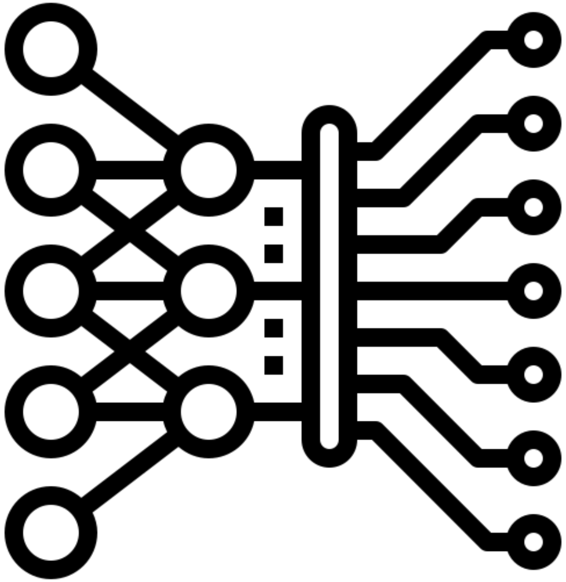
DeepMind

Graph Representation Learning for Algorithmic Reasoning

Petar Veličković

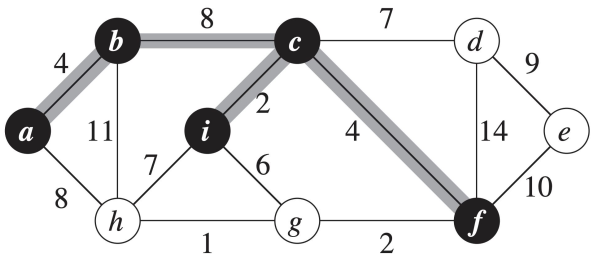
DL4G@WWW2020
21 April 2020

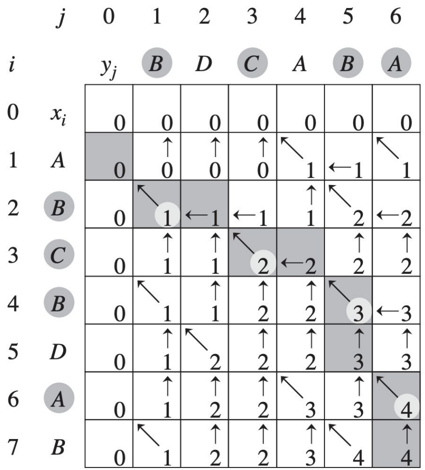
Problem-solving approaches

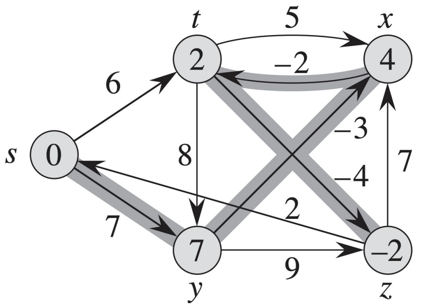


Neural networks

```
MERGE-SORT(A, p, r)
if p < r
    q = ⌊(p + r)/2⌋
    MERGE-SORT(A, p, q)
    MERGE-SORT(A, q + 1, r)
    MERGE(A, p, q, r)
```



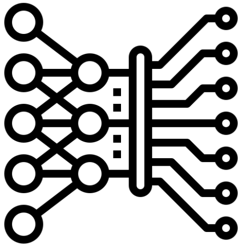


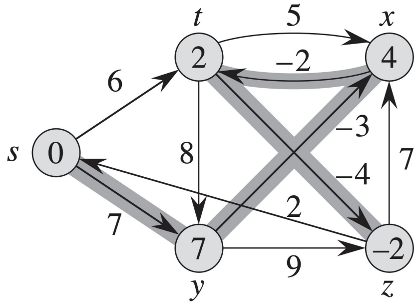


Algorithms

Algorithm figures: Cormen, Leiserson, Rivest and Stein. **Introduction to Algorithms.**

Problem-solving approaches





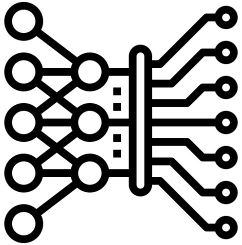
Neural networks

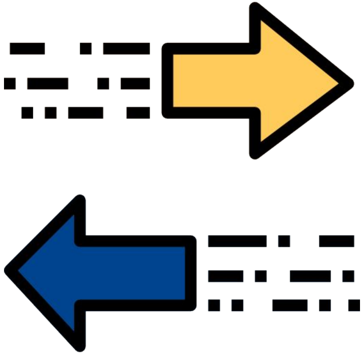
- \+ Operate on **raw** inputs
- \+ Generalise on **noisy** conditions
- \+ Models **reusable** across tasks
- – Require **big data**
- – Unreliable when **extrapolating**
- – Lack of **interpretability**

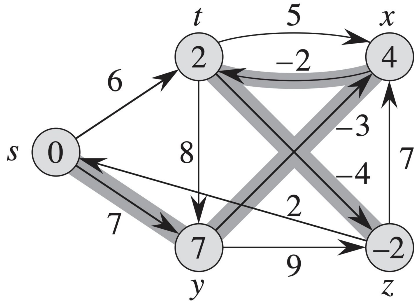
Algorithms

- \+ Trivially **strongly** generalise
- \+ **Compositional** (subroutines)
- \+ Guaranteed **correctness**
- \+ **Interpretable** operations
- – Inputs must match **spec**
- – Not **robust** to task variations

Problem-solving approaches







Neural networks

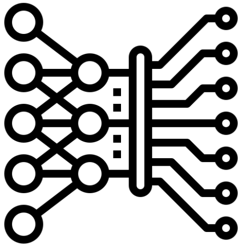
Algorithms

- \+ Operate
- \+ Genera
- \+ Models
- \- Require
- \- Unreliab
- \- Lack of

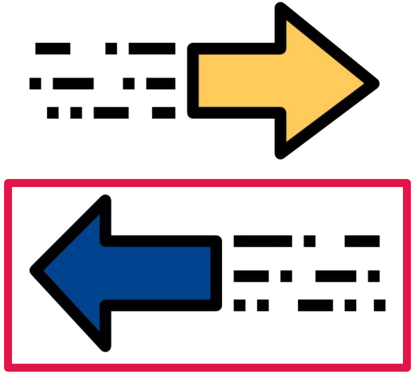
eralise
outines)
ess
ions
oec
riations

Is it possible to get the best of both worlds?

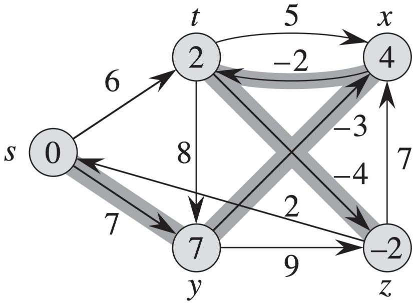
Problem-solving approaches



Neural networks



This talk!



Algorithms

+ Operate
+ Genera
+ Models
- Require
- Unreliab
- Lack of

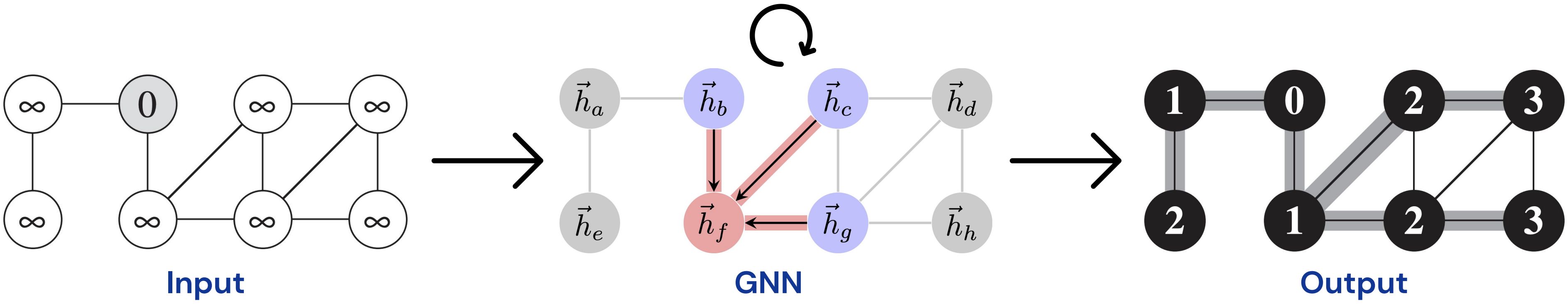
eralise
outines)
ness
ions
pec
riations

Is it possible to get the best of **both** worlds?

Neural Graph-Algorithmic Reasoning

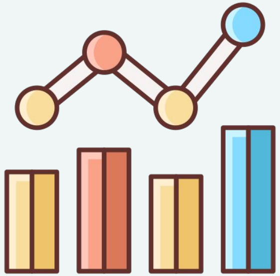
- *Can neural nets robustly **reason** like algorithms?*

- Algorithms manipulate (un)ordered sets of objects, and their relations.
  - ⇒ They operate over *graphs*.
  - Supervise **graph neural networks** on algorithm execution tasks!

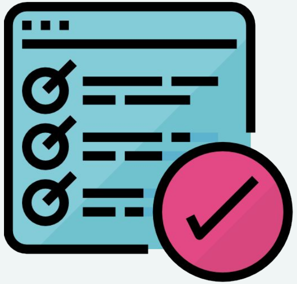


- Call this approach **neural graph algorithm execution**.

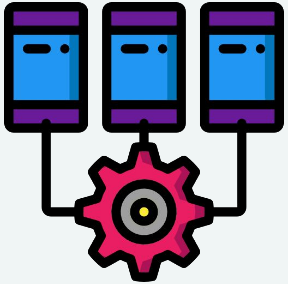
Why?



Benchmarking graph neural nets



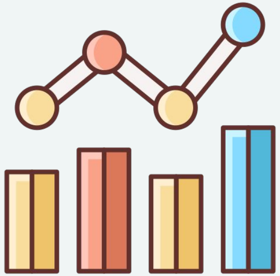
Strong generalisation



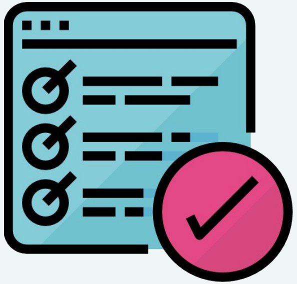
Multi-task learning

Algorithm discovery

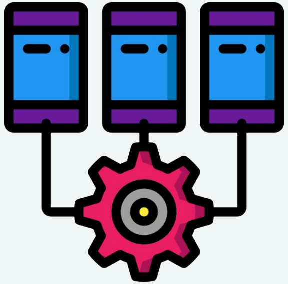
Why?



Benchmarking graph neural nets



Strong generalisation



Multi-task learning

Algorithm discovery

Benchmarking GNNs

- Popular GNN benchmark datasets often **unreliable**

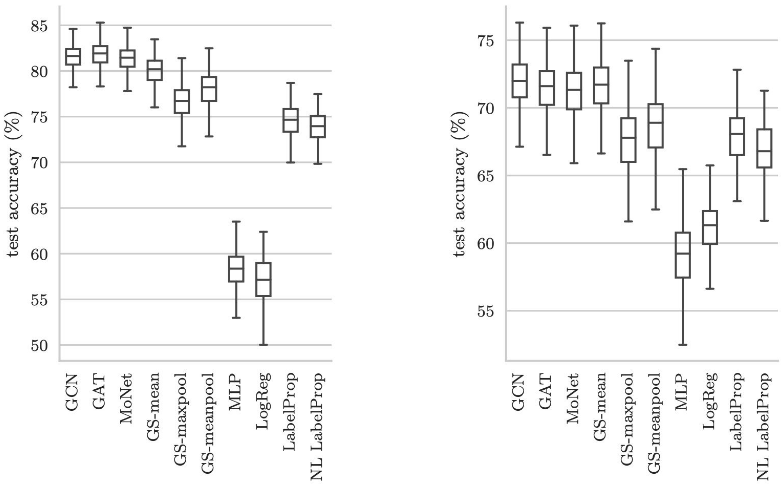
Pitfalls of Graph Neural Network Evaluation

Oleksandr Shchur,* Maximilian Mumme,* Aleksandar Bojchevski, Stephan Günnemann
Technical University of Munich, Germany
{shchur,mumme,a.bojchevski,guennemann}@in.tum.de

On Graph Classification Networks, Datasets and Baselines

Enxhell Luzhnica [* 1] **Ben Day** [* 1] **Pietro Lio** [1]

(a) CORA (b) CiteSeer



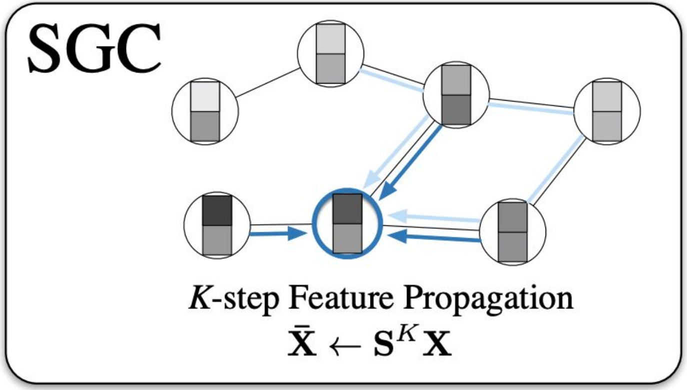
	Datasets			
Model	Reddit [5]	DD	Collab	Prot.
PatchySAN	41.32	76.27	72.60	75.00
GraphSAGE	42.24	75.42	68.25	70.48
ECC	41.73	74.10	67.79	72.65
Set2Set	43.49	78.12	71.75	74.29
SortPool	41.82	79.37	73.76	75.54
DiffPool-Det	46.18	75.47	**82.13**	75.62
DiffPool-NoLP	46.65	79.98	75.63	77.42
DiffPool	47.08	**81.15**	75.50	**78.10**
GU-Net/SHGC	-	78.59	74.54	75.46
MLP	40.96	80.22	74.00	75.74
GCN(R)-MLP	36.15	78.61	75.38	76.28
GCN-MLP	45.01	79.29	76.50	75.64
JK-Sum	**47.16**	79.02	77.00	75.82
JK-Sum-Decay	43.87	79.11	74.14	75.82
JK-Sum-ReInit	46.77	75.97	77.20	75.46

Benchmarking GNNs

- Popular GNN benchmark datasets often **unreliable**
  - **Complexity** not very high

Simplifying Graph Convolutional Networks

Felix Wu [*1] Tianyi Zhang [*1] Amauri Holanda de Souza Jr. [*12] Christopher Fifty [1] Tao Yu [1]
Kilian Q. Weinberger [1]



$$\hat{\mathbf{Y}}_{\text{SGC}} = \text{softmax}\left(\mathbf{S}^K \mathbf{X} \mathbf{\Theta}\right)$$

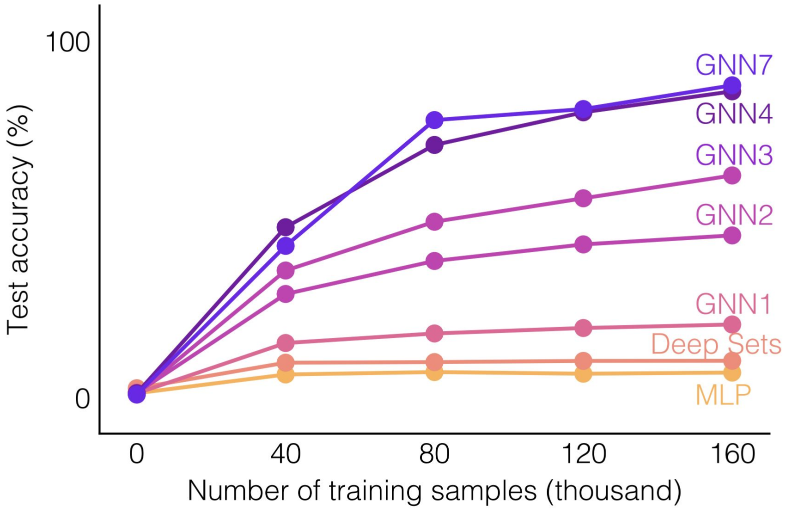
Our experiments:			
GCN	81.4 ± 0.4	70.9 ± 0.5	79.0 ± 0.4
GAT	83.3 ± 0.7	72.6 ± 0.6	78.5 ± 0.3
FastGCN	79.8 ± 0.3	68.8 ± 0.6	77.4 ± 0.3
GIN	77.6 ± 1.1	66.1 ± 0.9	77.0 ± 1.2
LNet	$80.2 \pm 3.0^{\dagger}$	67.3 ± 0.5	$78.3 \pm 0.6^{\dagger}$
AdaLNet	$81.9 \pm 1.9^{\dagger}$	$70.6 \pm 0.8^{\dagger}$	$77.8 \pm 0.7^{\dagger}$
DGI	82.5 ± 0.7	71.6 ± 0.7	78.4 ± 0.7
SGC	81.0 ± 0.0	71.9 ± 0.1	78.9 ± 0.0

Setting	Model	Test F1
Supervised	GaAN	96.4
	SAGE-mean	95.0
	SAGE-LSTM	95.4
	SAGE-GCN	93.0
	FastGCN	93.7
	GCN	**OOM**
Unsupervised	SAGE-mean	89.7
	SAGE-LSTM	90.7
	SAGE-GCN	90.8
	DGI	94.0
No Learning	Random-Init DGI	93.3
	SGC	94.9

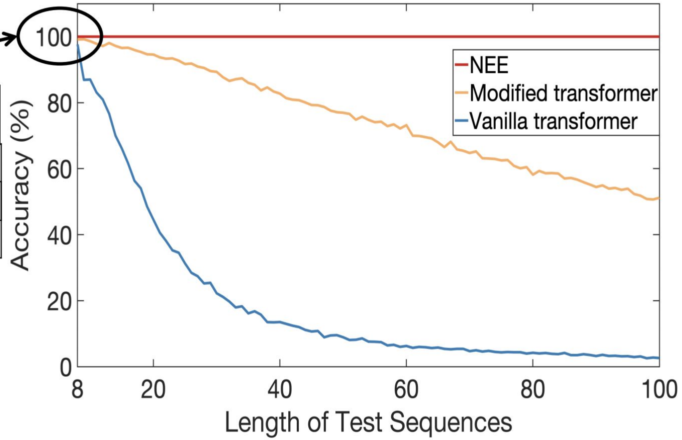
Benchmarking GNNs

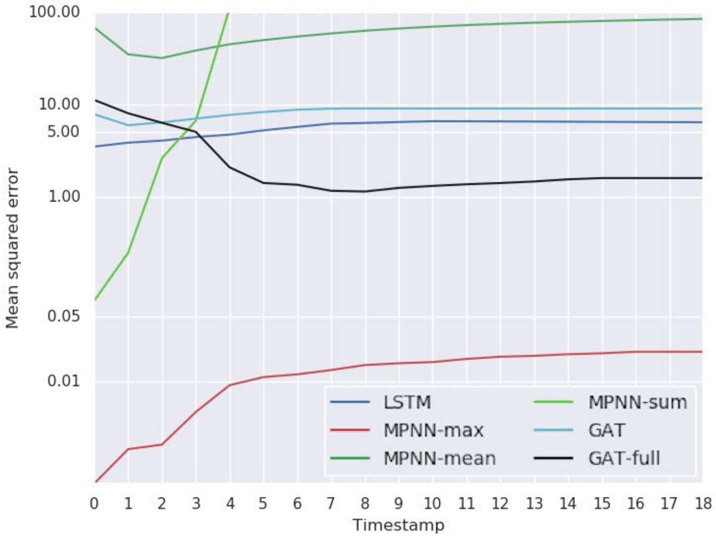
- Popular GNN benchmark datasets often **unreliable**
  - **Complexity** not very high

- Algorithms prove very **favourable**
  - **Infinite** data
  - Complex data **manipulation**
  - A clear **hierarchy** of models emerges!



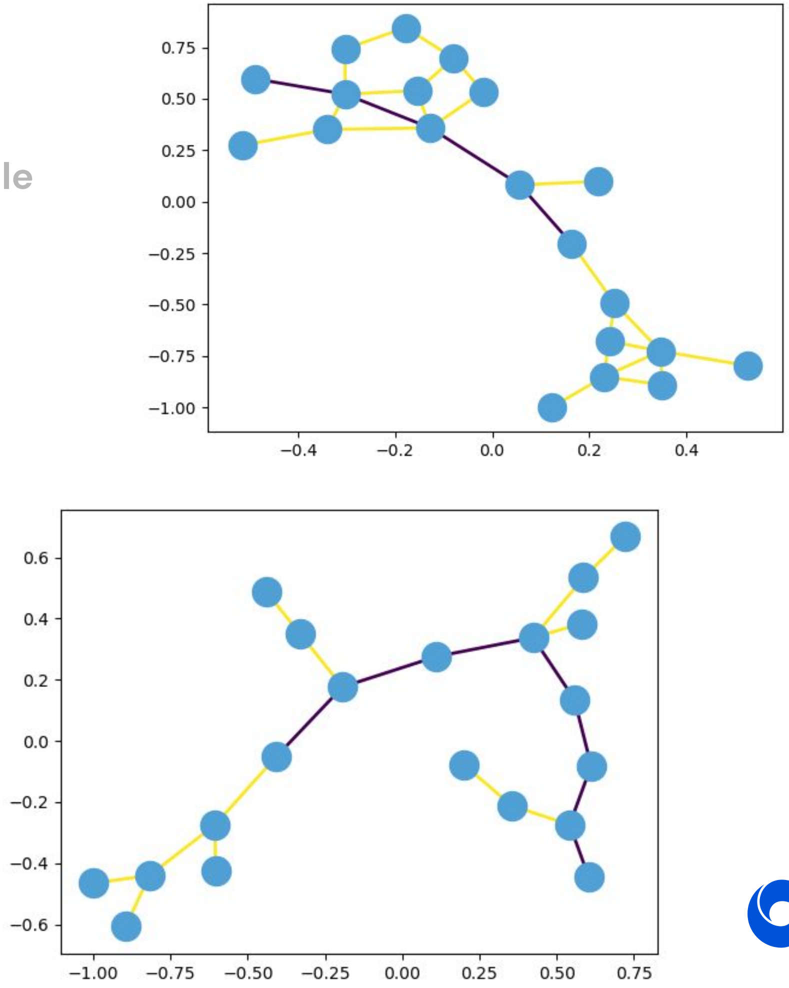
Models	Accuracy @ seq_len = 8
NEE	100.00%
Modified transformer	99.14%
Vanilla transformer	97.66%





Benchmarking GNNs

- Popular GNN benchmark datasets often **unreliable**
  - **Complexity** not very high
- Algorithms prove very **favourable**
  - **Infinite** data
  - Complex data **manipulation**
  - A clear **hierarchy** of models emerges!
- A clearly specified **generating** function
  - No **noise** in the data
  - Enabling rigorous **credit assignment**



Benchmarking GNNs

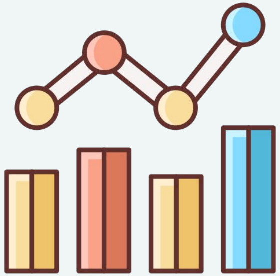
- Popular GNN benchmark datasets often **unreliable**
  - **Complexity** not very high

- Algorithms prove very **favourable**
  - **Infinite** data
  - Complex data **manipulation**
  - A clear **hierarchy** of models emerges!

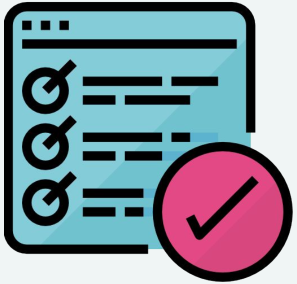
- A clearly specified **generating** function
  - No **noise** in the data
  - Enabling rigorous **credit assignment**

- The world is propped-up on *polynomial-time algorithms*
  - Applicable to NP-hard problems (see e.g. Joshi, Laurent and Bresson, NeurIPS'19 GRL)

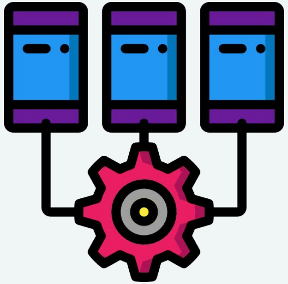
Why?



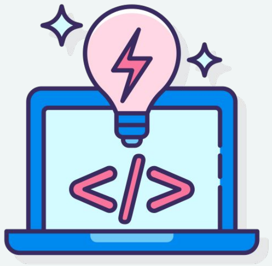
Benchmarking graph neural nets



Strong generalisation



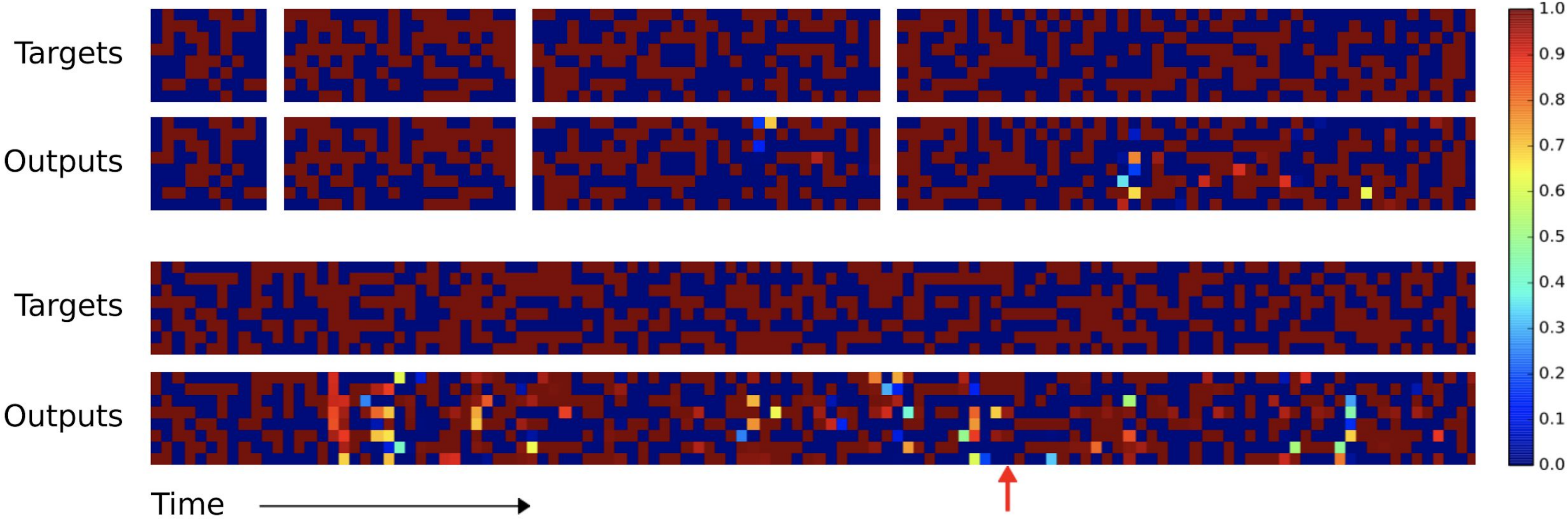
Multi-task learning



Algorithm discovery

Strong generalisation

- Learning an *algorithm* is **not** learning input-output *mapping*!



(Graves *et al.*, 2014)

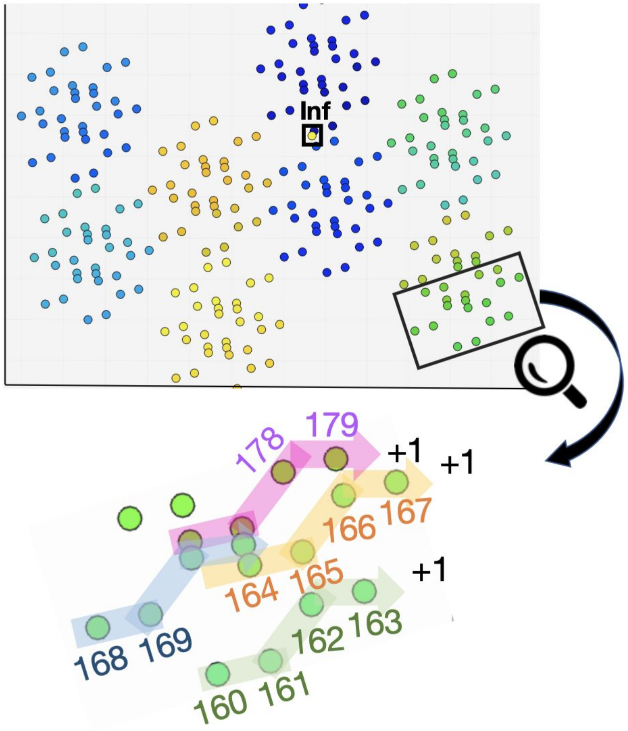
Strong generalisation

- Learning an *algorithm* is **not** learning input-output *mapping*!

- Imitating individual *operations* enables **strong** generalisation.
  - Consider how humans devise algorithms “by hand”.
  - Scales to much larger test graph sizes.

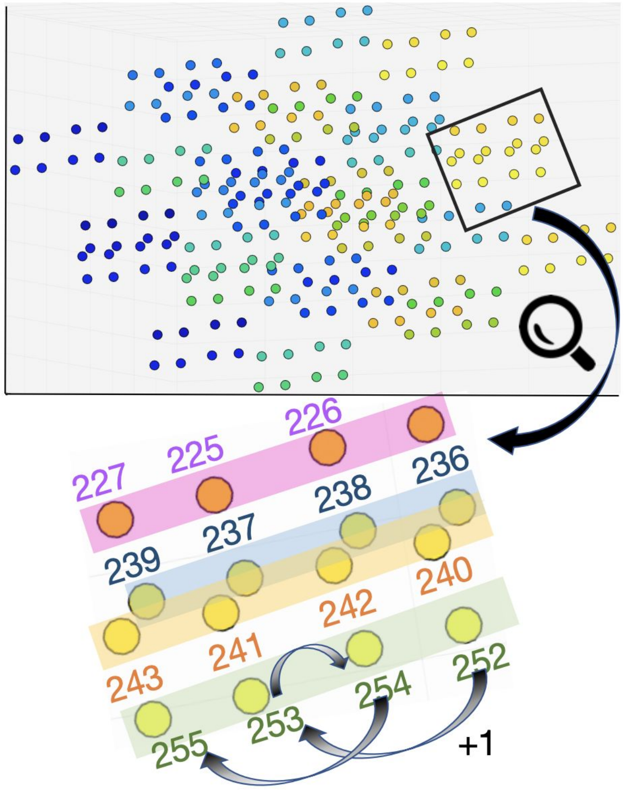
Table 1. Performance of different tasks on variable sizes of test examples (trained with examples of size 8)

Accuracy \ Sizes	25	50	75	100
Selection sort	100.00	100.00	100.00	100.00
Merge sort	100.00	100.00	100.00	100.00
Shortest paths	100.00	100.00	100.00	100.00*

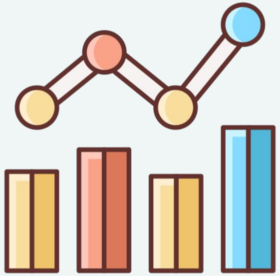


Strong generalisation

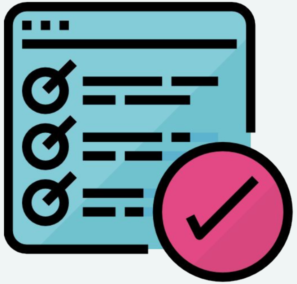
- Learning an *algorithm* is **not** learning input-output *mapping*!
- Imitating individual *operations* enables **strong** generalisation.
  - Consider how humans devise algorithms "by hand".
  - Scales to much larger test graph sizes.
- **Grounds** the GNN in the underlying algorithmic reasoning
  - Deep learning is about learning representations
  - Learn representations of **manipulations**!



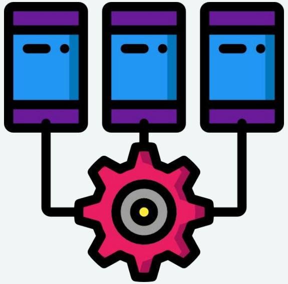
Why?



Benchmarking graph neural nets



Strong generalisation



Multi-task learning

Algorithm discovery

Multi-task learning

- Learning representations of **manipulations**
  - ⇒ lots of potential for representational *reuse*.
  - Many algorithms share **subroutines**.

```
MST-PRIM(G, w, s)
for each u ∈ G.V
    u.key = ∞
    u.π = NIL
s.key = 0
Q = G.V
while Q ≠ ∅
    u = EXTRACT-MIN(Q)
    for each v ∈ G.Adj[u]
        if v ∈ Q and w(u, v) < v.key
            DECREASE-KEY(Q, v, w(u, v))
            v.π = u
```

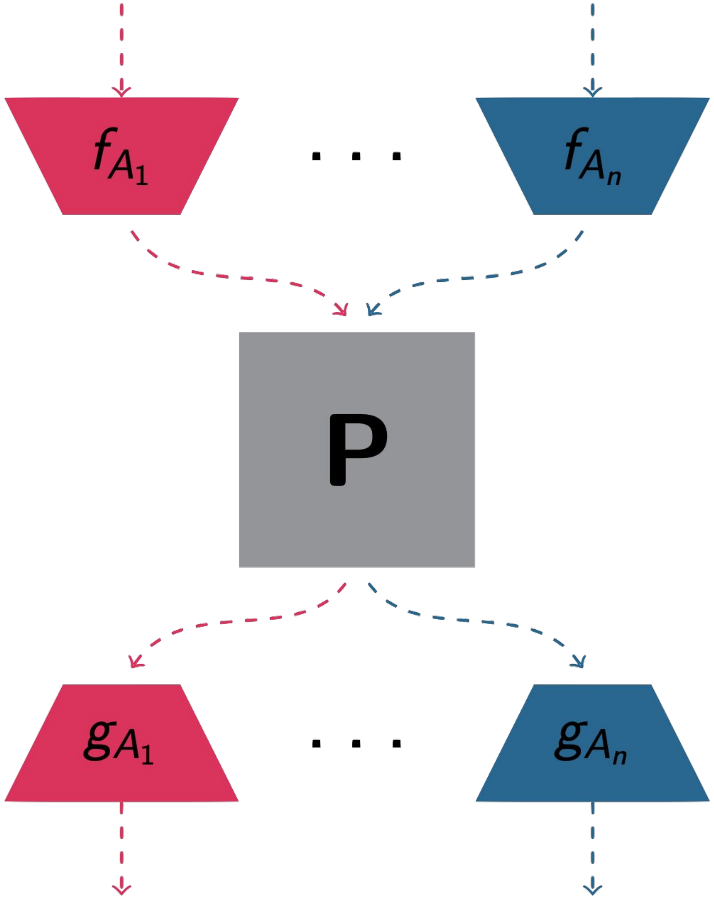
```
DIJKSTRA(G, w, s)
for each u ∈ G.V
    u.key = ∞
    u.π = NIL
s.key = 0
Q = G.V
while Q ≠ ∅
    u = EXTRACT-MIN(Q)
    for each v ∈ G.Adj[u]
        if u.key + w(u, v) < v.key
            DECREASE-KEY(Q, v, u.key + w(u, v))
            v.π = u
```

Multi-task learning

- Learning representations of **manipulations**
  - ⇒ lots of potential for representational *reuse*.
  - Many algorithms share **subroutines**.

- Representations can positively **reinforce** one another!
  - **Meta-representation** of algorithms.
  - Plentiful opportunity for:
    - *Multi-task* learning
    - *Meta*-learning
    - *Continual* learning

  with clearly defined task relations!



Multi-task learning

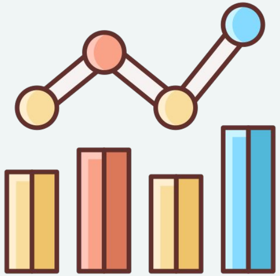
- Learning representations of **manipulations**
  - ⇒ lots of potential for representational *reuse*.
  - Many algorithms share **subroutines**.

- Representations can positively **reinforce** one another!
  - **Meta-representation** of algorithms.
  - Plentiful opportunity for:
    - *Multi-task* learning
    - *Meta*-learning
    - *Continual* learning

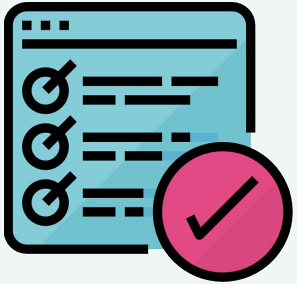
    with clearly defined task relations!

- Output of *easier* algorithm can be used as *input* for a harder one.

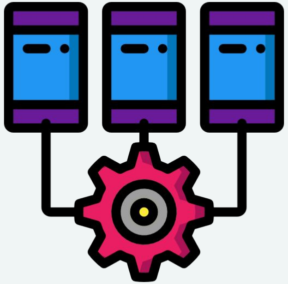
Why?



Benchmarking graph neural nets



Strong generalisation



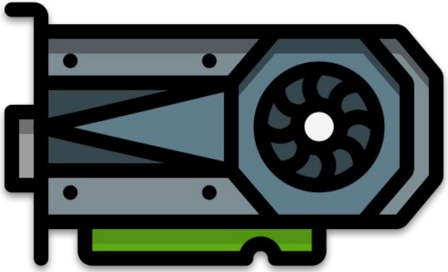
Multi-task learning

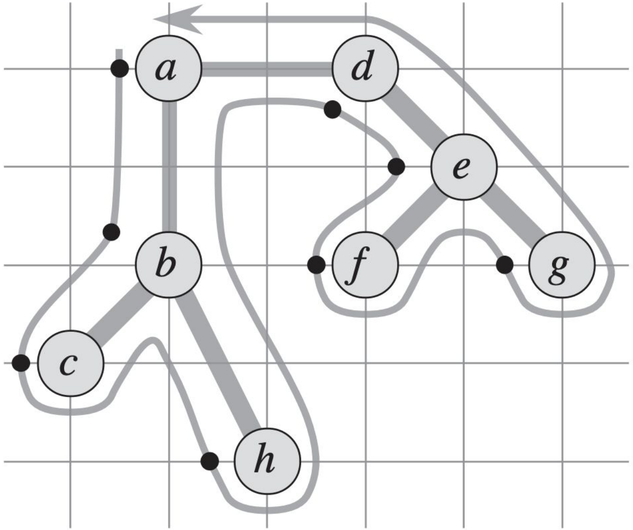
Algorithm discovery

Algorithm discovery

- Inspecting intermediate outputs of an algorithm can **decode** its behaviour!

- Opportunity for deriving **novel** algorithms, e.g.
  - Improved heuristics for *intractable* problems.
  - Optimising for GNN executors (e.g. GPU/TPU).





Algorithm discovery

- Inspecting intermediate outputs of an algorithm can **decode** its behaviour!

- Opportunity for deriving **novel** algorithms, e.g.
  - Improved heuristics for *intractable* problems.
  - Optimising for GNN executors (e.g. GPU/TPU).

- Machine learning ← **Competitive programming**!
  - My way into computer science :)

Sphere online judge

CODEFORCES

icpc International Collegiate Programming Contest

Algorithm discovery

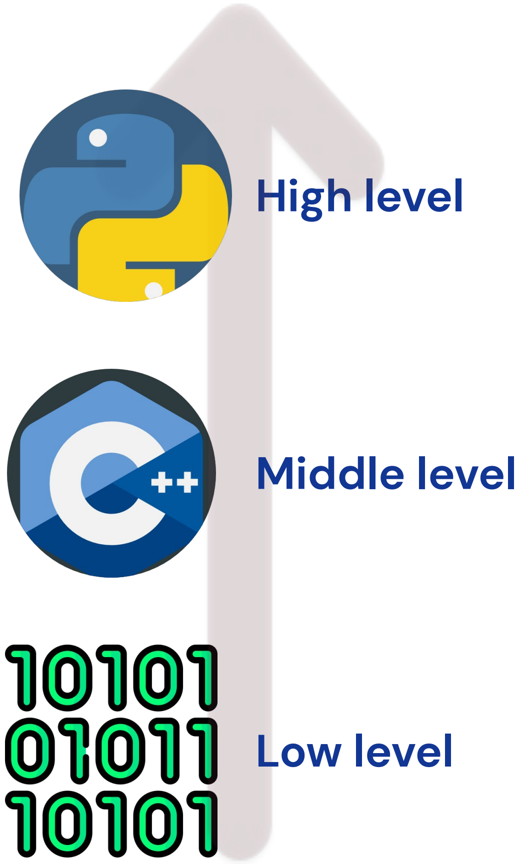
- Inspecting intermediate outputs of an algorithm can **decode** its behaviour!

- Opportunity for deriving **novel** algorithms, e.g.
  - Improved heuristics for *intractable* problems.
  - Optimising for GNN executors (e.g. GPU/TPU).

- Machine learning ← **Competitive programming**!
  - My way into computer science :)

- Conjecture: Can perform ***soft* subroutine reuse** from polynomial-time algorithms.

Programming language hierarchy



GNN-Algorithmic hierarchy

(Xu, Li, Zhang, Du, Kawarabayashi and Jegelka. ICLR 2020)

WHAT CAN NEURAL NETWORKS REASON ABOUT? **Algo–level**

(Veličković, Ying, Padovano, Hadsell and Blundell. ICLR 2020)

NEURAL EXECUTION OF GRAPH ALGORITHMS **Step–level**

(Yan, Swersky, Koutra, Ranganathan and Hashemi. 2020)

NEURAL EXECUTION ENGINES **Unit–level**

GNN-Algorithmic hierarchy

(Xu, Li, Zhang, Du, Kawarabayashi and Jegelka. ICLR 2020)

Algo-level

- Learns an **algorithm** end-to-end only
- Strong theoretical link between **generalisation power** and **algorithmic alignment**
- GNNs align well with *dynamic programming!*

(Veličković, Ying, Padovano, Hadsell and Blundell. ICLR 2020)

Step-level

- Supervises on atomic **steps** of an algorithm
- Out-of-distribution testing of various GNNs
- *Multi-task learning + maximisation aggregators* generalise stronger!

(Yan, Swersky, Koutra, Ranganathan and Hashemi. 2020)

Unit-level

- Learns to execute tiny **operations**, then composes them
- Binary encoding and conditional masking
- Achieves *perfect* strong generalisation!

GNN-Algorithmic hierarchy

(Xu, Li, Zhang, Du, Kawarabayashi and Jegelka. ICLR 2020)

WHAT CAN NEURAL NETWORKS REASON ABOUT? **Algo–level**

(Veličković, Ying, Padovano, Hadsell and Blundell. ICLR 2020)

NEURAL EXECUTION OF GRAPH ALGORITHMS **Step–level**

(Yan, Swersky, Koutra, Ranganathan and Hashemi. 2020)

NEURAL EXECUTION ENGINES **Unit–level**

(Xu, Li, Zhang, Du, Kawarabayashi and Jegelka. ICLR 2020)

What Can Neural Networks Reason About?

- Which networks are best suited for certain types of **reasoning**?
  - **Theorem**: better *structural alignment* implies better *generalisation*!
  - GNNs ~ dynamic programming

Graph Neural Network

for k = 1 ... GNN iter:

for u in S:

$h_u^{(k)} = \Sigma_v$ 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:

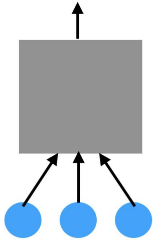
d[k][u] = $\min_v$ d[k-1][v] + cost (v, u)

Learns a simple reasoning step

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

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

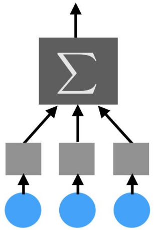
Architectures under study



MLPs

~ feature extraction

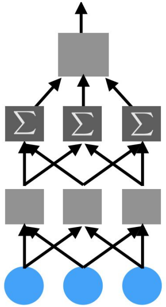
$$y = \mathrm{MLP}(\|_{s\in S} X_s)$$



Deep Sets (Zaheer *et al.*, NeurIPS 2017)

~ summary statistics

$$y = \mathrm{MLP}_2\left(\sum_{s\in S} \mathrm{MLP}_1(X_s)\right)$$



GNNs

~ (pairwise) relations

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

$$y = \mathrm{MLP}_2\left(\sum_{s\in S} h_s^{(K)}\right)$$

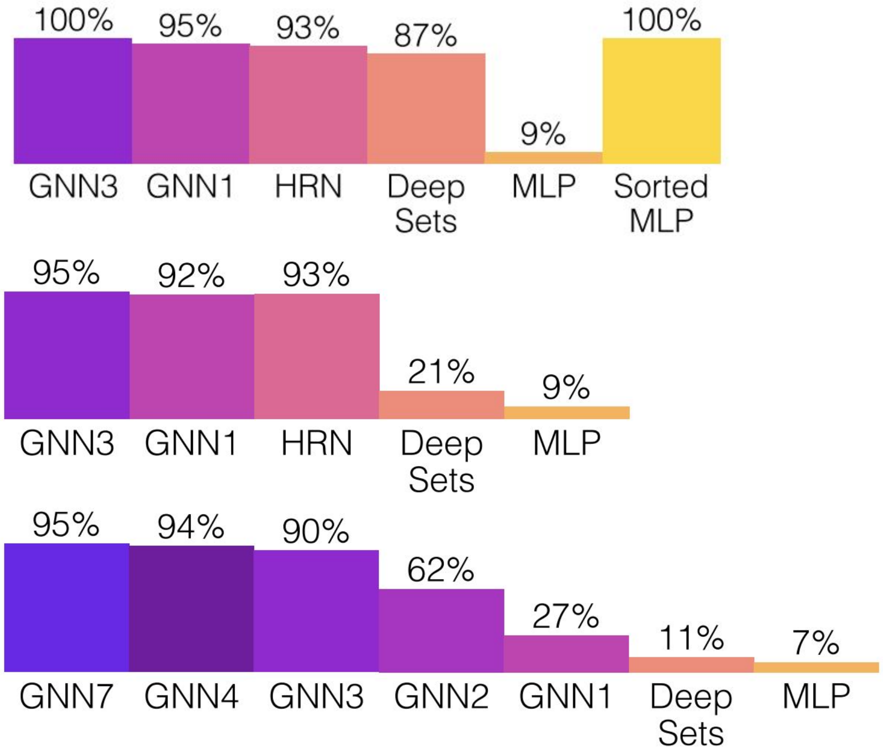
(Xu, Li, Zhang, Du, Kawarabayashi and Jegelka. ICLR 2020)

Empirical results

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?



GNN-Algorithmic hierarchy

(Xu, Li, Zhang, Du, Kawarabayashi and Jegelka. ICLR 2020)

WHAT CAN NEURAL NETWORKS REASON ABOUT? **Algo–level**

(Veličković, Ying, Padovano, Hadsell and Blundell. ICLR 2020)

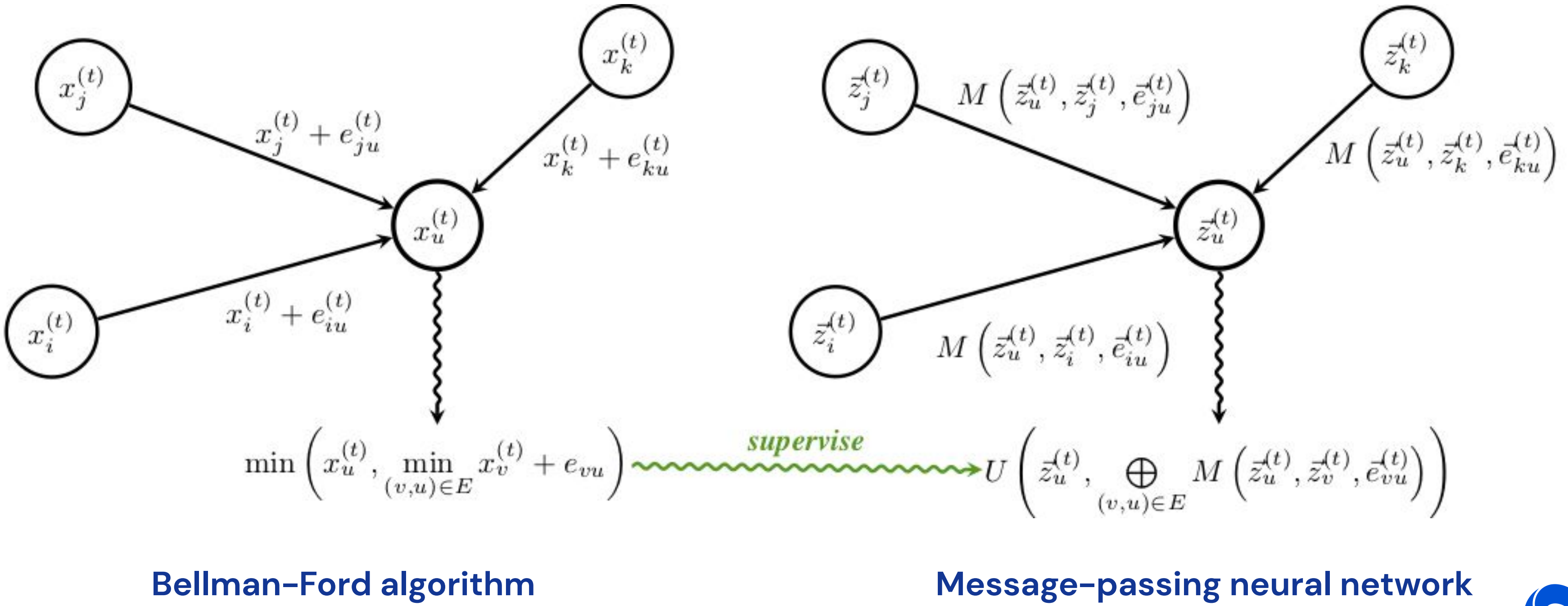
NEURAL EXECUTION OF GRAPH ALGORITHMS **Step–level**

(Yan, Swersky, Koutra, Ranganathan and Hashemi. 2020)

NEURAL EXECUTION ENGINES **Unit–level**

Neural Execution of Graph Algorithms

Supervise on appropriate output values **at every step.**

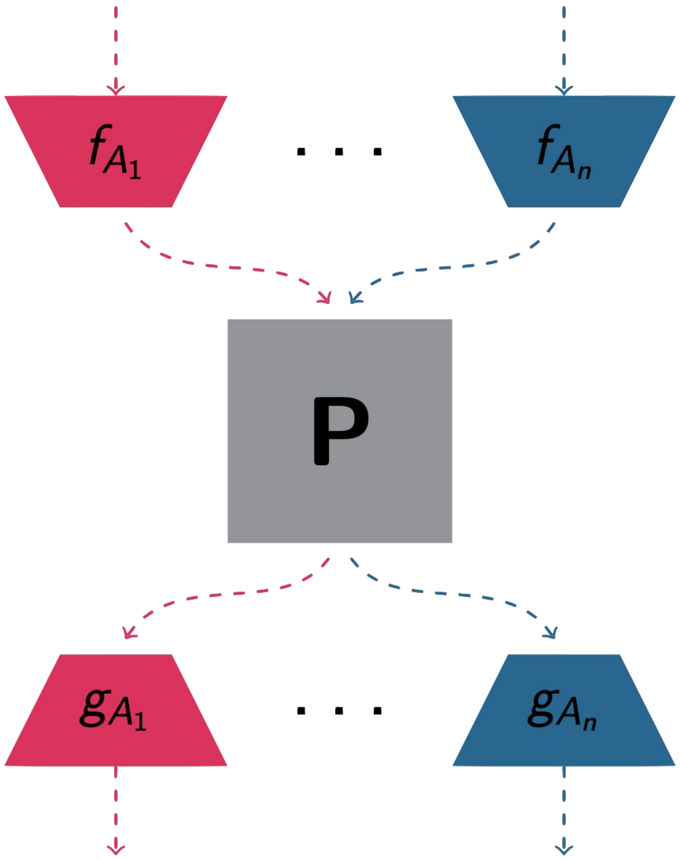


(Veličković, Ying, Padovano, Hadsell and Blundell. ICLR 2020)

Components of the executor

- **Encoder** network* $\vec{z}_i^{(t)} = f_A(\vec{x}_i^{(t)}, \vec{h}_i^{(t-1)})$
- **Processor** network $\mathbf{H}^{(t)} = P(\mathbf{Z}^{(t)}, \mathbf{E}^{(t)})$
- **Decoder** network* $\vec{y}_i^{(t)} = g_A(\vec{z}_i^{(t)}, \vec{h}_i^{(t)})$
- **Termination** network* $\tau^{(t)} = \sigma(T_A(\mathbf{H}^{(t)}))$
- **Repeat** as long as $\tau^{(t)} > 0.5$

*algorithm-specific



- Hypothesis: **MPNN-max** is a highly suitable processor

(Veličković, Ying, Padovano, Hadsell and Blundell. ICLR 2020)

Evaluation

- Evaluate on *parallel* and *sequential* algorithms.
  - Parallel: *Reachability* (**BFS**), *Shortest paths* (**Bellman-Ford**)
  - Sequential: *Minimal spanning trees* (**Prim**)
  - Explicit inductive bias on sequentiality (learnable mask!)

- Generate **graphs** from a wide variety of distributions:
  - Ladder, Grid, Tree, 4-Caveman, 4-Community, Erdős-Rényi, Barabási-Albert
  - Attach random-valued weights to each edge

- Study the “human-programmer” perspective: test generalisation from small graphs (20 nodes) to larger graphs (50/100 nodes).

- Learn to execute BFS and Bellman-Ford with **same** processor!

(Veličković, Ying, Padovano, Hadsell and Blundell. ICLR 2020)

Evaluation: Shortest paths (+ Reachability)

Model	Predecessor (mean step accuracy / last-step accuracy) 20 nodes	50 nodes	100 nodes
LSTM (Hochreiter & Schmidhuber, 1997)	47.20% / 47.04%	36.34% / 35.24%	27.59% / 27.31%
GAT* (Veličković et al., 2018)	64.77% / 60.37%	52.20% / 49.71%	47.23% / 44.90%
GAT-full* (Vaswani et al., 2017)	67.31% / 63.99%	50.54% / 48.51%	43.12% / 41.80%
MPNN-mean (Gilmer et al., 2017)	93.83% / 93.20%	58.60% / 58.02%	44.24% / 43.93%
MPNN-sum (Gilmer et al., 2017)	82.46% / 80.49%	54.78% / 52.06%	37.97% / 37.32%
MPNN-max (Gilmer et al., 2017)	**97.13% / 96.84%**	**94.71% / 93.88%**	**90.91% / 88.79%**
MPNN-max (*curriculum*)	95.88% / 95.54%	91.00% / 88.74%	84.18% / 83.16%
MPNN-max (*no-reach*)	82.40% / 78.29%	78.79% / 77.53%	81.04% / 81.06%
MPNN-max (*no-algo*)	78.97% / 95.56%	83.82% / 85.87%	79.77% / 78.84%

Trained on 20-node graphs!

Trained without reachability objective

Trained without step-wise supervision

(Veličković, Ying, Padovano, Hadsell and Blundell. ICLR 2020)

Evaluation: Sequential execution

Model	Accuracy (next MST node / MST predecessor)		
	20 nodes	*50 nodes*	*100 nodes*
LSTM (Hochreiter & Schmidhuber, 1997)	11.29% / 52.81%	3.54% / 47.74%	2.66% / 40.89%
GAT* (Veličković et al., 2018)	27.94% / 61.74%	22.11% / 58.66%	10.97% / 53.80%
GAT-full* (Vaswani et al., 2017)	29.94% / 64.27%	18.91% / 53.34%	14.83% / 51.49%
MPNN-mean (Gilmer et al., 2017)	**90.56% / 93.63%**	52.23% / 88.97%	20.63% / 80.50%
MPNN-sum (Gilmer et al., 2017)	48.05% / 77.41%	24.40% / 61.83%	31.60% / 43.98%
MPNN-max (Gilmer et al., 2017)	87.85% / 93.23%	**63.89% / 91.14%**	**41.37% / 90.02%**
MPNN-max (*no-algo*)	— / 71.02%	— / 49.83%	— / 23.61%

The sequential inductive bias is very **helpful**!

GNN-Algorithmic hierarchy

(Xu, Li, Zhang, Du, Kawarabayashi and Jegelka. ICLR 2020)

What Can Neural Networks Reason About? **Algo–level**

(Veličković, Ying, Padovano, Hadsell and Blundell. ICLR 2020)

Neural Execution of Graph Algorithms **Step–level**

(Yan, Swersky, Koutra, Ranganathan and Hashemi. 2020)

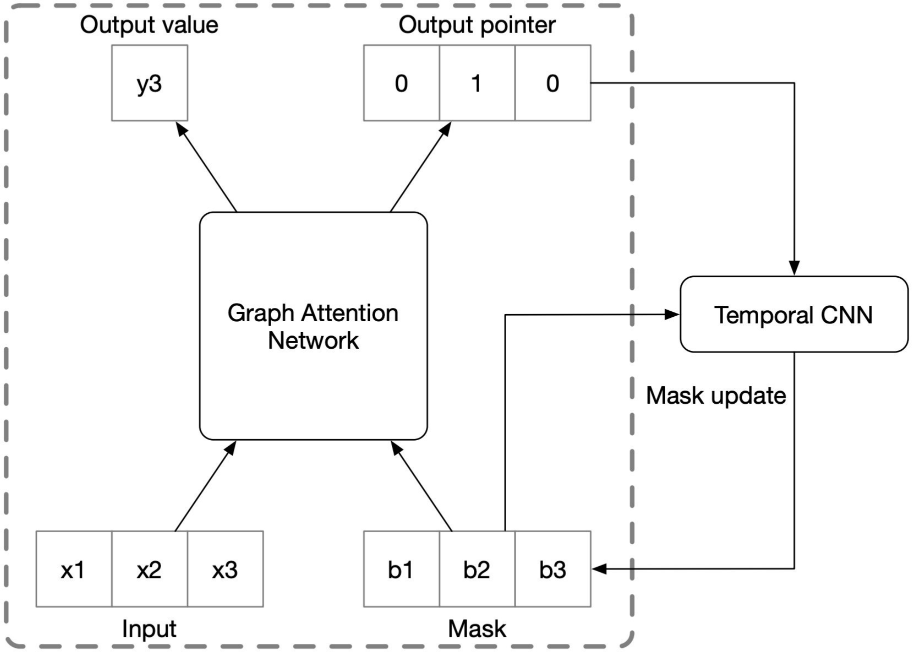
Neural Execution Engines **Unit–level**

(Yan, Swersky, Koutra, Raganathan and Hashemi. 2020)

Neural Execution Engines

- Teach a neural net to **strongly** perform *tiny* tasks (e.g. sum, product, argmin)
  - **Compose** tasks to specify algorithms
  - The building blocks must stay robust with long/OOD rollouts!

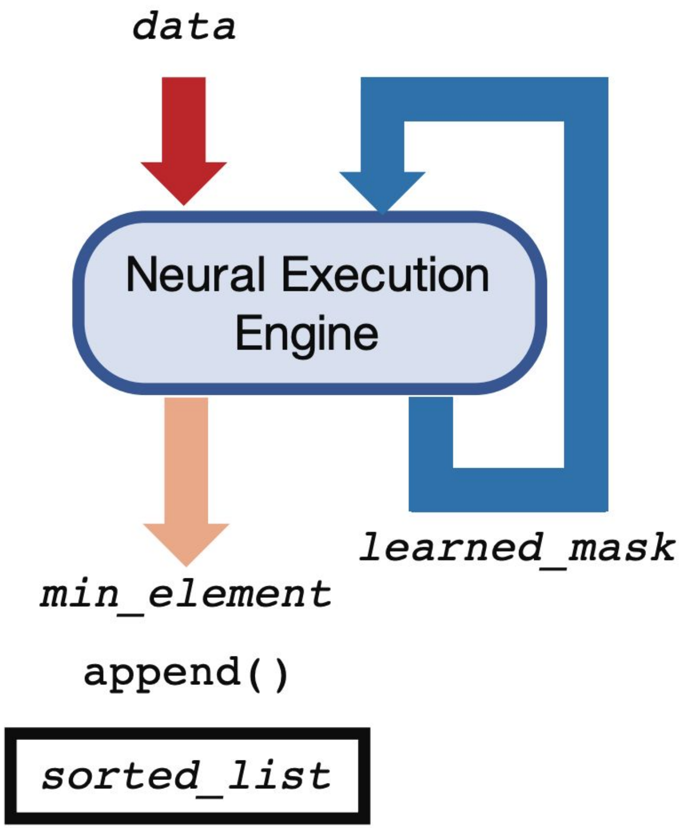
- Key components:
  - Bitwise embeddings
  - Transformers
  - Conditional masking



(Yan, Swersky, Koutra, Raganathan and Hashemi. 2020)

Learning to **selection** sort by composing **argmin**

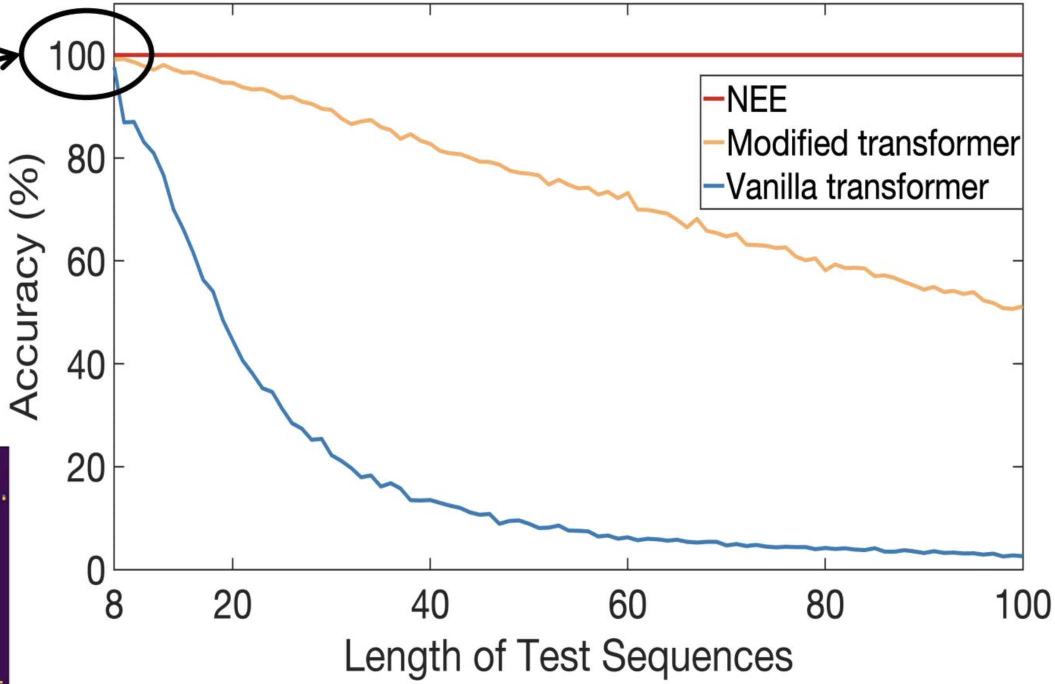
```
selection_sort(data):
  sorted_list = []
  while (len(data) > 0):
    min_index, min_element = find_min(data)
    data.delete(min_index)
    sorted_list.append(min_element)
  return sorted_list

find_min(data):
    min_element = -1
    min_index = -1
    for index, element in enumerate(data):
      if (element < min_element):
        min_element = element
        min_index = index

  return [min_index, min_element]
```

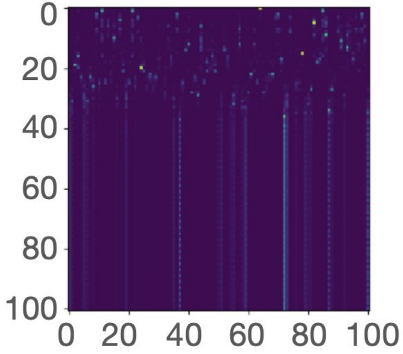


(Yan, Swersky, Koutra, Raganathan and Hashemi. 2020)

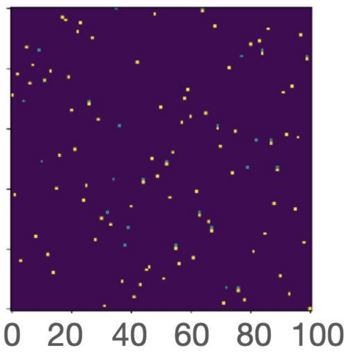
Learning to **selection** sort by composing **argmin**

Models	Accuracy @ seq_len = 8
NEE	100.00%
Modified transformer	99.14%
Vanilla transformer	97.66%





(a) Fuzzy Attention (seq2seq)



(b) Clear Attention (Selection Sort NEE)

(Yan, Swersky, Koutra, Raganathan and Hashemi. 2020)

Composing subroutines (Dijkstra)

shortest_path(*graph*, *source_node*, *shortest_path*):
  dists = []
  nodes = []
  anchor_node = *source_node*
  node_list = *graph*.get_nodes()

  while node_list:

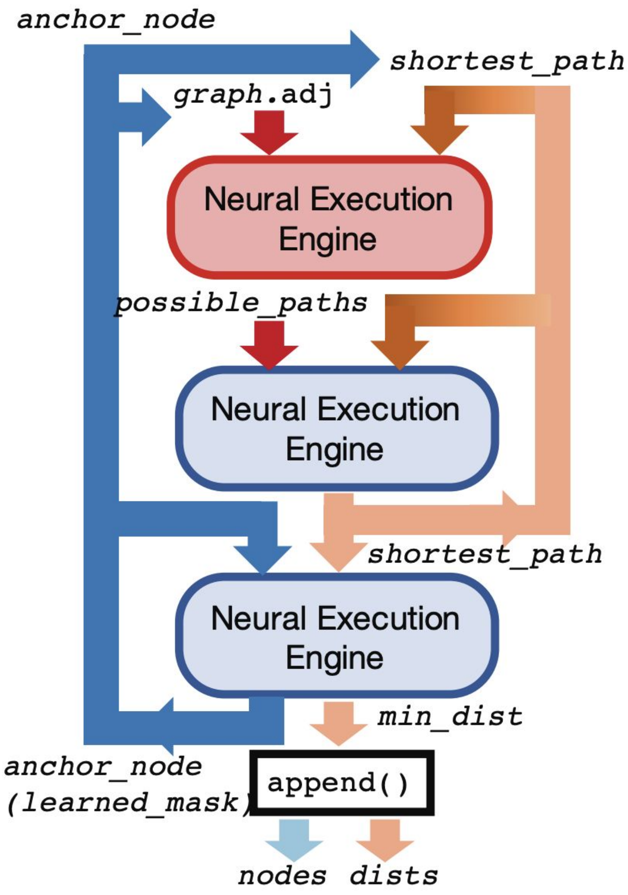
   possible_paths = **sum**(*graph.adj(anchor_node)*, *shortest_path(anchor_node)*)

   shortest_path = **min**(*possible_paths*, *shortest_path*)

   anchor_node, *min_dist* = **min**(*shortest_path*)

   node_list.delete(*anchor_node*)
   nodes.append(*anchor_node*)
   dists.append(*min_dist*)

  return *dists*, *nodes*



(Yan, Swersky, Koutra, Raganathan and Hashemi. 2020)

Recursive subroutines (Merge sort)

```
merge_sort(data, start, end):
  if (start < end):
    mid = (start + end ) / 2

    merge_sort(data, start, mid)
    merge_sort(data, mid+1, end)

    merge(data, start, mid, end)
```

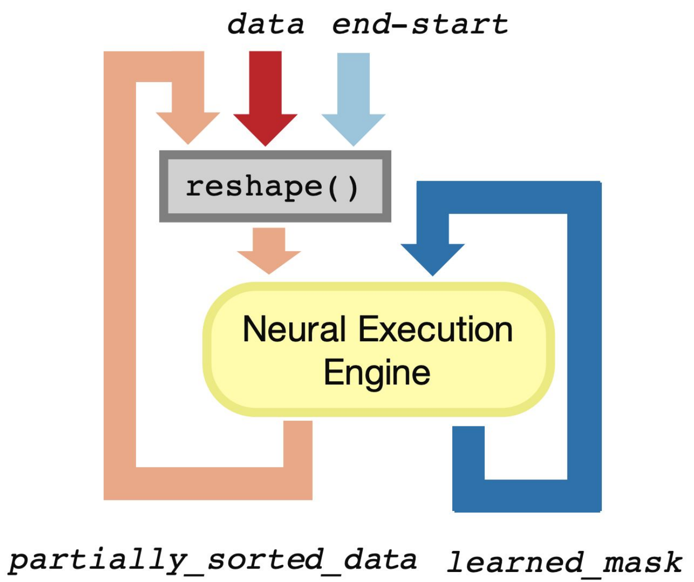


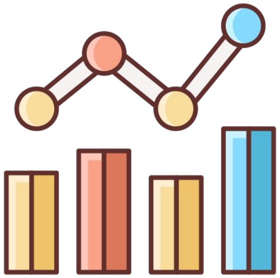
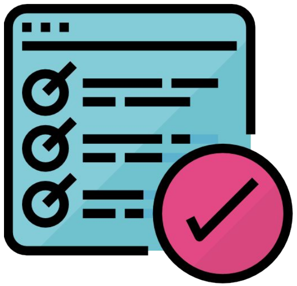
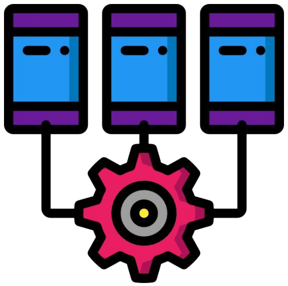
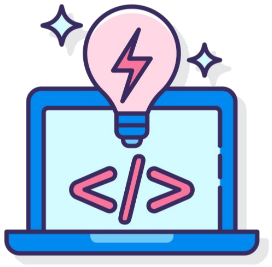
Table 1. Performance of different tasks on variable sizes of test examples (trained with examples of size 8)

Accuracy \ Sizes	25	50	75	100
Selection sort	100.00	100.00	100.00	100.00
Merge sort	100.00	100.00	100.00	100.00
Shortest paths	100.00	100.00	100.00	100.00*

Conclusions

- **Algorithmic reasoning** is an exciting novel area for **graph representation learning**!
  - Three concurrent works explore it at different levels:
    - Algo-level *(Xu, Li, Zhang, Du, Kawarabayashi and Jegelka. ICLR 2020)*
    - Step-level *(Veličković, Ying, Padovano, Hadsell and Blundell. ICLR 2020)*
    - Unit-level *(Yan, Swersky, Koutra, Raganathan and Hashemi. 2020)*

- Many questions left to be answered, at *all* levels of the hierarchy!
  - `<Your contribution here/>`

DeepMind

Thank you!

Questions?

petarv@google.com | https://petar-v.com

In collaboration with Charles Blundell, Raia Hadsell, Rex Ying, Matilde Padovano, Lars Buesing, Matt Overlan, Razvan Pascanu and Oriol Vinyals