

# What can neural networks reason about?

**Shashank Shekhar**

MLRG Talk, January 12, 2021

# Credits

Eric J Taylor for presentation layout

Parts of slides borrowed from [Anushree Hede](#)

# Key takeaways (spoiler alert!)

Published as a conference paper at ICLR 2020

---

## WHAT CAN NEURAL NETWORKS REASON ABOUT?

**Keyulu Xu<sup>†</sup>, Jingling Li<sup>‡</sup>, Mozhi Zhang<sup>‡</sup>, Simon S. Du<sup>§</sup>, Ken-ichi Kawarabayashi<sup>¶</sup>,  
Stefanie Jegelka<sup>†</sup>**

<sup>†</sup>Massachusetts Institute of Technology (MIT)

<sup>‡</sup>University of Maryland

<sup>§</sup>Institute for Advanced Study (IAS)

<sup>¶</sup>National Institute of Informatics (NII)

{keyulu, stefje}@mit.edu

\* Theoretical paper but I will be talking at a high level about the key idea proposed and its implications

# Key takeaways

- Reasoning  $\leq$  Algorithmic structure
- Neural Network's ability to learn to reason : Architecture  $\Leftrightarrow$  Structure
- **Intuition:** As architecture's alignment with the algorithm increases, the network itself has to learn simple functions (and not the whole algorithm)
- Theoretical measure: ALGORITHMIC ALIGNMENT
- Empirical usefulness:  
**Reasoning process:** summary statistics, relational argmax, dynamic programming  
**NN architectures:** MLP, Deep Sets, GNN

# Presentation outline

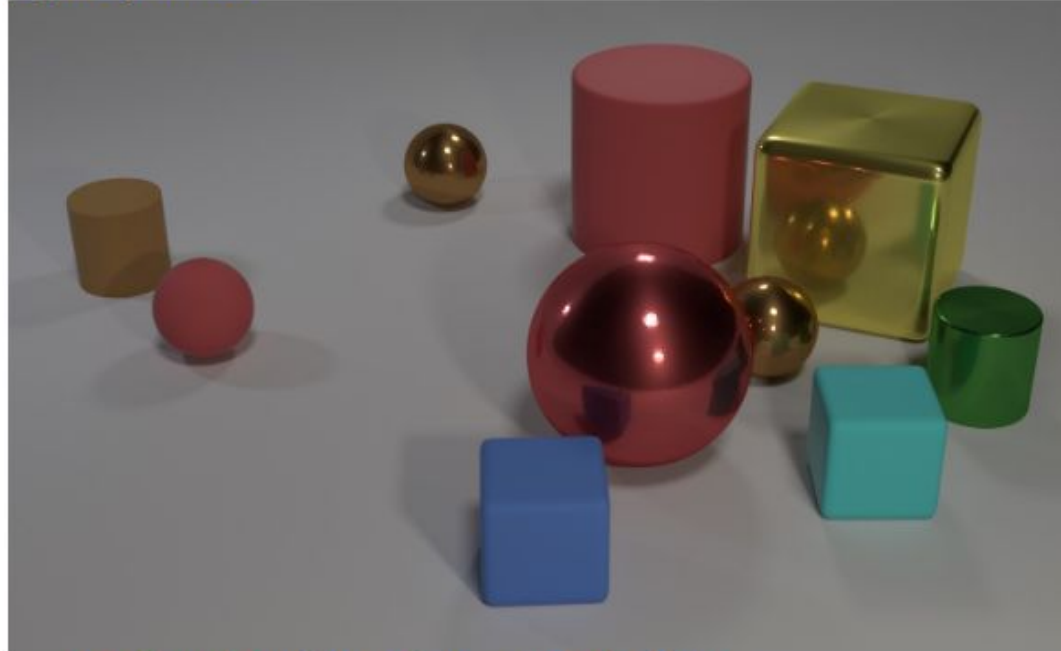
1. Reasoning and algorithms
2. Neural Network structure
3. Algorithmic Alignment
4. Results

# Presentation outline

1. Reasoning and algorithms
2. Neural Network structure
3. Algorithmic Alignment
4. Results

# Reasoning

Questions in CLEVR test various aspects of visual reasoning including **attribute identification**, **counting**, **comparison**, **spatial relationships**, and **logical operations**.



Q: Are there an **equal number** of **large things** and **metal spheres**?

Q: **What size** is the **cylinder that is left of** the **brown metal** thing **that is left of** the **big sphere**?

Q: There is a **sphere** with the **same size as** the **metal cube**; is it **made of the same material as** the **small red sphere**?

Q: **How many** objects are **either small cylinders** or **red** things?

# Reasoning: Problem Formalization

- Universe  $S$  = set of objects to reason about
- Each object  $s \in S$  is represented by a feature vector  $X = [h_1, h_2, \dots, h_k]$
- Given a set of universes  $\{S_1, S_2, \dots, S_m\}$  and answer labels  $y = \{y_1, y_2, \dots, y_m\}$

learn a function which can answer questions about unseen universe  $y = g(S)$

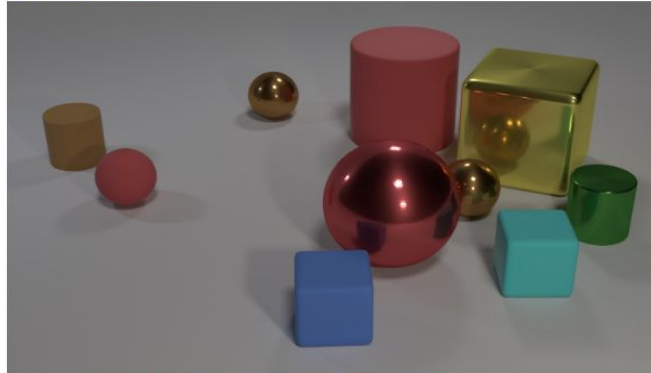


# Reasoning: Problem Formalization

- VQA
  - **Universe:** Images/Questions
  - **Objects:** Objects in the image/question
  - **Answers:** Answer
- Shortest Path
  - **Universe:** Graph
  - **Objects:** Nodes/Edges
  - **Answers:** Shortest path

# Reasoning: summary statistics

Questions in CLEVR test various aspects of visual reasoning including **attribute identification**, **counting**, **comparison**, **spatial relationships**, and **logical operations**.



Q: Are there an **equal number** of **large things** and **metal spheres**?

Q: **What size** is the **cylinder that is left of** the **brown metal** thing **that is left of** the **big sphere**?

Q: There is a **sphere** with the **same size as** the **metal cube**; is it **made of the same material as** the **small red sphere**?

Q: **How many** objects are **either small cylinders or red things**?

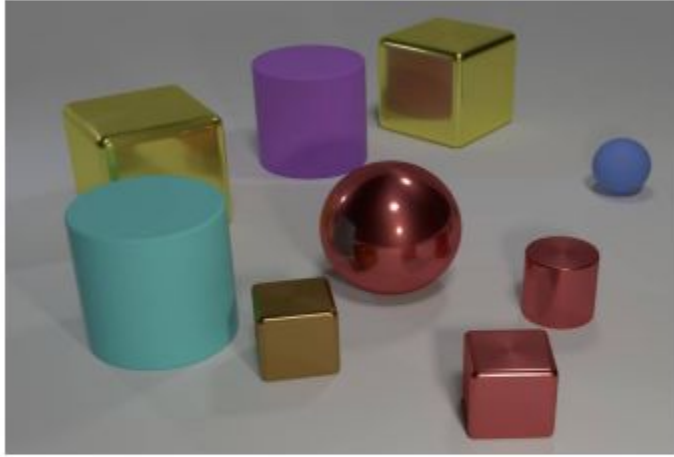


*Summary statistics*

What is the maximum value difference among treasures?

Max/Min/Sum etc of (features of) all objects

# Reasoning: relational argmax



Compare pairwise relations between objects and answer a question about those pairwise results

*Relational argmax*

What are the colors of the furthest pair of objects?

# Reasoning: dynamic programming



## *Dynamic programming*

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

Several relational reasoning tasks can  
be solved using a dynamic  
programming algorithm

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

e.g. Shortest path can be solved using Bellman-Ford

# Reasoning: dynamic programming

**VQA:** “Starting at object X, if each time we jump to the closest object, which object is K jumps away?” (sort-of-CLEVR dataset)

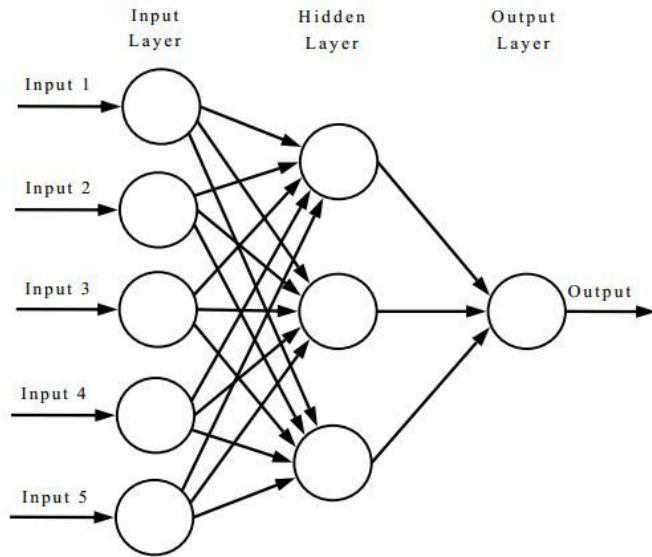
$\text{closest}[1][i] = \arg \min_j d(i, j), \text{closest}[k][i] = \text{closest}[k - 1][\text{closest}[1][i]]$

**Intuitive physics:** Authors also show how moving objects and force interactions, which are another popular area of AI reasoning research, can be modelled as dynamic programming updates (refer paper)

# Presentation outline

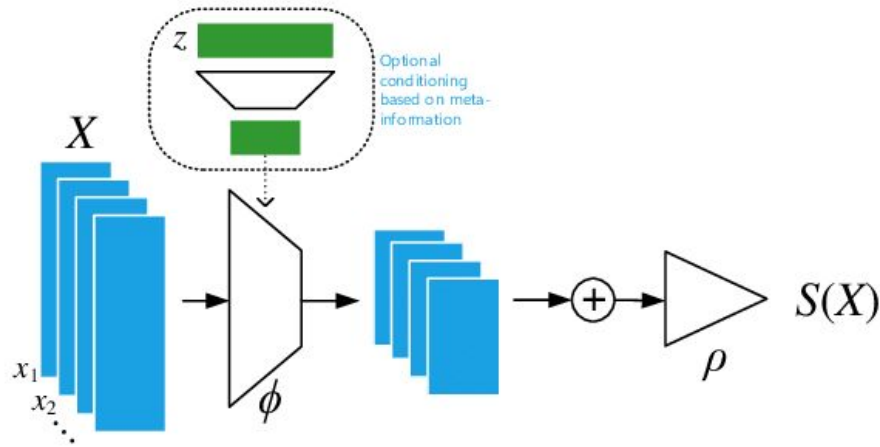
1. Reasoning and algorithms
2. Neural Network structure
3. Algorithmic Alignment
4. Results

# Network Structure: MLP



- No inherent relational structure
- Works well for single object universes (image classification)
- Has a hard time generalizing to multi-object universes if trained on concatenated object representations

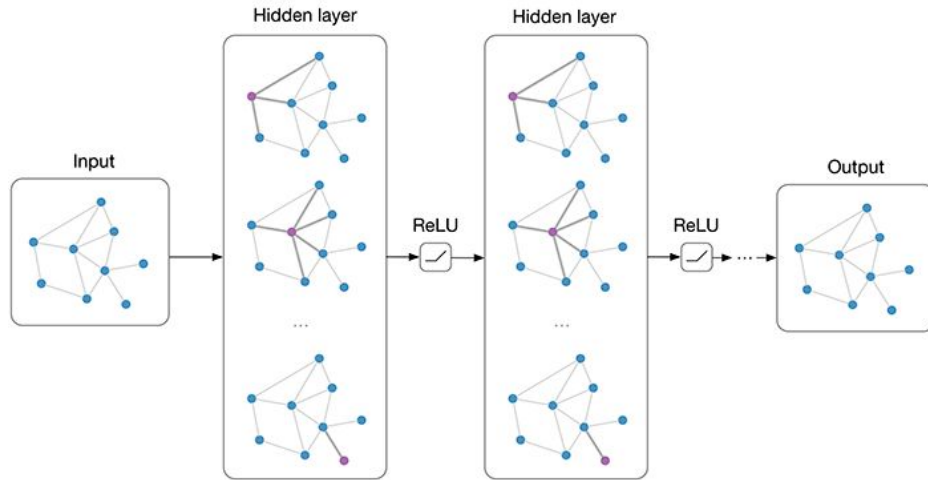
# Network Structure: Deep Sets



- $y = \text{MLP}_2 [\sum_{s \in S} \text{MLP}_1 (X_s)]$
- Can learn permutation invariant functions of objects
- Reasoning problems often require learning functions of unordered sets



# Network Structure: GNNs



- Message passing scheme where at iteration  $k$  the representation  $h_s^{(k)}$  is recursively updated by aggregating representations of neighbouring states

- $$h_s^{(k)} = \sum_{t \in S} \text{MLP}_1^{(k)} [h_s^{(k-1)}, h_t^{(k-1)}]$$

$$h_s = \text{MLP}_2(\sum_{s \in S} h_s^{(k-1)})$$

$$h_s = \text{output}, K = \# \text{layers}, h_s^{(0)} = X_s$$

- Also permutation invariant. GNNs can also focus on pairwise relations unlike deep sets

# Network structure

- Empirically,

$\text{GNN} > \text{Deep Sets} > \text{MLP}$

- Theoretically,

$\text{GNN} \Leftrightarrow \text{Deep Sets} \Leftrightarrow \text{MLP}$

- Therefore,

$\text{Difference in accuracy} = f(\text{difference in generalization ability})$

# Presentation outline

1. Reasoning and algorithms
2. Neural Network structure
3. Algorithmic Alignment
4. Results

# Network structure and algorithms

- The inductive bias of *'the neural network's architecture induces a computational structure on the function it computes'*  
e.g. CNNs perform great on images because convolution filters are translation invariant for objects in the images
- Intuitively, a network may generalize better if its able to represent a function more 'easily'
- Battaglia et al already discussed this idea about GNNs being better at relation learning due to their structure (without formalization)

# Network structure and algorithms

## Bellman-Ford algorithm

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

```
  for u in S:
```

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

- e.g. Bellman-Ford algorithm outlines the correct reasoning process to solve a shortest path problem

# Network structure and algorithms

## Graph Neural Network

for  $k = 1 \dots \text{GNN iter:}$

for  $u$  in  $S$ :

*No need to learn for-loops*

$$h_u^{(k)} = \sum_v \text{MLP}(h_v^{(k-1)}, h_u^{(k-1)})$$

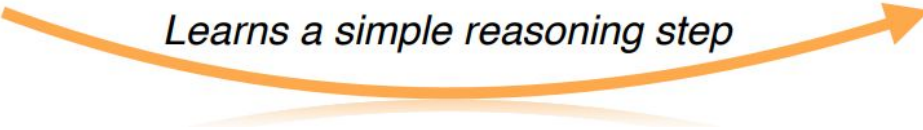
## Bellman-Ford algorithm

for  $k = 1 \dots |S| - 1$ :

for  $u$  in  $S$ :

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

*Learns a simple reasoning step*



- e.g. Bellman-Ford algorithm outlines the correct reasoning process to solve a shortest path problem
- GNN can simulate Bellman Ford if it's able to learn the relaxation step in the last line (sum->min over neighboring nodes  $v$ ) via its aggregation operation
- However, an MLP or Deep Set would have to learn the structure of the entire for loop

# PAC learning framework

**Definition 3.3. (PAC learning and sample complexity).** Fix an error parameter  $\epsilon > 0$  and failure probability  $\delta \in (0, 1)$ . Suppose  $\{x_i, y_i\}_{i=1}^M$  are i.i.d. samples from some distribution  $\mathcal{D}$ , and the data satisfies  $y_i = g(x_i)$  for some underlying function  $g$ . Let  $f = \mathcal{A}(\{x_i, y_i\}_{i=1}^M)$  be the function generated by a learning algorithm  $\mathcal{A}$ . Then  $g$  is  $(M, \epsilon, \delta)$ -*learnable* with  $\mathcal{A}$  if

$$\mathbb{P}_{x \sim \mathcal{D}} [\|f(x) - g(x)\| \leq \epsilon] \geq 1 - \delta. \quad (3.1)$$

The *sample complexity*  $\mathcal{C}_{\mathcal{A}}(g, \epsilon, \delta)$  is the minimum  $M$  so that  $g$  is  $(M, \epsilon, \delta)$ -learnable with  $\mathcal{A}$ .

# Algorithmic Alignment

**Definition 3.4. (Algorithmic alignment).** Let  $g$  be a reasoning function and  $\mathcal{N}$  a neural network with  $n$  modules  $\mathcal{N}_i$ . The module functions  $f_1, \dots, f_n$  generate  $g$  for  $\mathcal{N}$  if, by replacing  $\mathcal{N}_i$  with  $f_i$ , the network  $\mathcal{N}$  simulates  $g$ . Then  $\mathcal{N}$   $(M, \epsilon, \delta)$ -algorithmically aligns with  $g$  if (1)  $f_1, \dots, f_n$  generate  $g$  and (2) there are learning algorithms  $\mathcal{A}_i$  for the  $\mathcal{N}_i$ 's such that  $n \cdot \max_i C_{\mathcal{A}_i}(f_i, \epsilon, \delta) \leq M$ .



# Algorithmic Alignment

Put more simply, a neural network aligns with an algorithm if

1. It can simulate it via a limited number of modules ( $n$ )
2. Each module is simple i.e. has low sample complexity ( $M/n$ )

*'Good algorithmic alignment, i.e., small  $M$ , implies that all algorithm steps  $f_i$  to simulate the algorithm  $g$  are easy to learn.'*

# Measuring Algorithmic Alignment

**Theorem 3.5. (Sample complexity for overparameterized MLP modules).** Let  $\mathcal{A}$  be an overparameterized and randomly initialized two-layer MLP trained with gradient descent for a sufficient number of iterations. Suppose  $g : \mathbb{R}^d \rightarrow \mathbb{R}^m$  with components  $g(x)^{(i)} = \sum_j \alpha_j^{(i)} (\beta_j^{(i)\top} x)^{p_j^{(i)}}$ , where  $\beta_j^{(i)} \in \mathbb{R}^d$ ,  $\alpha \in \mathbb{R}$ , and  $p_j^{(i)} = 1$  or  $p_j^{(i)} = 2l$  ( $l \in \mathbb{N}_+$ ). The sample complexity  $\mathcal{C}_{\mathcal{A}}(g, \epsilon, \delta)$  is

$$\mathcal{C}_{\mathcal{A}}(g, \epsilon, \delta) = O\left(\frac{\max_i \sum_{j=1}^K p_j^{(i)} |\alpha_j^{(i)}| \cdot \|\beta_j^{(i)}\|_2^{p_j^{(i)}} + \log(m/\delta)}{(\epsilon/m)^2}\right). \quad (3.2)$$

- Functions that are “simple” when expressed as a polynomial (e.g. via a Taylor expansion) can be sample-efficiently learned by an MLP.
- Complexity increases with #number of objects in the universe as  $\|\mathbf{B}_j\|$  increases and also #modules required to represent  $g(x)$  as  $K$  increases
- Algorithm steps which perform computations over many objects (e.g. for loops) lead to higher sample complexity for MLPs

# Presentation outline

1. Reasoning and algorithms
2. Neural Network structure
3. Algorithmic Alignment
4. Results

# Theoretical result

**Corollary 3.7.** *Suppose universe  $S$  has  $\ell$  objects  $X_1, \dots, X_\ell$ , and  $g(S) = \sum_{i,j} (X_i - X_j)^2$ . In the setting of Theorem 3.6, the sample complexity bound for MLP is  $O(\ell^2)$  times larger than for GNN.*

Theoretically, for a pairwise relation learning task (sum of pairwise squared differences), the sample complexity bound for MLP is  $O(\text{objects}^2)$  times larger than for GNN

# Empirical Settings

- **Summary Statistics**

- **Maximum value difference** : Each object is a treasure  $X = [h_1, h_2, h_3]$  with location  $h_1$ , value  $h_2$  and color  $h_3$ . Models have to predict difference in value between most and least valuable treasure

$$y(S) = \max_{s \in S} h_2(X_s) - \min_{s \in S} h_2(X_s).$$

- **Relational argmax**

- **Furthest pair**: same object setting as before. Train models to find colors of objects with the largest distance (encoded as an integer category representing pair of colors)

$$y(S) = (h_3(X_{s_1}), h_3(X_{s_2})) \quad \text{s.t.} \quad \{X_{s_1}, X_{s_2}\} = \arg \max_{s_1, s_2 \in S} \|h_1(X_{s_1}) - h_1(X_{s_2})\|_{\ell_1}$$

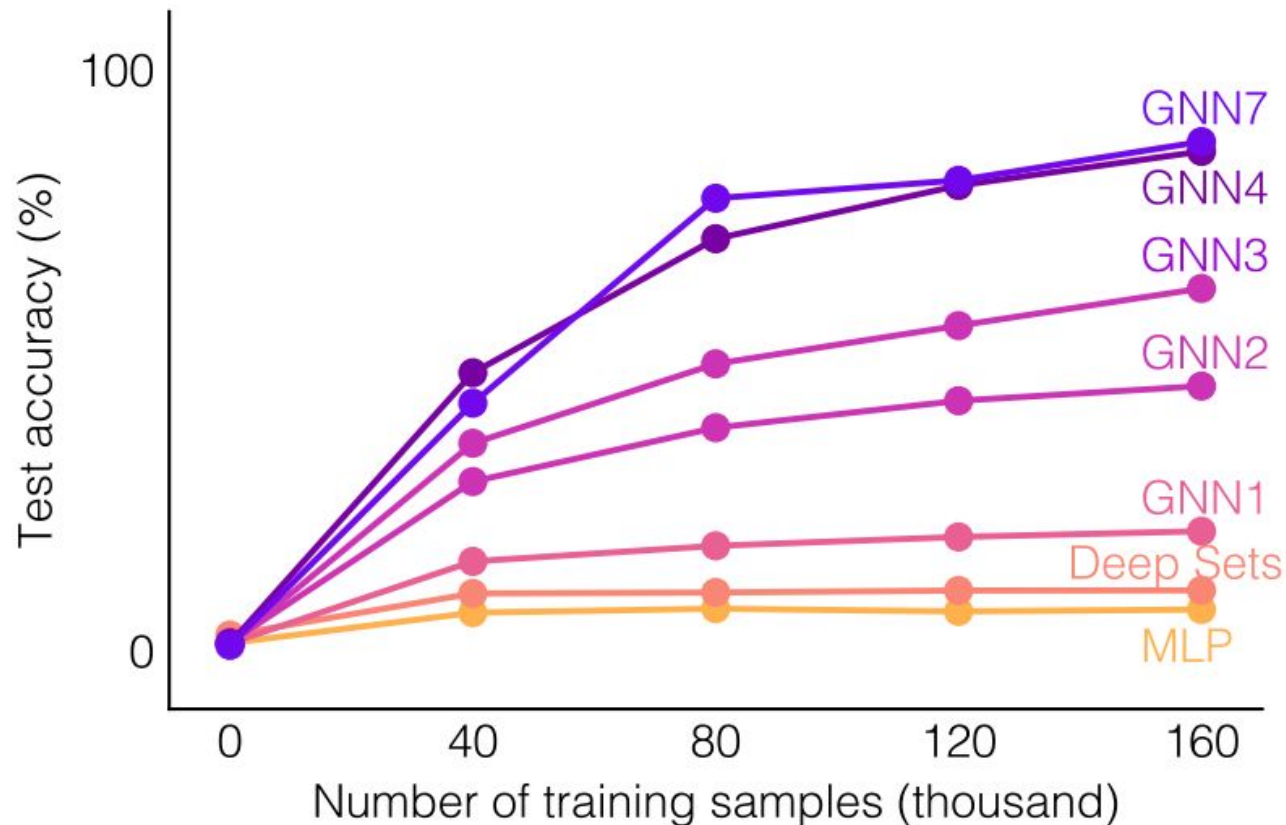
- **Dynamic programming**

- **Shortest Path**: solved using Bellman-Ford previously discussed

- **NP-Hard problems**

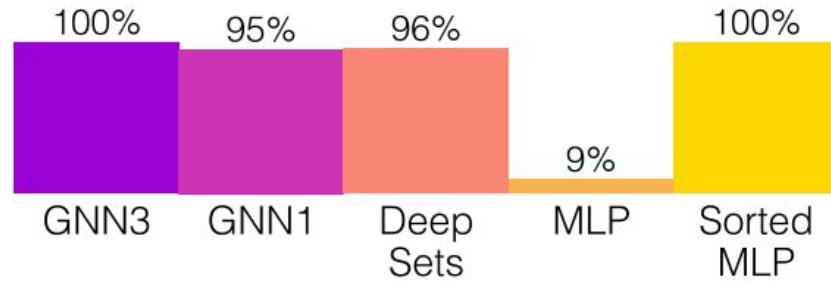
- **Subset sum**: Given a set of numbers, does there exist a subset of numbers (values of treasures here) which sums to zero?

# Empirical Results: Sample efficiency (DP task)

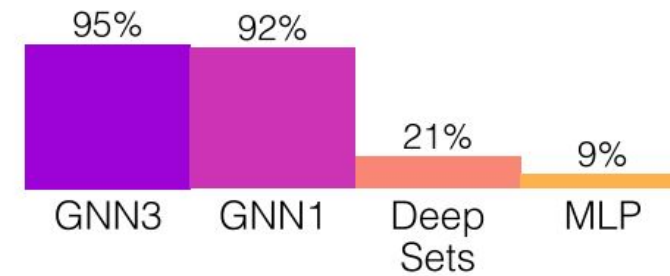


- GNNs align well with DP, Deep Sets and MLP don't
- In this case Bellman-Ford algorithm needs 7 iterations to converge to a solution
- Authors show that an optimized version of the algorithm needs 4 iterations, which might explain why only GNNs with 4 iterations or above generalise well

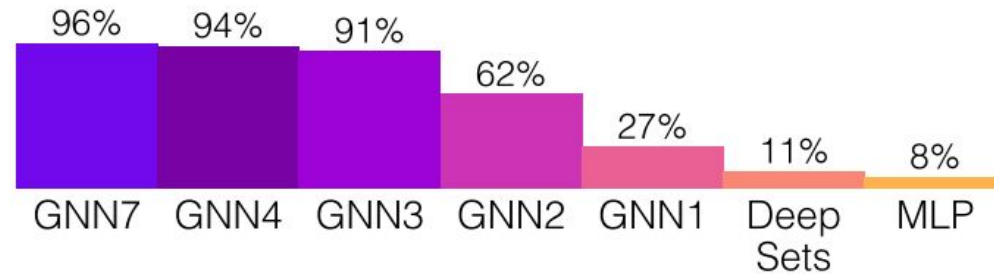
# Empirical Results: Test Accuracy



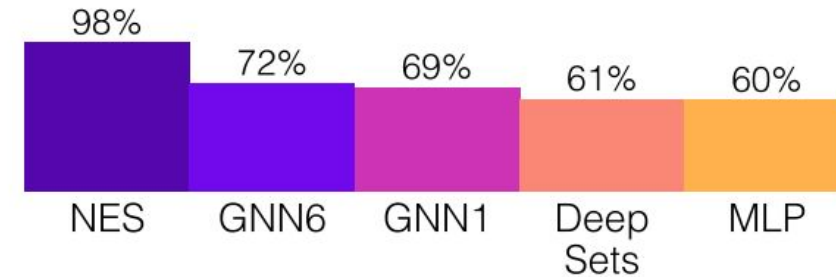
(a) Maximum value difference.



(b) Furthest pair.



(c) Monster trainer.



(d) Subset sum. Random guessing yields 50%.

# Future works

1. Building neural networks with better algorithmic alignment for reasoning
2. Neural architectures that learn algorithmic paradigms other than dynamic programming
3. Solving other algorithmic tasks (like combinatorial optimization) using neural networks designed to align with algorithmic solutions



# Thank You