Provably Powerful Graph Networks

Authors: Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, Yaron Lipman
Presenters: Lucas Tecot, Difan Zou, Weitong Zhang

Main Idea

We want to create networks that are as expressive as k-WL tests.

$$F = h \circ m \circ L_d \circ \sigma \circ \cdots \circ \sigma \circ L_1$$

Permutation Group

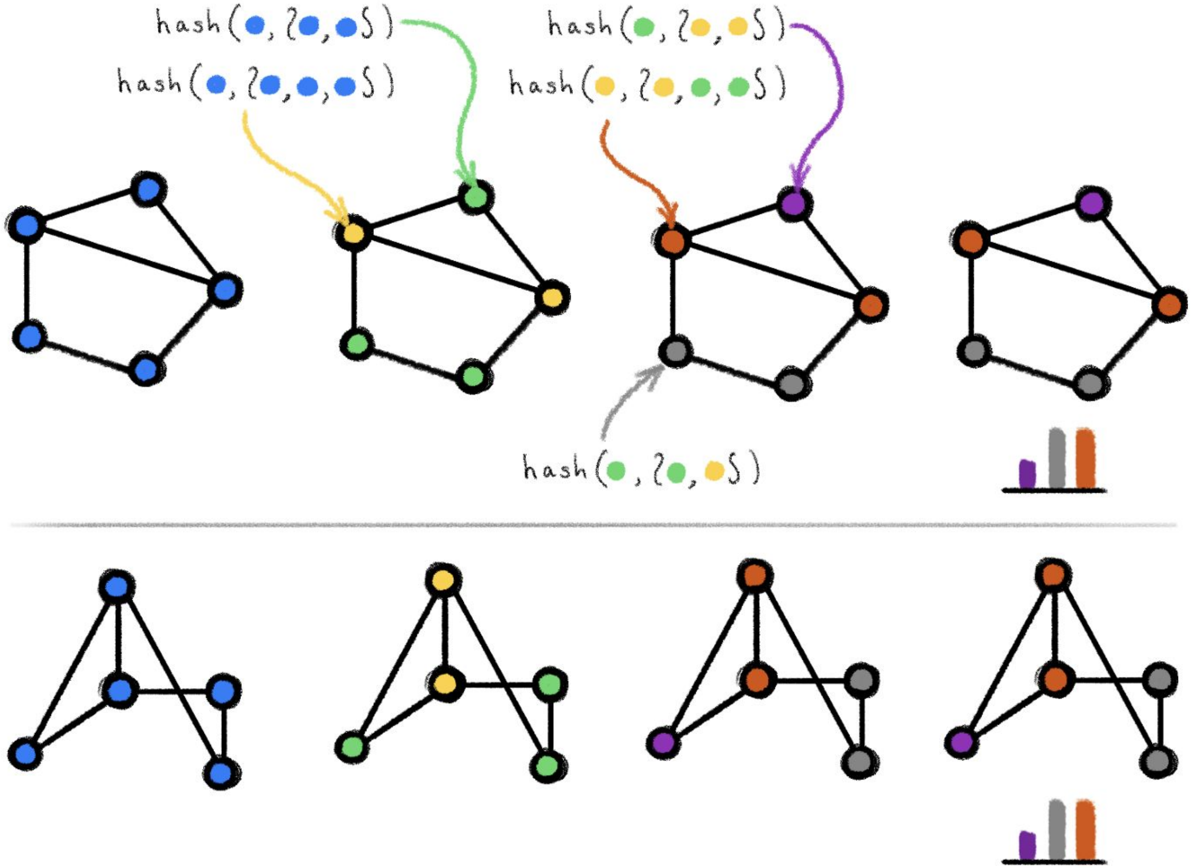
Equivariance $\longrightarrow$ $$L_i(g \cdot \mathbf{X}) = g \cdot L_i(\mathbf{X}), \qquad \forall g \in S_n$$

Invariance $\longrightarrow$ $$h(g \cdot \mathbf{X}) = h(\mathbf{X}), \qquad \forall g \in S_n$$

$$F(g \cdot \mathbf{X}) = m(\cdots(L_1(g \cdot \mathbf{X}))\cdots) = m(\cdots(g \cdot L_1(\mathbf{X}))\cdots) = \cdots = m(h(g \cdot L_d(\cdots))) = F(\mathbf{X}).$$

If we construct our network with equivariant / invariant layers that follow the k-WL test, then we can distinguish graphs that are different according to k-WL while producing the same output for all isomorphic graphs.

The Weisfeiler-Lehman Graph Isomorphism Test



k-WL and k-FWL

Instead of looking at individual vertices, let's look at **k-tuples of vertices.**

$$N_j(\boldsymbol{i}) = \left\{ (i_1, \ldots, i_{j-1}, i', i_{j+1}, \ldots, i_k) \;\middle|\; i' \in [n] \right\}$$

$$N_j^F(\boldsymbol{i}) = \Big((j, i_2, \ldots, i_k), (i_1, j, \ldots, i_k), \ldots, (i_1, \ldots, i_{k-1}, j) \Big)$$

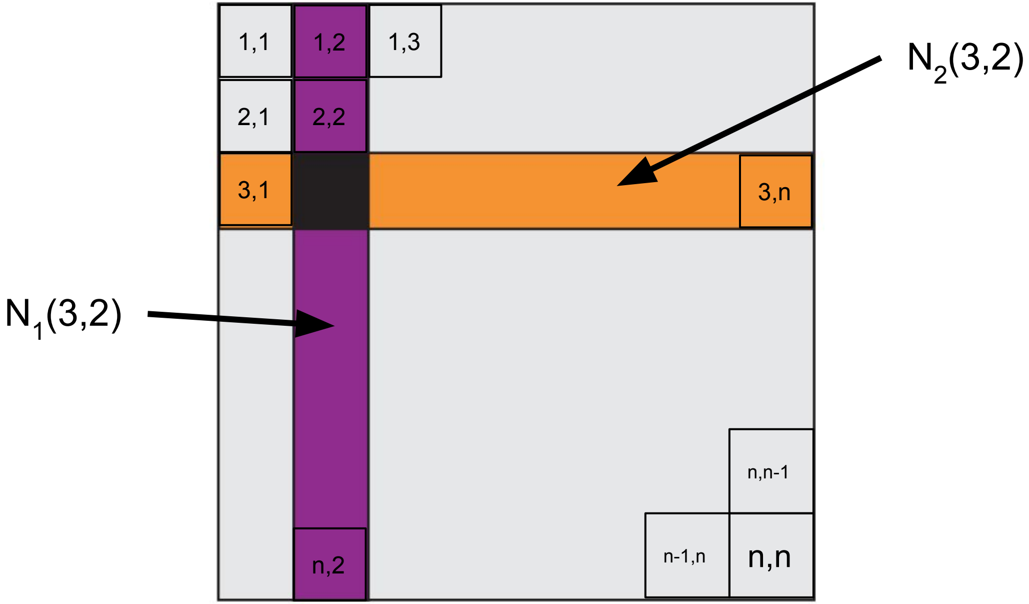
$$\text{WL:} \quad \mathbf{C}_{\boldsymbol{i}}^{l} = \operatorname{enc}\Big(\mathbf{C}_{\boldsymbol{i}}^{l-1}, \Big(\; \{\mathbf{C}_{\boldsymbol{j}}^{l-1} \mid \boldsymbol{j} \in N_j(\boldsymbol{i})\} \;\Big|\; j \in [k] \Big) \; \Big)$$

$$\text{FWL:} \quad \mathbf{C}_{\boldsymbol{i}}^{l} = \operatorname{enc}\Big(\mathbf{C}_{\boldsymbol{i}}^{l-1}, \Big\{\!\!\Big\{ (\mathbf{C}_{\boldsymbol{j}}^{l-1} \mid \boldsymbol{j} \in N_j^F(\boldsymbol{i})) \;\Big|\; j \in [n] \Big\}\!\!\Big\} \; \Big)$$

k-WL Example for **k=2**

$$N_j(\boldsymbol{i}) = \left\{ (i_1, \ldots, i_{j-1}, i', i_{j+1}, \ldots, i_k) \,\middle|\, i' \in [n] \right\}$$

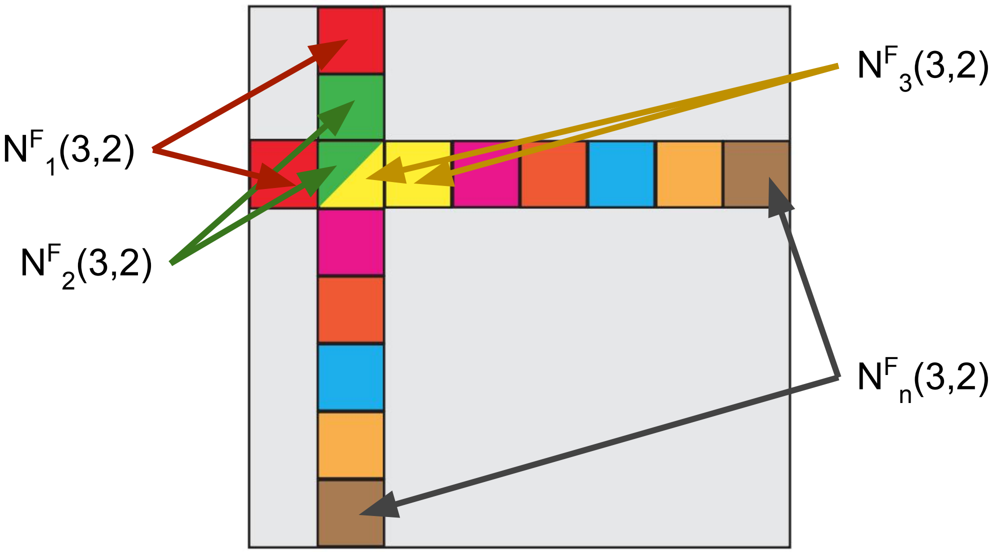
$$\mathbf{C}_{\boldsymbol{i}}^{l} = \mathrm{enc}\Big(\mathbf{C}_{\boldsymbol{i}}^{l-1}, \Big(\{\!\{ \mathbf{C}_{\boldsymbol{j}}^{l-1} \mid \boldsymbol{j} \in N_j(\boldsymbol{i}) \}\!\} \,\Big|\, j \in [k] \Big) \Big)$$



k-FWL Example for **k=2**

$$N_j^F(\boldsymbol{i}) = \Big((j, i_2, \ldots, i_k), (i_1, j, \ldots, i_k), \ldots, (i_1, \ldots, i_{k-1}, j)\Big)$$

$$\mathbf{C}_{\boldsymbol{i}}^l = \mathrm{enc}\Big(\mathbf{C}_{\boldsymbol{i}}^{l-1}, \Big\{\!\!\Big\{ \big(\mathbf{C}_{\boldsymbol{j}}^{l-1} \mid \boldsymbol{j} \in N_j^F(\boldsymbol{i})\big) \;\Big|\; j \in [n] \Big\}\!\!\Big\} \Big)$$



Initialization

Based on the **isomorphism type** of the k-tuple.

$$\mathbf{C}_{\boldsymbol{i}} = \mathbf{C}_{\boldsymbol{i}'} \text{ if for all } q, r \in [k]:$$

1. $v_{i_q} = v_{i_r} \iff v_{i'_q} = v_{i'_r}$
2. $d(v_{i_q}) = d(v_{i'_q})$
3. $(v_{i_r}, v_{i_q}) \in E \iff (v_{i'_r}, v_{i'_q}) \in E$

Input vertex color-assigning function

The K-Ladder

1. 1-WL and 2-WL have equivalent discrimination power.
2. k-FWL is equivalent to $(k+1)$-WL for $k \geq 2$.
3. For each $k \geq 2$ there is a pair of non-isomorphic graphs distinguishable by $(k+1)$-WL but not by k-WL.

Power-sum Multi-symmetric Polynomials

Colors are represented as vectors of length "a", and encoding can be done by concatenation.
We need to find a way to represent multisets of these colors. (While staying equivariant.)

$$\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_a) \in [n]^a$$

$$y^{\boldsymbol{\alpha}} = y_1^{\alpha_1} \cdot y_2^{\alpha_2} \cdots y_a^{\alpha_a}$$

$$p_{\boldsymbol{\alpha}}(\boldsymbol{X}) = \sum_{i=1}^{n} x_i^{\boldsymbol{\alpha}}, \quad \boldsymbol{X} \in \mathbb{R}^{n \times a}$$

Proposition 1

Proposition 1. *For arbitrary* $\boldsymbol{X}, \boldsymbol{X}' \in \mathbb{R}^{n \times a}$*:* $\exists g \in S_n$ *so that* $\boldsymbol{X}' = g \cdot \boldsymbol{X}$ *if and only if* $u(\boldsymbol{X}) = u(\boldsymbol{X}')$.

$$u(\boldsymbol{X}) := \left(p_{\boldsymbol{\alpha}}(\boldsymbol{X}) \mid |\boldsymbol{\alpha}| \leq n\right)$$

$$|\boldsymbol{\alpha}| = \sum_{j=1}^{a} \alpha_j$$

PMP generates the "ring" of MP.
For an arbitrary Multi-symmetric Polynomial ***q****,* there exists a polynomial ***r*** such that:

$$q(\boldsymbol{X}) = r\left(u(\boldsymbol{X})\right)$$

Prop 1 Proof

$$p_{\boldsymbol{\alpha}}(\boldsymbol{X}) = \sum_{i=1}^{n} x_i^{\boldsymbol{\alpha}}$$

$$u(\boldsymbol{X}) := \left(p_{\boldsymbol{\alpha}}(\boldsymbol{X}) \;\middle|\; |\boldsymbol{\alpha}| \leq n\right)$$

$$\boldsymbol{X}' = g \cdot \boldsymbol{X} \longrightarrow u(\boldsymbol{X}) = u(\boldsymbol{X}')$$

True by inspection. (Changing order of summation does nothing)

$$u(\boldsymbol{X}) = u(\boldsymbol{X}') \longrightarrow \boldsymbol{X}' = g \cdot \boldsymbol{X}$$

Proof by contradiction

$$u(\boldsymbol{X}) = u(\boldsymbol{X}') \qquad g \cdot \boldsymbol{X} \neq \boldsymbol{X}'$$

Let $K \subset \mathbb{R}^{n \times a}$ be a compact set containing $[\boldsymbol{X}], [\boldsymbol{X}']$

"Orbit" of X under permutation group

$$[\boldsymbol{X}] = \{g \cdot \boldsymbol{X} \mid g \in S_n\}$$

$$g \cdot \boldsymbol{X} \neq \boldsymbol{X}'$$

$$\downarrow$$

$$[\boldsymbol{X}] \cap [\boldsymbol{X}'] = \emptyset$$

We can construct a continuous function that separates inputs from X and X'.
Via the Stone–Weierstrass Theorem applied to real continuous functions on K, we can approximate this function with a polynomial ***f***.

Prop 1 Proof Cont.

$$q(\boldsymbol{X}) = \frac{1}{n!}\sum_{g\in S_n} f(g\cdot\boldsymbol{X}) \longleftarrow f|_{[\boldsymbol{X}]} \geq 1 \text{ and } f|_{[\boldsymbol{X}']} \leq 0$$

$$q(g\cdot\boldsymbol{X}) = q(\boldsymbol{X}), \text{ for all } g \in S_n$$

q is a **multi-symmetric polynomial**

$$q(\boldsymbol{X}) = r(u(\boldsymbol{X}))$$

Assumption from beginning of proof by contradiction

$$1 \leq q(\boldsymbol{X}) = \boxed{r(u(\boldsymbol{X})) = r(u(\boldsymbol{X}'))} = q(\boldsymbol{X}') \leq 0$$

K-order Graph Networks vs. k-WL

Theorem 1. *Given two graphs $G = (V, E, d)$, $G' = (V', E', d')$ that can be distinguished by the k-WL graph isomorphism test, there exists a k-order network F so that $F(G) \neq F(G')$. On the other direction for every two isomorphic graphs $G \cong G'$ and k-order network F, $F(G) = F(G')$.*

Showing that K-order Graph Networks are as powerful as k-WL

K-order Graph Networks vs. k-WL

Construction of the input tensor X (for consistency we will use the notation B).

First, an input graph $G = (V, E, d)$ is represented using a tensor of the form $\mathbf{B} \in \mathbb{R}^{n^2 \times (e+1)}$, as follows. The last channel of $\mathbf{B}$, namely $\mathbf{B}_{:,:,e+1}$ (':' stands for all possible values $[n]$) encodes the adjacency matrix of G according to E. The first e channels $\mathbf{B}_{:,:,1:e}$ are zero outside the diagonal, and $\mathbf{B}_{i,i,1:e} = d(v_i) \in \mathbb{R}^e$ is the color of vertex $v_i \in V$.

First e channels: representing the initial color of all vertex
Last channel: representing the adjacency matrix of the graph

Properties of the input tensor X (i.e., Tensor B constructed above)

For any two isomorphic graphs G and G', we have X = g(X') for some permutation function g

K-order Graph Networks vs. k-WL

Proof of the argument: if G = G’, then F(G)=F(G’)

Recall the property of k-order graph networks

$$F(g \cdot \mathbf{X}) = m(\cdots(L_1(g \cdot \mathbf{X}))\cdots) = m(\cdots(g \cdot L_1(\mathbf{X}))\cdots) = \cdots = m(h(g \cdot L_d(\cdots))) = F(\mathbf{X})$$

Note that G=G’ implies X = g(X’), then F(G) = F(X) = F(g(X’)) = F(X’) = F(G’)

K-order Graph Networks vs. k-WL

Proof of the argument: if G and G' can be distinguished by the k-WL test, there exists a k-order network F so that F(G) != F(G')

Initialization/First layer

Recall the input tensor X has dimension **n^2 * (e+1)**, then the **linear equivariant operator** in the first layer is defined by

$$L(\mathbf{X})_{\boldsymbol{i},r,s,w} = \mathbf{X}_{i_r,i_s,w}, \quad w \in [e+1]$$

subtensors of X defined by the k-tuple of vertices

$$L(\mathbf{X})_{\boldsymbol{i},r,s,e+2} = \begin{cases} 1 & i_r = i_s \\ 0 & \text{otherwise} \end{cases}$$

equality pattern of the k-tuple i

$$L : \mathbb{R}^{n^2 \times (e+1)} \to \mathbb{R}^{n^k \times k^2 \times (e+2)}$$

K-order Graph Networks vs. k-WL

Initialization/First layer

$$L(\mathbf{X})_{\boldsymbol{i},r,s,w} = \mathbf{X}_{i_r,i_s,w}, \quad w \in [e+1]$$

$$L(\mathbf{X})_{\boldsymbol{i},r,s,e+2} = \begin{cases} 1 & i_r = i_s \\ 0 & \text{otherwise} \end{cases}$$

$$L : \mathbb{R}^{n^2 \times (e+1)} \to \mathbb{R}^{n^k \times k^2 \times (e+2)}$$

Verifying its equivariant property

L is equivariant with respect to the permutation action. Indeed, for $w \in [e+1]$,

$$(g \cdot L(\mathbf{X}))_{\boldsymbol{i},r,s,w} = L(\mathbf{X})_{g^{-1}(\boldsymbol{i}),r,s,w} = \mathbf{X}_{g^{-1}(i_r),g^{-1}(i_s),w} = (g \cdot \mathbf{X})_{i_r,i_s,w} = L(g \cdot \mathbf{X})_{\boldsymbol{i},r,s,w}.$$

For $w = e+2$ we have

$$(g \cdot L(\mathbf{X}))_{\boldsymbol{i},r,s,w} = L(\mathbf{X})_{g^{-1}(\boldsymbol{i}),r,s,w} = \begin{cases} 1 & g^{-1}(i_r) = g^{-1}(i_s) \\ 0 & \text{otherwise} \end{cases} = \begin{cases} 1 & i_r = i_s \\ 0 & \text{otherwise} \end{cases} = L(g \cdot \mathbf{X})_{\boldsymbol{i},r,s,w}.$$

K-order Graph Networks vs. k-WL

k-WL update step

Recall the definitions of neighborhoods and coloring representation update rules of k-WL

$$\boxed{N_j(\boldsymbol{i}) = \left\{ (i_1, \ldots, i_{j-1}, i', i_{j+1}, \ldots, i_k) \,\middle|\, i' \in [n] \right\}}$$

For any j we have n neighborhood tuples

$$\text{WL:} \quad \mathbf{C}^l_{\boldsymbol{i}} = \operatorname{enc}\left(\mathbf{C}^{l-1}_{\boldsymbol{i}}, \left(\boxed{\{\mathbf{C}^{l-1}_{\boldsymbol{j}} \mid \boldsymbol{j} \in N_j(\boldsymbol{i})\} \,\middle|\, j \in [k]} \right) \right)$$

Neighborhood aggregation

K-order Graph Networks vs. k-WL

k-WL update step

The dimension of color representations increases hugely (b > n^a) (although in practice one can set b=a).

Let B be the input tensor with dimension **n^k * a**

First, apply the polynomial function $\tau : \mathbb{R}^a \rightarrow \mathbb{R}^b$, $\boxed{b = \binom{n+a}{a}}$ entrywise to $\mathbf{B}$, where τ is defined by $\tau(x) = (x^{\boldsymbol{\alpha}})_{|\boldsymbol{\alpha}| \leq n}$ (note that b is the number of multi-indices $\boldsymbol{\alpha}$ such that $|\boldsymbol{\alpha}| \leq n$). This gives $\mathbf{Y} \in \mathbb{R}^{n^k \times b}$ where $\mathbf{Y}_{\boldsymbol{i},:} = \tau(\mathbf{B}_{\boldsymbol{i},:}) \in \mathbb{R}^b$.

Second, apply the linear operator

$$\boxed{\mathbf{C}^j_{\boldsymbol{i},r} := L_j(\mathbf{Y})_{\boldsymbol{i},r} = \sum_{i'=1}^{n} \mathbf{Y}_{i_1,\cdots,i_{j-1},i',i_{j+1},\ldots,i_k,r}, \quad \boldsymbol{i} \in [n]^k, r \in [b].}$$

Calculate the power-sum symmetric polynomials for the neighborhood set $N_j(i)$

K-order Graph Networks vs. k-WL

Verifying the equivariant property

L_j is equivariant with respect to the permutation action. Indeed, $L_j(g \cdot \mathbf{Y})_{\boldsymbol{i},r} = \sum_{i'=1}^{n} (g \cdot \mathbf{Y})_{i_1, \cdots, i_{j-1}, i', i_{j+1}, \ldots, r} = \sum_{i'=1}^{n} \mathbf{Y}_{g^{-1}(i_1) \cdots, g^{-1}(i_{j-1}), i', g^{-1}(i_{j+1}), \ldots, r} = L_j(\mathbf{Y})_{g^{-1}(\boldsymbol{i}), r} = (g \cdot L_j(\mathbf{Y}))_{\boldsymbol{i},r}$.

Verifying the bijective property

Now, note that

$$\mathbf{C}^j_{\boldsymbol{i},:} = L_j(\mathbf{Y})_{\boldsymbol{i},:} = \sum_{i'=1}^{n} \tau(\mathbf{B}_{i_1, \cdots, i_{j-1}, i', i_{j+1}, \ldots, i_k, :}) = \sum_{\boldsymbol{j} \in N_j(\boldsymbol{i})} \tau(\mathbf{B}_{\boldsymbol{j},:}) = u(\boldsymbol{X}),$$

where $\boldsymbol{X} = \mathbf{B}_{i_1, \ldots, i_{j-1}, :, i_{j+1}, \ldots, i_k, :}$ as desired.

Concatenation

Third, the k-WL update step is the concatenation: $(\mathbf{B}, \mathbf{C}^1, \ldots, \mathbf{C}^k)$ Dimension: kb+a

K-order Graph Networks vs. k-WL

Histogram Computation

- The output coloring tensor $H(\mathbf{B}) \in \mathbb{R}^{n^k \times a}$ Here **a** denotes the dimension of the output color representation vector (extremely large).
- The set of initial colors and graphs are finite (assumption).
- Let **b** denote the size of the set of all possible colors. Use one-hot encoding (denoted by **m**) to each color in H(B), the obtained tensor, denoted by **Y**, has dimension **n^k * b.**
- Using summing invariance operator **h** defined as follows to generate the desired histogram.

$$h(\mathbf{Y})_j = \sum_{\boldsymbol{i} \in [n]^k} \mathbf{Y}_{\boldsymbol{i},j}, \; j \in [b]$$

Finally, the k-order invariant network is defined by

$$F = h \circ m \circ L_d \circ \sigma \circ \cdots \circ \sigma \circ L_1$$

Pros and Cons of K-order Graph Networks

Pros

- The k-order graph network is at least as powerful as k-WL test, thus has higher expressive power than message passing graph networks.

Cons

- The k-order graph network is computationally inefficient since it involves the calculation of high-order tensors.

A Simple Network with 3-WL Discrimination Power

Block structure

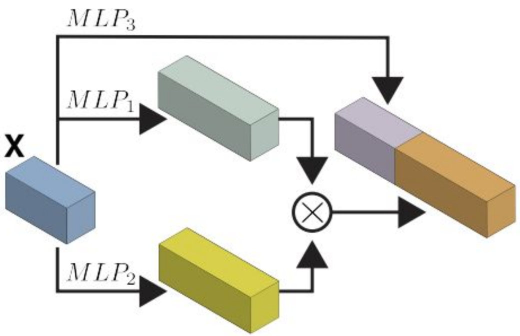
$$F = m \circ h \circ B_d \circ B_{d-1} \cdots \circ B_1$$

$B_1, \ldots, B_d$ are blocks

$$m_3 : \mathbb{R}^a \to \mathbb{R}^{b'}$$

Input tensor

$$\mathbf{X} \in \mathbb{R}^{n \times n \times a}$$



$$m_1, m_2 : \mathbb{R}^a \to \mathbb{R}^b$$

Figure 2: Block structure.

Matrix product

$$\mathbf{W}_{:,:,j} := m_1(\mathbf{X})_{:,:,j} \cdot m_2(\mathbf{X})_{:,:,j},\ j \in [b]$$

A Simple Network with 3-WL Discrimination Power

Lemma 1. *The model F described above is invariant, i.e., $F(g \cdot \mathsf{B}) = F(\mathsf{B})$, for all $g \in S_n$, and B.*

Proof. Note that matrix multiplication is equivariant: for two matrices $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{n \times n}$ and $g \in S_n$ one has $(g \cdot \boldsymbol{A}) \cdot (g \cdot \boldsymbol{B}) = g \cdot (\boldsymbol{A} \cdot \boldsymbol{B})$. This makes the basic building block B_i equivariant, and consequently the model F invariant, i.e., $F(g \cdot \mathsf{B}) = F(\mathsf{B})$. $\square$

Permutation on the graph does not affect matrix production

A Simple Network with 3-WL Discrimination Power

Theorem 2. *Given two graphs $G = (V, E, d)$, $G' = (V', E', d')$ that can be distinguished by the 3-WL graph isomorphism test, there exists a network F (equation 6) so that $F(G) \neq F(G')$. On the other direction for every two isomorphic graphs $G \cong G'$ and F (Equation 6), $F(G) = F(G')$.*

Showing that the constructed Graph Networks are as powerful as 3-WL

Main idea: proving that the network can be as powerful as 2-FWL

A Simple Network with 3-WL Discrimination Power

Construction of the input tensor X (for consistency we will use the notation B).

Input. We assume our input tensors have the form $\mathbf{B} \in \mathbb{R}^{n^2 \times (e+2)}$. The first $e+1$ channels are as before, namely encode vertex colors (features) and adjacency information. The $e+2$ channel is simply taken to be the identity matrix, that is $\mathbf{B}_{:,:,e+2} = I_d$.

Properties of the input tensor X (i.e., Tensor B constructed above)

For any two isomorphic graphs G and G', we have X = g(X') for any g

Then it is easy to see that if G = G', we have F(G)=F(G')

A Simple Network with 3-WL Discrimination Power

Initialization/First layer

$A := \mathbf{B}_{:,:,e+1}$ Adjacency matrix $\quad\quad$ $\mathbf{Y} := \mathbf{B}_{:,:,1:e}$ Input color tensor

The output of the first layer $\mathbf{C} \in \mathbb{R}^{n^2 \times (4e+1)}$ is the concatenation of the following matrices

$$A \cdot \mathbf{Y}_{:,:,j}, \quad (\mathbf{1}\mathbf{1}^T - A) \cdot \mathbf{Y}_{:,:,j}, \quad \mathbf{Y}_{:,:,j} \cdot A, \quad \mathbf{Y}_{:,:,j} \cdot (\mathbf{1}\mathbf{1}^T - A), \quad j \in [e],$$

A Simple Network with 3-WL Discrimination Power

2-FWL update

Recall the definitions of neighborhoods and coloring update rules of k-WL

$$N_j^F(\boldsymbol{i}) = \Big((j, i_2, \dots, i_k), (i_1, j, \dots, i_k), \dots, (i_1, \dots, i_{k-1}, j) \Big)$$

$$\text{FWL:} \quad \mathbf{C}_{\boldsymbol{i}}^{l} = \mathrm{enc}\Big(\mathbf{C}_{\boldsymbol{i}}^{l-1}, \Big\{\!\!\Big\{ (\mathbf{C}_{\boldsymbol{j}}^{l-1} \mid \boldsymbol{j} \in N_j^F(\boldsymbol{i})) \;\Big|\; j \in [n] \Big\}\!\!\Big\} \Big)$$

A Simple Network with 3-WL Discrimination Power

2-FWL update

when k = 2, we have

$$\mathbf{C}_{\boldsymbol{i}} = \mathrm{enc}\Big(\mathbf{B}_{\boldsymbol{i}}, \Big\{\!\!\Big\{(\mathbf{B}_{j,i_2}, \mathbf{B}_{i_1,j}) \Big| j \in [n]\Big\}\!\!\Big\} \Big)$$

To implement this we will need to compute a tensor $\mathbf{Y}$, where the coloring $\mathbf{Y}_{\boldsymbol{i}}$ encodes the multiset $\Big\{\!\!\Big\{(\mathbf{B}_{j,i_2,:}, \mathbf{B}_{i_1,j,:}) \Big| j \in [n]\Big\}\!\!\Big\}$.

Let X be the concatenation of the neighborhoods of the multiset i = (i_1, i_2)

$$\boldsymbol{X}_{j,:} = (\mathbf{B}_{j,i_2,:}, \mathbf{B}_{i_1,j,:}), \quad j \in [n].$$

Our goal is to compute an output tensor $\mathbf{W} \in \mathbb{R}^{n^2 \times b}$, where $\mathbf{W}_{i_1,i_2,:} = u(\boldsymbol{X})$

A Simple Network with 3-WL Discrimination Power

Calculation of W

Consider the multi-index set $\{\boldsymbol{\alpha} \mid \boldsymbol{\alpha} \in [n]^{2a}, |\boldsymbol{\alpha}| \leq n\}$ of cardinality $b = \binom{n+2a}{2a}$, and write it in the form $\{(\boldsymbol{\beta}_l, \boldsymbol{\gamma}_l) \mid \boldsymbol{\beta}, \boldsymbol{\gamma} \in [n]^a, |\boldsymbol{\beta}_l| + |\boldsymbol{\gamma}_l| \leq n, l \in b\}$. Now define polynomial maps $\tau_1, \tau_2 : \mathbb{R}^a \to \mathbb{R}^b$ by $\tau_1(x) = (x^{\boldsymbol{\beta}_l} \mid l \in [b])$, and $\tau_2(x) = (x^{\boldsymbol{\gamma}_l} \mid l \in [b])$. We apply τ_1 to the features of $\mathbf{B}$, namely $\mathbf{Y}_{i_1,i_2,l} := \tau_1(\mathbf{B})_{i_1,i_2,l} = (\mathbf{B}_{i_1,i_2,:})^{\boldsymbol{\beta}_l}$; similarly, $\mathbf{Z}_{i_1,i_2,l} := \tau_2(\mathbf{B})_{i_1,i_2,l} = (\mathbf{B}_{i_1,i_2,:})^{\boldsymbol{\gamma}_l}$. Now,

$$
\begin{aligned}
\mathbf{W}_{i_1,i_2,l} &:= (\mathbf{Z}_{:,:,l} \cdot \mathbf{Y}_{:,:,l})_{i_1,i_2} = \sum_{j=1}^{n} \mathbf{Z}_{i_1,j,l} \mathbf{Y}_{j,i_2,l} = \sum_{j=1}^{n} \tau_1(\mathbf{B})_{j,i_2,l}\, \tau_2(\mathbf{B})_{i_1,j,l} \\
&= \sum_{j=1}^{n} \mathbf{B}_{j,i_2,:}^{\boldsymbol{\beta}_l} \mathbf{B}_{i_1,j,:}^{\boldsymbol{\gamma}_l} = \sum_{j=1}^{n} (\mathbf{B}_{j,i_2,:}, \mathbf{B}_{i_1,j,:})^{(\boldsymbol{\beta}_l, \boldsymbol{\gamma}_l)},
\end{aligned}
$$

hence $\mathbf{W}_{i_1,i_2,:} = u(\boldsymbol{X})$

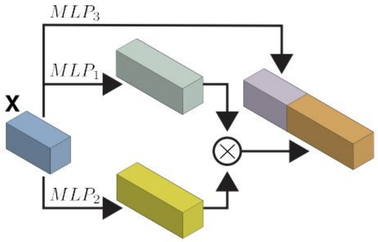


Figure 2: Block structure.

Let m1 and m2 be corresponding to polynomials tau_1 and tau_2, and m3 be the identity mapping, we can get the desired output via concatenating (B, m1(B), m2(B))

A Simple Network with 3-WL Discrimination Power

Why using 2-FWL to construct the network rather than using 3-WL (used in Morris et al., 2018)?

2-FWL only needs to deal with **n^2 multisets** while 3-WL needs to deal with n^3 multisets. Thus the 2-FWL based graph networks have lower space and time complexities

Experiments

- GNN models are implemented as block struct
- Classification tasks
  - Social network, Bioinformatics
  - Parameter search: 10-fold cross validation
    - hyper parameters: learning rate, decay..
    - structure (a, b, etc.) for the block structure
  - Results: best averaged accuracy across the 10-folds
-

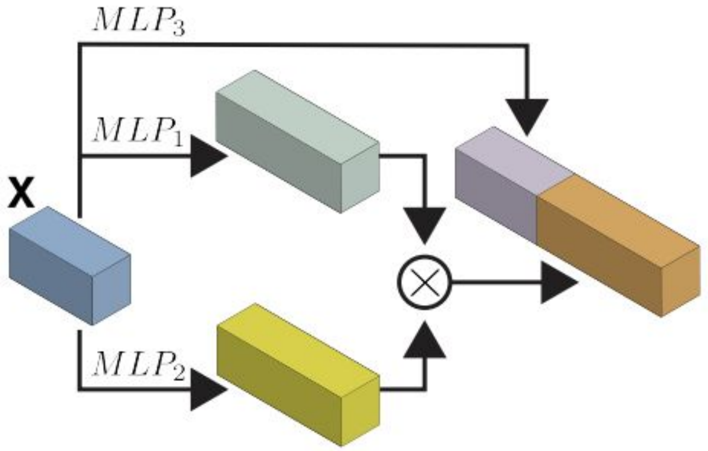


Figure 2: Block structure.

Table 1: Graph Classification Results on the datasets from Yanardag and Vishwanathan (2015)

dataset	MUTAG	PTC	PROTEINS	NCI1	NCI109	COLLAB	IMDB-B	IMDB-M
size	188	344	1113	4110	4127	5000	1000	1500
classes	2	2	2	2	2	3	2	3
avg node #	17.9	25.5	39.1	29.8	29.6	74.4	19.7	13
				Results				
GK (Shervashidze et al., 2009)	81.39±1.7	55.65±0.5	71.39±0.3	62.49±0.3	62.35±0.3	NA	NA	NA
RW (Vishwanathan et al., 2010)	79.17±2.1	55.91±0.3	59.57±0.1	> 3 days	NA	NA	NA	NA
PK (Neumann et al., 2016)	76±2.7	59.5±2.4	73.68±0.7	82.54±0.5	NA	NA	NA	NA
WL (Shervashidze et al., 2011)	84.11±1.9	57.97±2.5	**74.68±0.5**	**84.46±0.5**	**85.12±0.3**	NA	NA	NA
FGSD (Verma and Zhang, 2017)	**92.12**	**62.80**	73.42	79.80	78.84	**80.02**	73.62	**52.41**
AWE-DD (Ivanov and Burnaev, 2018)	NA	NA	NA	NA	NA	73.93±1.9	**74.45 ± 5.8**	51.54 ±3.6
AWE-FB (Ivanov and Burnaev, 2018)	87.87±9.7	NA	NA	NA	NA	70.99 ± 1.4	73.13 ±3.2	51.58 ± 4.6
DGCNN (Zhang et al., 2018)	85.83±1.7	58.59±2.5	75.54±0.9	74.44±0.5	NA	73.76±0.5	70.03±0.9	47.83±0.9
PSCN (Niepert et al., 2016)(k=10)	88.95±4.4	62.29±5.7	75±2.5	76.34±1.7	NA	72.6±2.2	71±2.3	45.23±2.8
DCNN (Atwood and Towsley, 2016)	NA	NA	61.29±1.6	56.61± 1.0	NA	52.11±0.7	49.06±1.4	33.49±1.4
ECC (Simonovsky and Komodakis, 2017)	76.11	NA	NA	76.82	75.03	NA	NA	NA
DGK (Yanardag and Vishwanathan, 2015)	87.44±2.7	60.08±2.6	75.68±0.5	80.31±0.5	80.32±0.3	73.09±0.3	66.96±0.6	44.55±0.5
DiffPool (Ying et al., 2018)	NA	NA	**78.1**	NA	NA	75.5	NA	NA
CCN (Kondor et al., 2018)	**91.64±7.2**	**70.62±7.0**	NA	76.27±4.1	75.54±3.4	NA	NA	NA
Invariant Graph Networks (Maron et al., 2019a)	83.89±12.95	58.53±6.86	76.58±5.49	74.33±2.71	72.82±1.45	78.36±2.47	72.0±5.54	48.73±3.41
GIN (Xu et al., 2019)	89.4±5.6	64.6±7.0	76.2±2.8	82.7±1.7	NA	80.2±1.9	**75.1±5.1**	**52.3±2.8**
1-2-3 GNN (Morris et al., 2018)	86.1±	60.9±	75.5±	76.2±	NA	NA	74.2±	49.5±
Ours 1	90.55±8.7	66.17±6.54	77.2±4.73	**83.19±1.11**	**81.84±1.85**	80.16±1.11	72.6±4.9	50±3.15
Ours 2	88.88±7.4	64.7±7.46	76.39±5.03	81.21±2.14	**81.77±1.26**	**81.38±1.42**	72.2±4.26	44.73±7.89
Ours 3	89.44±8.05	62.94±6.96	76.66±5.59	80.97±1.91	**82.23±1.42**	**80.68±1.71**	73±5.77	50.46±3.59
Rank	$\mathbf{3^{rd}}$	$\mathbf{2^{nd}}$	$\mathbf{2^{nd}}$	$\mathbf{2^{nd}}$	$\mathbf{2^{nd}}$	$\mathbf{1^{st}}$	$\mathbf{6^{th}}$	$\mathbf{5^{th}}$

Experiments

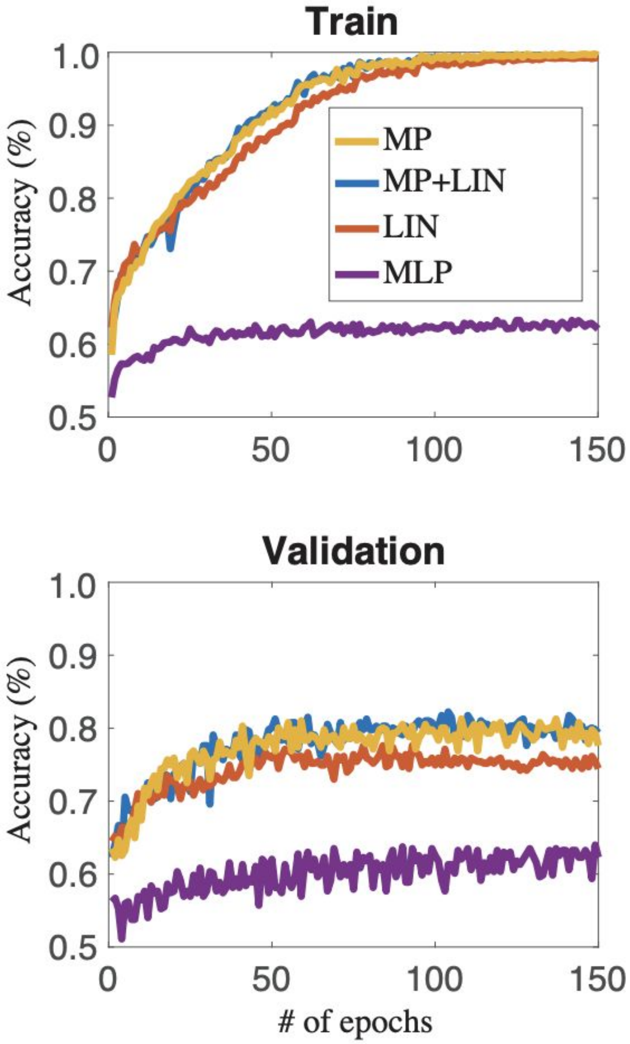
- Regression task
    - QM9, physical quantities prediction
    - 80% - 10% - 10% train-val-test split
    - predict 12 quantities using one network v.s. using 12 networks seperately

Table 2: Regression, the QM9 dataset.

Target	DTNN	MPNN	123-gnn	Ours 1	Ours 2
μ	0.244	0.358	0.476	**0.231**	**0.0934**
α	0.95	0.89	**0.27**	0.382	0.318
ϵ_{homo}	0.00388	0.00541	0.00337	**0.00276**	**0.00174**
ϵ_{lumo}	0.00512	0.00623	0.00351	**0.00287**	**0.0021**
Δ_ϵ	0.0112	0.0066	0.0048	**0.00406**	**0.0029**
$\langle R^2 \rangle$	17	28.5	22.9	**16.07**	**3.78**
$ZPVE$	0.00172	0.00216	**0.00019**	0.00064	0.000399
U_0	-	-	0.0427	0.234	**0.022**
U	-	-	0.111	0.234	**0.0504**
H	-	-	0.0419	0.229	**0.0294**
G	-	-	0.0469	0.238	**0.024**
C_v	0.27	0.42	**0.0944**	0.184	0.144

Equivariant layer evaluation

- Comparing with other models
  - Proposed model (MP, this work)
  - Matrix product + full linear basis from (Maron et al., 2019a) (
  - only full linear basis (LIN)
  - MLP applied to feature dimension (without graph knowledge)
- Results:
  - All can achieve 0 train error excl. MLP
  - MP, MP + LIN has better generalization performance
  - MP is more efficient than MP + LIN



Experiment code

- Our verification code available at Google Colab https://colab.research.google.com/drive/1V4DRDXj9UtfULdrGhDQbuX4S-KEFOJk_?usp=sharing
- Original GitHub code available at hadarser/ProvablyPowerfulGraphNetworks (https://github.com/hadarser/ProvablyPowerfulGraphNetworks)

Thank you!