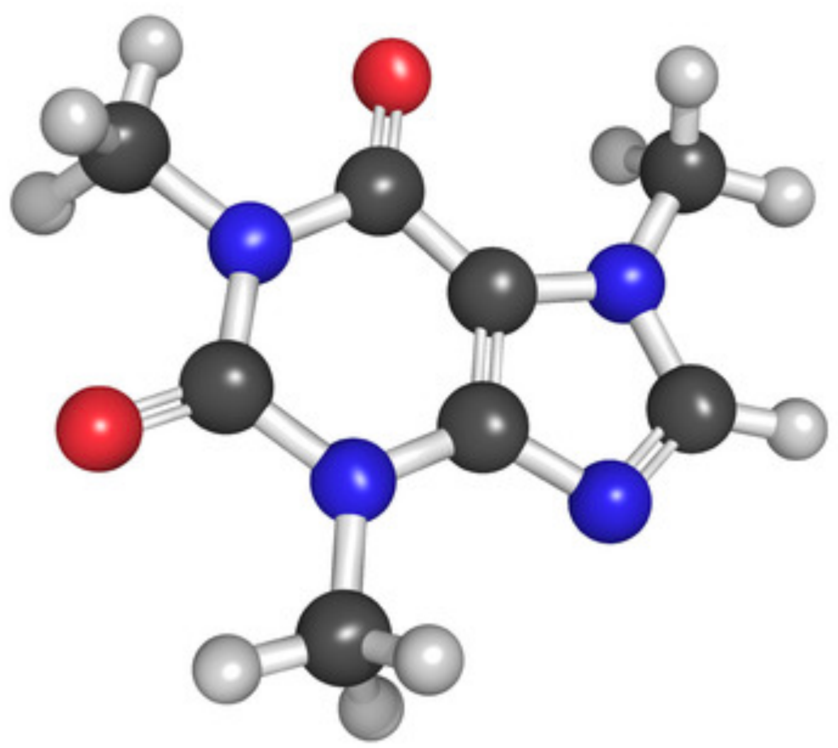
Graph Neural Networks: Expressive Power, Generalization, Extrapolation

Keyulu Xu

MIT

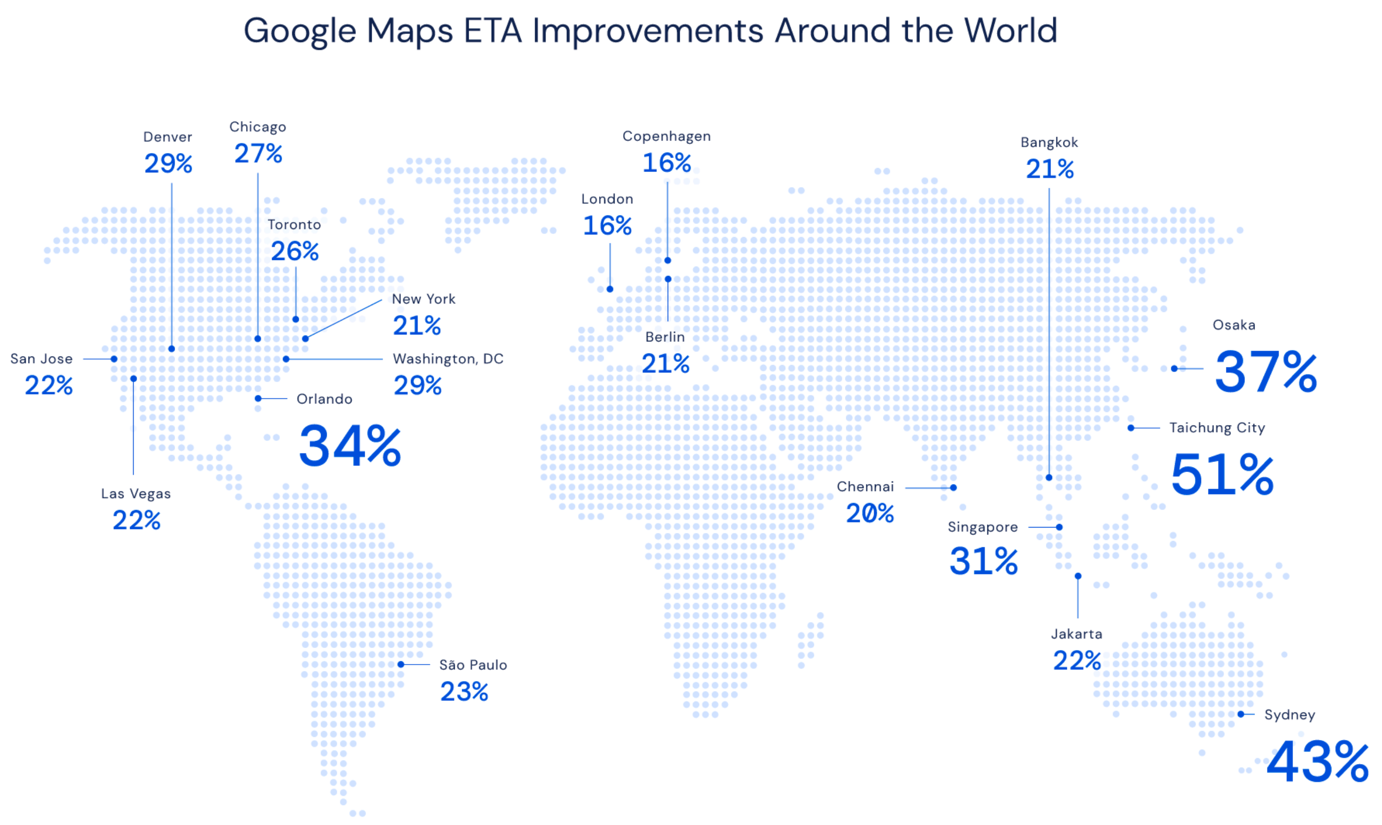
joint work with S. Jegelka, K. Kawarabayashi, S. S. Du, J. Leskovec, R. Salakhutdinov, W. Hu, M. Zhang, J. Li, R. Wang, K. Hou, B. Poczos

Applications



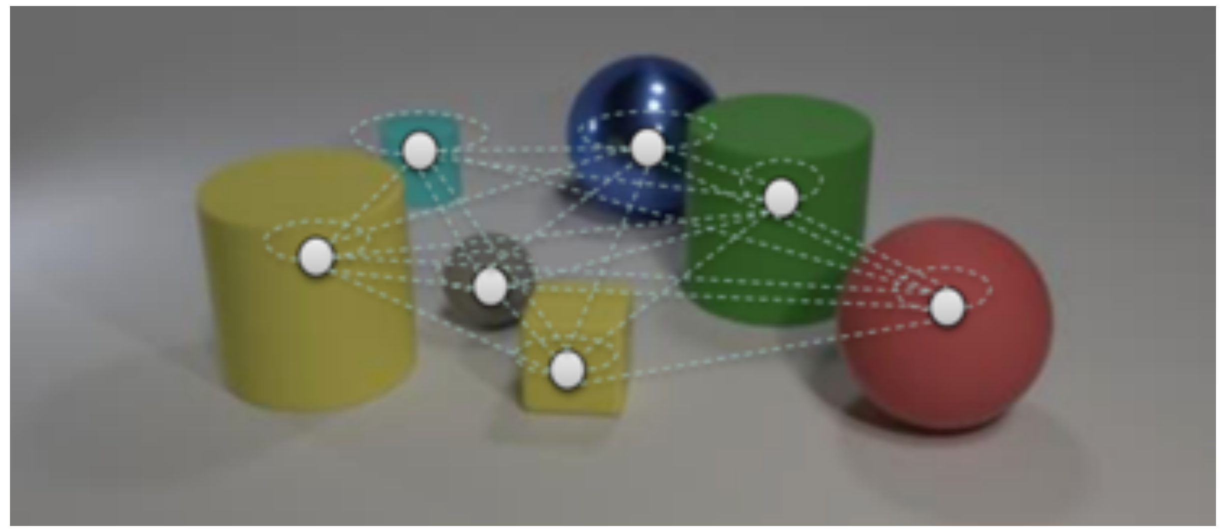
Drug discovery
(Duvenaud et al. 2015)

Recommender system
(Ying et al. 2018)



Google Map ETA
(Lange et al. 2020)

Physical reasoning
(Wu et al. 2017)



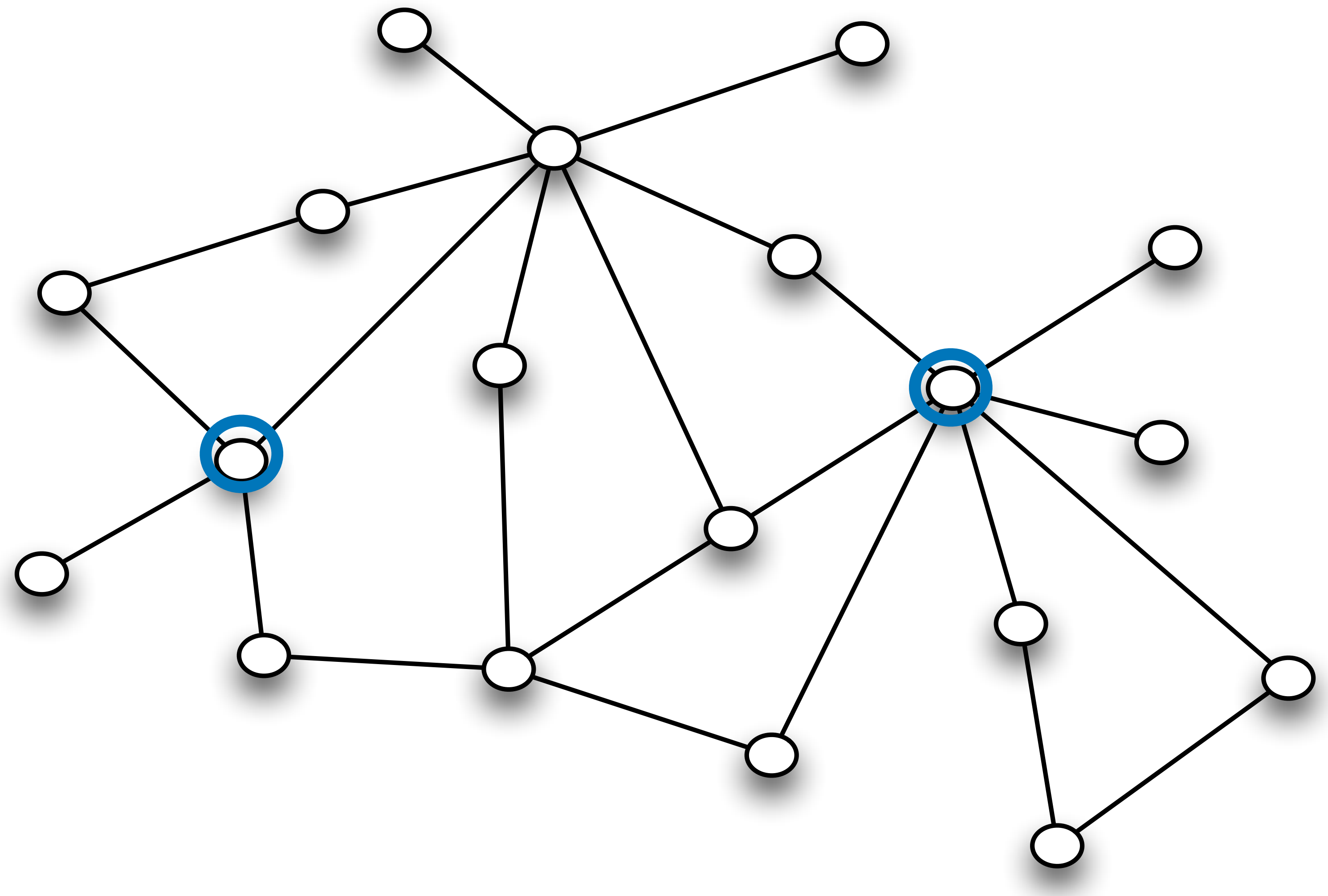
Visual reasoning
(Santoro et al. 2017)

Learning with graphs

Input: graph $G = (V, E)$

X_u node features

$w_{u,v}$ edge features (optional)



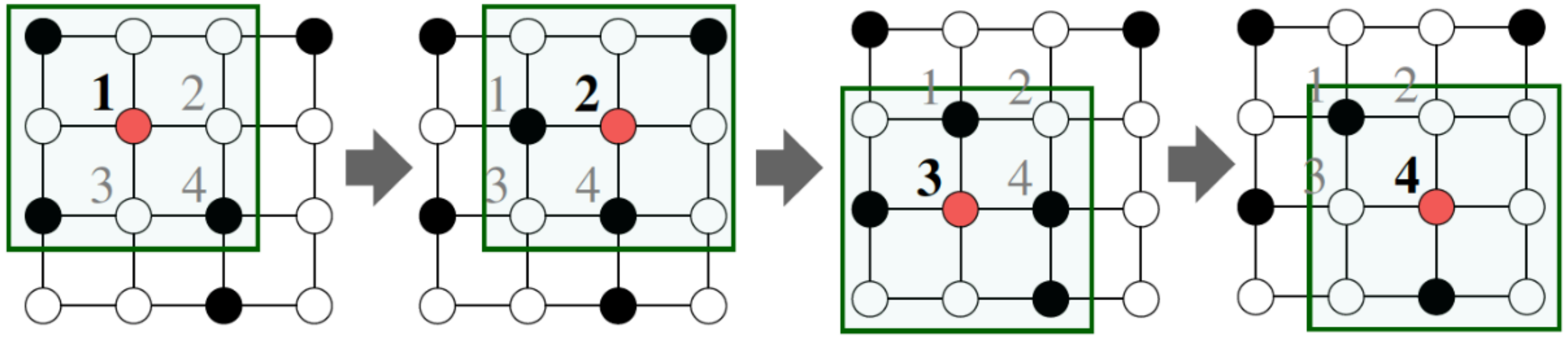
Goal: learn node & graph representations

$$h_u \in \mathbb{R}^d \qquad h_G \in \mathbb{R}^l$$

Task: forecast on graphs, nodes, or edges

How to represent graphs?

CNNs for images



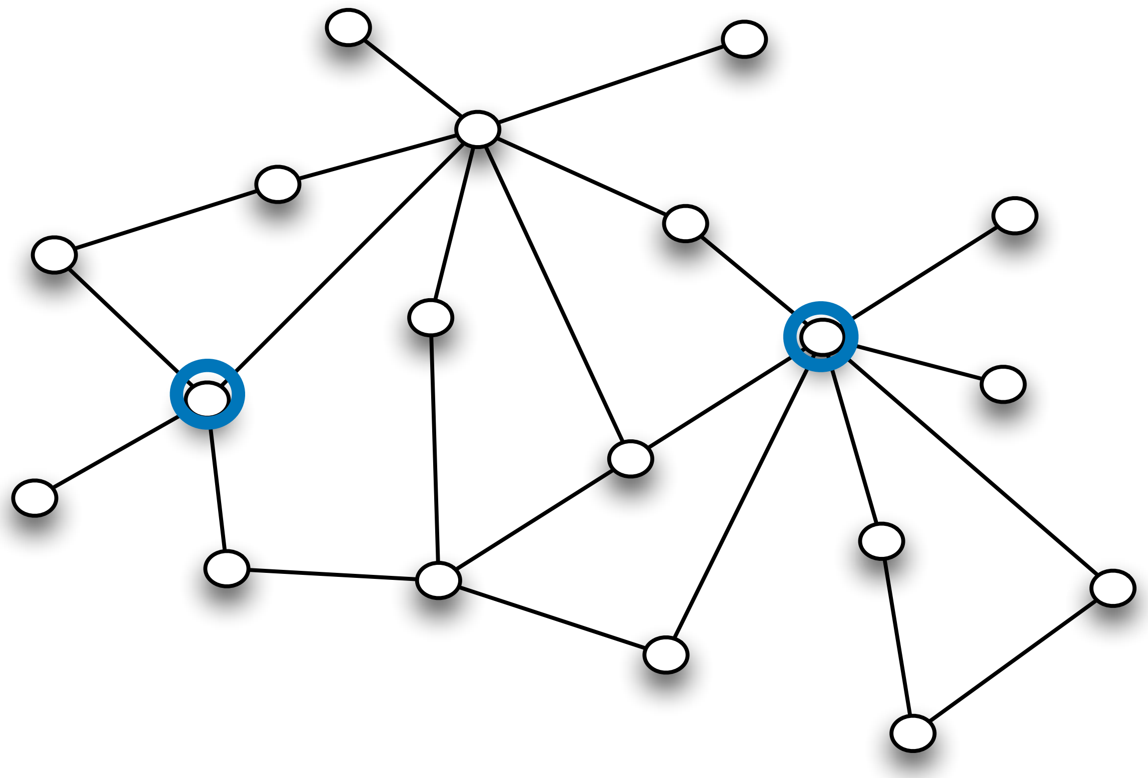
(illustration: J. Leskovec)

(@kyokofukada_official)

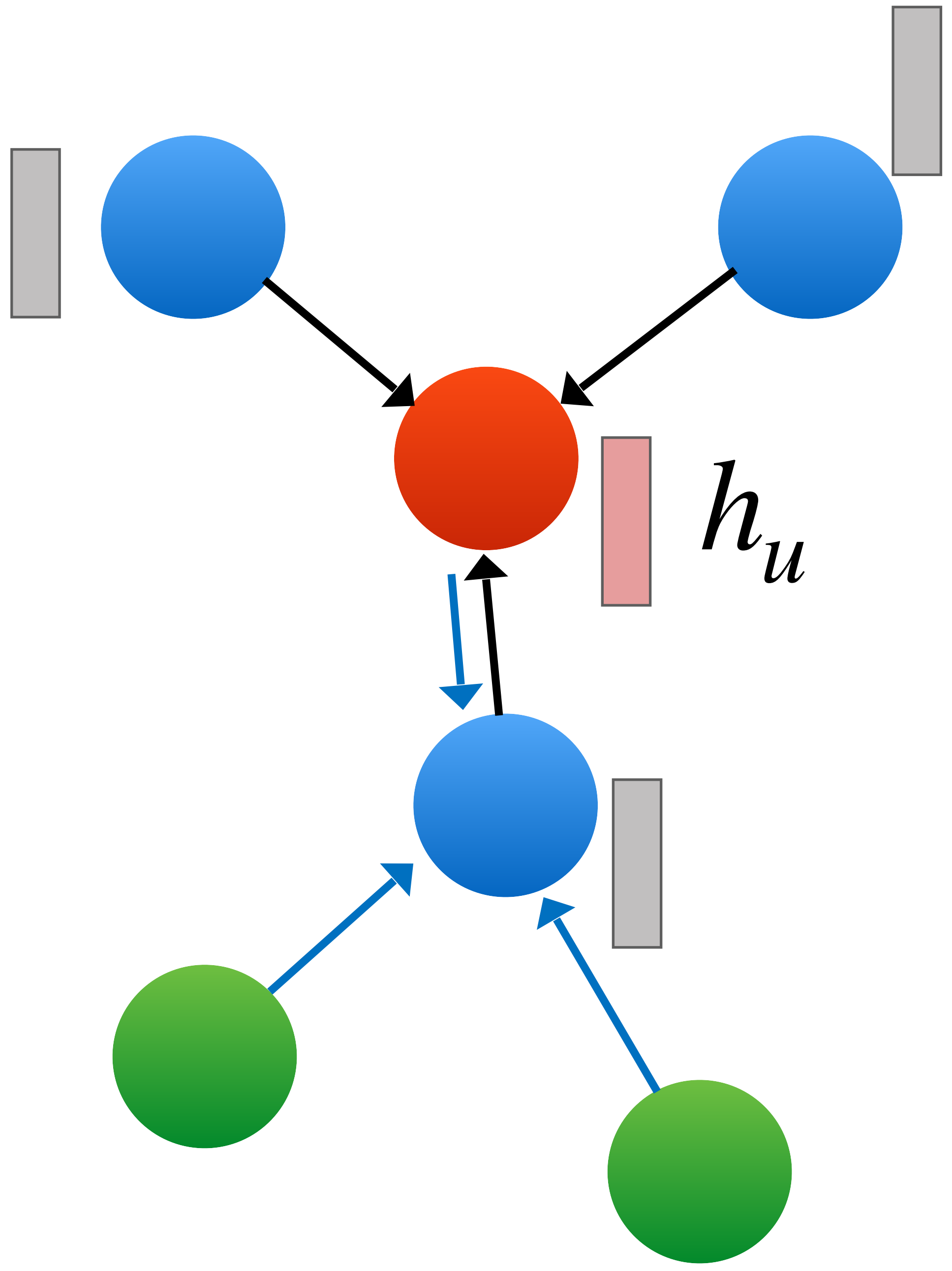
vs. general graphs

no ordering of neighbors

different node degrees



Graph Neural Networks (GNNs)



In each round:

For $u \in V$ concurrently:

Aggregate over neighbors

Representation of neighbor node v in round $k-1$

$$h_u^{(k)} = \text{AGGREGATE}^{(k)}\left(\left\{\left(h_v^{(k-1)}, h_u^{(k-1)}\right)\right\} \,\middle|\, v \in \mathcal{N}(u)\right)$$

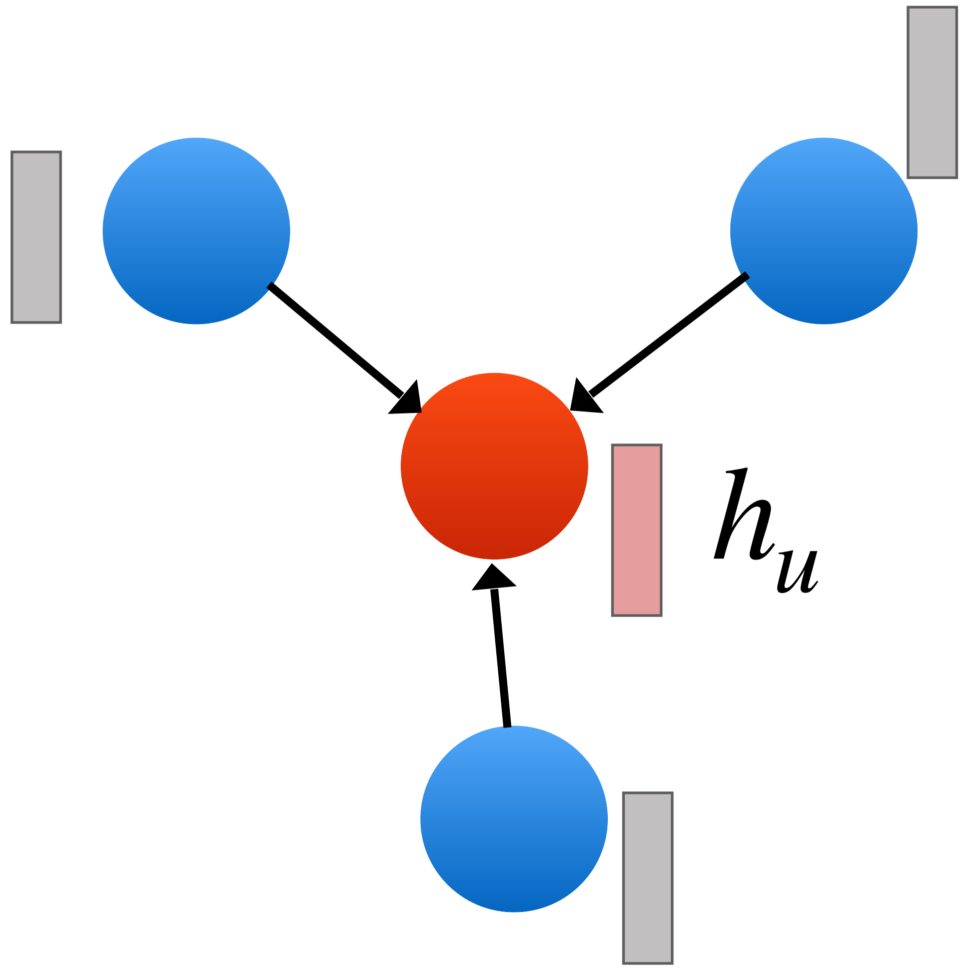
....

Graph-level **readout**

$$h_G = \text{READOUT}\left(\left\{h_u^{(K)}\right\} \,\middle|\, u \in V\right)$$

(Gori et al. 2005, Merkwirth & Lengauer 2005, Scarselli et al 2009, Duvenaud et al., 2015, Battaglia et al., 2016, Dai et al., 2016, Defferrard et al., 2016, Kearnes et al., 2016, Li et al., 2016, Gilmer et al., 2017, Hamilton et al., 2017, Kipf & Welling, 2017, Velickovic et al., 2018, Xu et al., 2018)

Training



1. Parameterize AGGREGATE$^{(k)}$ and READOUT

Can also recover ConvNets, Transformer etc

2. Specify a loss on node/graph/edge representations

3. Train on data points with SGD

Roadmap

- **Expressive power**

  How Powerful are Graph Neural Networks?

- Generalization

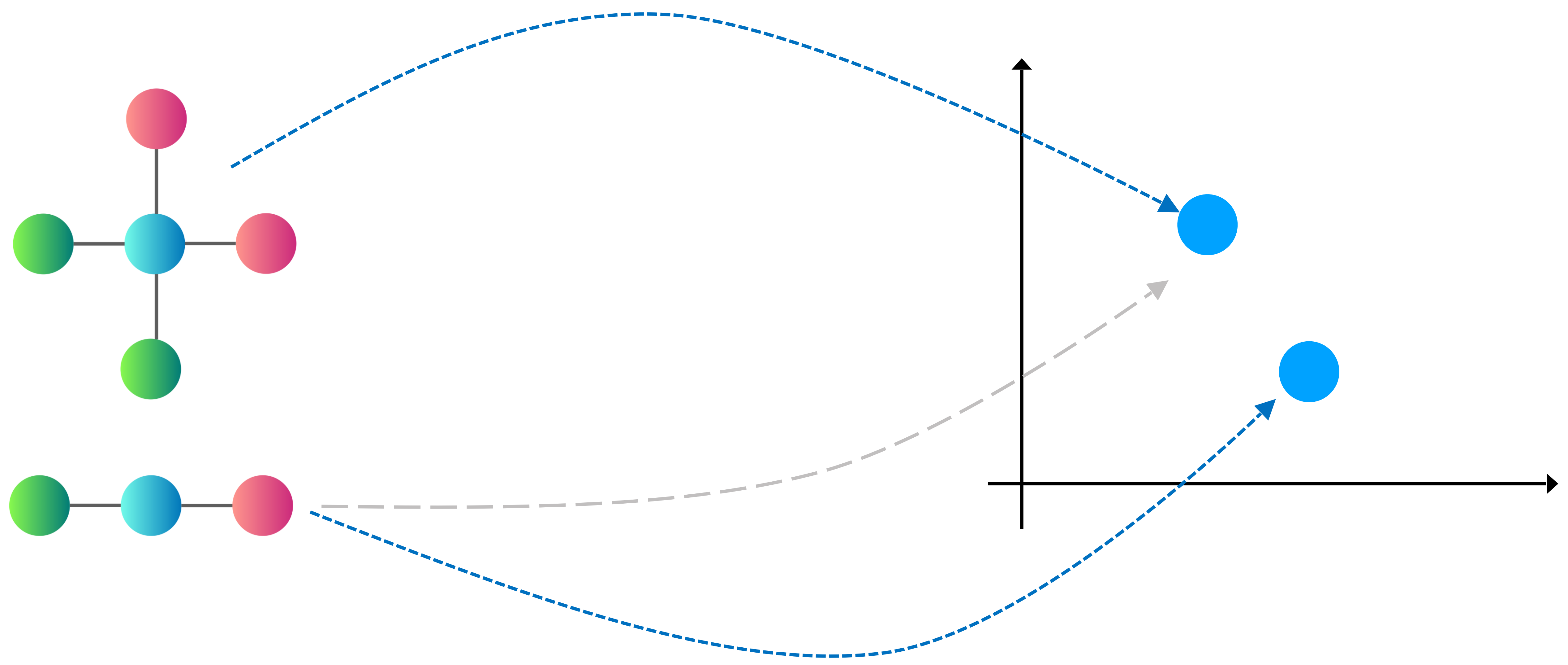
  Graph Neural Tangent Kernel

  What Can Neural Networks Reason About?

- Extrapolation

  How Neural Networks Extrapolate: From Feedforward to Graph Neural Networks

Expressive power

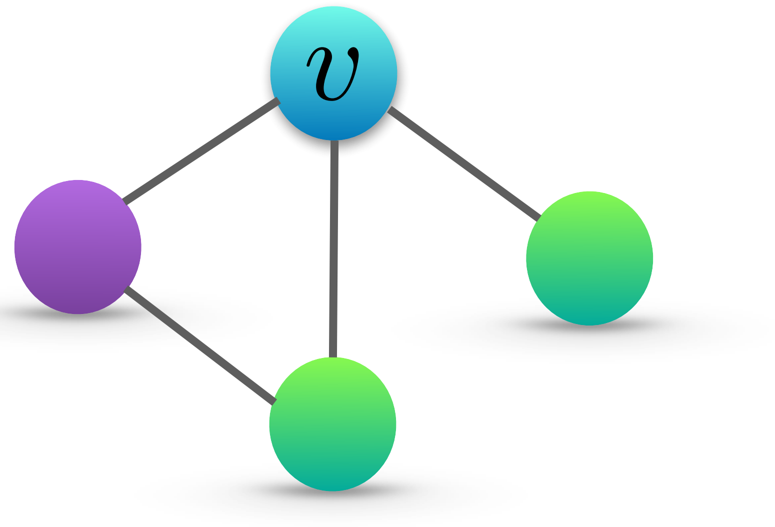


Which graphs can GNNs distinguish?

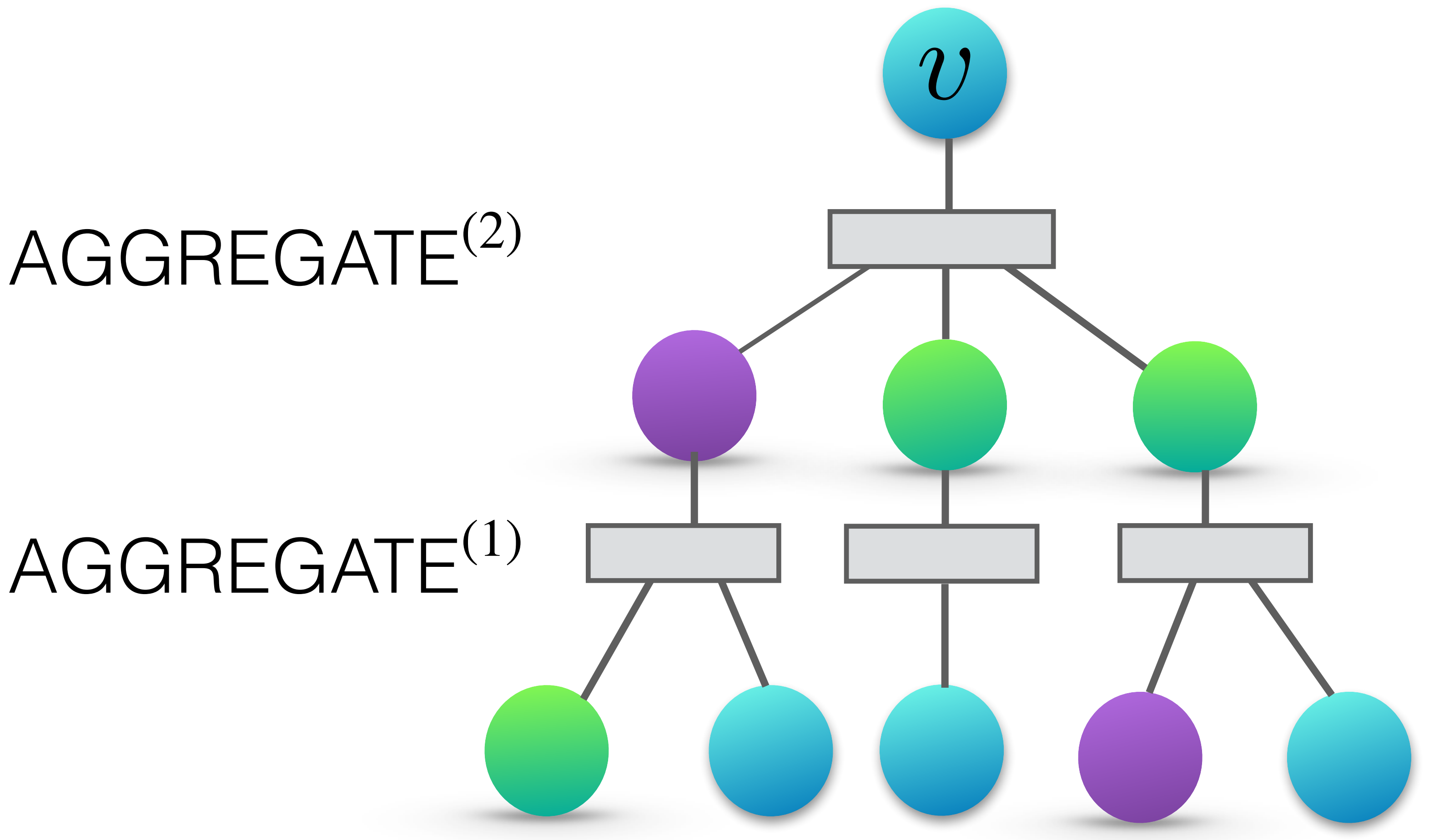
What does GNN discriminative power depend on?

Assume countable node input features

Intuition



Input graph



GNN computation graph

Observation I: Expressive power of a GNN depends on that of AGGREGATE

by a recursive argument

Observation II: Powerful GNNs have **injective** AGGREGATE

How powerful are GNNs?

Theorem (XHLJ’19)
GNNs are at most as powerful as a Weisfeiler-Lehman graph isomorphism test*.

This upper bound is achieved if AGGREGATE and READOUT are injective multiset functions.

neighborhood - multiset

**(Weisfeiler & Lehman 1968, Babai, Erdös, Selkow 1980, Babai & Kucera 1979, Cai, Furer, Immerman 1992, Evdokimov & Ponomarenko 1999, Douglas 2011)*

failure cases: certain regular graphs

A maximally powerful GNN

Lemma (XHLJ'19)
Any (injective) multi-set function g can be decomposed as

$$g(X) = \phi\left(\sum_{x \in X} f(x)\right)$$

(generalizing Zaheer et al 2017, Ravanbakhsh et al 2016, Qi et al 2017,…)

- Graph Isomorphism Network (GIN): **sum & universal approximator MLP**

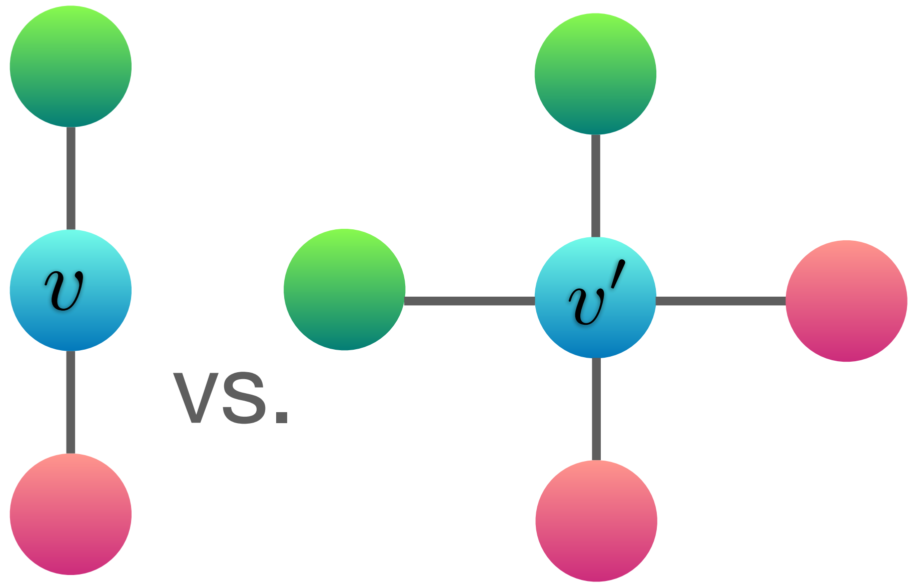
$$h_v^{(k)} = \mathrm{MLP}^{(k)}\left(\left(1 + \epsilon^{(k)}\right) \cdot h_v^{(k-1)} + \sum_{u \in \mathcal{N}(v)} h_u^{(k-1)}\right)$$

Less powerful GNNs

$$g\left(X\right) = \phi\left(\sum_{x \in X} f(x)\right)$$

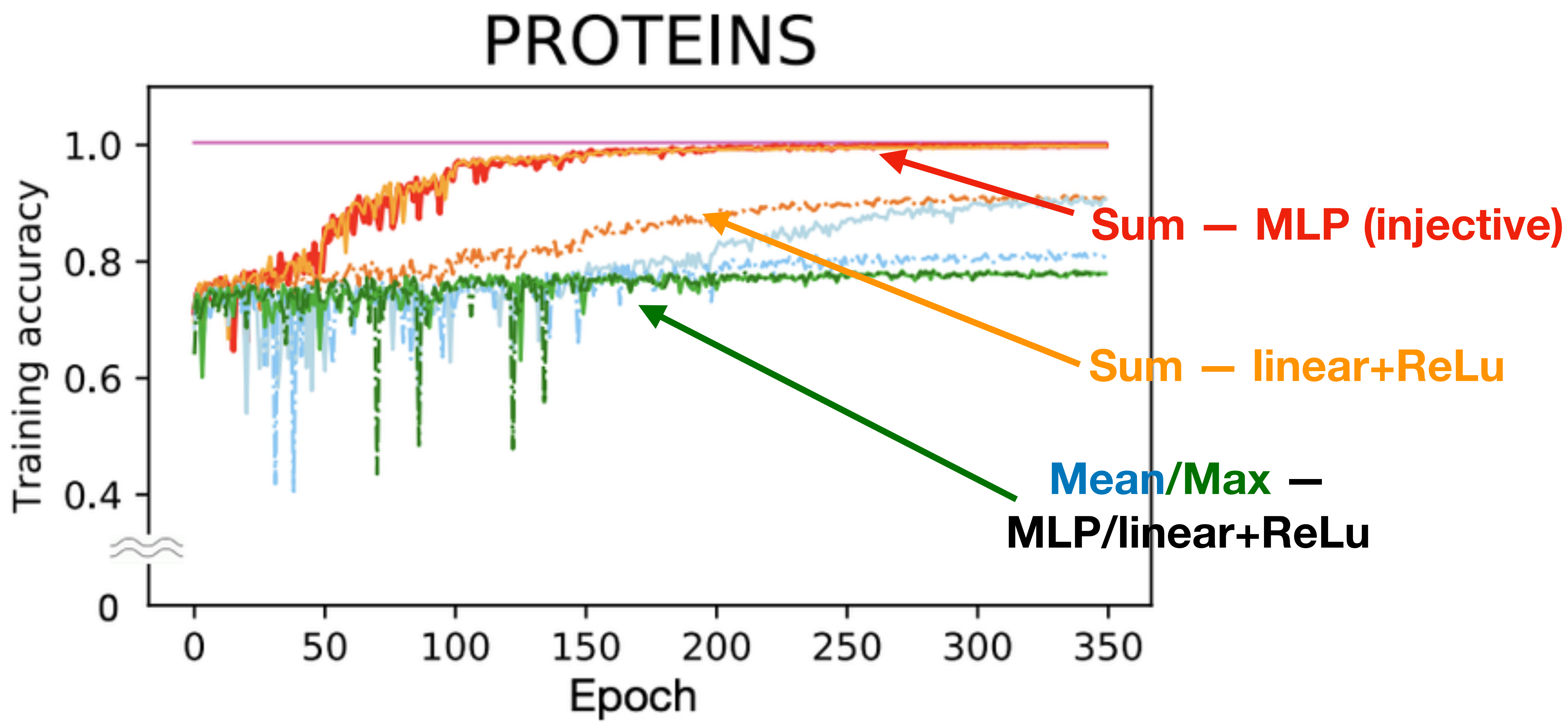
MEAN: $g(X) = \phi\left(\mathrm{MEAN}\{f(x) : x \in X\}\right)$

MAX: $g(X) = \phi\left(\mathrm{MAX}\{f(x) : x \in X\}\right)$ *not injective*



Linear+ReLU vs. MLP for ϕ

not universal approximator



Training Accuracy

Selected related & follow-up work

Pursuing more power

1. Consider higher-order structure and tensors

(Kondor et al. 2018, Keriven et al. 2019, Maron et al. 2019, Morris et al. 2019, Murphy et al. 2019)

2. Add auxiliary node identifitation

(Sato et al. 2019, 2020; Vignac et al. 2020)

3. Incorporate domain-specific structure & features

(Barceló et al. 2020, Bouritsas et al. 2020, Corso et al. 2020, Klicpera et al. 2020, Zhang et al. 2020)

Approximation

1. Equivalence of graph isomorphism test and function approximation, lower bound on width, counting substructures

(Scarselli et al. 2009, Chen et al. 2019, 2020; Garg et al. 2020, Loukas et al. 2020ab)

Roadmap

- Expressive power

  How Powerful are Graph Neural Networks?

- **Generalization**

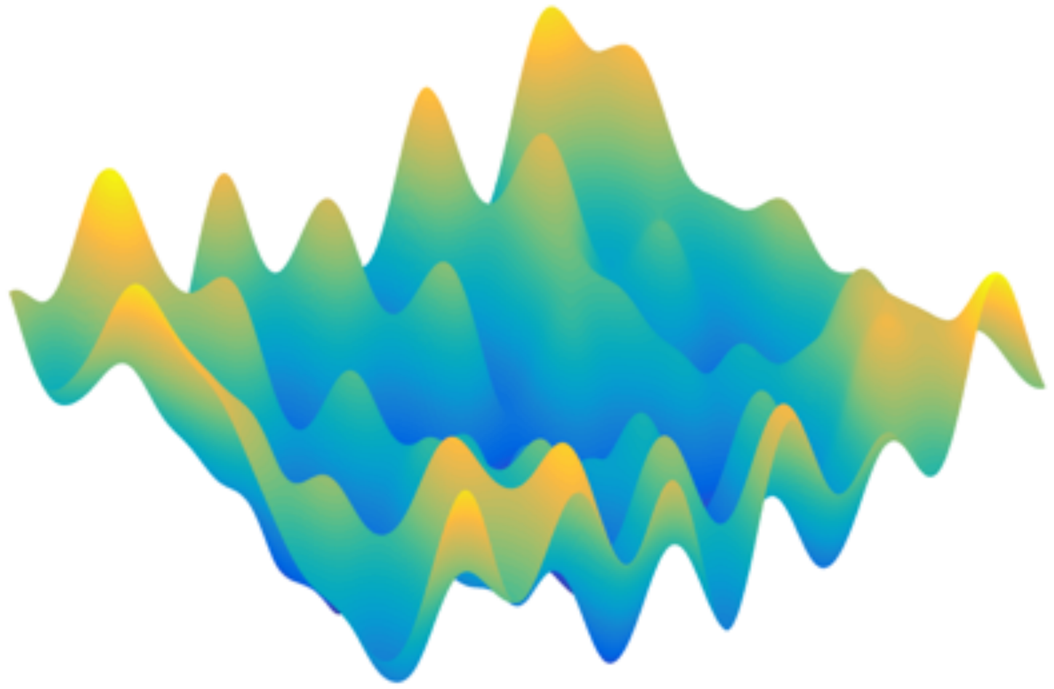
  Graph Neural Tangent Kernel

  What Can Neural Networks Reason About?

- Extrapolation

  How Neural Networks Extrapolate: From Feedforward to Graph Neural Networks

Generalization



non-convex landscape

GNN with sufficient expressive power

▶ There *exists* a GNN parameter that fits all data*

** except graphs that GNNs cannot distinguish*

Q1 Can gradient descent find such a parameter, i.e., **global minima**? ✓

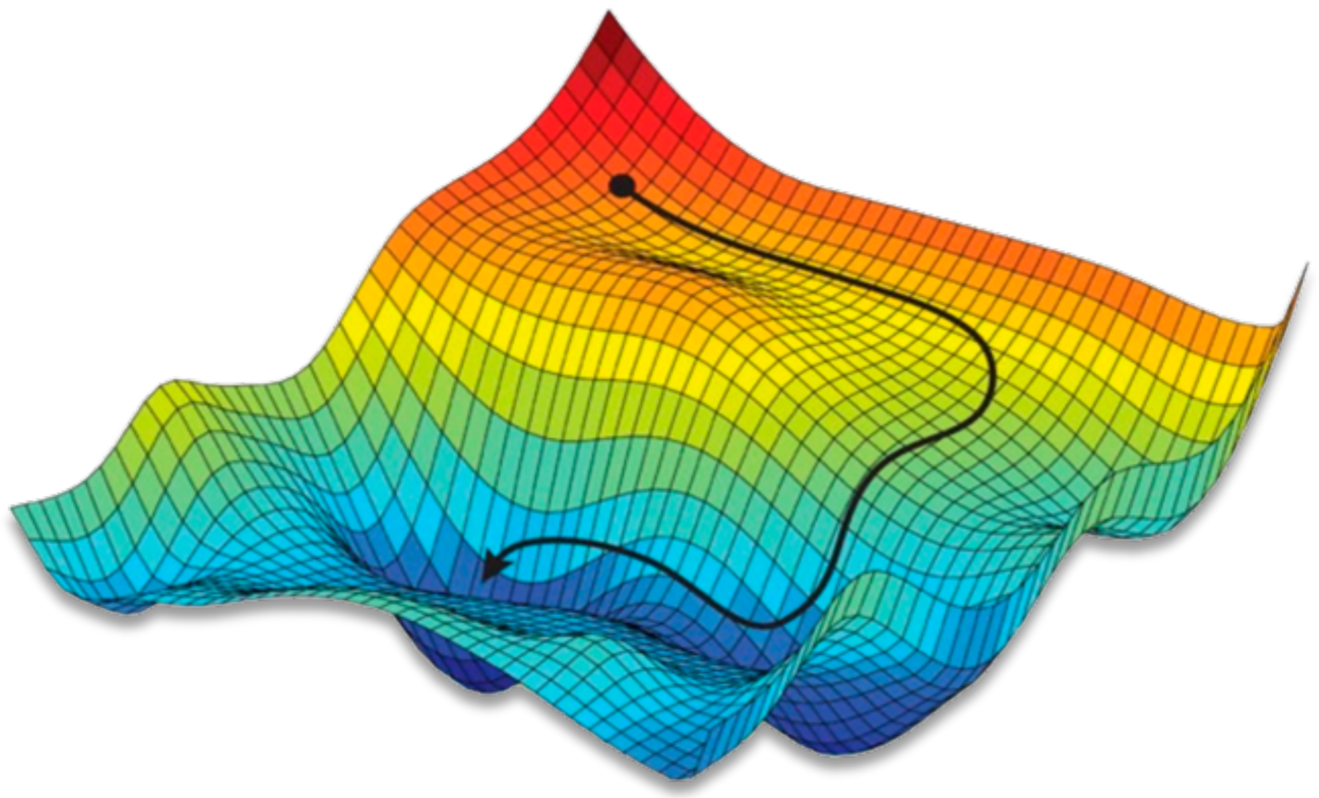
Q2 Why do GNNs trained by GD **generalize** despite high complexity? ✓

Graph Neural Tangent Kernel. DHPSWX'19

Q3 What **tasks** can GNNs generalize well in? How to **design** architectures? ✓

What Can Neural Networks Reason About? XLZDKJ'20

Graph Neural Tangent Kernel



Parameter trajectory

$\theta_{GNN}(t)$

GNN output $f(\theta_{GNN}, G)$

DHPSWX'19

Over-parameterized GNNs trained by GD is equivalent to that of kernel regression with Graph NTK:

$$k(G, G') = \mathbb{E}_{\theta_{GNN} \sim \mathcal{W}} \left[\left\langle \frac{\partial f(\theta_{GNN}, G)}{\partial \theta_{GNN}}, \frac{\partial f(\theta_{GNN}, G')}{\partial \theta_{GNN}} \right\rangle \right]$$

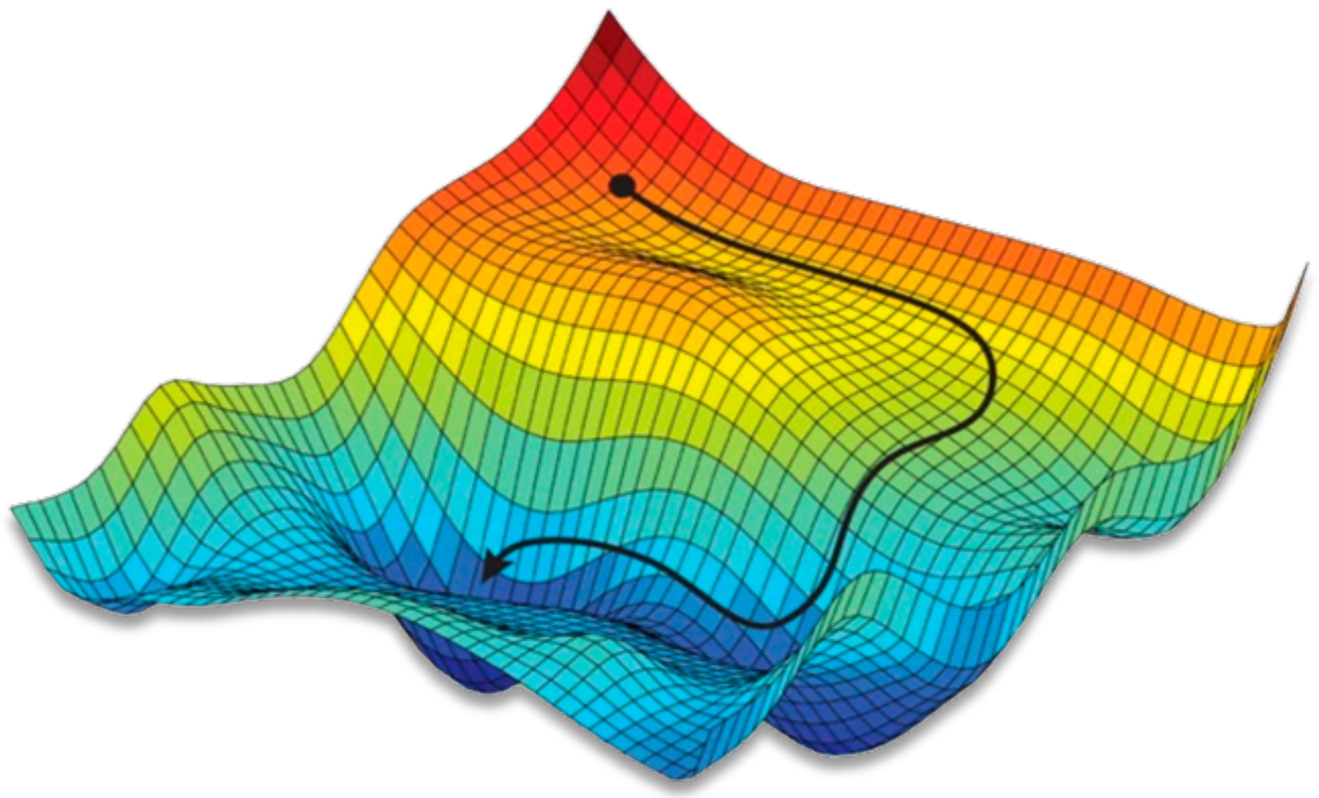
Refer to paper for exact formula of Graph NTK

Assumptions: very wide, infinitesimally small learning rate, initialization with certain scaling.

(NTK theory: Jacot et al 2018, Li and Liang 2018, Allen-Zhu et al 2019, Arora et al 2019ab, Cao and Gu 2019, Du et al 2019ab…)

Intuition of NTK

Introduced in Jacot et al 2018; concurrently developed by Li and Liang 2018, Allen-Zhu et al 2019, Arora et al 2019, Du et al 2019…



Parameter trajectory

$\theta(t)$

NN output $f(\theta, x)$

Network output $u(t) = (f(\theta(t), x_i))_{i=1}^n$

Dynamic follows $\dfrac{du}{dt} = -\boldsymbol{H}(t)(u(t) - y)$

$$\boldsymbol{H}(t)_{ij} = \left\langle \frac{\partial f(\theta(t), x_i)}{\partial \theta}, \frac{\partial f(\theta(t), x_j)}{\partial \theta} \right\rangle \text{ for } (i, j) \in [n] \times [n]$$

When width $\to \infty$, $\boldsymbol{H}(t) \approx \boldsymbol{H}(0)$ **a fixed kernel NTK**

Optimization & generalization error

Over-parameterized GNNs trained by GD = Graph NTK

Q1 Can GD find a global minimum for GNN?

Yes, Graph NTK is convex

Q2 Why do GNNs trained by GD generalize despite high complexity?

Generalization bound $$\frac{\sqrt{\boldsymbol{y}^\top \boldsymbol{H}^{-1} \boldsymbol{y} \cdot \operatorname{tr}(\boldsymbol{H})}}{n}.$$

(Bartlett and Mendelson 2002)

$\boldsymbol{H}$ - *graph NTK matrix; we provide an analytical form*

$\boldsymbol{y}$ - *training labels*

n - *number of training data*

Approaches of generalization analysis

	Predict Performance	Explanation	Assumptions	Examples
Norm based	✗	Norm *unknown* before training	Less	*Scarselli et al 2018, Garg et al 2020…*
Trajectory based (GNTK*)	✓	Fine-grained analysis of *simple* functions	Medium	DHPSWX'19
▶ Inductive biases (algo alignment)	✓	Structured functions e.g., *algorithms*	Medium+	XLZDKJ'20

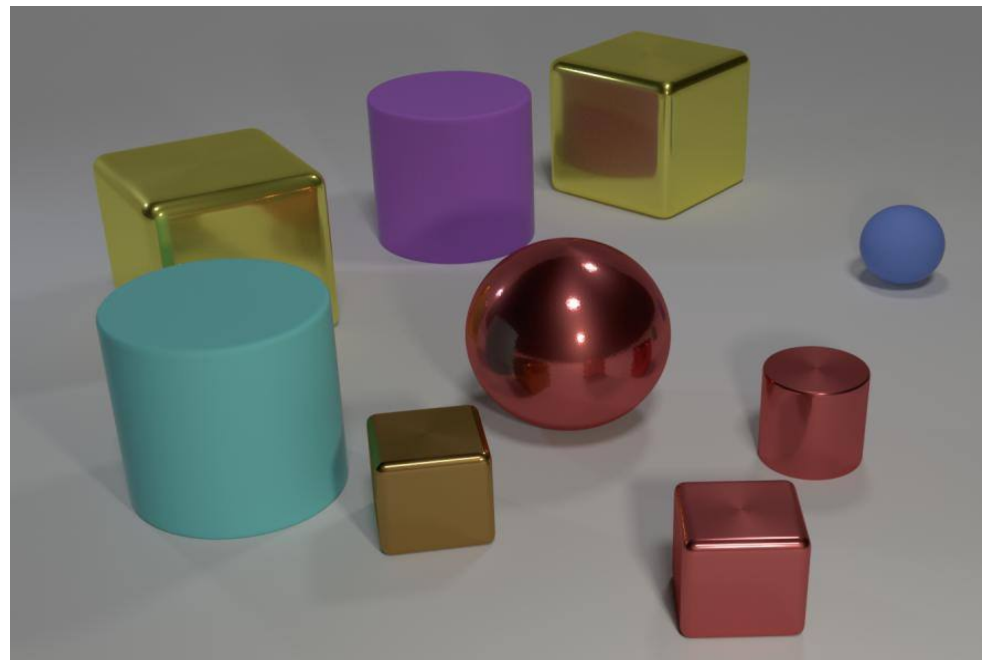
(Other trajectory based regimes for non-GNN: Chizat and Bach 2018, Mei et al 2018…)

Reasoning and perception

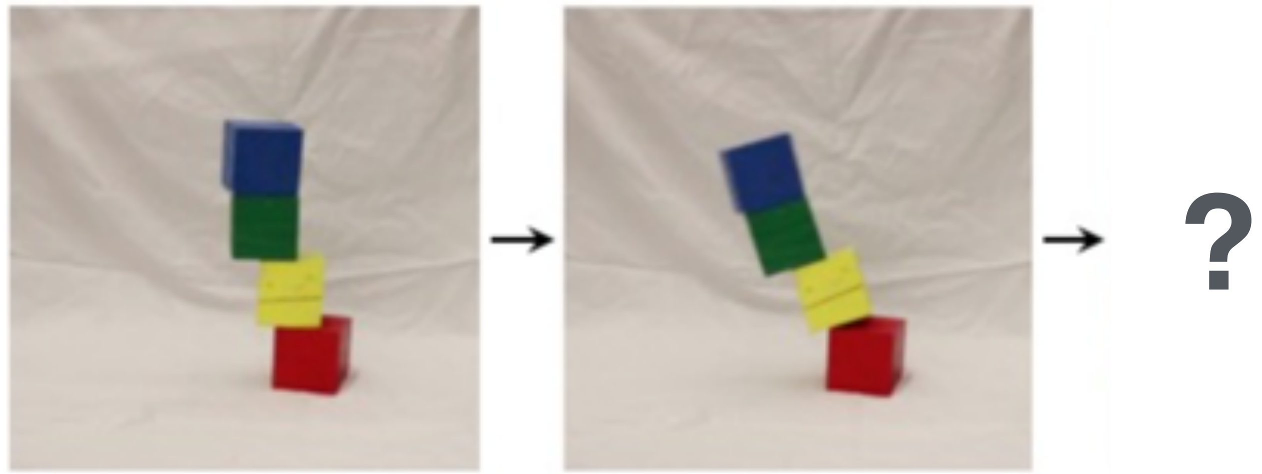
Color of her sweater?

Perception

Reasoning



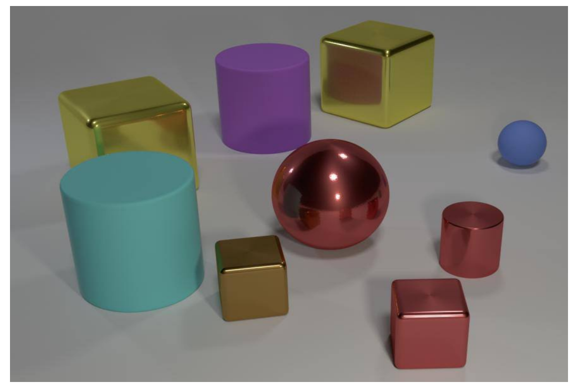
Furthest pair of objects?



Next position of the block?

Specialized architectures for learning different algorithms

a collection of
object features

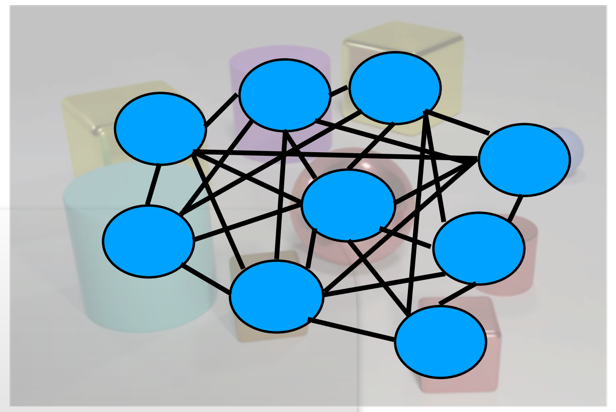


ConvNets

learns
perception algorithm

h_1
…
h_n

?



GNN
(or other model)

learns
reasoning algorithm

answer

(Weston et al., 2015; Johnson et al., 2017a; Wu et al. 2017, Fleuret et al., 2011; Antol et al., 2015; Battaglia et al., 2016, 2018; Watters et al., 2017; Fragkiadaki et al., 2016; Chang et al., 2017, 2019; Saxton et al., 2019; Santoro et al., 2018…)

Formalizing inductive bias of architectures

Algorithmic alignment (XLZDKJ'20)
Network can simulate algorithm via *few, easy-to-learn* "modules".

Claim: Better algo alignment implies better generalization.

Graph Neural Network

for k = 1 ... GNN iter:

for u in S:

$h_u^{(k)} = \Sigma_v \; \mathrm{MLP}(h_v^{(k-1)}, h_u^{(k-1)})$

No need to learn for-loops

Bellman-Ford algorithm

for k = 1 ... |S| - 1:

for u in S:

$d[k][u] = \min_v d[k-1][v] + \mathrm{cost}(v, u)$

Learns a simple reasoning step

Without good alignment -> need to learn complicated functions e.g., for-loop

Algo alignment measure

Algorithmic alignment (XLZDKJ'20)
A neural network (M, ϵ, δ)-aligns with an algorithm if it can simulate the algorithm via n weight-shared modules, each of which is (ϵ, δ) PAC-learnable with M/n samples.

$$\mathbb{P}_{x \sim \mathcal{D}} \left[\| \underbrace{f(x)}_{\text{learned function}} - \underbrace{g(x)}_{\text{true function (algorithm)}} \| \leq \epsilon \right] \geq 1 - \delta$$

(Valiant 1984)

* *Sample complexity of learning simple modules can be estimated via e.g., NTK (Arora et al. 2019)*

Better alignment implies better generalization

Theorem (XLZDKJ'20)
If a neural network and a task algorithm (M, ϵ, δ)-align, then, under assumptions*, the task is $(O(\epsilon), O(\delta))$ PAC-learnable by the network with M examples.

- * *Lipschitznes and SGD sequential training*
- * *Related work experimenting assumptions: Veličković et al 2020*

GNNs sample-efficiently learn dynamic programming

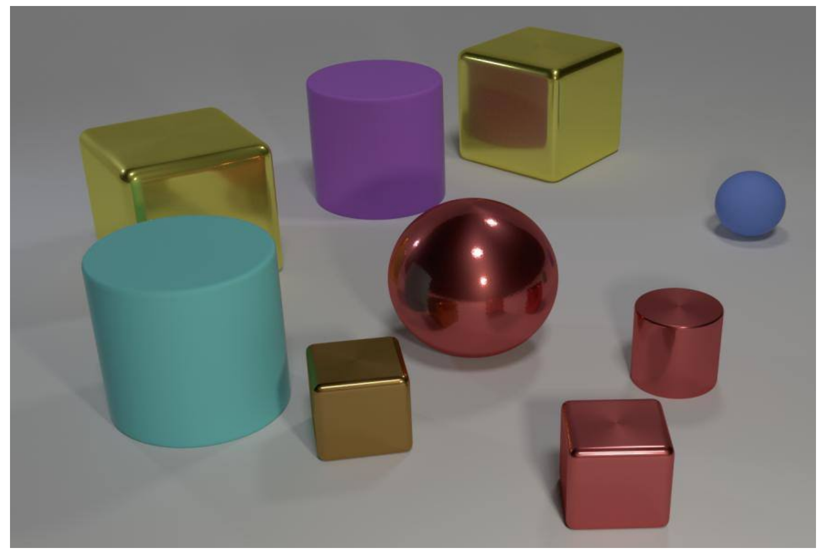
DP-Update: simple module easily learned by GNN's MLP modules

$$\text{Answer}[k][i] = \text{DP-Update}(\{\text{Answer}[k-1][j],\ j = 1 \ldots n\})$$

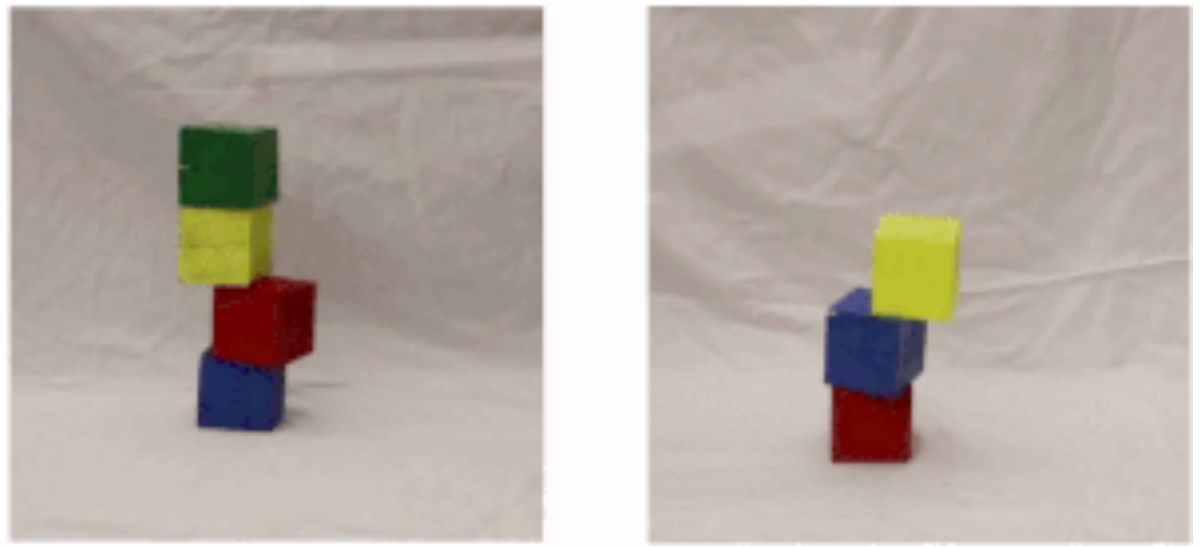
$$h_s^{(k)} = \sum\nolimits_{t \in S} \mathbf{MLP}_1^{(k)}\left(h_s^{(k-1)}, h_t^{(k-1)}\right)$$

Reasoning tasks as DP:

many graph algorithms



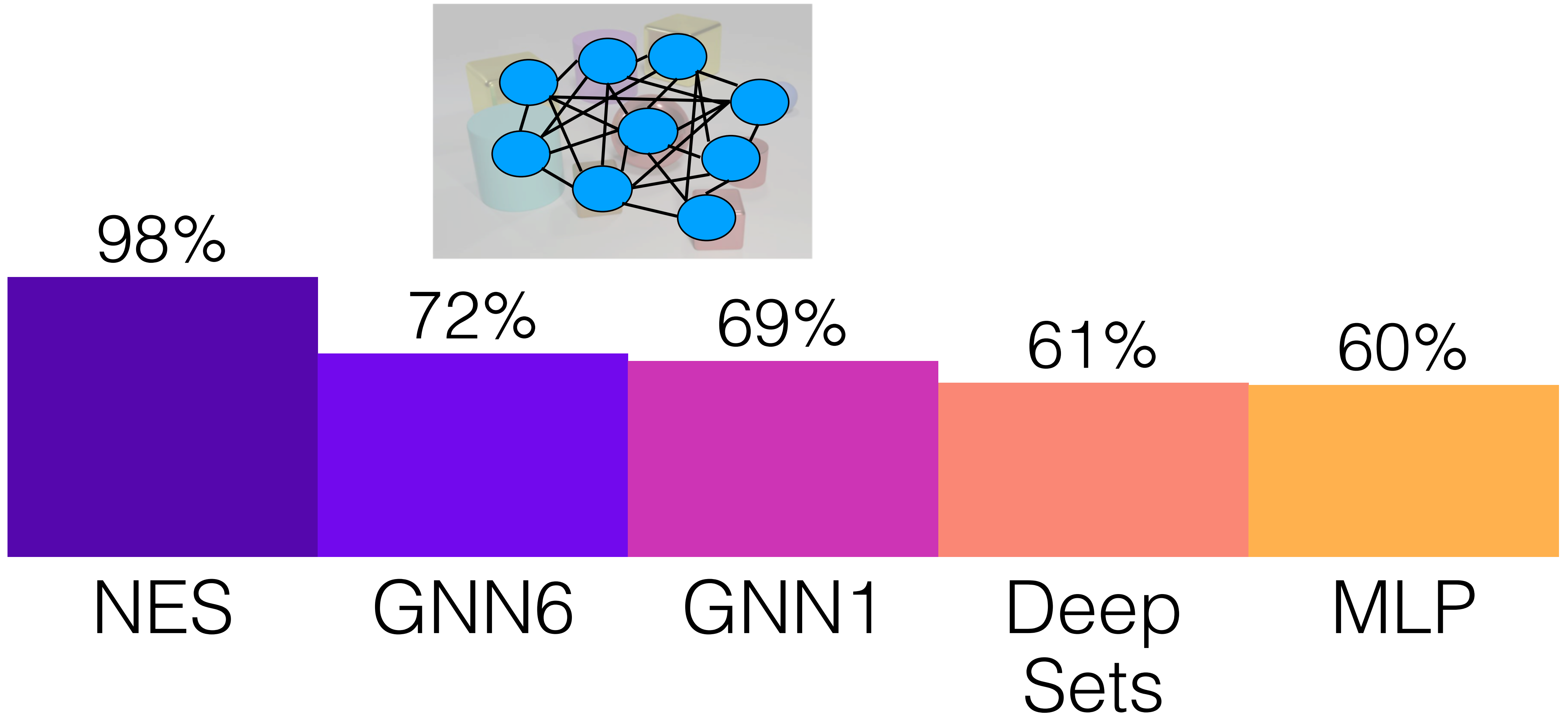
visual question answering



Intuitive physics

Limits of GNN: NP-hard problem

Subset sum: Can any subset of a set of numbers sum to zero?

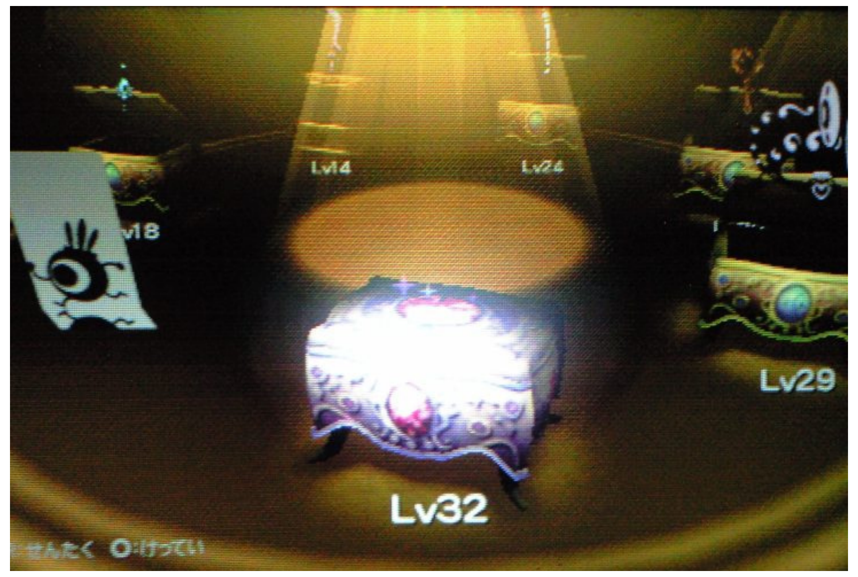


NES (Neural Exhaustive Search) - based on **algo alignment**

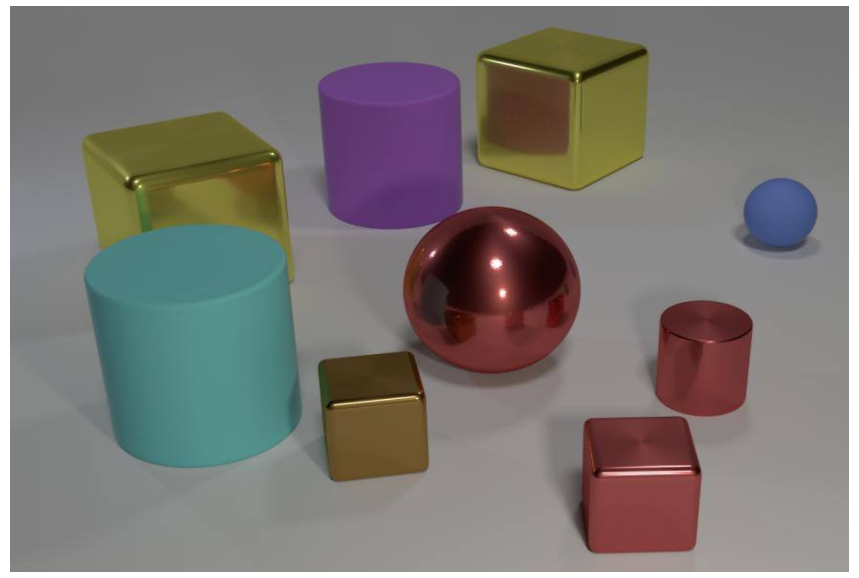
$$\mathrm{MLP}_2(\max_{\tau \subseteq S} \mathrm{MLP}_1 \circ \mathrm{LSTM}(X_1, ..., X_{|\tau|} : X_1, ..., X_{|\tau|} \in \tau))$$

$$y = \max_S 1[\, h(S) = 0 \,], \quad h(S) = \Sigma_{x \text{ in } S} \; X$$

A hierarchy of tasks

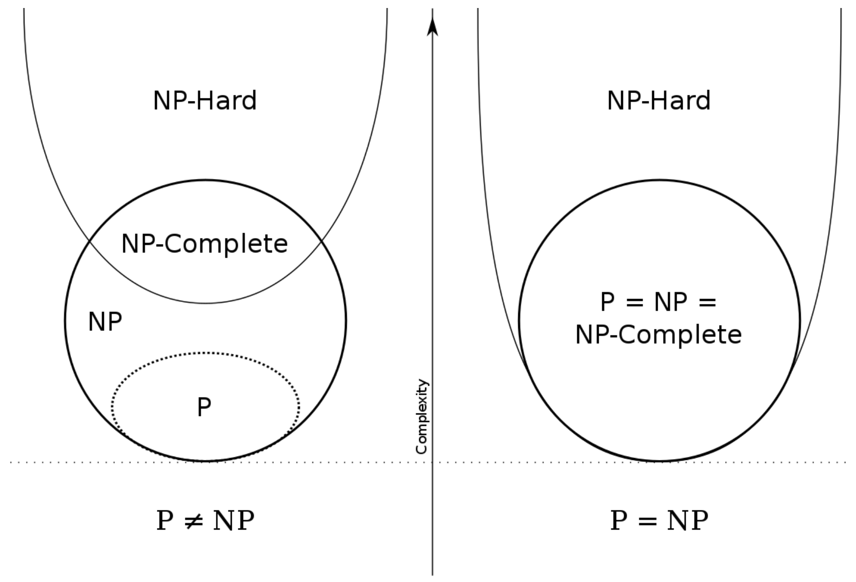


Summary statistics
What is the maximum value difference among treasures?



Relational argmax
What are the colors of the furthest pair of objects?

Dynamic programming
What is the cost to defeat monster X by following the optimal path?



NP-hard problem
Subset sum: Is there a subset that sums to 0?

	Summary statistics	Relational argmax	Dynamic programming	NP-hard problem
Graph Neural Network (GNN)	✓	✓	✓	✗
DeepSets *(Zaheer et al. 2017)*	✓	✗	✗	✗
MLP	✗	✗	✗	✗
Neural Exhaustive Search (NES)				✓

Roadmap

- Expressive power

  How Powerful are Graph Neural Networks?

- Generalization

  Graph Neural Tangent Kernel

  What Can Neural Networks Reason About?

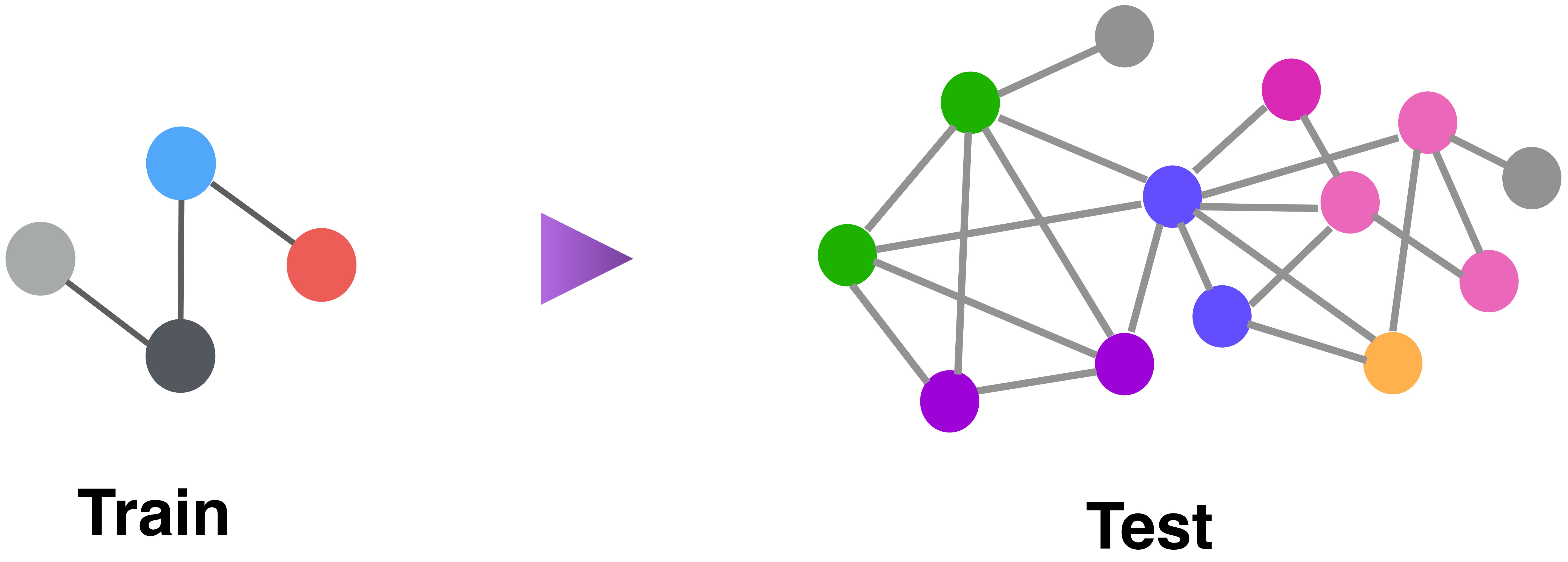
- **Extrapolation**

  How Neural Networks Extrapolate: From Feedforward to Graph Neural Networks

Extrapolation

What function does a neural network trained by GD implement outside the support of the training distribution?

** In-distribution generalization: train = test distribution*



Generalize across graph structure, size, node & edge features?

Puzzle

Prior works report **GNNs successfully extrapolate** to larger graphs

(Battaglia et al. 2016, 2018; Lample and Charton 2020, Velickovic et al., 2020 …)

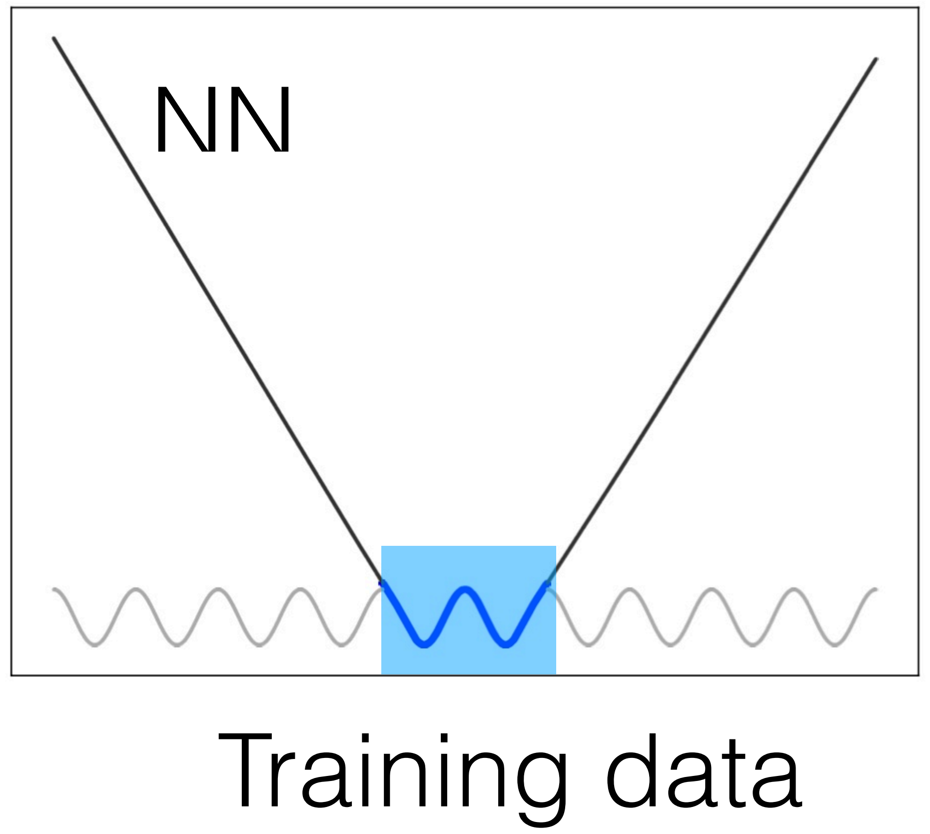
Prior works also report **MLPs** and ConvNets **fail out-of-distribution**

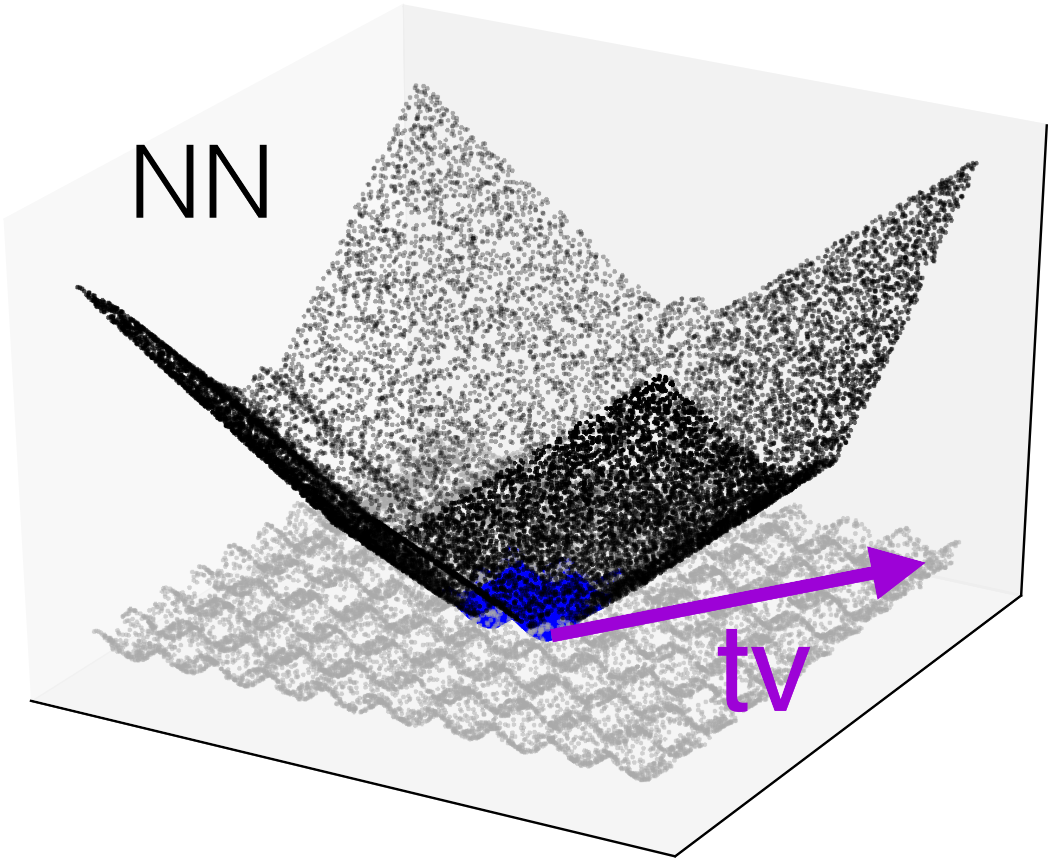
(Barnard and Wessels,1992; Haley and Soloway, 1992; Santoro et al. 2018; Arjovsky et al. 2019…)

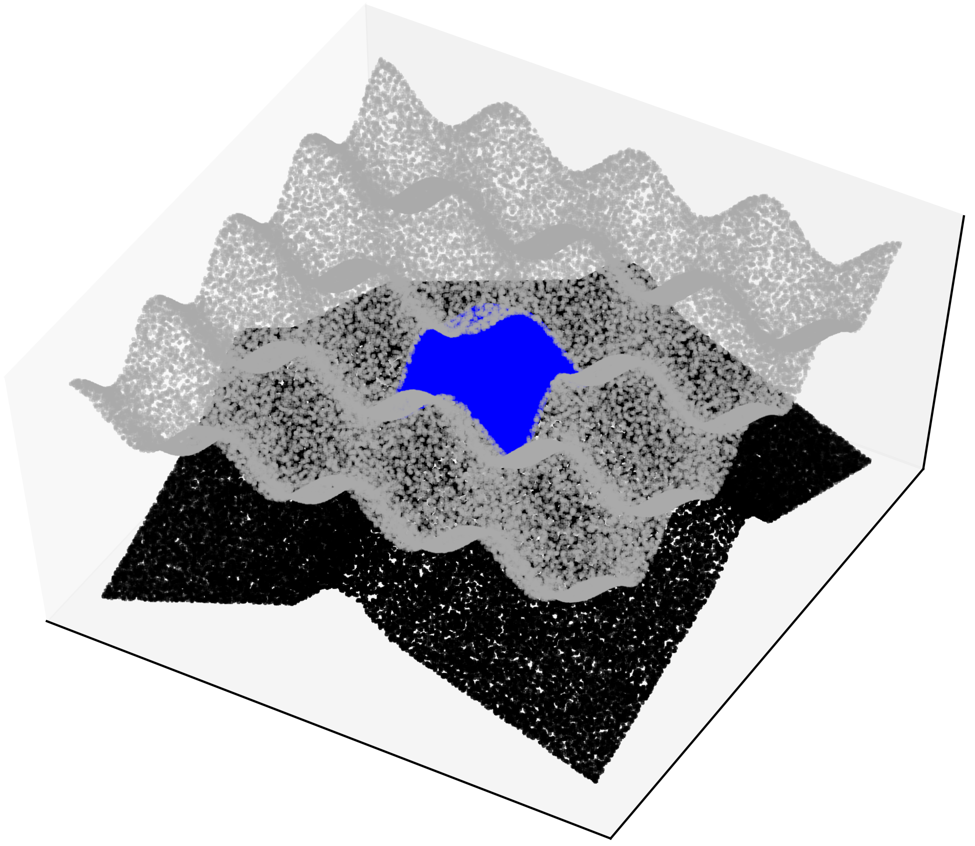
But GNNs have MLP modules…

$$\boldsymbol{h}_u^{(k)} = \sum_{v \in \mathcal{N}(u)} \mathrm{MLP}^{(k)}\Big(\boldsymbol{h}_u^{(k-1)}, \boldsymbol{h}_v^{(k-1)}, \boldsymbol{w}_{(v,u)}\Big), \quad \boldsymbol{h}_G = \mathrm{MLP}^{(K+1)}\Big(\sum_{u \in G} \boldsymbol{h}_u^{(K)}\Big)$$

Linear extrapolation of ReLU MLPs







Theorem (XZLDKJ'20)
Let f be a two-layer ReLU MLP trained by GD*. For any direction $v \in \mathbb{R}^d$, let $x = tv$. For any $h > 0$, as $t \to \infty$, $f(x + hv) - f(x) \to \beta_v h$ with rate $O(1/t)$

* *Assumption: NTK regime*

Implications for GNNs

Shortest Path: $d[k][u] = \boxed{\min_{v \in \mathcal{N}(u)}} d[k-1][v] + \boldsymbol{w}(v,u)$

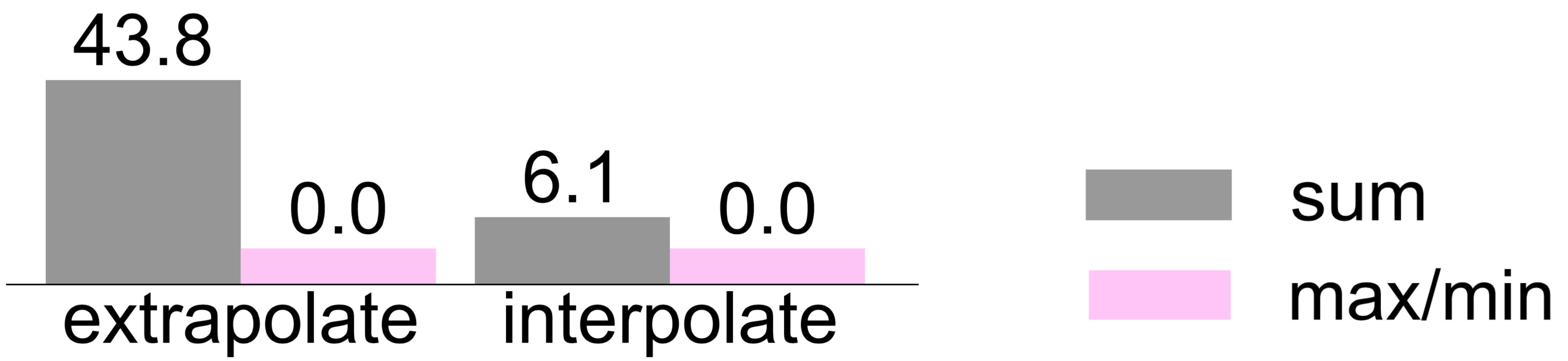
GNN (sum): $\boldsymbol{h}_u^{(k)} = \boxed{\sum_{v \in \mathcal{N}(u)} \text{MLP}^{(k)}} \left(\boldsymbol{h}_u^{(k-1)}, \boldsymbol{h}_v^{(k-1)}, \boldsymbol{w}_{(v,u)}\right)$

Requires non-linear extrapolation

Battaglia et al 2018; Velickovic et al 2020 extrapolate with:

$$\boldsymbol{h}_u^{(k)} = \boxed{\min_{v \in \mathcal{N}(u)} \text{MLP}^{(k)}} \left(\boldsymbol{h}_u^{(k-1)}, \boldsymbol{h}_v^{(k-1)}, \boldsymbol{w}_{(v,u)}\right)$$

Need to extrapolate linear function



Encoding non-linearity in architecture and features

Linear algorithmic alignment (XZLDKJ'20)
Network can simulate algorithm via ~~*easy-to-learn*~~ *linear* "modules".

Claim: Linear algo alignment helps *extrapolation.*

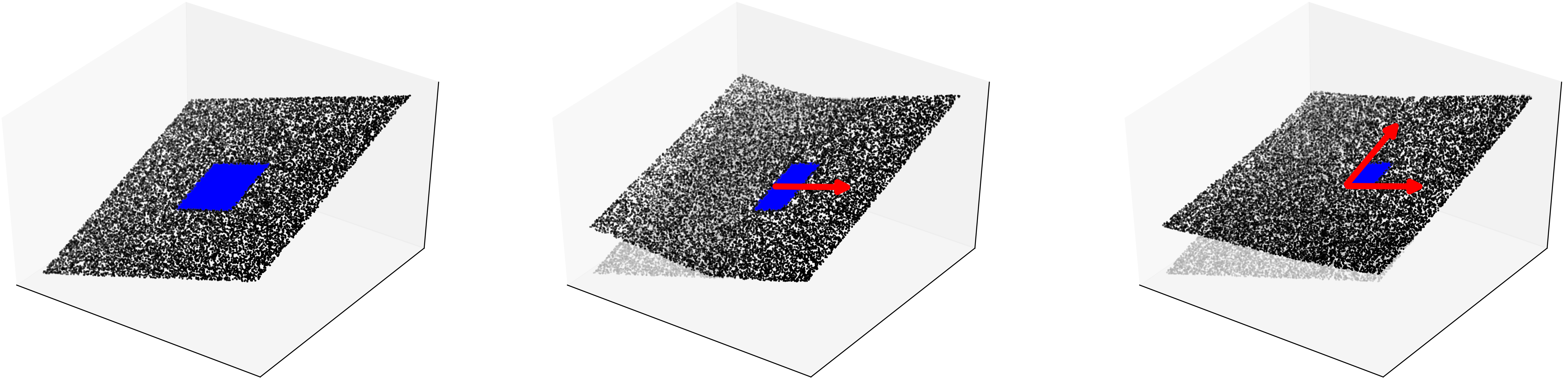
Architecture: symbolic modules, activation, architecture search

(Johnson et al 2017, Battaglia et al 2018, Yi et al 2018, Velickovic et al 2020)

Features: feature engineering, pre-training on out-of-distribution data (e.g., BERT)

(Devlin et al., 2019, Chen et al., 2020, Lample and Charton 2020, Hu et al 2020, Hendrycks et al 2020)

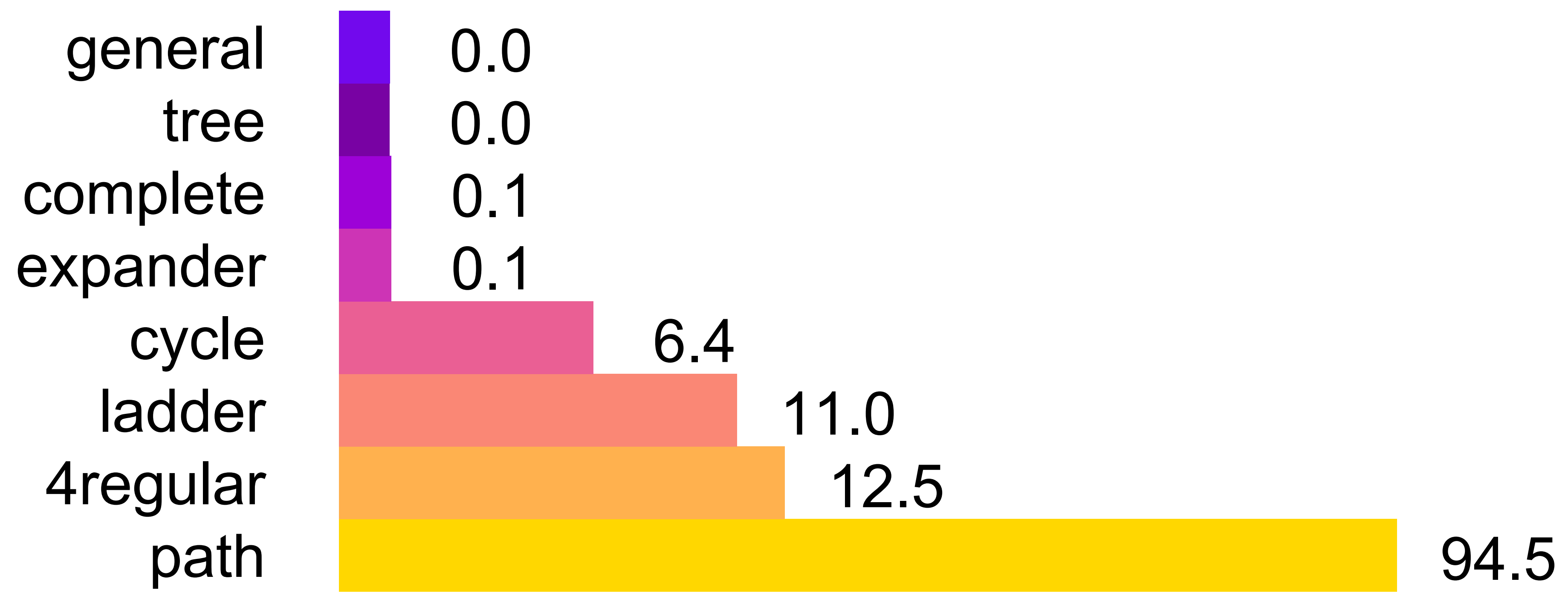
Requirement on geometry of training distribution



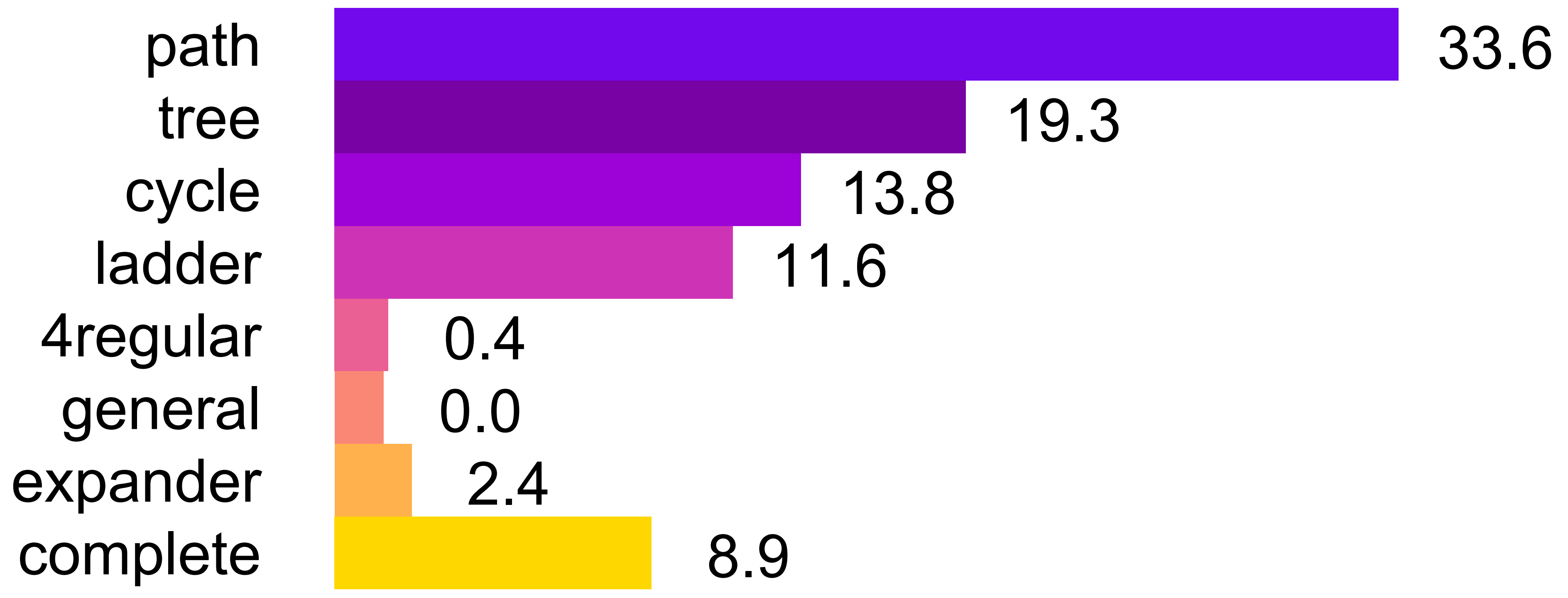
Theorem (XZLDKJ'20)
Let f be a two-layer ReLU MLP trained by GD*. Suppose target function is $\beta^\intercal x$ and support of training distribution covers all directions. As the number of training examples $n \to \infty$, $f(x) \to \beta^\intercal x$.

* *Assumption: NTK regime*

Requirement on training graph structure



Max Degree



Shortest Path

Proposition (XZLDKJ'20)

A max-aggregation GNN trained by GD* learns max degree if training data $\{\deg_{\max}(G_i), \deg_{\min}(G_i), N_i^{\max}\deg_{\max}(G_i), N_i^{\min}\deg_{\min}(G_i)\}_{i=1}^{n}$ spans $\mathbb{R}^4$.

* *Assumption: NTK regime*

Summary

- **Expressive power:** How to build powerful GNNs
- **Generalization:** Trajectory analysis, Inductive bias of architectures
- **Extrapolation:** Non-linearities matter

How Powerful are Graph Neural Networks? K. Xu, W. Hu, J. Leskovec and S. Jegelka. ICLR 2019.

Graph Neural Tangent Kernel: Fusing Graph Neural Networks with Graph Kernels. S. S. Du, K. Hou, B. Poczos, R. Salakhutdinov, R. Wang, K. Xu. NeurIPS 2019.

What Can Neural Networks Reason About? K. Xu, J. Li, M. Zhang, S. S. Du, K. Kawarabayashi, S. Jegelka. ICLR 2020.

How Neural Networks Extrapolate: From Feedforward to Graph Neural Networks. K. Xu, M. Zhang, J. Li, S. S. Du, K. Kawarabayashi, S. Jegelka. arXiv 2009.11848