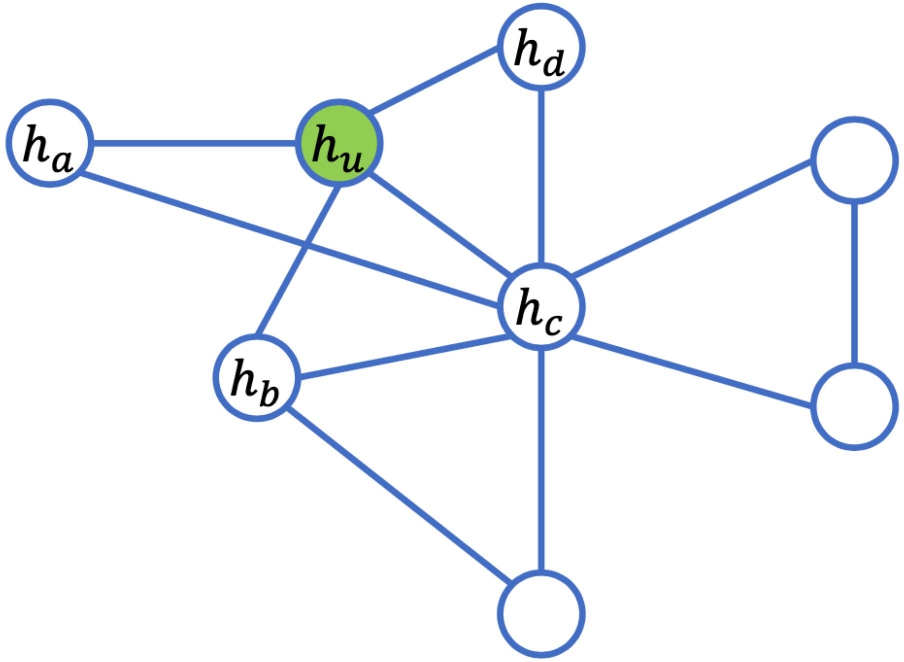
Autobahn:

Automorphism-based Graph Neural Nets

MPNNs are not expressive enough



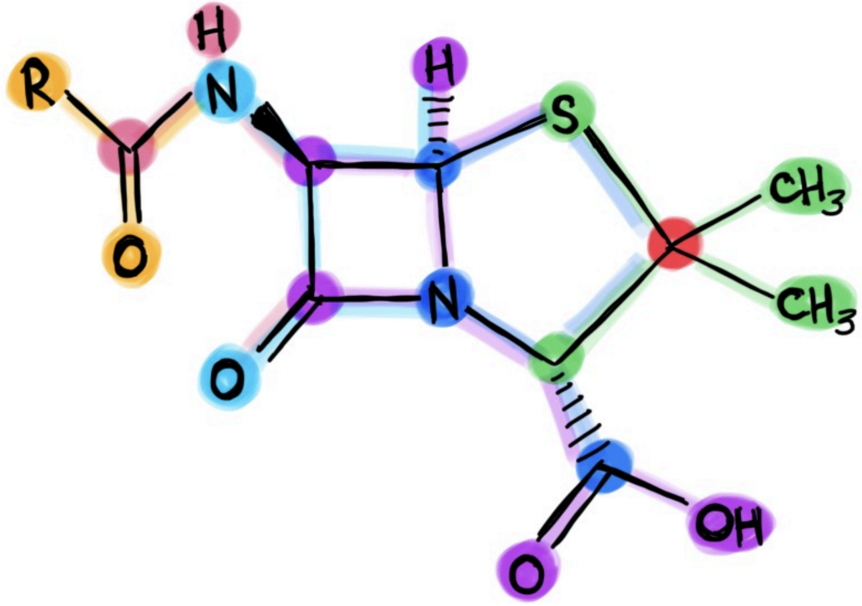
$h_u^{(k)} = embedding\ of\ u\ at\ k^{th} - iteration$

$$m_{N(u)}^{(k)} = AGGREGATE^{(k)}\left(\left\{h_v^{(k)}, \forall\, v \in N(u)\right\}\right)$$

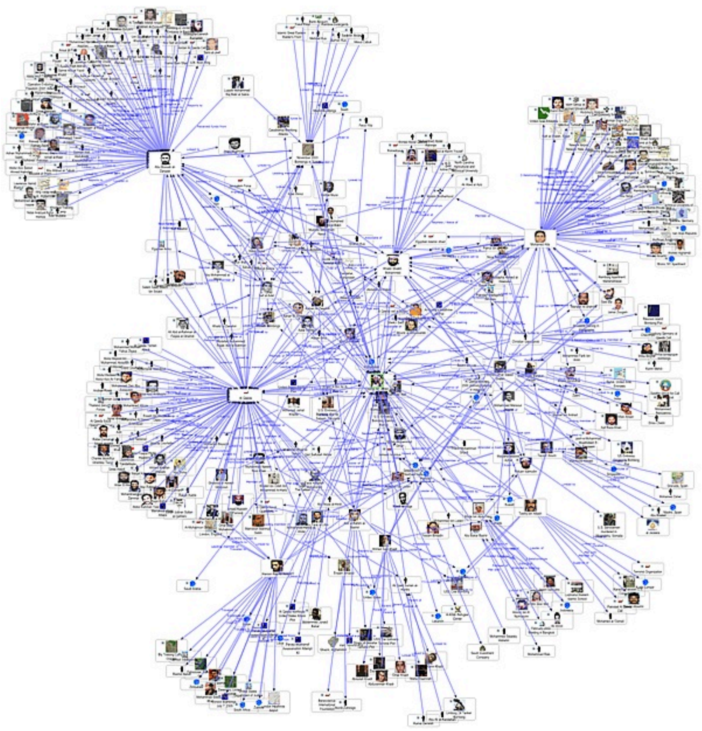
$$h_u^{(k+1)} \leftarrow UPDATE^{(k)}\left(h_u^{(k)}, m_{N(u)}^{(k)}\right)$$

$$f_i^{(l+1)} = \nu\left(w_{self} f_i^l + w_{neigh} \sum_{v \in N(i)} f_v^l\right)$$

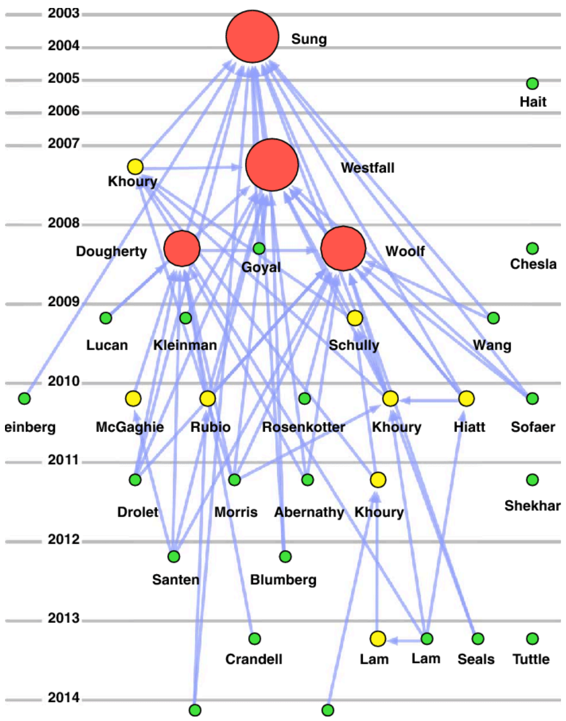
Different Problem – Same Architecture



Molecular Graph



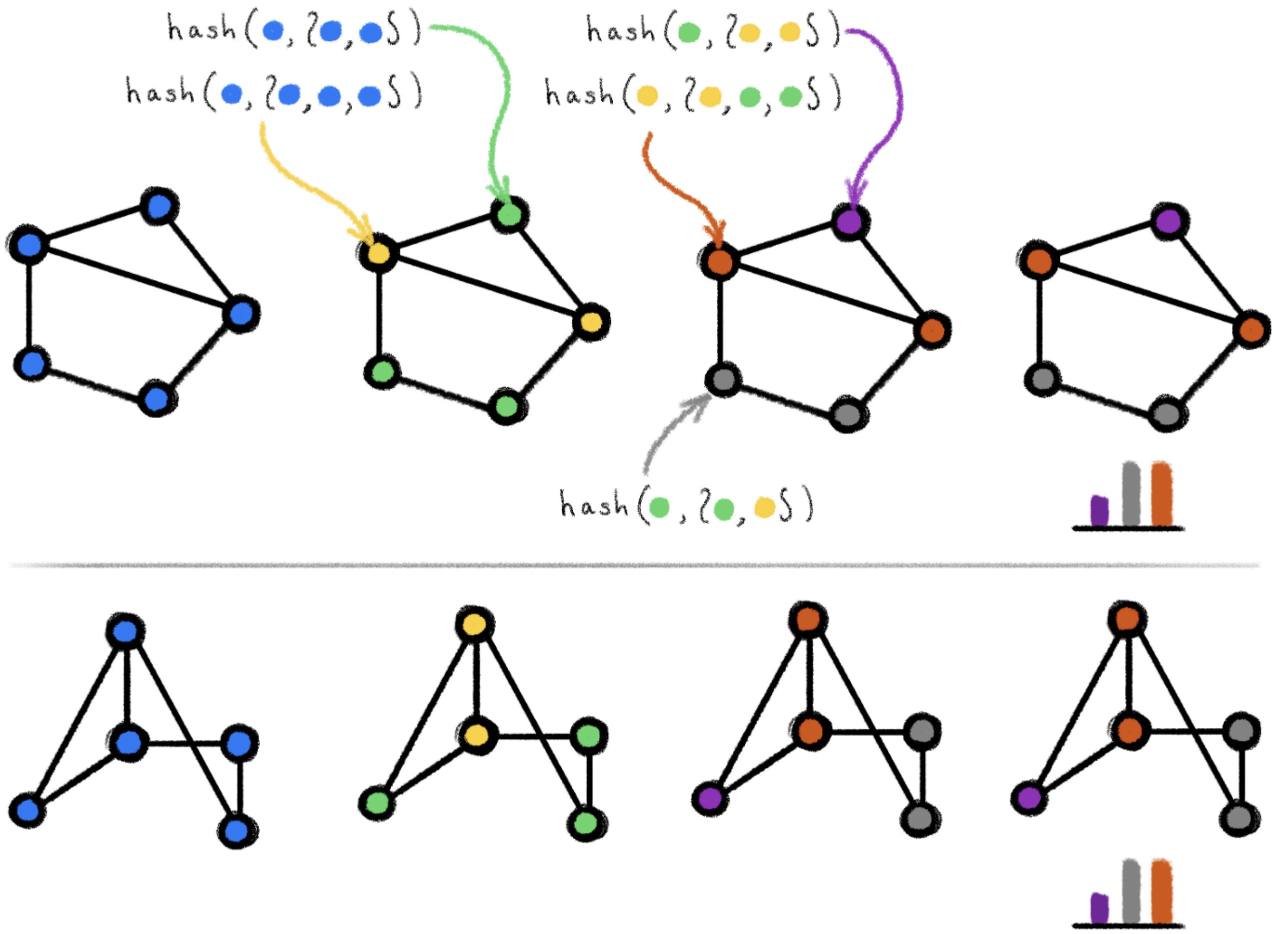
Social Network Graph



Citation Graph

How can GNN expressivity be ranked?

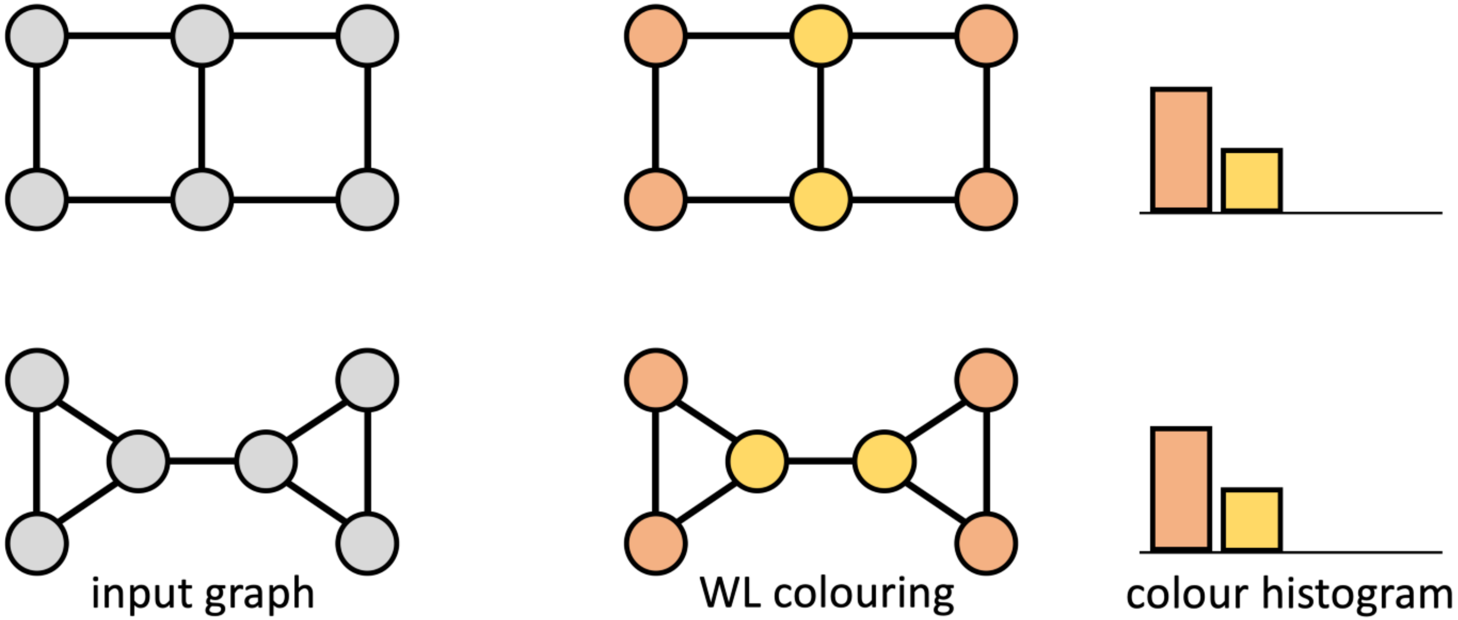
Weisfeiler-Lehman Test



How can GNN expressivity be ranked?

Weisfeiler-Lehman Test

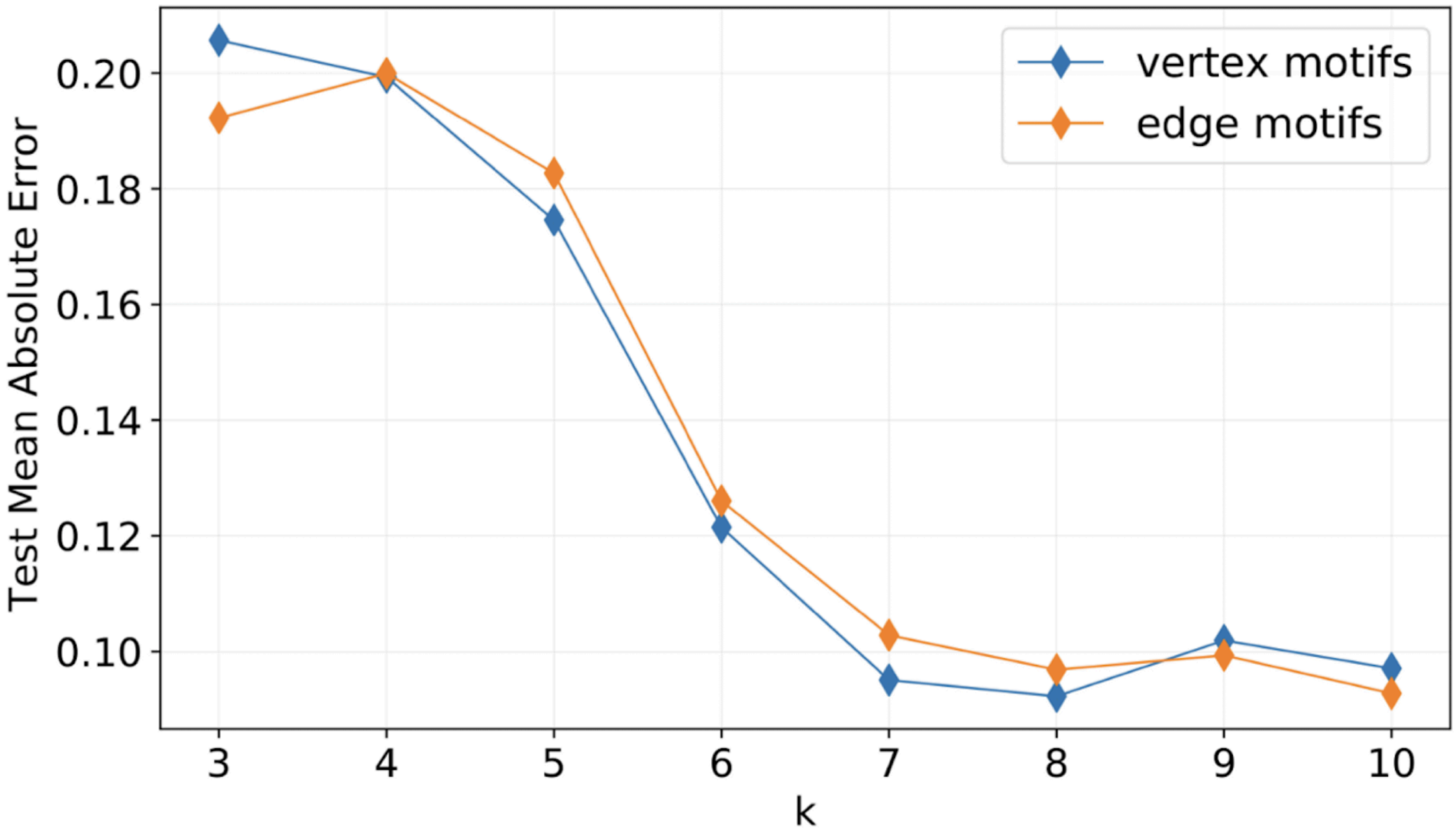
1-WL Test cannot detect cycles:



Weisfeiler-Lehman Hierarchy

- k-WL is a higher-order extension of WL test
  - Determine color for each k-tuple instead of individual node
  - Can count substructures up to size k
  - $\mathcal{O}(n^k)$
- (k+1)-WL strictly stronger than k-WL
- MPNNs ≤ 1-WL expressive
- k-GNN extension possible, but k>3 computationally infeasible

Performance on molecular graphs



Goals of Autobahn's Architecture

1. Computationally feasible
2. Able to incorporate Domain Knowledge
3. Invariant to Input Graph Permutations
   "Same Input => Same Output"

$\mathfrak{G}$ - invariance

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

$$\rho(x,\ y) = (x + t_1,\ y + t_2)$$

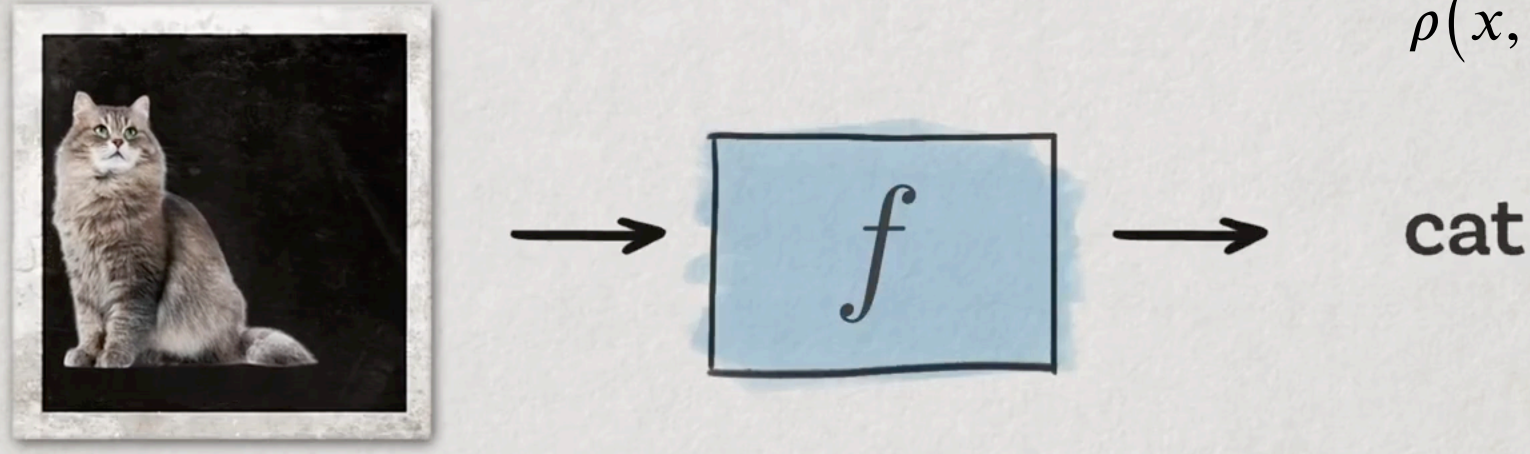


image classification

$\mathfrak{G}$ - equivariance

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

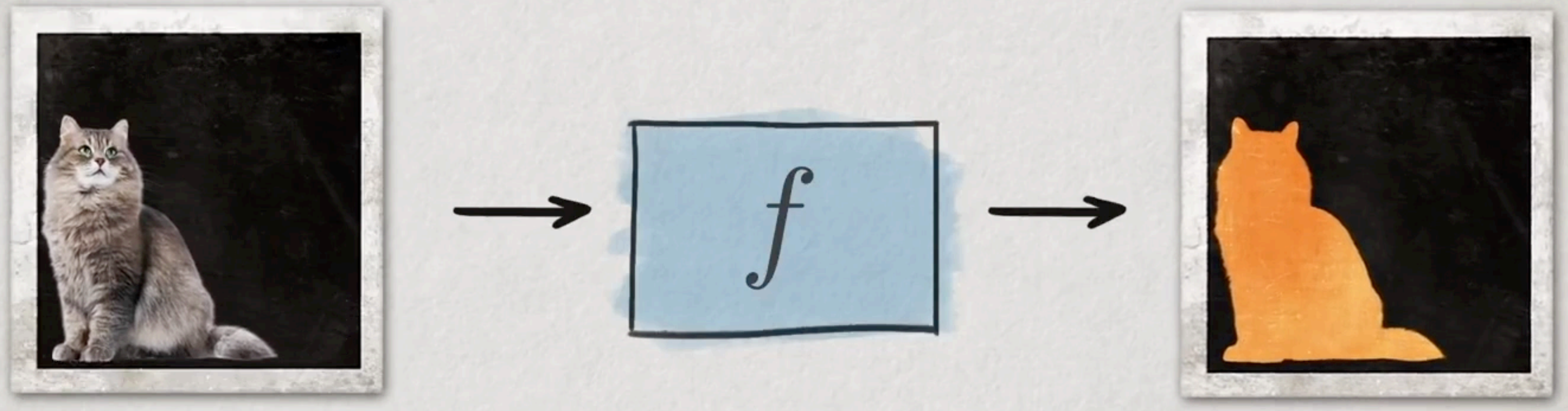
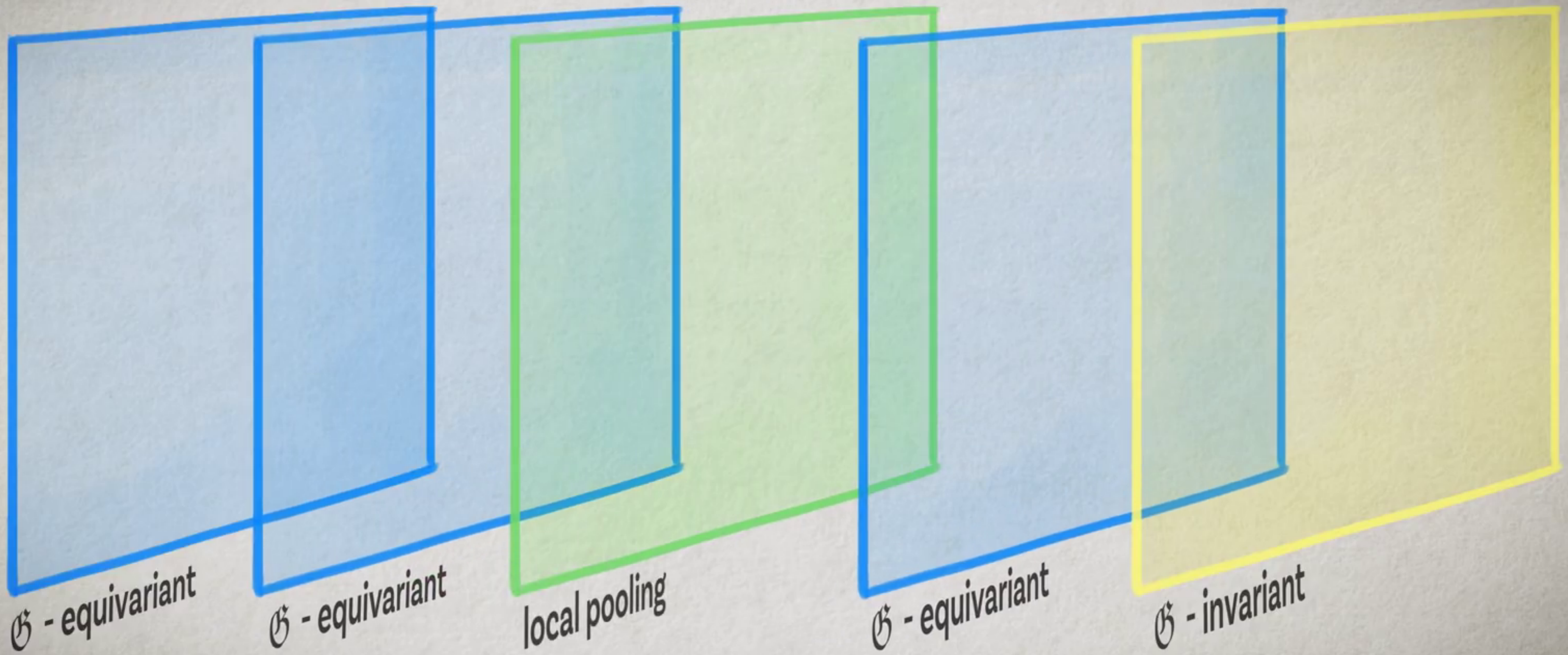
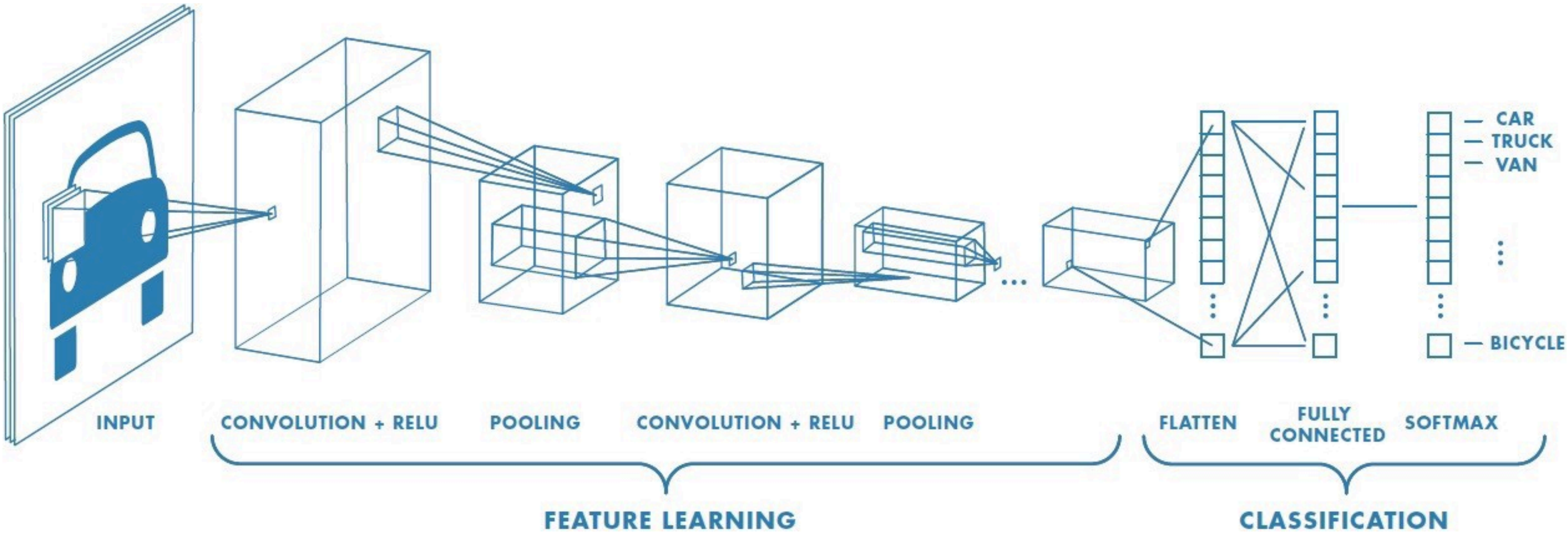


image segmentation

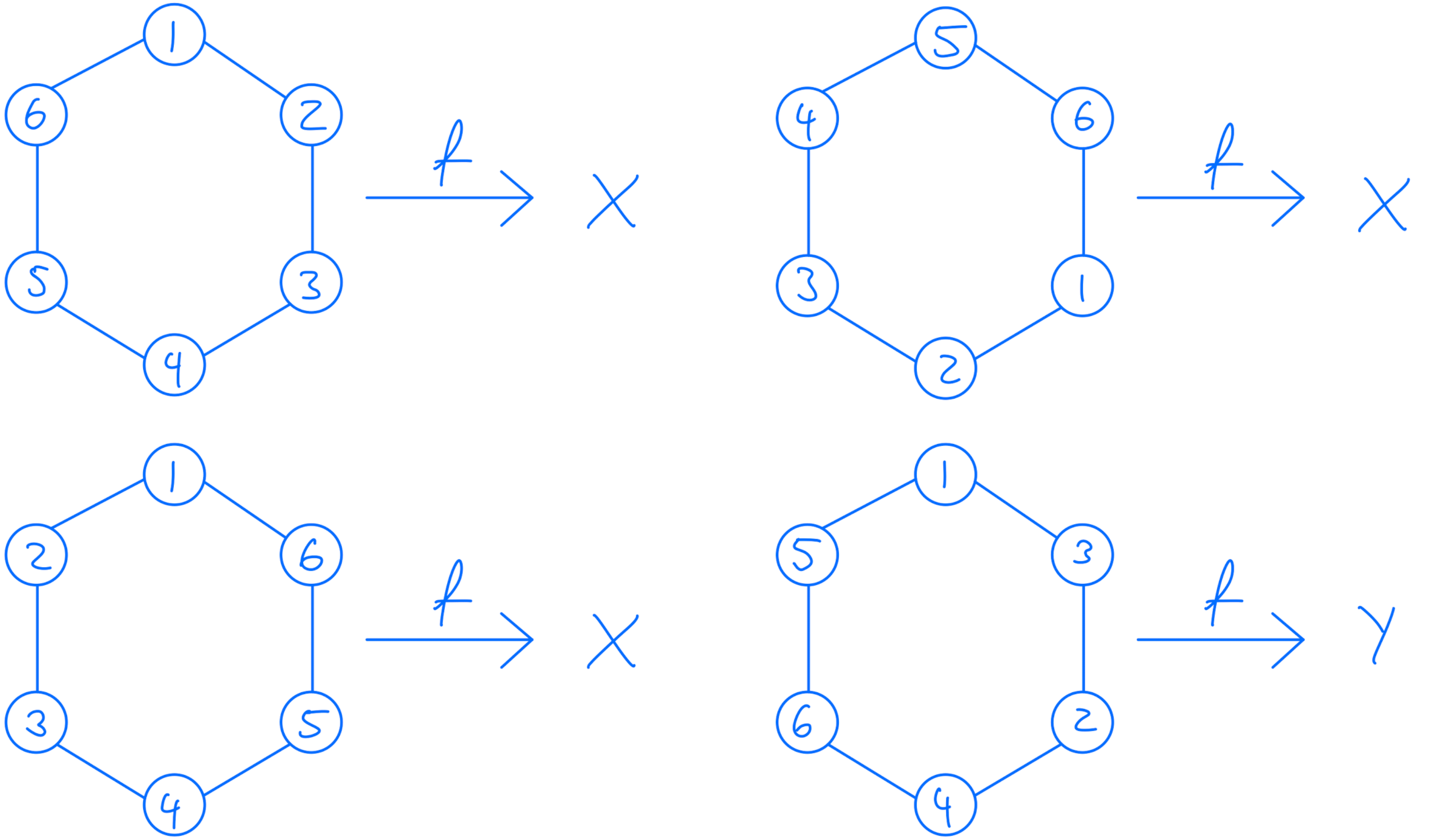
Layer Hierarchy



Layer Hierarchy – CNN Example

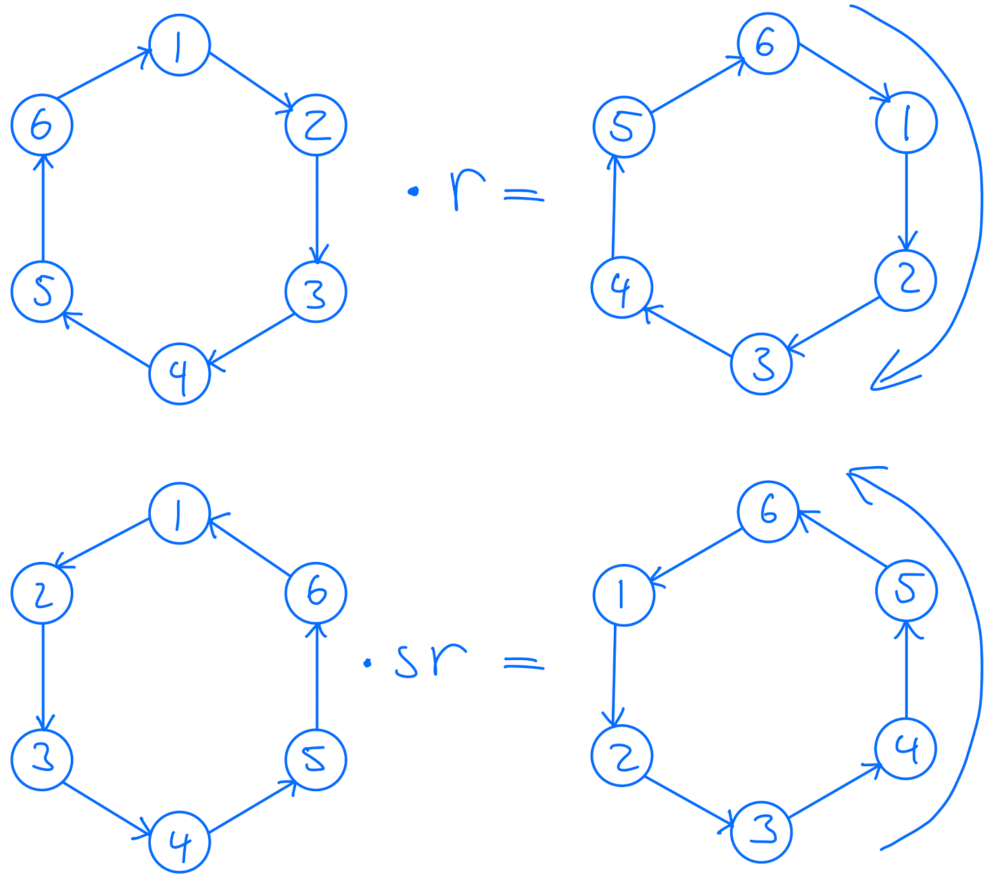


Permutation Invariance Examples

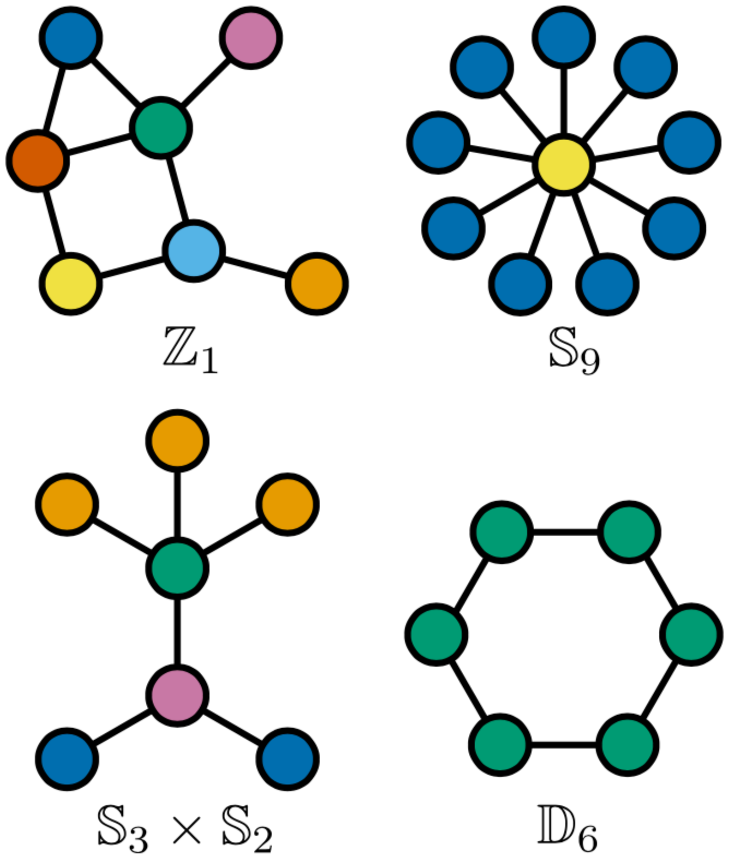


Graph Automorphism Group

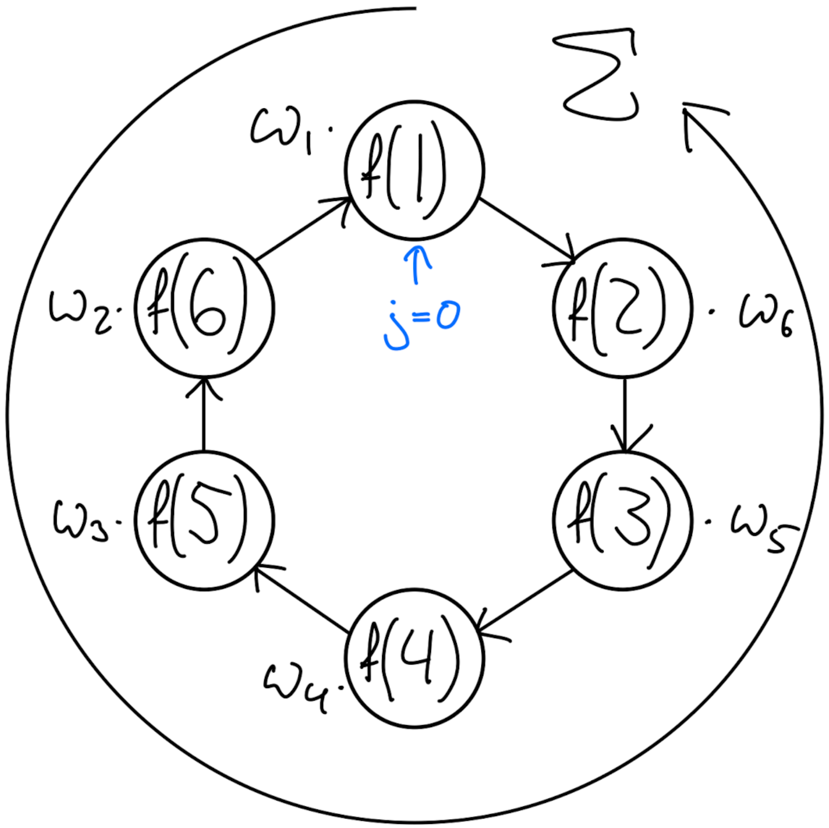
- Definition of Algebraic Group:
  - Set + Binary Operation
  - Closure
  - Associativity
  - Identity Element
  - Inverse
- Def. Automorphism Group: Group of Permutations, which preserve edge-vertex connectivity



Graph Automorphism Group



Convolutions on Automorphism Groups



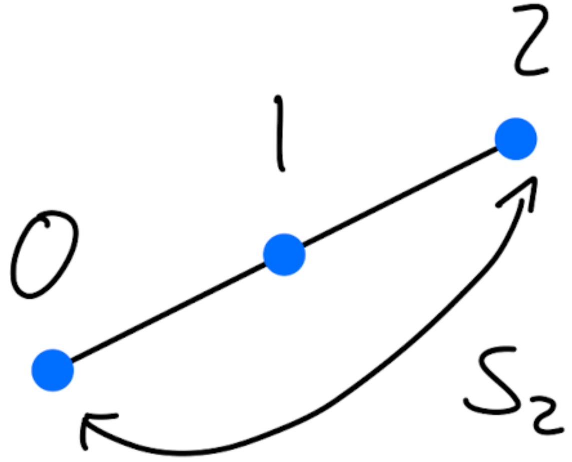
$$w = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}$$

$$(f * w)_j = \sum_{k=0}^{n} f(r^j r^{-k}) \cdot w(r^k)$$

$$(f * w)_{j=0} = f(r^0 r^{-0}) \cdot w(r^0) + f(r^{-1}) \cdot w(r^1) + \ldots + f(r^{-5}) \cdot w(r^5)$$

$$= w_1 f(1) + w_2 f(6) + w_3 f(5) + \ldots + w_6 f(6)$$

Derivate MPNN from Star Graph



$$\begin{aligned}(f * w)_i &= w_0 f(0) + w_1 f(1) + w_2 f(2) \\ &+ w_0 f(2) + w_1 f(1) + w_2 f(0) \\ &= \underbrace{2w_1}_{\alpha_1} f(1) + \underbrace{(w_0 + w_2)}_{\alpha_2}(f(0) + f(2)) \\ &= \alpha_1 f(1) + \alpha_2 \sum_{j=1}^{n} f(j)\end{aligned}$$

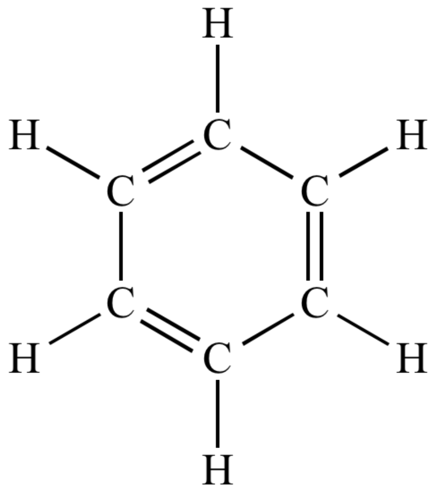
Benefits of Automorphisms

- Enables new subgraph structures
- Computation over Automorphism Group preserves more information:
    - Aggregating over a cycle preserves more structure
        - => more information
        - => faster computation: $|\text{Aut}(S_6)| = 120, |\text{Aut}(C_6)| = 12$
- Most expressive Aggregation step: Convolution over Aut(G)

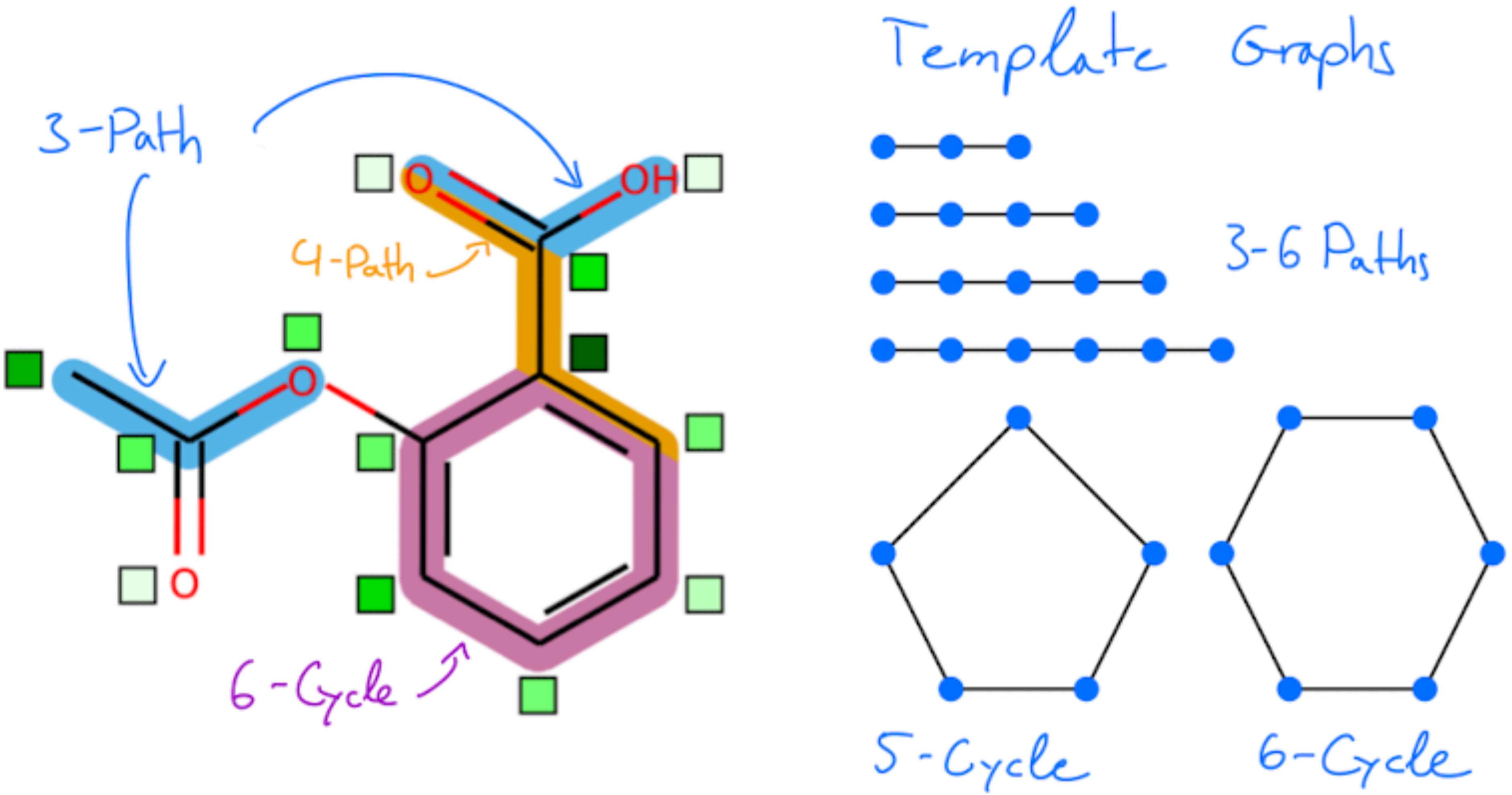
Foundational Realizations

- Convolutions work well in the image domain
- Automorphisms of the substructures form the basis for calculation

- Example: Benzene Ring – natural notion of convolution

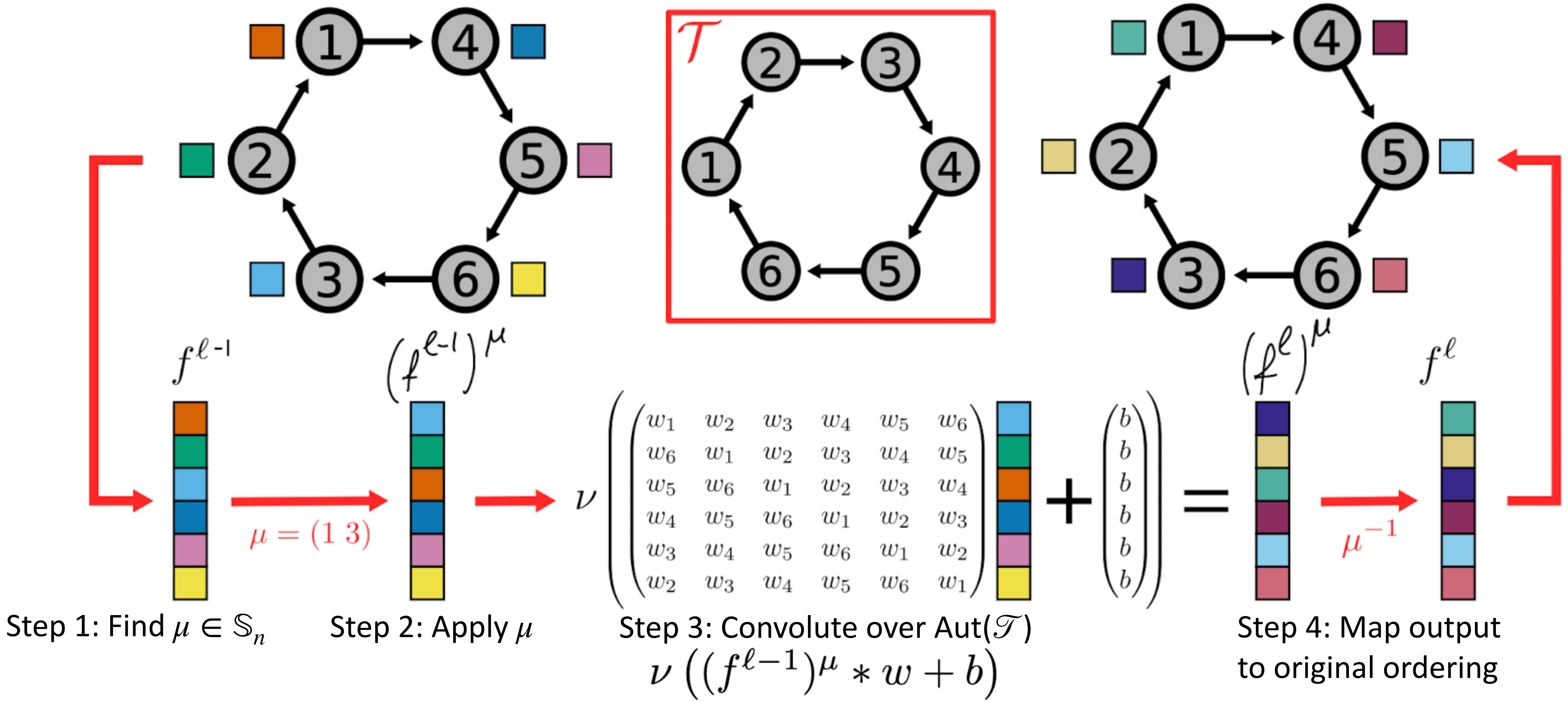


Overview of Architecture



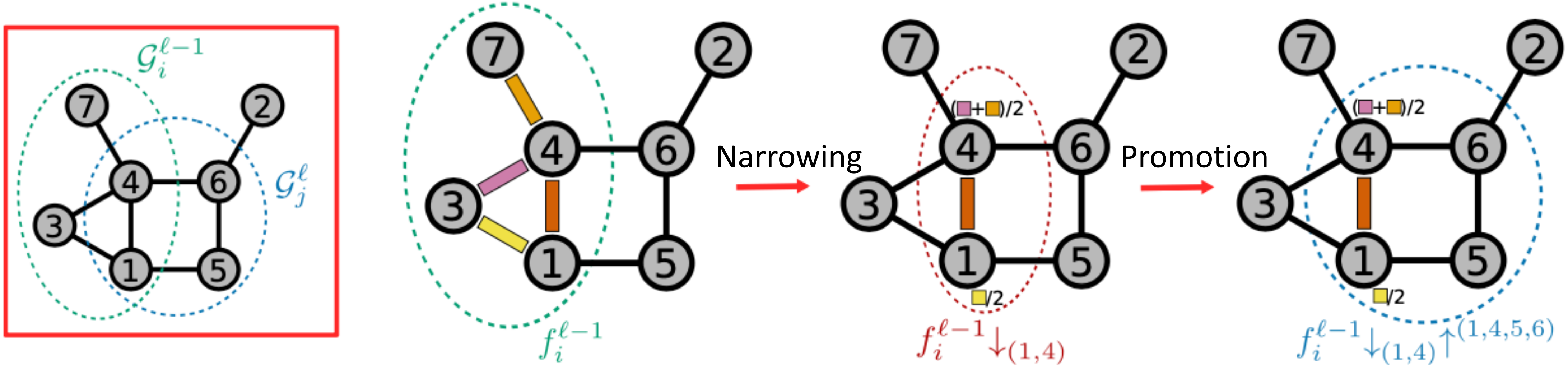
Computing on Substructures

Automorphism-based Neurons



Narrowing & Promotion

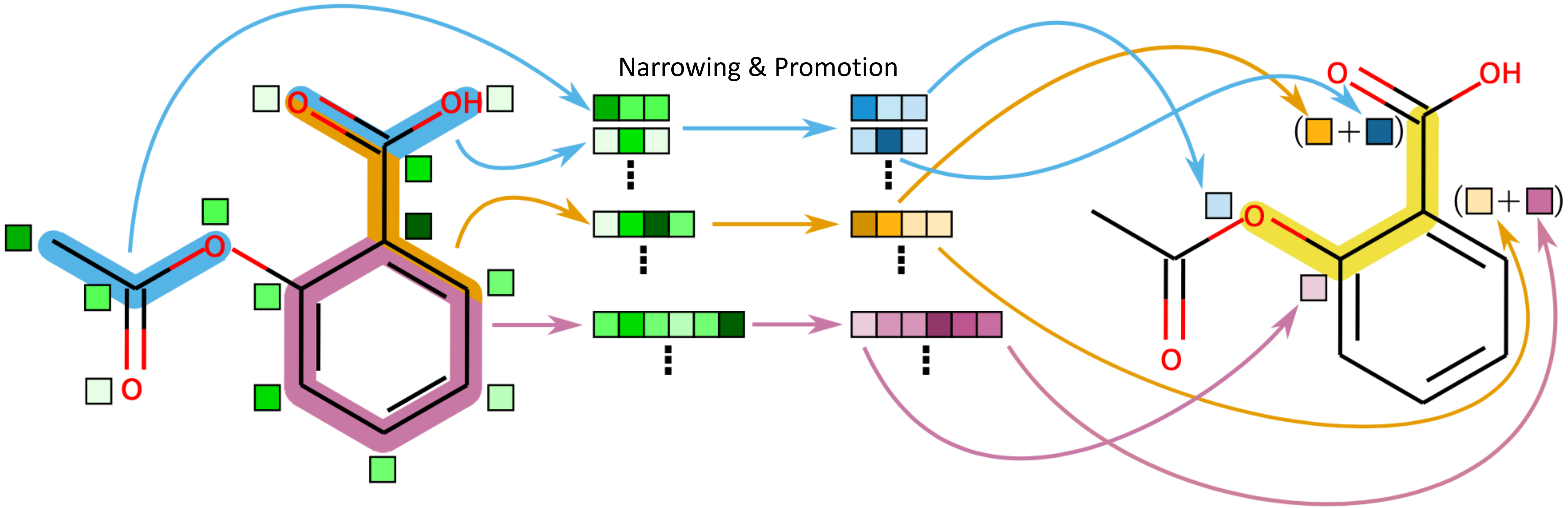
Transfer information between subgraphs



- Narrowing: Project each incoming activation to the corresponding intersection
- Promotion: Extend the activation to not involved nodes

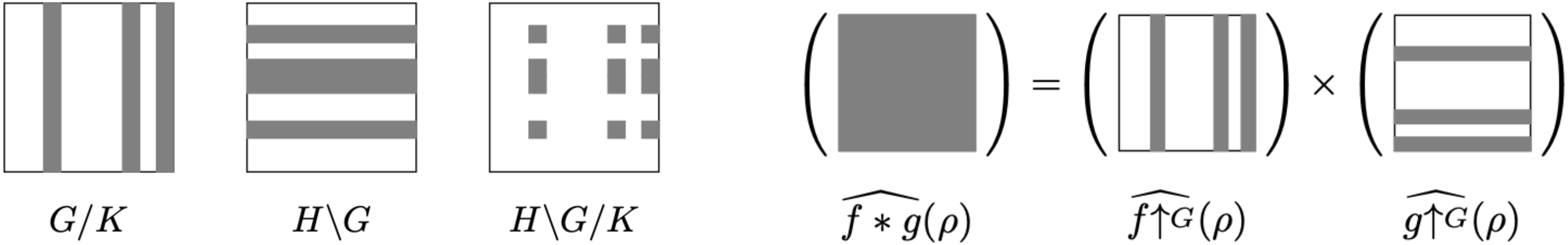
Narrowing & Promotion

Transfer information between subgraphs



Some Implementation Details

- Efficiently find graph substructures? => yes!
- How is convolution efficiently computed?

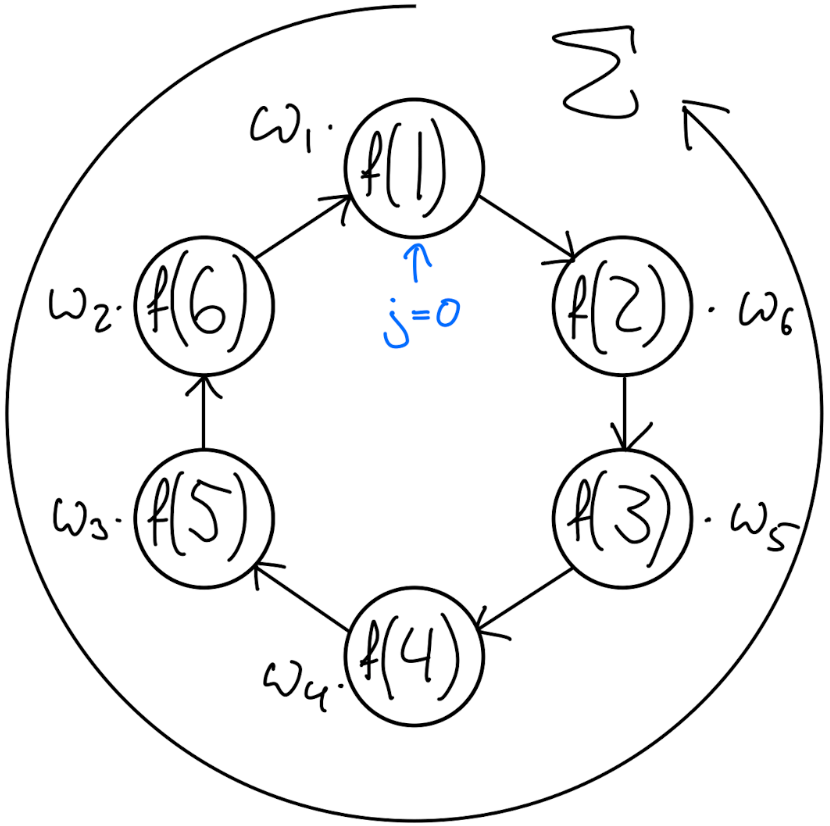


- https://github.com/risilab/Autobahn

Representing Convolution as Matrix Mult.

$$\nu\left(\begin{pmatrix} w_1 & w_2 & w_3 & w_4 & w_5 & w_6 \\ w_6 & w_1 & w_2 & w_3 & w_4 & w_5 \\ w_5 & w_6 & w_1 & w_2 & w_3 & w_4 \\ w_4 & w_5 & w_6 & w_1 & w_2 & w_3 \\ w_3 & w_4 & w_5 & w_6 & w_1 & w_2 \\ w_2 & w_3 & w_4 & w_5 & w_6 & w_1 \end{pmatrix}\right.$$

$$\left. + \begin{pmatrix} b \\ b \\ b \\ b \\ b \\ b \end{pmatrix}\right)$$



Outlook

- Published 3rd Feb 2022
- “Only” on-par performance compare to state of the art
- Many Experiments left:
  - Different Problem Domains
  - Different Substructures
  - Graph Coarsening
  - Different Activations