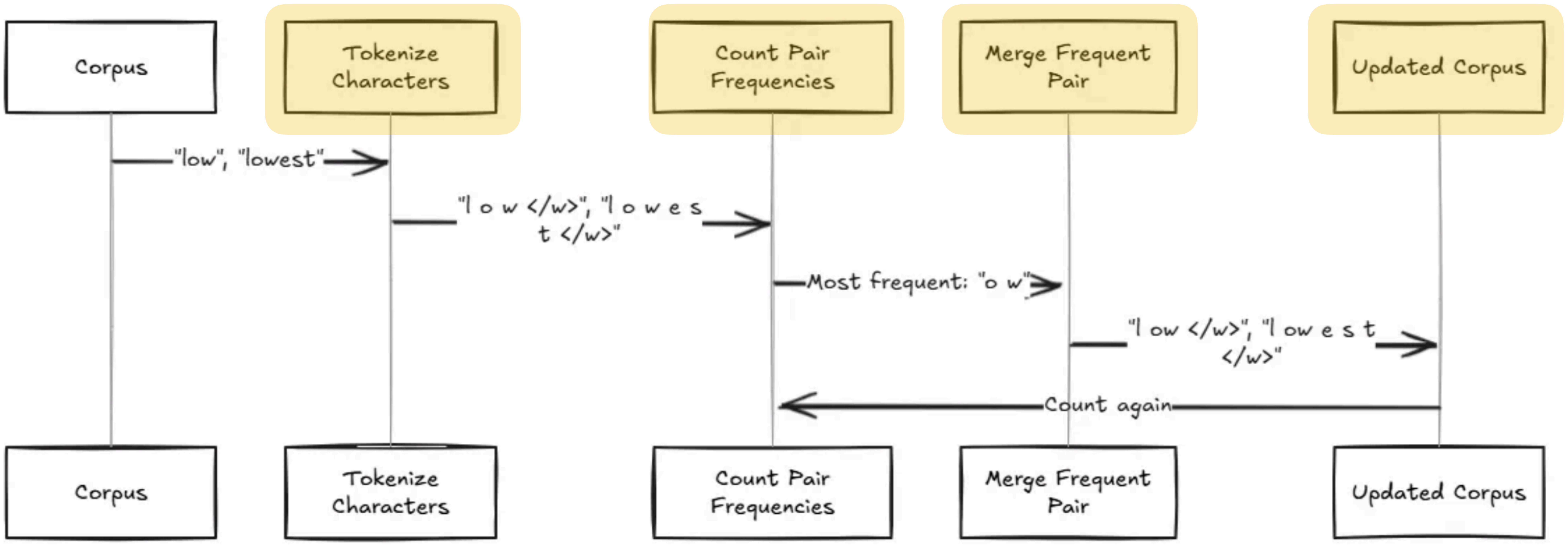
Byte pair encoding:




https://vizuara.substack.com/p/understanding-byte-pair-encoding

# Tokenization looks a lot like compression…

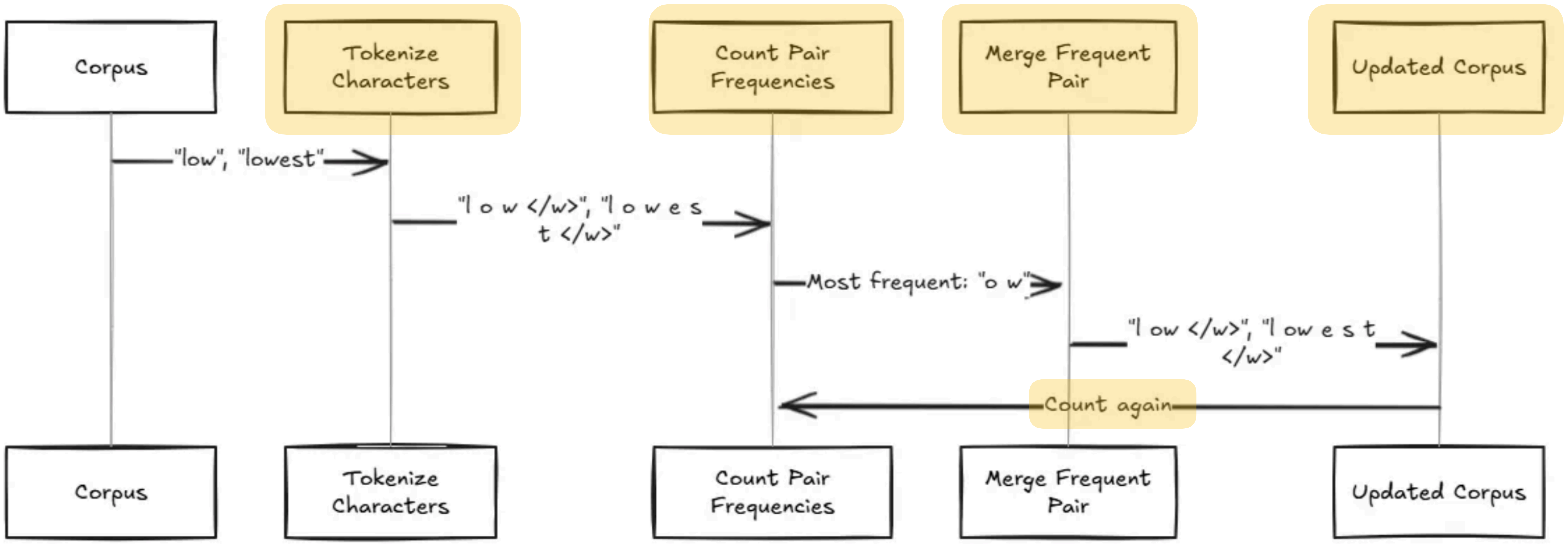
Byte pair encoding:




https://vizuara.substack.com/p/understanding-byte-pair-encoding

# Tokenization looks a lot like compression…

Byte pair encoding:




https://vizuara.substack.com/p/understanding-byte-pair-encoding

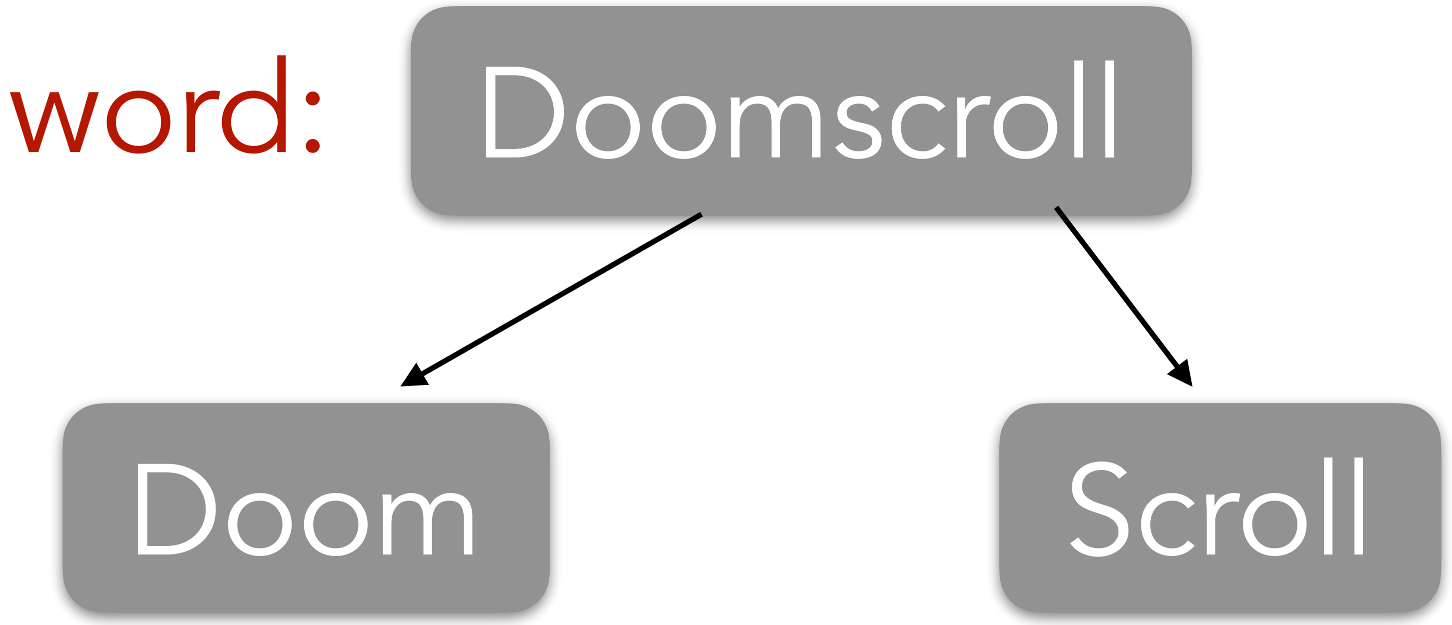
# But tokenization $\neq$ compression alone

- Want to generalize out of distribution (both in meaning and compressibility): part of idea of subword tokenization

# But tokenization ≠ compression alone

- Want to generalize out of distribution (both in meaning and compressibility): part of idea of subword tokenization

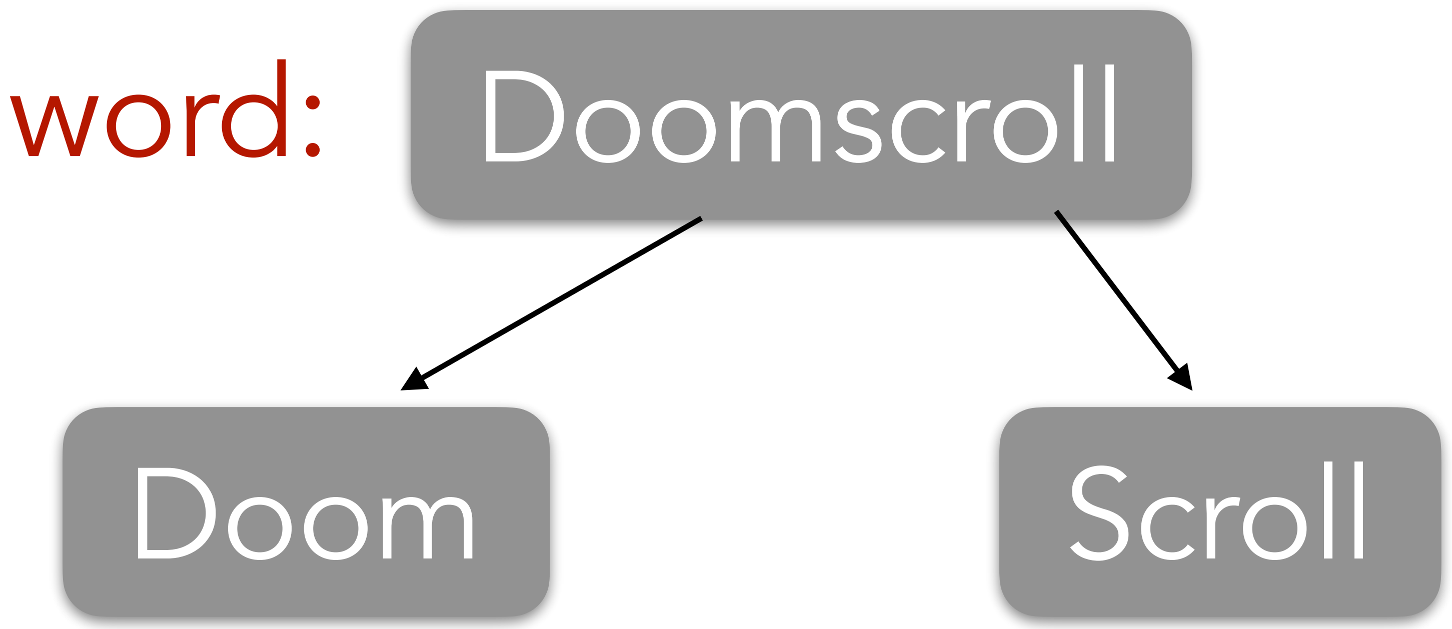
E.g. unseen word:



# But tokenization ≠ compression alone

- Want to generalize out of distribution (both in meaning and compressibility): part of idea of subword tokenization

E.g. unseen word:

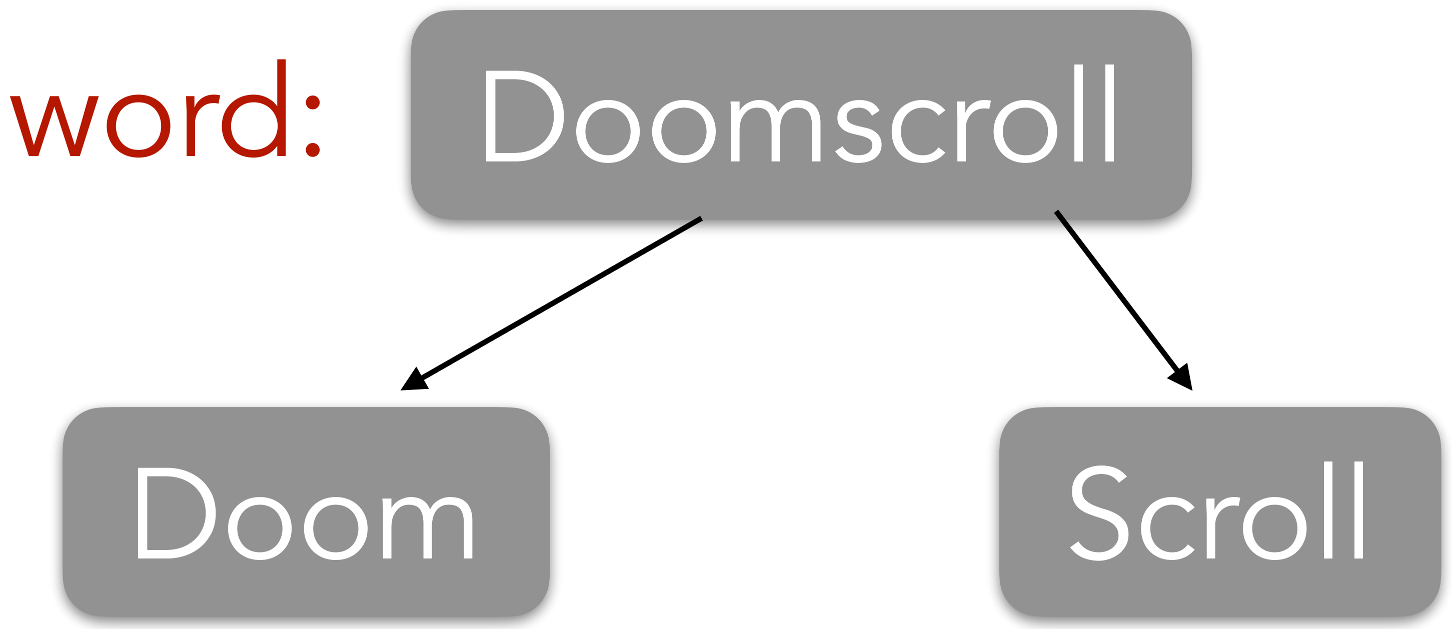



- LLMs don't train well on neurally compressed text ("Training LLMs over Neurally Compressed Text", Lester et al 2024)

# But tokenization ≠ compression alone

- Want to generalize out of distribution (both in meaning and compressibility): part of idea of subword tokenization

E.g. unseen word:

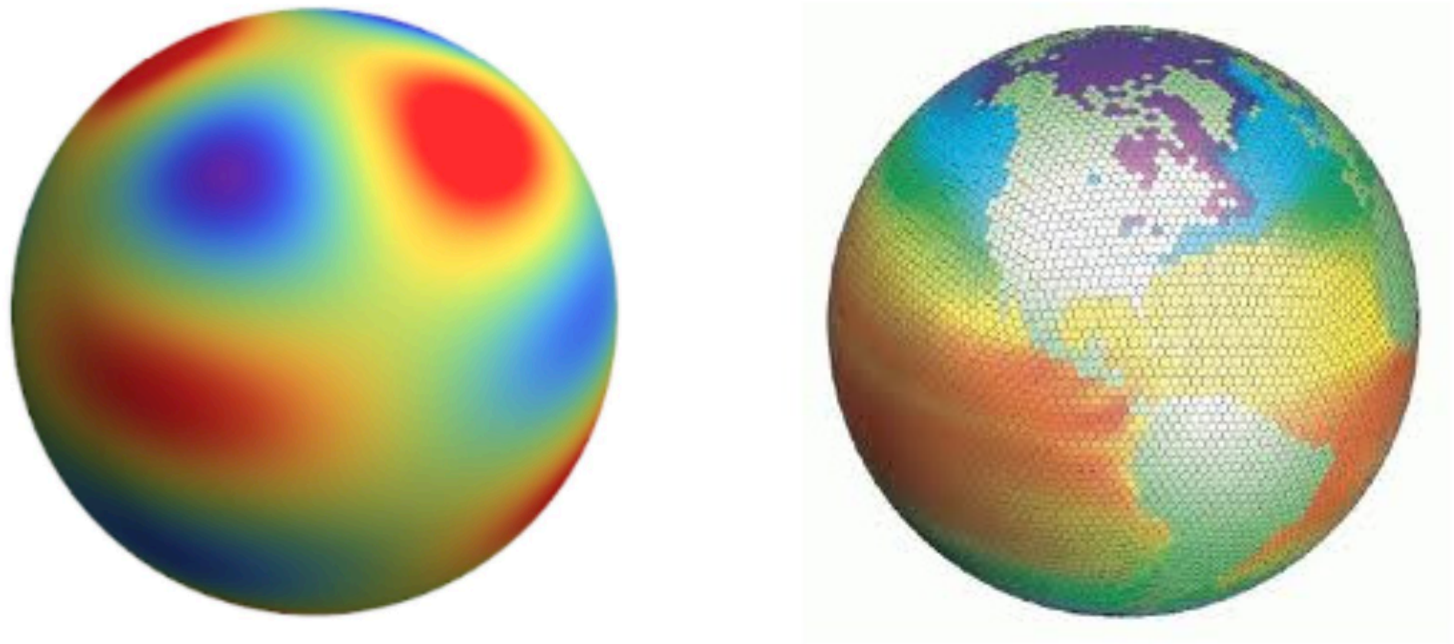



- LLMs don't train well on neurally compressed text ("Training LLMs over Neurally Compressed Text", Lester et al 2024)

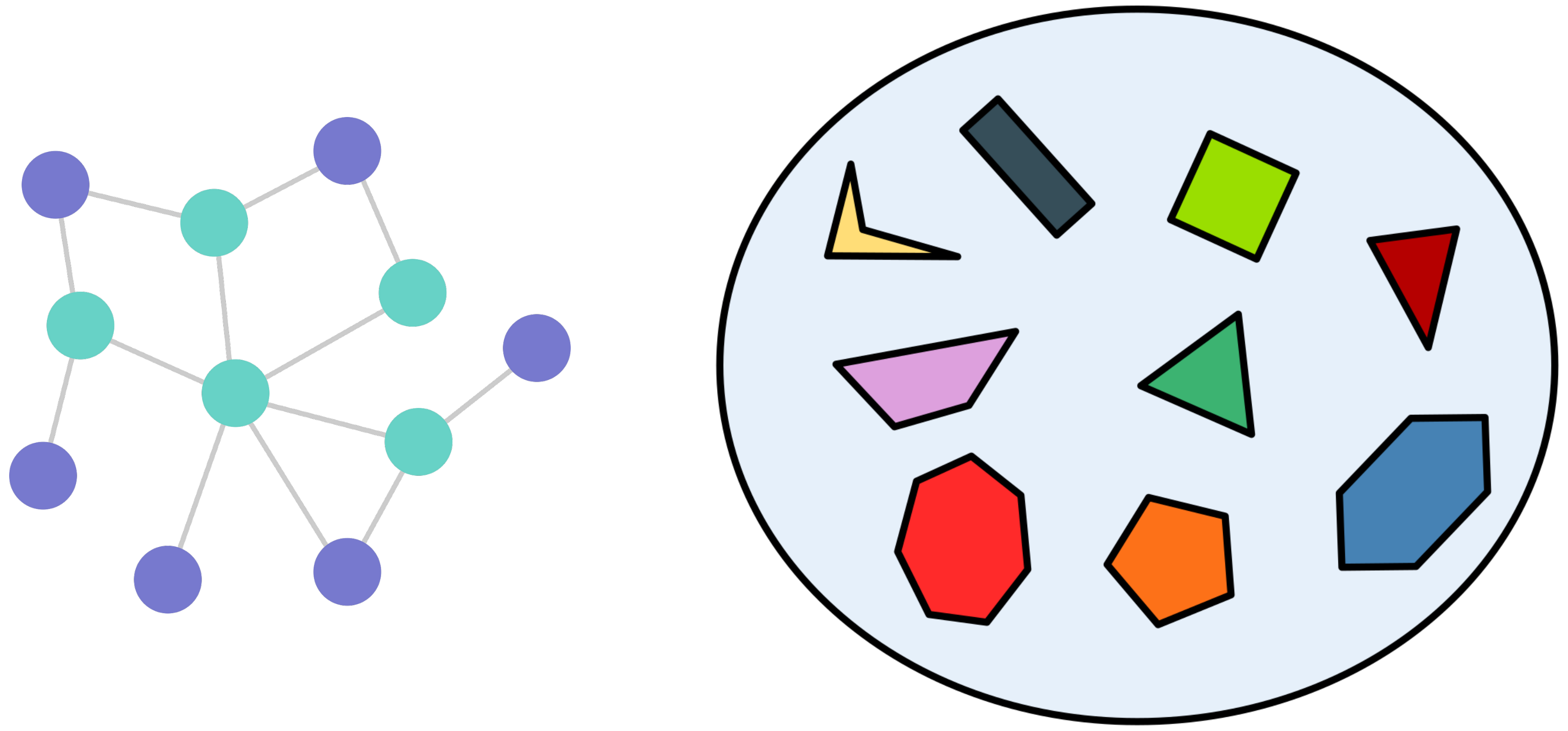
- Tokenizations that are equivalently good "compressors" train very differently (e.g. numbers, "Tokenization counts: the impact of tokenization on arithmetic in frontier LLMs", Singh & Strouse 2024)

How to apply these ideas to tokenizing other types of data?

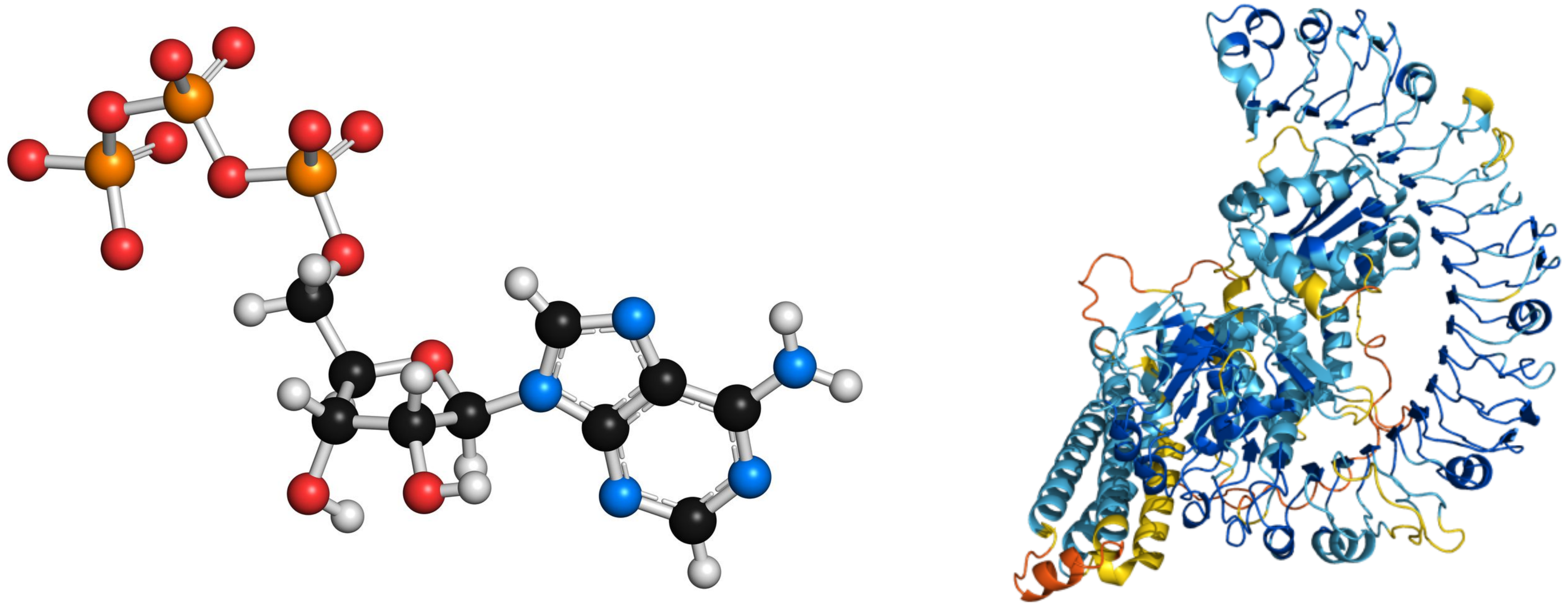
# New, geometric data types



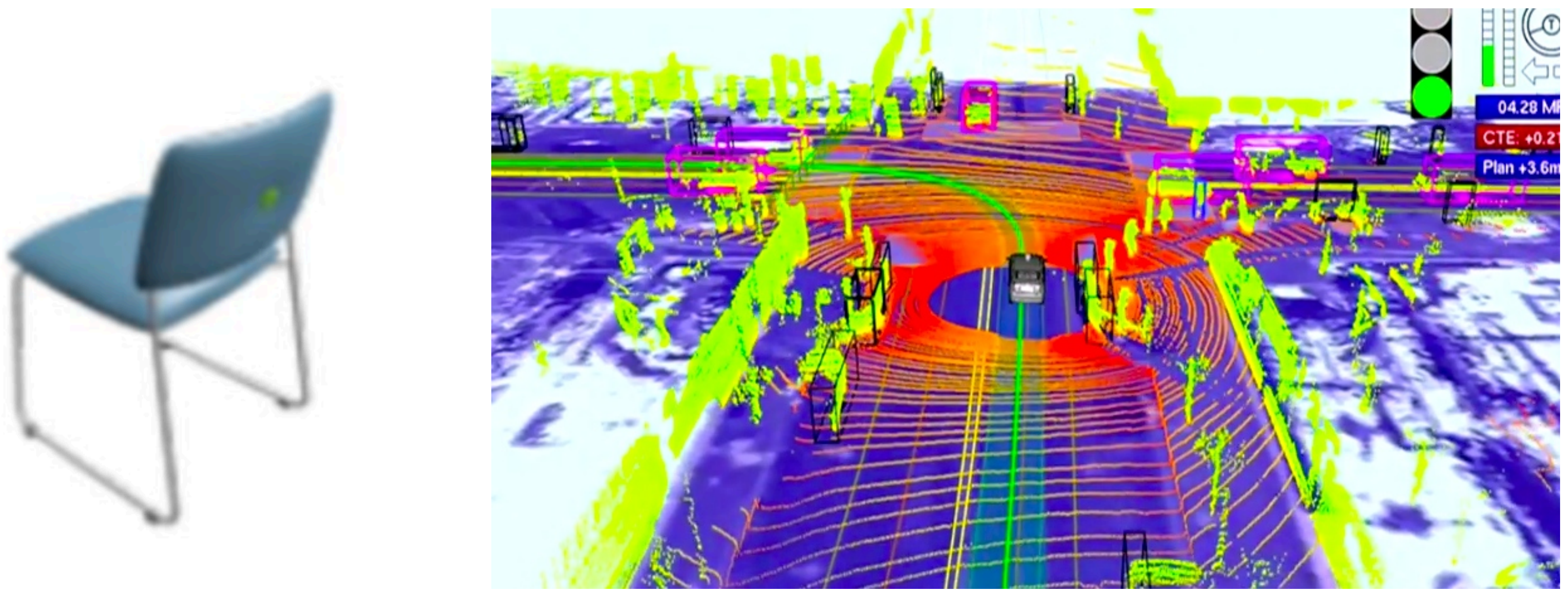
Spherical functions



Graphs and sets



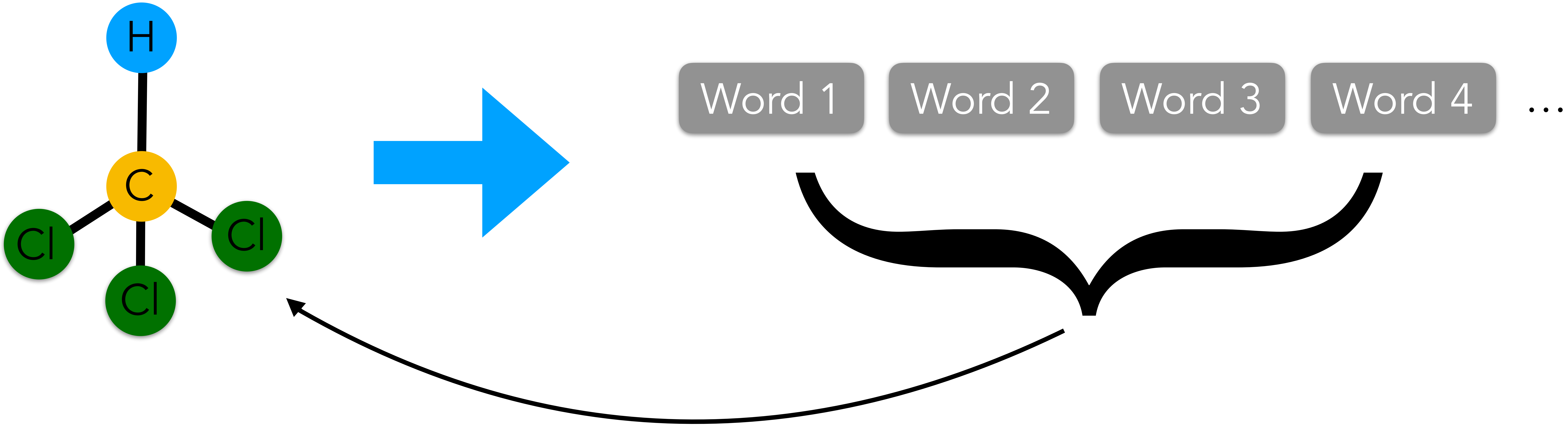
Small molecules and proteins



3D scans and objects

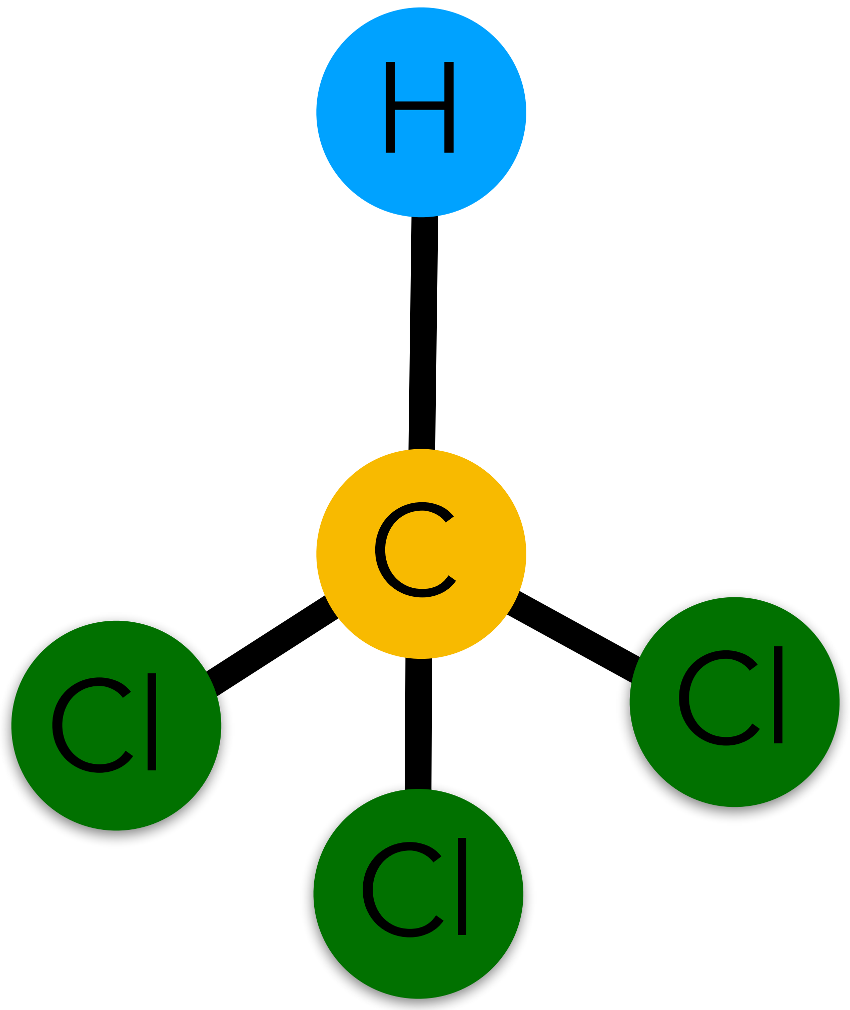
# How do we convert a molecule into “words”?

# How do we convert a molecule into “words”?




*Must be able to recover the molecule (to high accuracy) from the sequence of tokens*

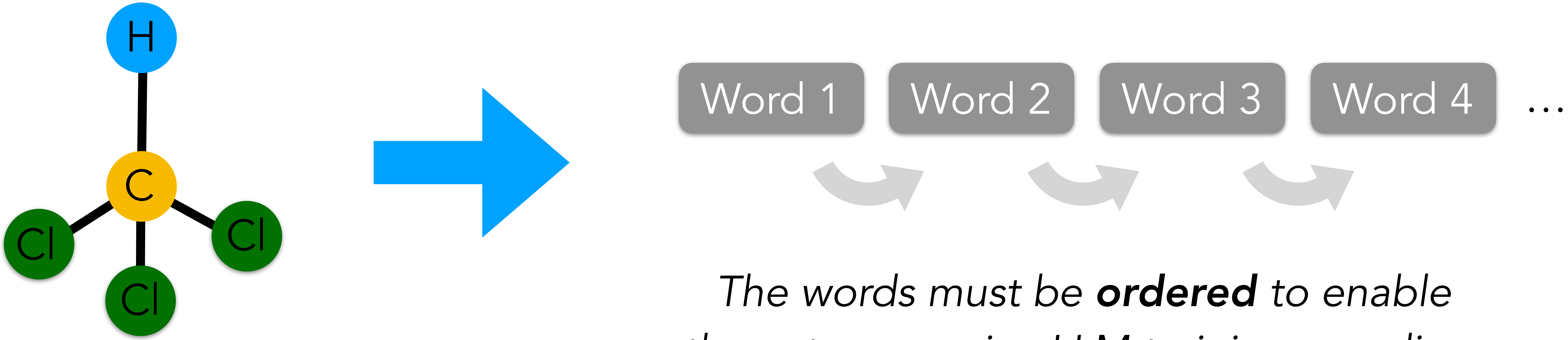
# How do we convert a molecule into “words”?




Word 1 Word 2 Word 3 Word 4 …

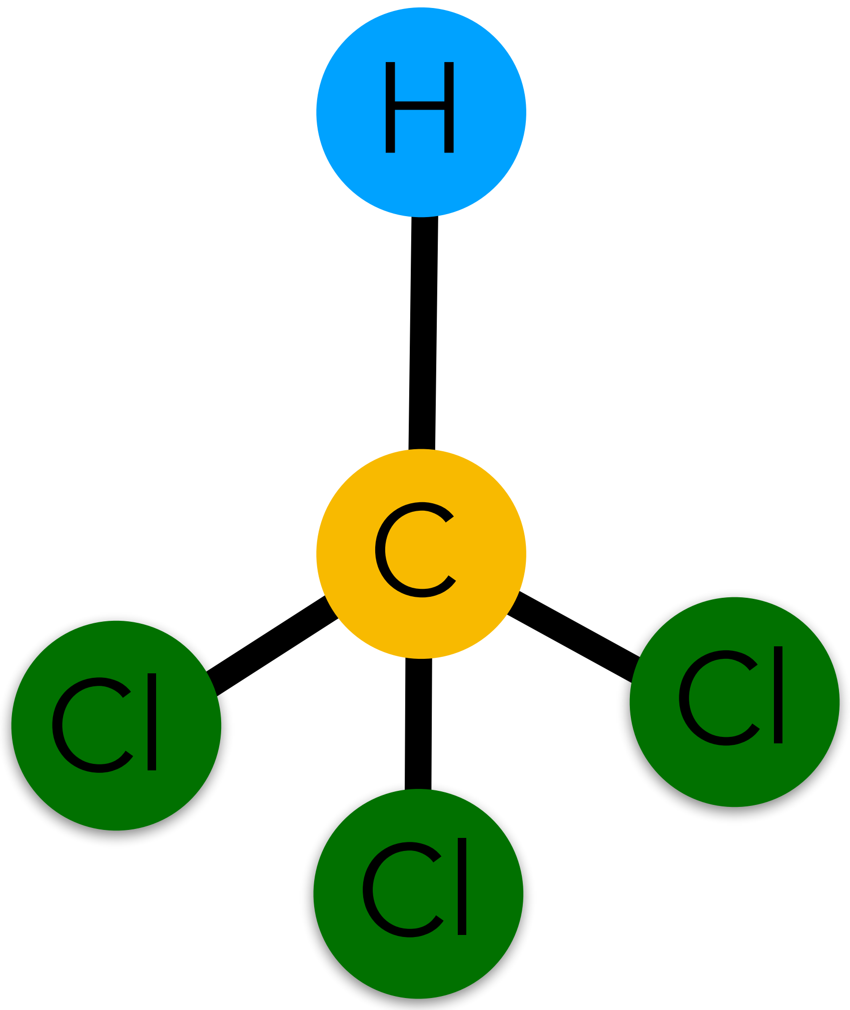
*The words must come from a **discrete** vocabulary, so that the LLM can learn a separate embedding for each*

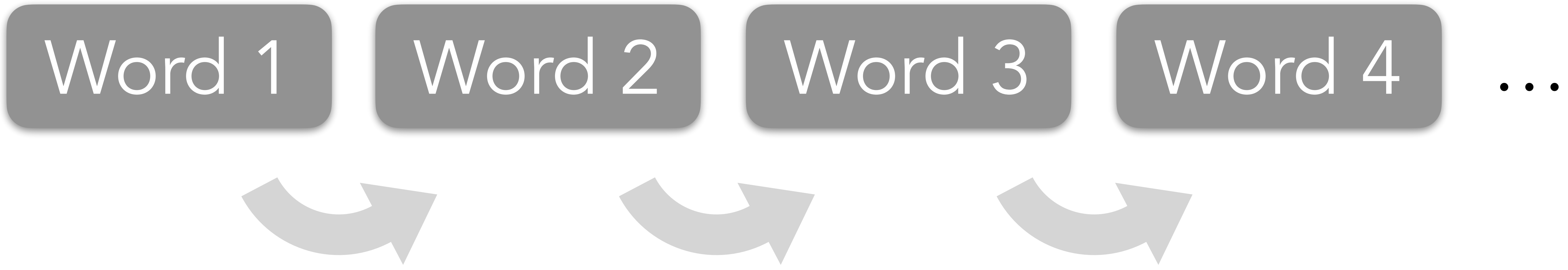
# How do we convert a molecule into "words"?




*The words must be* ***ordered*** *to enable the autoregressive LLM training paradigm (next-token prediction)*

# How do we convert a molecule into "words"?



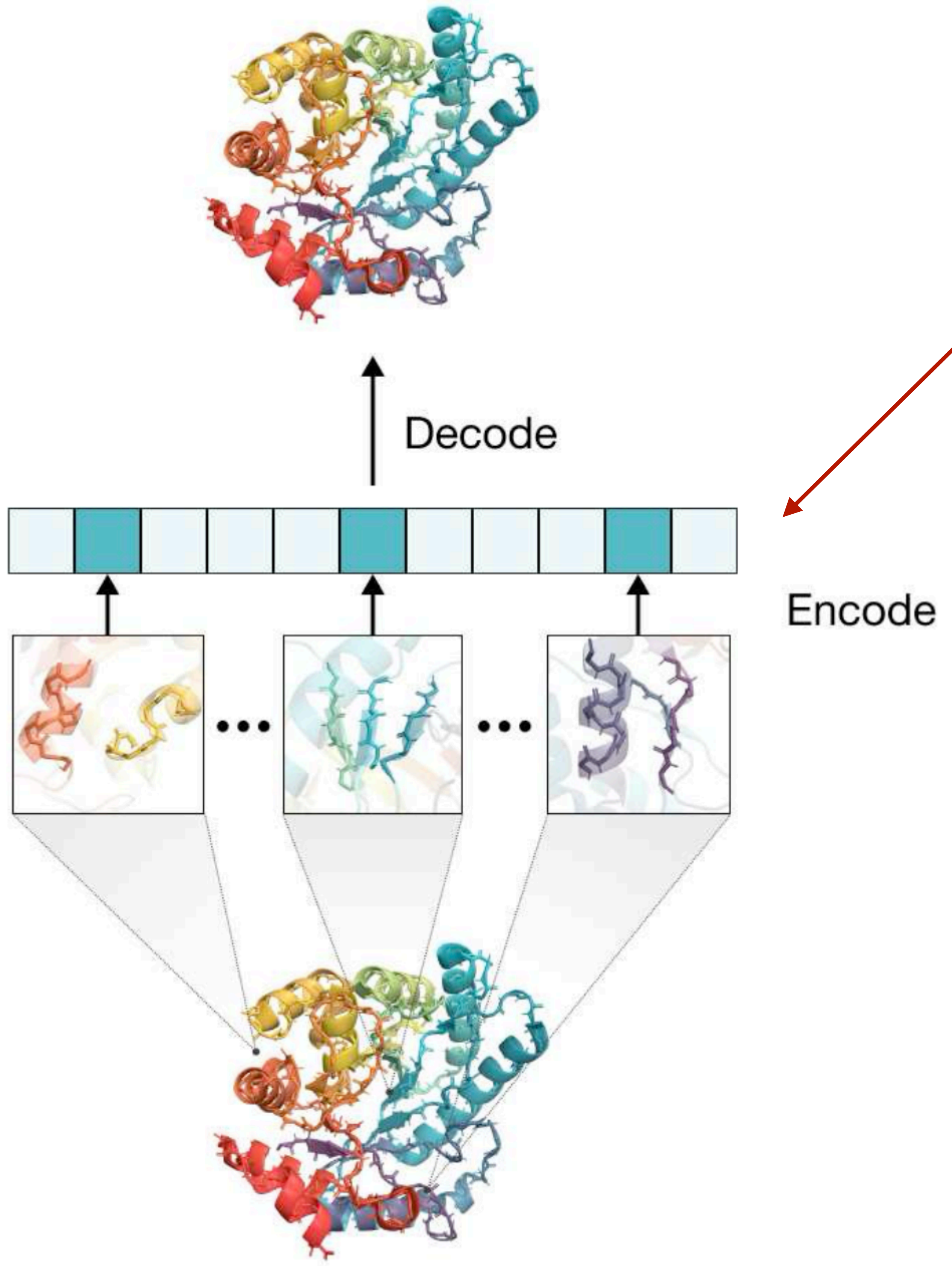




*The words must be* ***ordered*** *to enable the autoregressive LLM training paradigm (next-token prediction)*

**Permutation canonicalization returns!**

# Molecule Tokenization Uses Equivariance

ESM3 uses geometric attention to encode local structure token per residue; decode from all residues at once

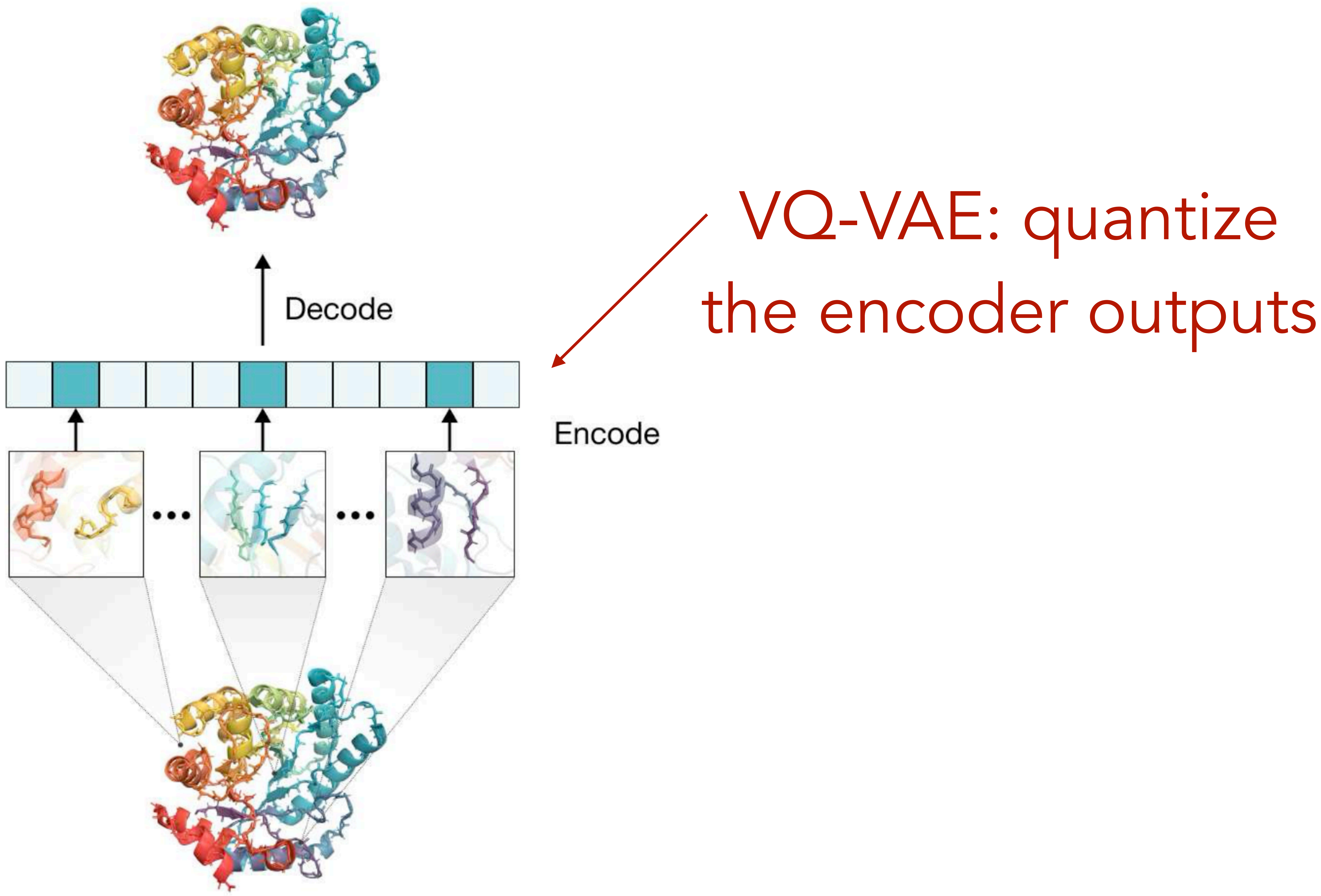



VQ-VAE: quantize the encoder outputs

# Molecule Tokenization Uses Equivariance

Encoder is rotation invariant:

- Uses ordering of amino acids to define neighborhoods
- All-to-all "geometric attention" within neighborhoods
  - Means: define local coordinate frame using backbone, then convert to global frame before performing attention



# Open questions

- How to tokenize molecules — both proteins and others?
  - Want: generalizability outside training data, efficiency, learnability
  - Inductive bias - invariance to (local) rotation, permutation (relevant for non-proteins)?
-  Byte Latent Transformer is recent, tokenization-free method: is it effective for multi-modal and/or non-text data too?
  - Note: still requires canonicalization!

# Concluding thoughts

- Inductive bias isn't dead! But: emphasis on flexibility + incorporation into highly scalable methods
-  Even if the specific methods (tokenization, positional encodings, canonicalization) are eventually replaced by learnable substitutes: guides the search space
- Other directions not discussed: learnable symmetries, soft loss objectives

Thanks! Questions?