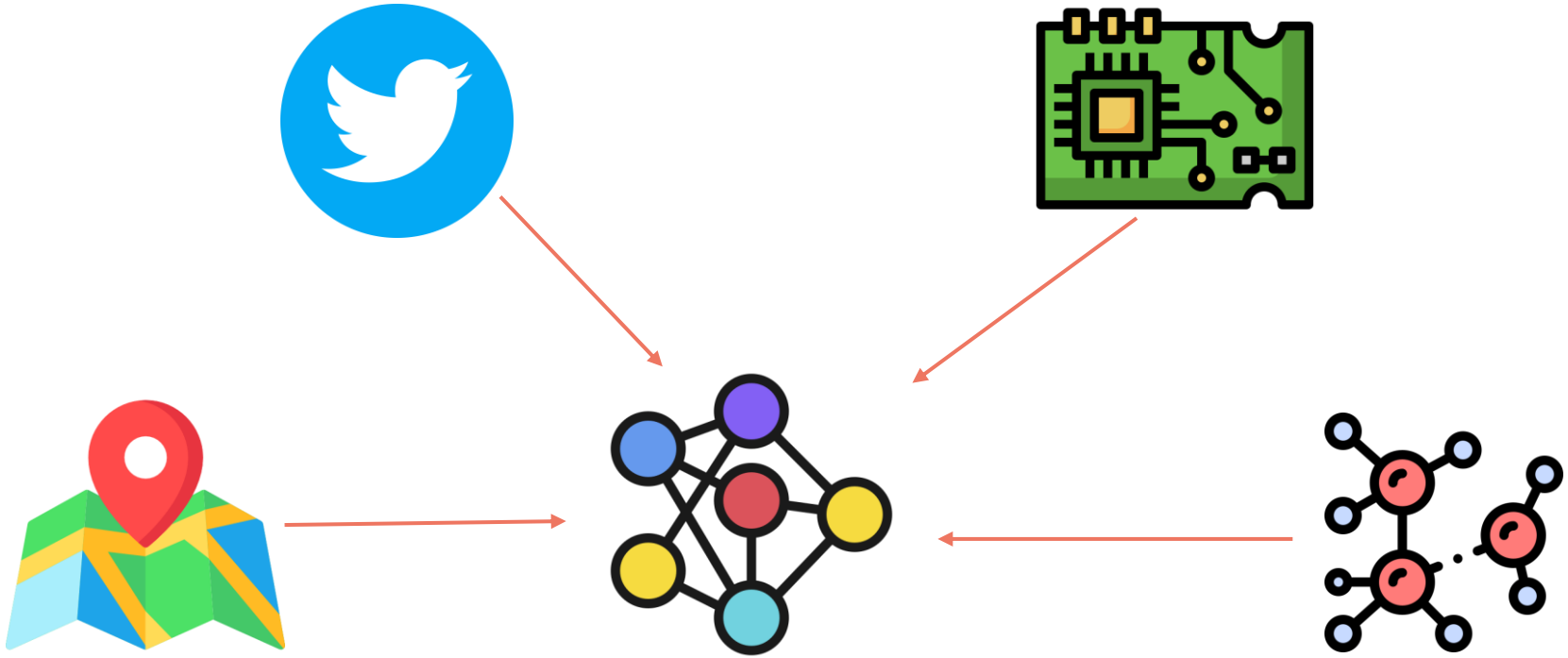
A generalist neural algorithmic learner

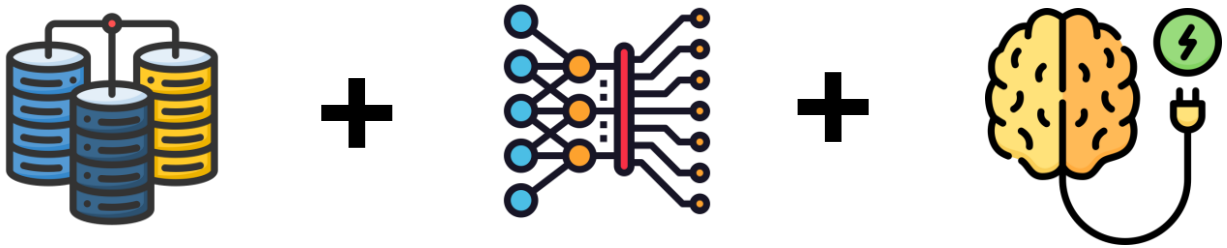
Presented by Dennis Vilgertshofer

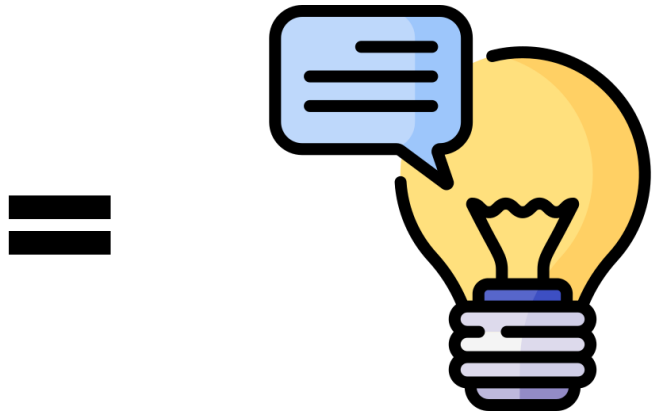
23.05.2023

Algorithms



Neural Networks

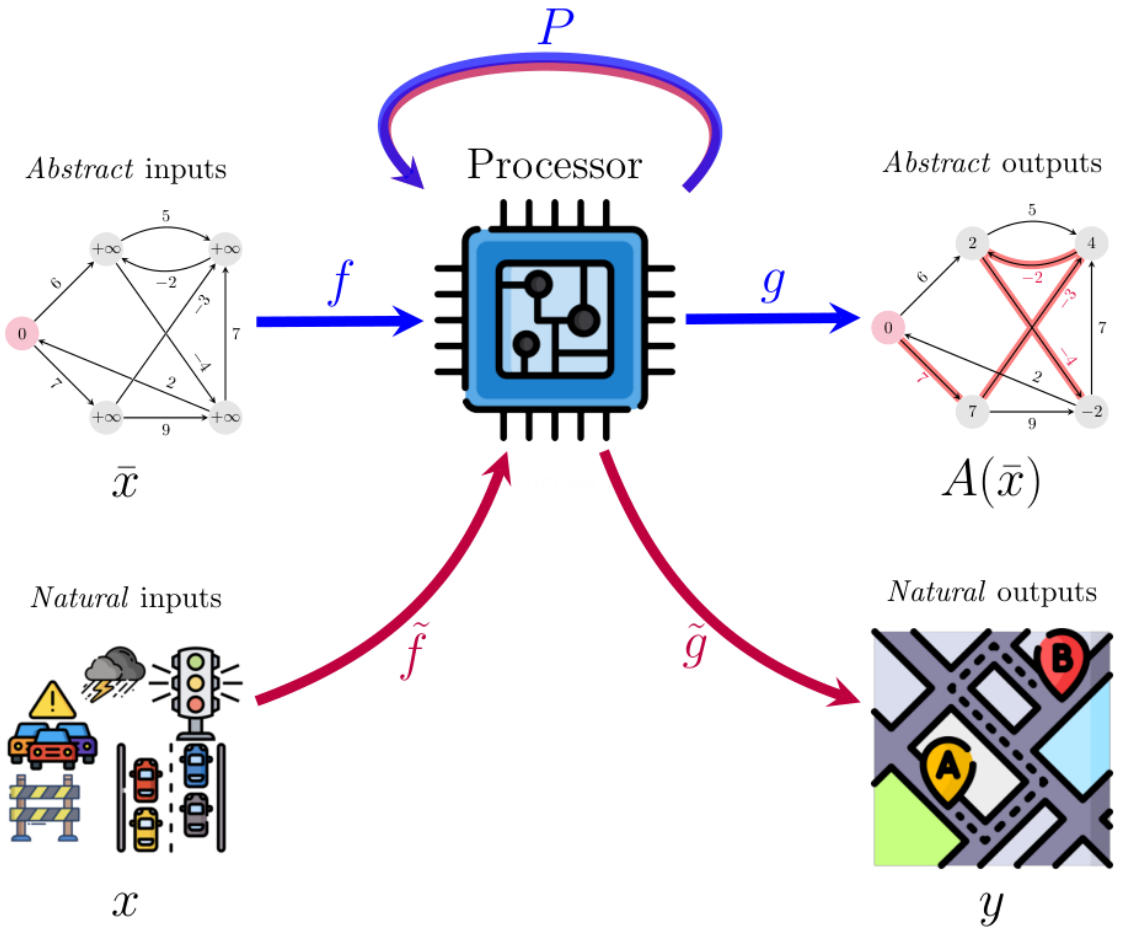




Neural Networks vs Algorithms

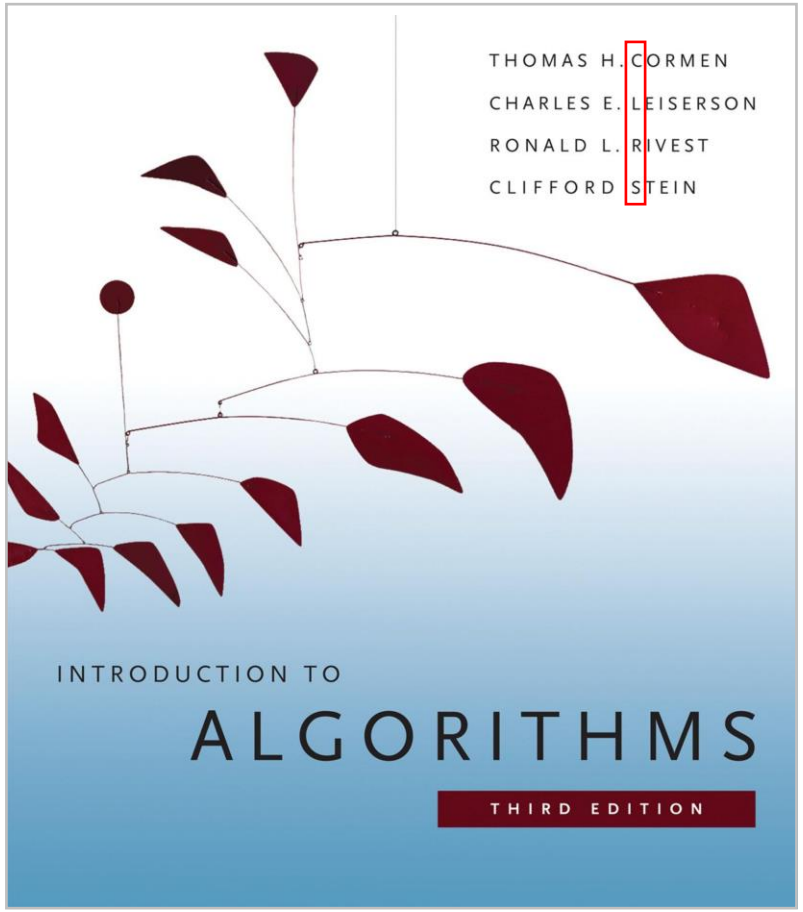
Neural Networks	Algorithms
Requires massive amounts of data	Does not require data
Challenging to generalize OOD	Trivially strongly generalize
Hard to interpret	Interpretable stepwise operations
Unreliable	Provable correctness
Adaptable to a wide range of problems	Inflexible

Motivation


P
Processor
Abstract inputs
Abstract outputs
f
g
$\bar{x}$
$A(\bar{x})$
Natural inputs
Natural outputs
$\tilde{f}$
$\tilde{g}$
x
y

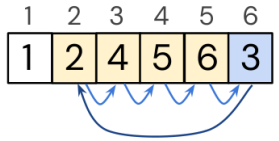
Related Work

CLRS-30 Benchmark

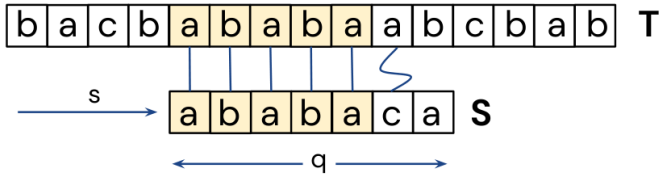


- Aligns closely to pseudocode from the book
- Automatically generate input/output pairs
- Capture intermediate algorithm trajectory

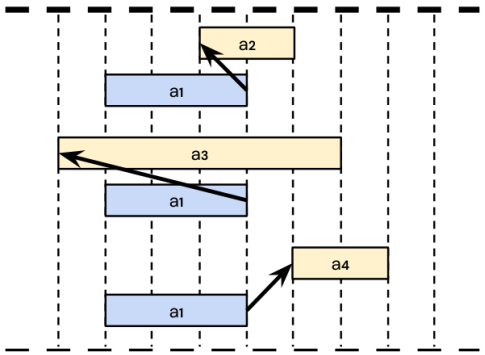
A)



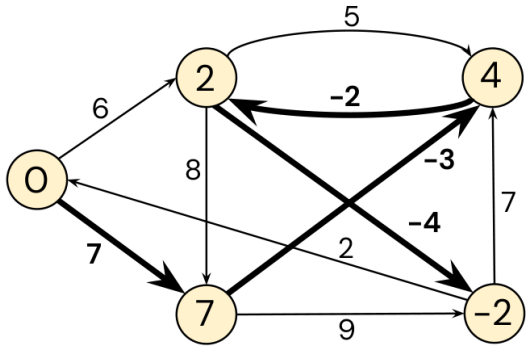
B)



C)



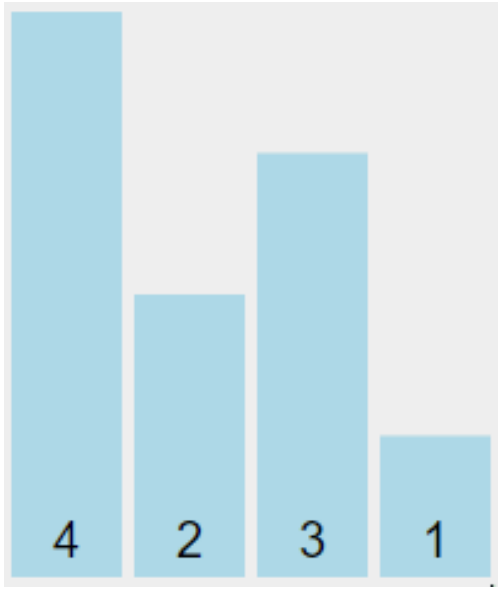
D)



CLRS-30 Benchmark

- Data represented by n vertices
- Static set of probes `(stage, loc, type, values)` for each algorithm
- 5 Feature Types:
  - `scalar`
  - `categorical`
  - `mask`
  - `mask_one`
  - `pointer`

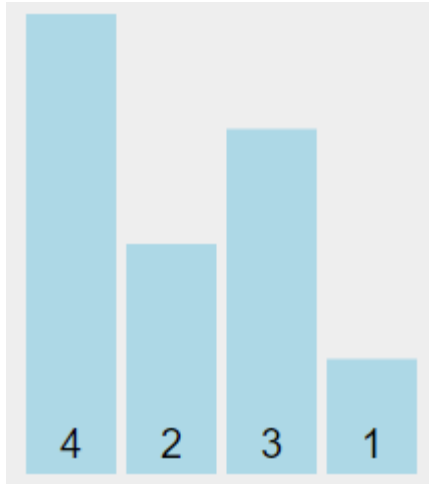
CLRS-30 Example: Bubble Sort



CLRS-30 Example: Bubble Sort

```
pos:    [0,0.33,0.67,1]
key:    [4,  2 , 3  ,1]
```

- Input
- Hint
- Output

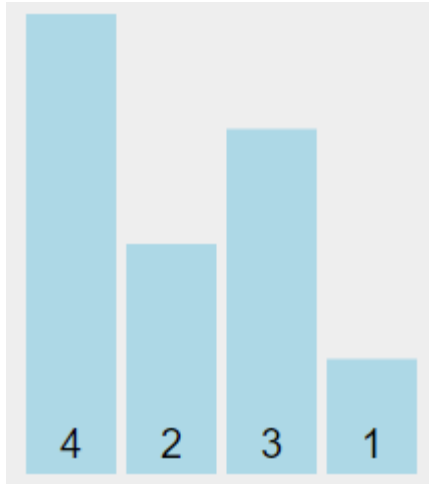


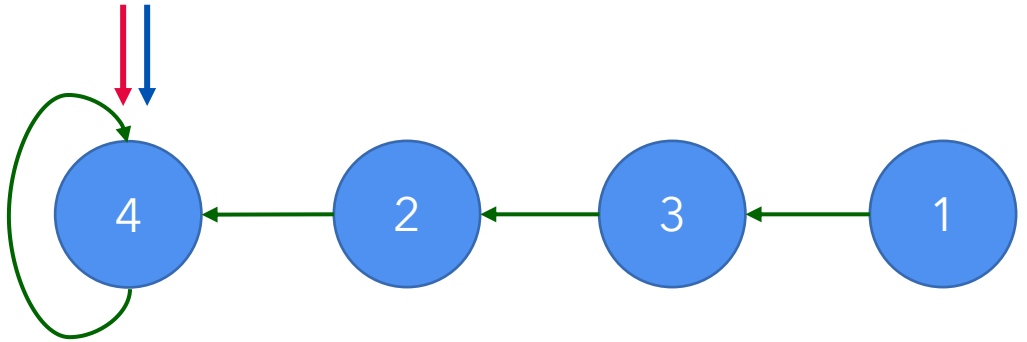
CLRS-30 Example: Bubble Sort

```
pos:    [0,0.33,0.67,1]
key:    [4,  2 , 3  ,1]
```

- Input
- Hint
- Output

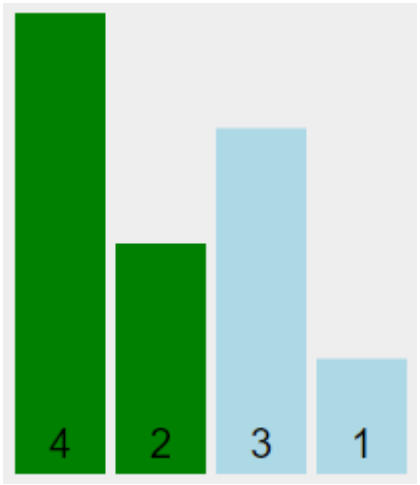
```
i:      [1,0,0,0]
j:      [1,0,0,0]
pred:   [0,0.33,0.67,1]
```





- i (mask_one)
- j (mask_one)
- List State Specification (pointer)

CLRS-30 Example: Bubble Sort

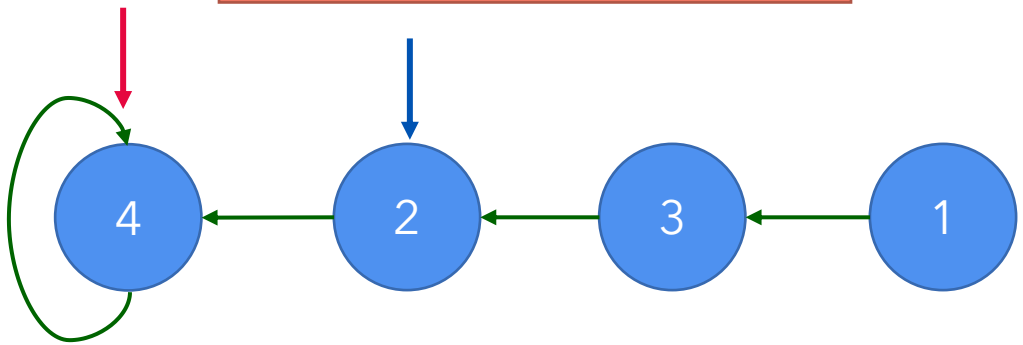



```
pos:    [0,0.33,0.67,1]
key:    [4,  2 , 3  ,1]
```

Input
Hint
Output

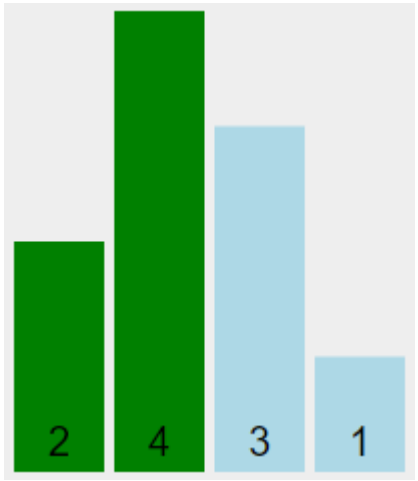
```
i:      [1,0,0,0]
j:      [1,0,0,0]
pred:   [0,0.33,0.67,1]
```

```
i:      [1,0,0,0]
j:      [0,1,0,0]
pred:   [0,0.33,0.67,1]
```



i (mask_one)
j (mask_one)
List State Specification (pointer)

CLRS-30 Example



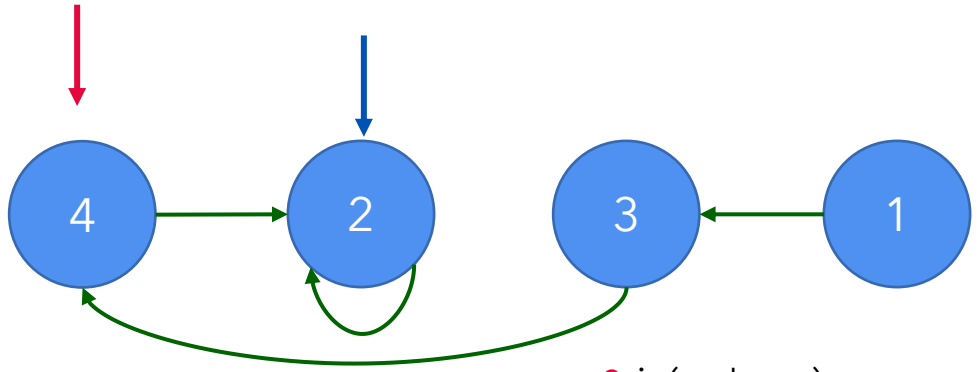

```
pos:    [0,0.33,0.67,1]
key:    [4,  2 , 3  ,1]
```

```
i:      [1,0,0,0]
j:      [1,0,0,0]
pred:   [0,0.33,067,1]
```

```
i:      [1,0,0,0]
j:      [0,1,0,0]
pred:   [0,0.33,067,1]
```

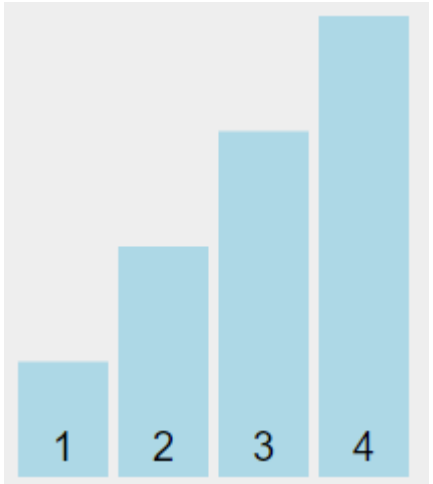
```
i:      [1,0,0,0]
j:      [0,1,0,0]
pred:   [0.33,0,0.67,1]
```

- Input
- Hint
- Output



- i (mask_one)
- j (mask_one)
- List State Specification (pointer)

CLRS-30 Example




```
pos:    [0,0.33,0.67,1]
key:    [4,  2 , 3  ,1]
```

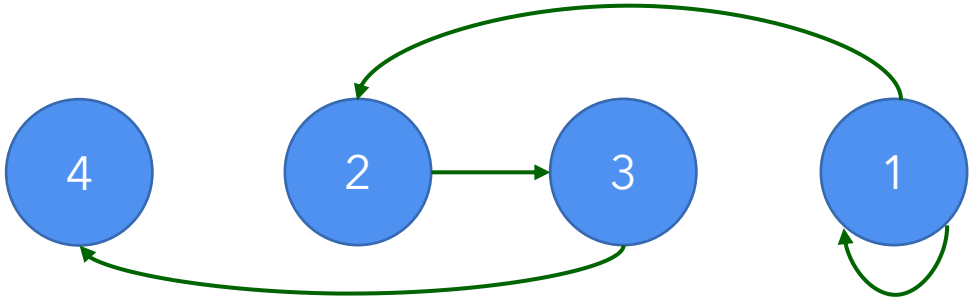
```
i:      [1,0,0,0]
j:      [1,0,0,0]
pred:   [0,0.33,067,1]
```

```
i:      [1,0,0,0]
j:      [0,1,0,0]
pred:   [0,0.33,067,1]
```

```
i:      [1,0,0,0]
j:      [0,1,0,0]
pred:   [0.33,0,0.67,1]
```

- Input
- Hint
- Output

→ ... → `pos: [1,0.33,0.67,0]`

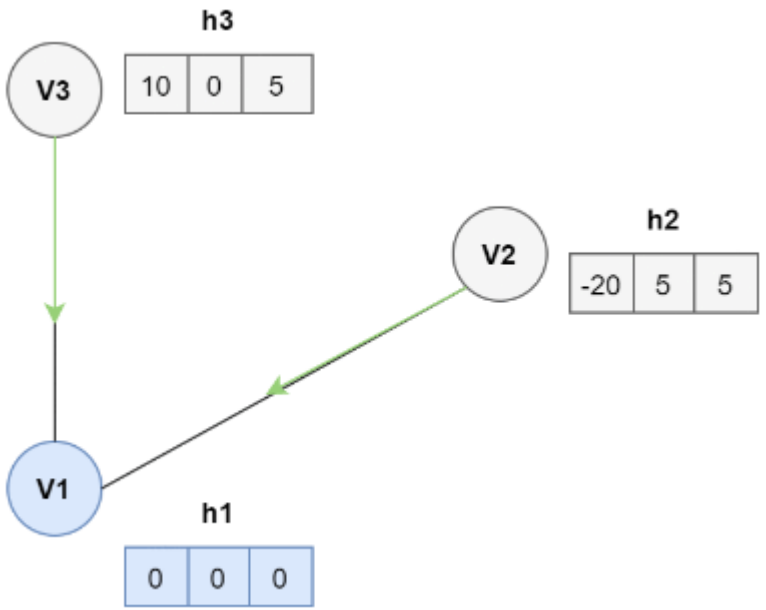


- i (mask_one)
- j (mask_one)
- List State Specification (pointer)

Message Passing Neural Network

$$m_v^{(t+1)} = \sum_{w \in N(v)} h_w^{(t)}$$

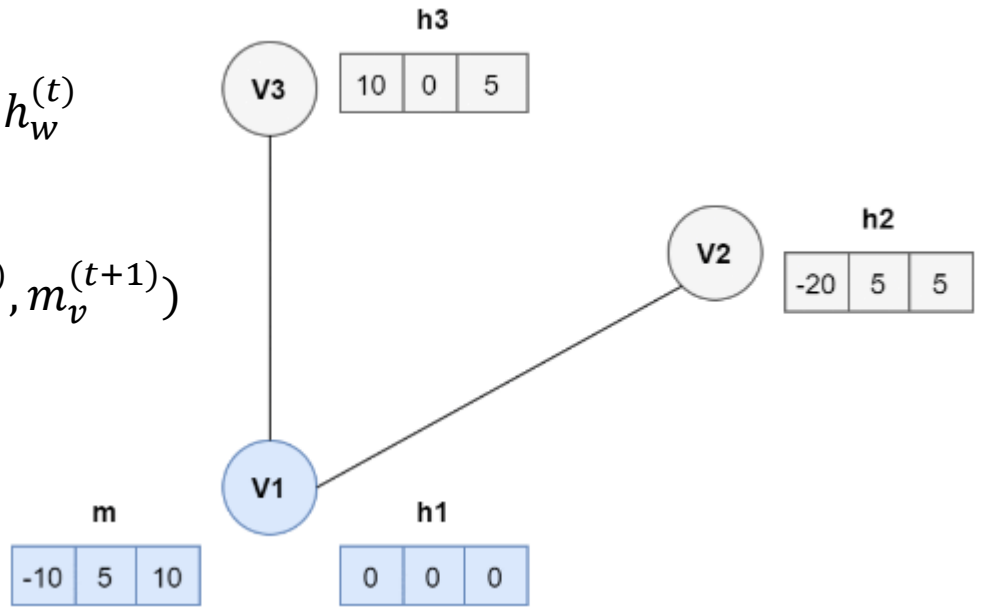
$$h_v^{(t+1)} = average(h_v^{(t)}, m_v^{(t+1)})$$



Message Passing Neural Network

$$m_v^{(t+1)} = \sum_{w \in N(v)} h_w^{(t)}$$

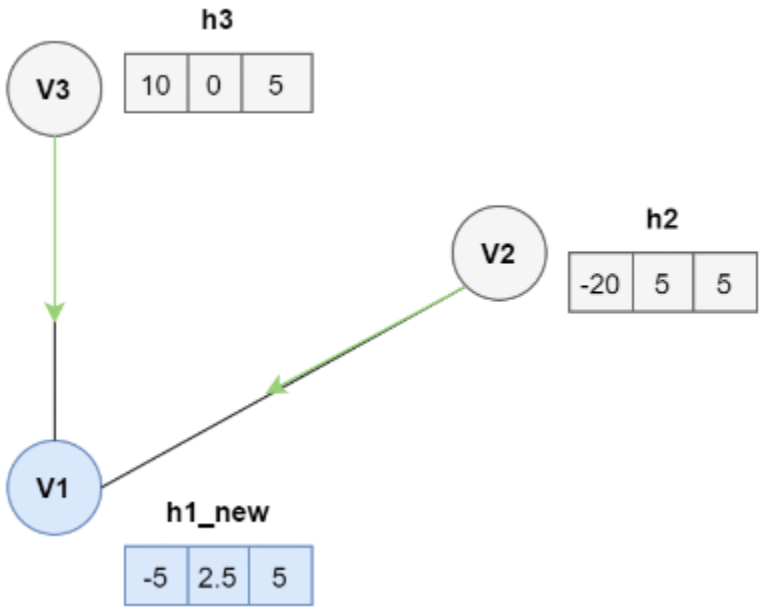
$$h_v^{(t+1)} = average(h_v^{(t)}, m_v^{(t+1)})$$



Message Passing Neural Network

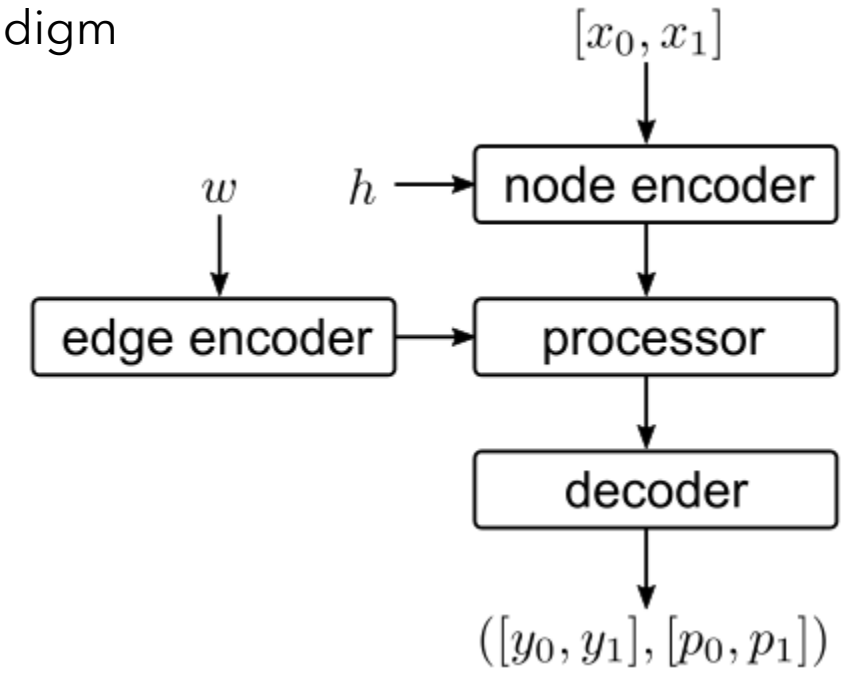
$$m_v^{(t+1)} = \sum_{w \in N(v)} h_w^{(t)}$$

$$h_v^{(t+1)} = average(h_v^{(t)}, m_v^{(t+1)})$$

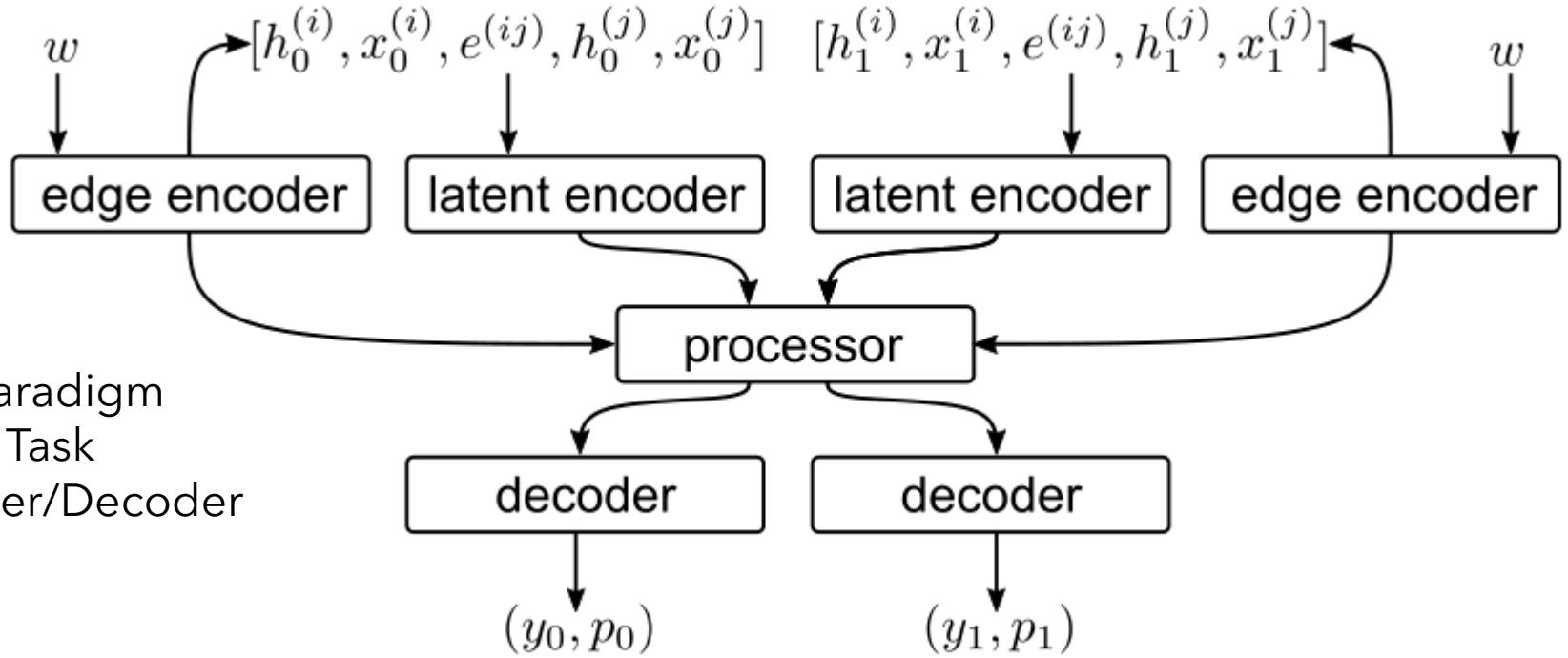


NeuralExecutor

- Encode-Process-Decode Paradigm
- Tasks share Encoder
- Linear Encoder/Decoder



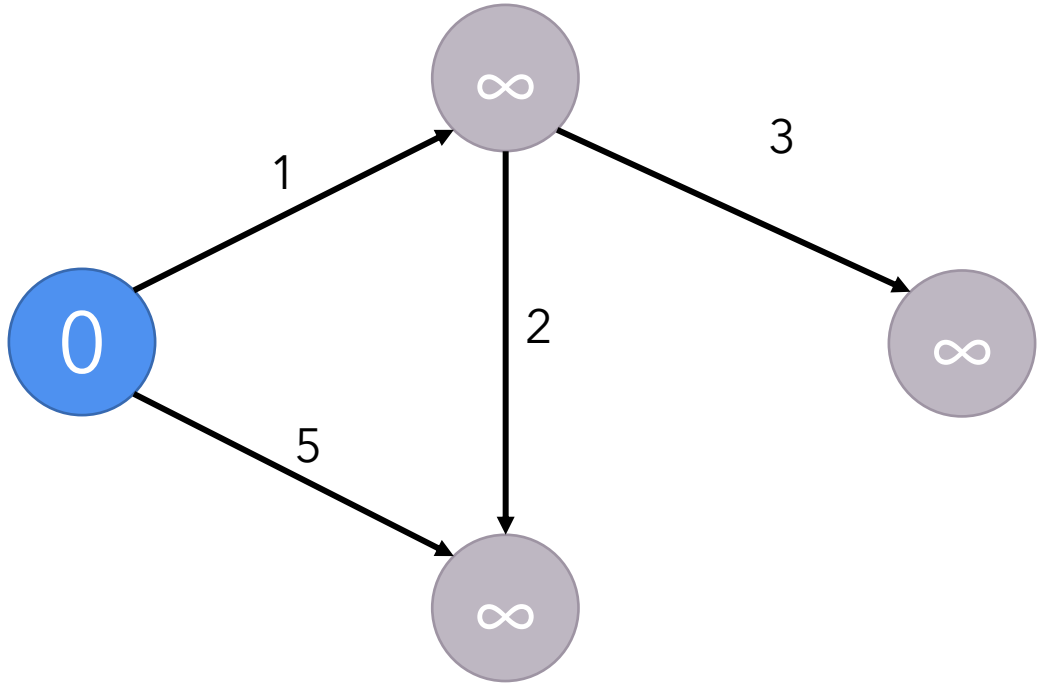
NeuralExecutor++



- Encode-Process-Decode Paradigm
- Separate Encoder for each Task
- Linear + Non-Linear Encoder/Decoder

Architecture

Example: Bellman-Ford

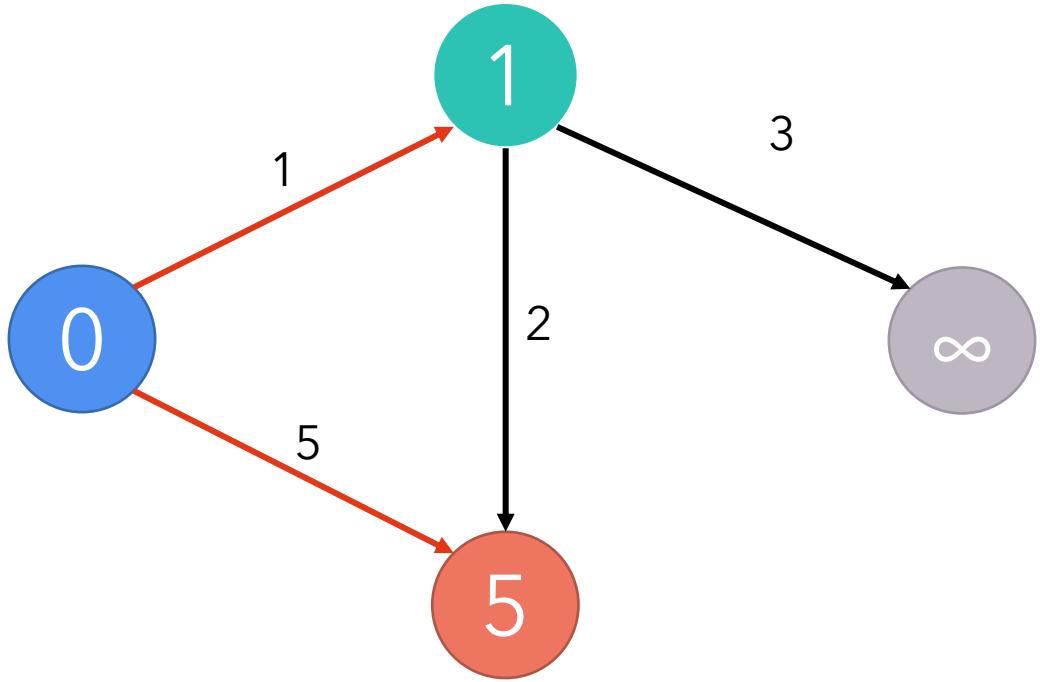


Position	(Node, Scalar)
Start Node	(Node, Mask_One)
Adj_Matrix	(Edge, Mask)
Edge Weights	(Edge, Scalar)

Distance	(Node, Scalar)
Mask	(Node, Mask_One)
Predecessor	(Edge, Mask)

Node Distances	(Node, Pointer)

Example: Bellman-Ford

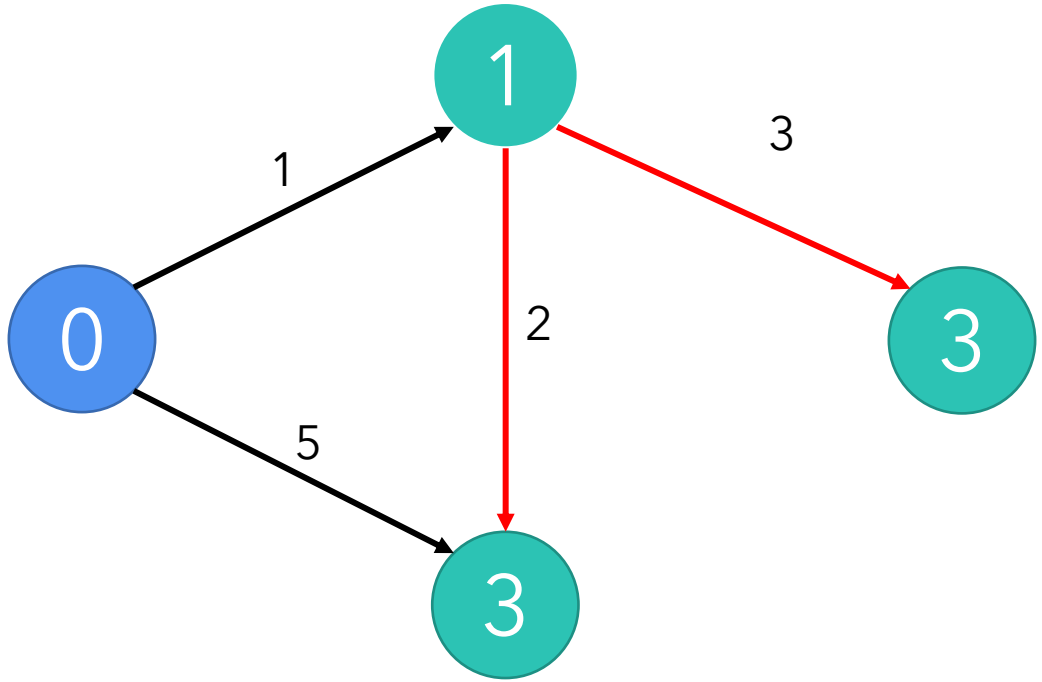


Position	(Node, Scalar)
Start Node	(Node, Mask_One)
Adj_Matrix	(Edge, Mask)
Edge Weights	(Edge, Scalar)

Distance	(Node, Scalar)
Mask	(Node, Mask_One)
Predecessor	(Edge, Mask)

Node Distances	(Node, Pointer)

Example: Bellman-Ford



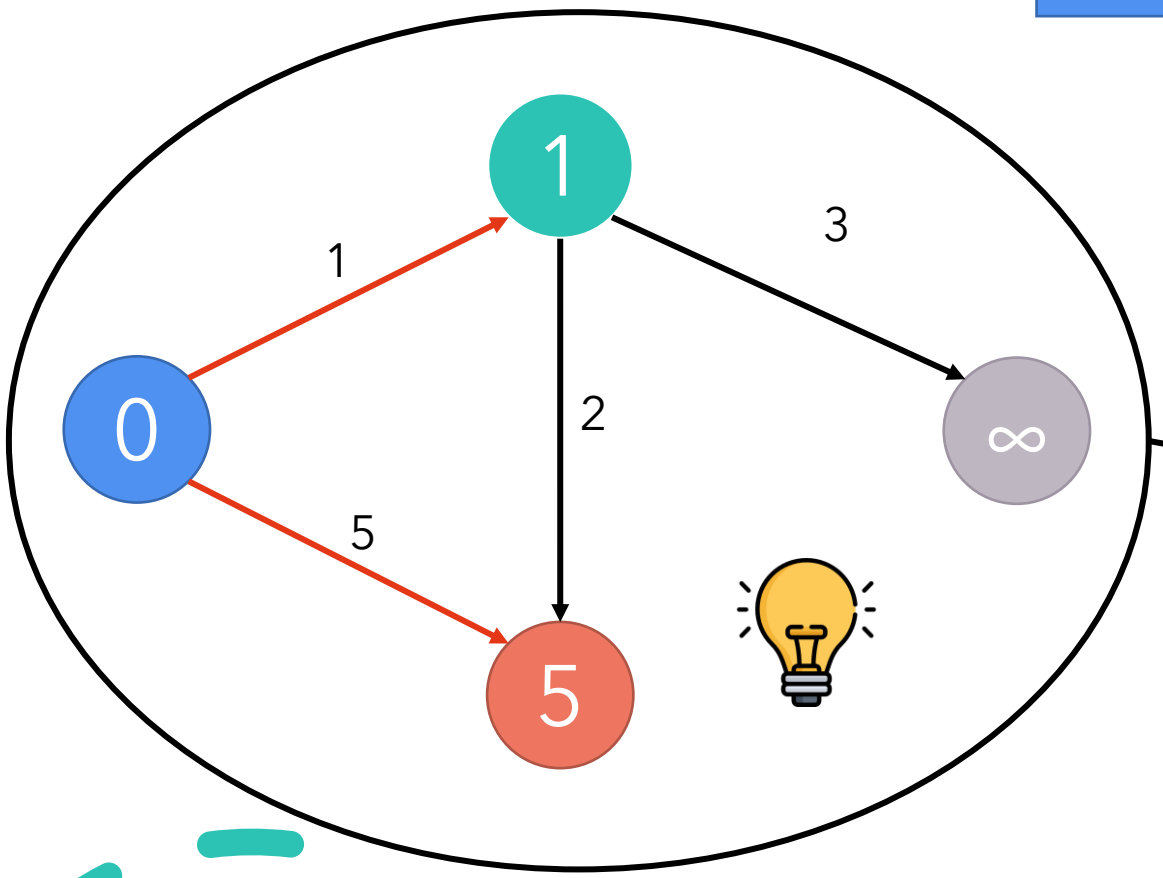
Position	(Node, Scalar)
Start Node	(Node, Mask_One)
Adj_Matrix	(Edge, Mask)
Edge Weights	(Edge, Scalar)

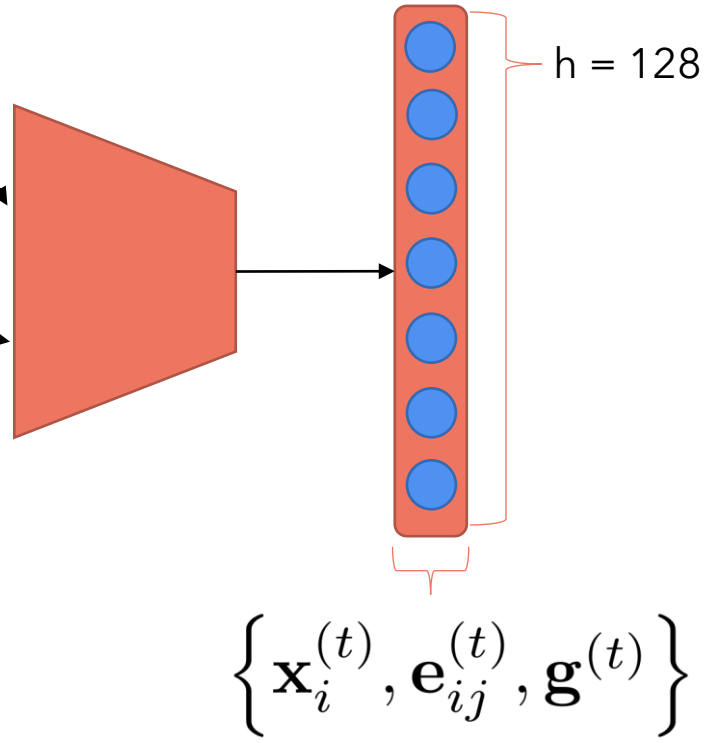
Distance	(Node, Scalar)
Mask	(Node, Mask_One)
Predecessor	(Edge, Mask)

Node Distances	(Node, Pointer)

Encoder

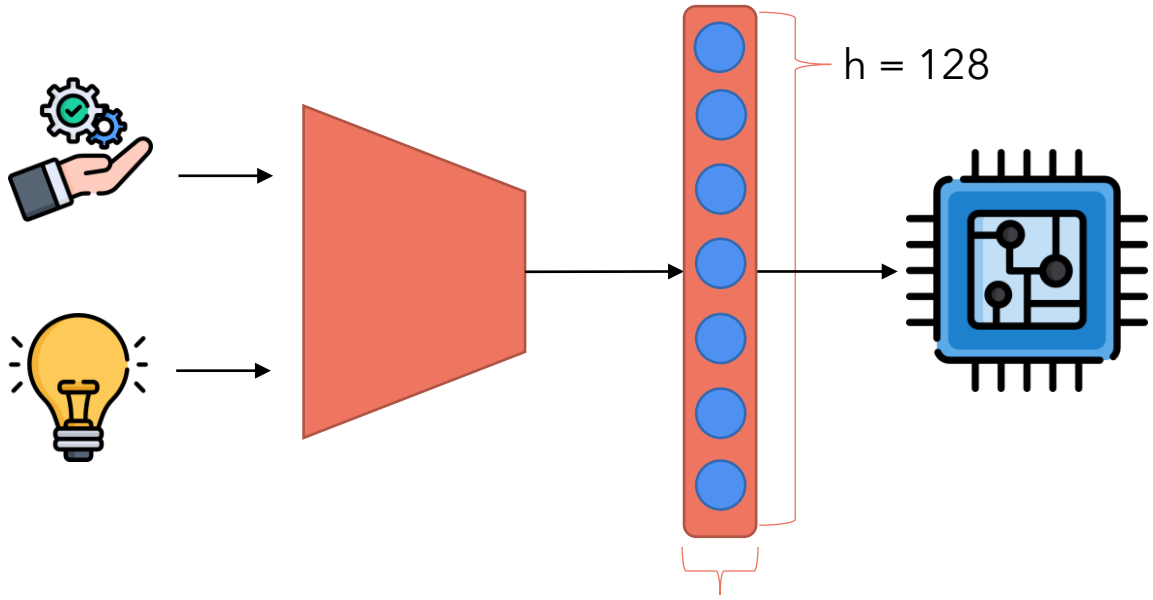
Position	(Node, Scalar)
Start Node	(Node, Mask_One)
Adj_Matrix	(Edge, Mask)
Edge Weights	(Edge, Scalar)





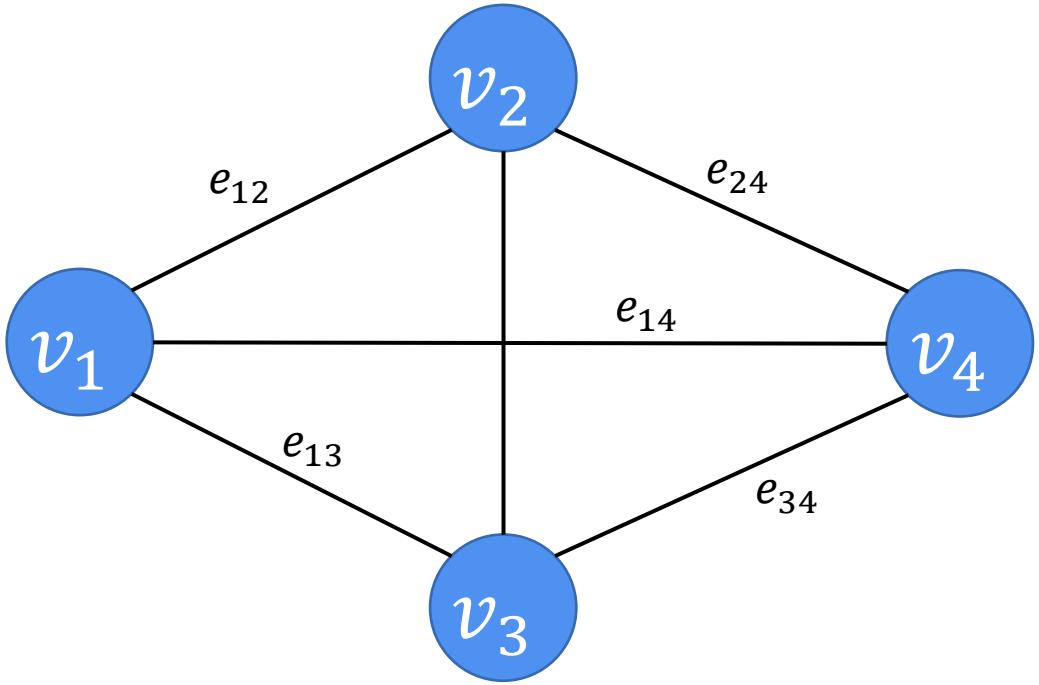
$$\left\{ \mathbf{x}_i^{(t)}, \mathbf{e}_{ij}^{(t)}, \mathbf{g}^{(t)} \right\}$$

Processor

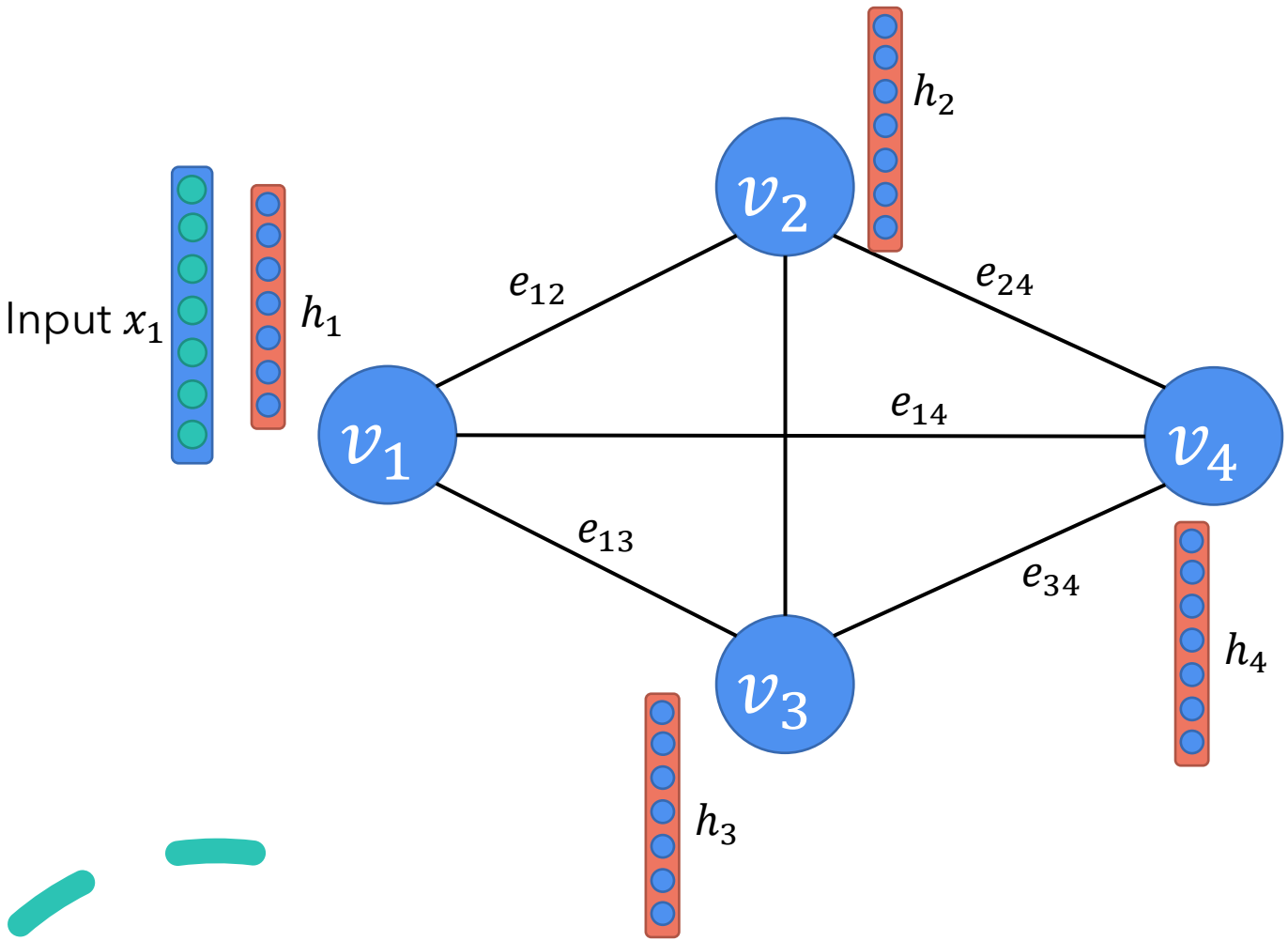


$$\left\{ \mathbf{x}_i^{(t)}, \mathbf{e}_{ij}^{(t)}, \mathbf{g}^{(t)} \right\}$$

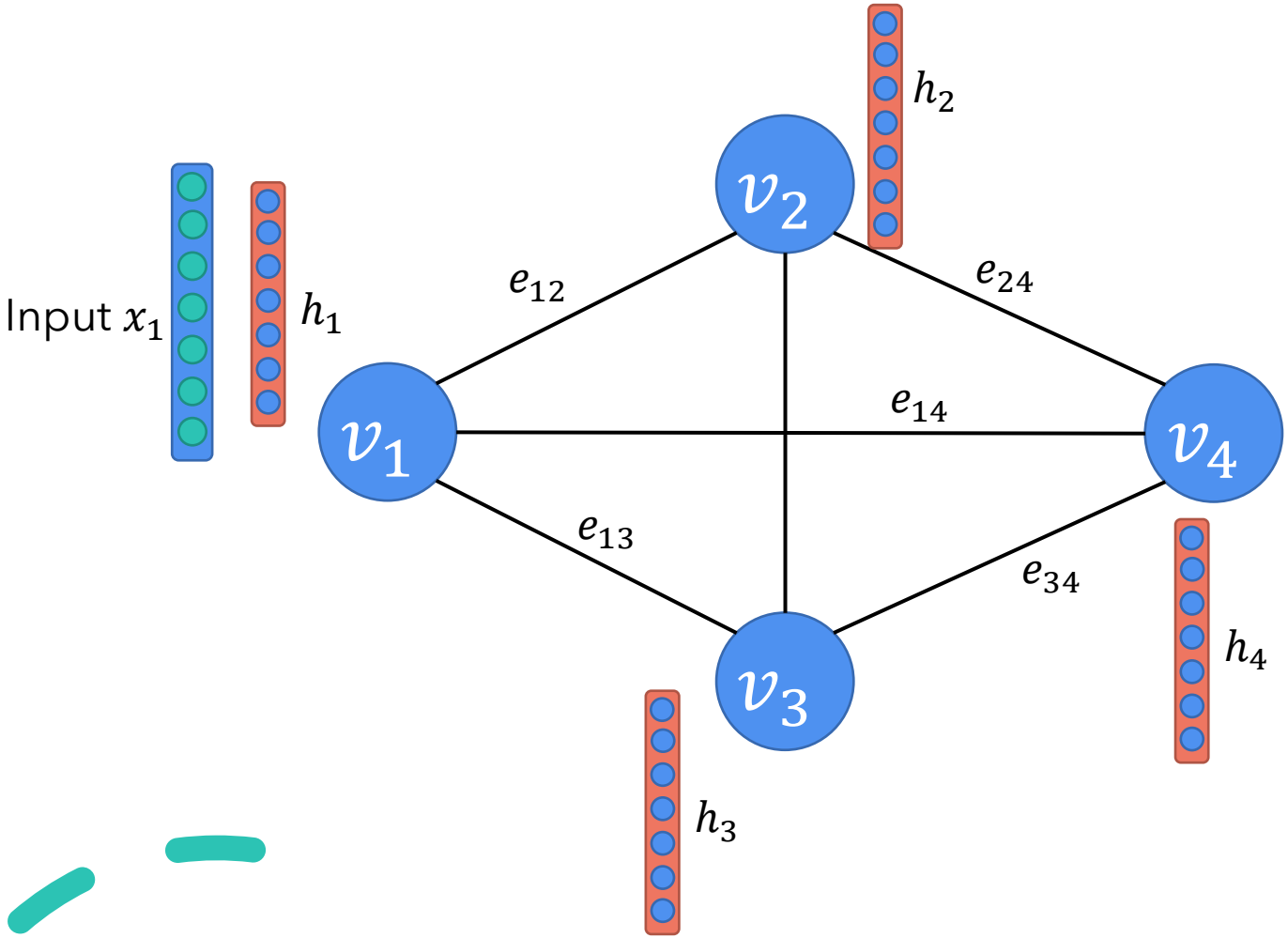
Inside the MPNN



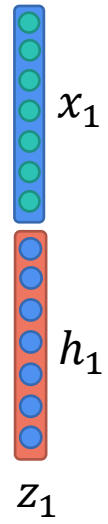
Inside the MPNN



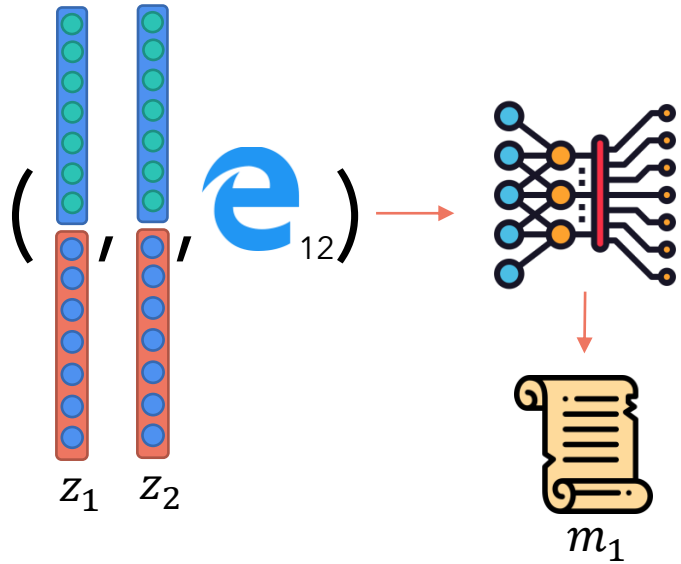
Inside the MPNN



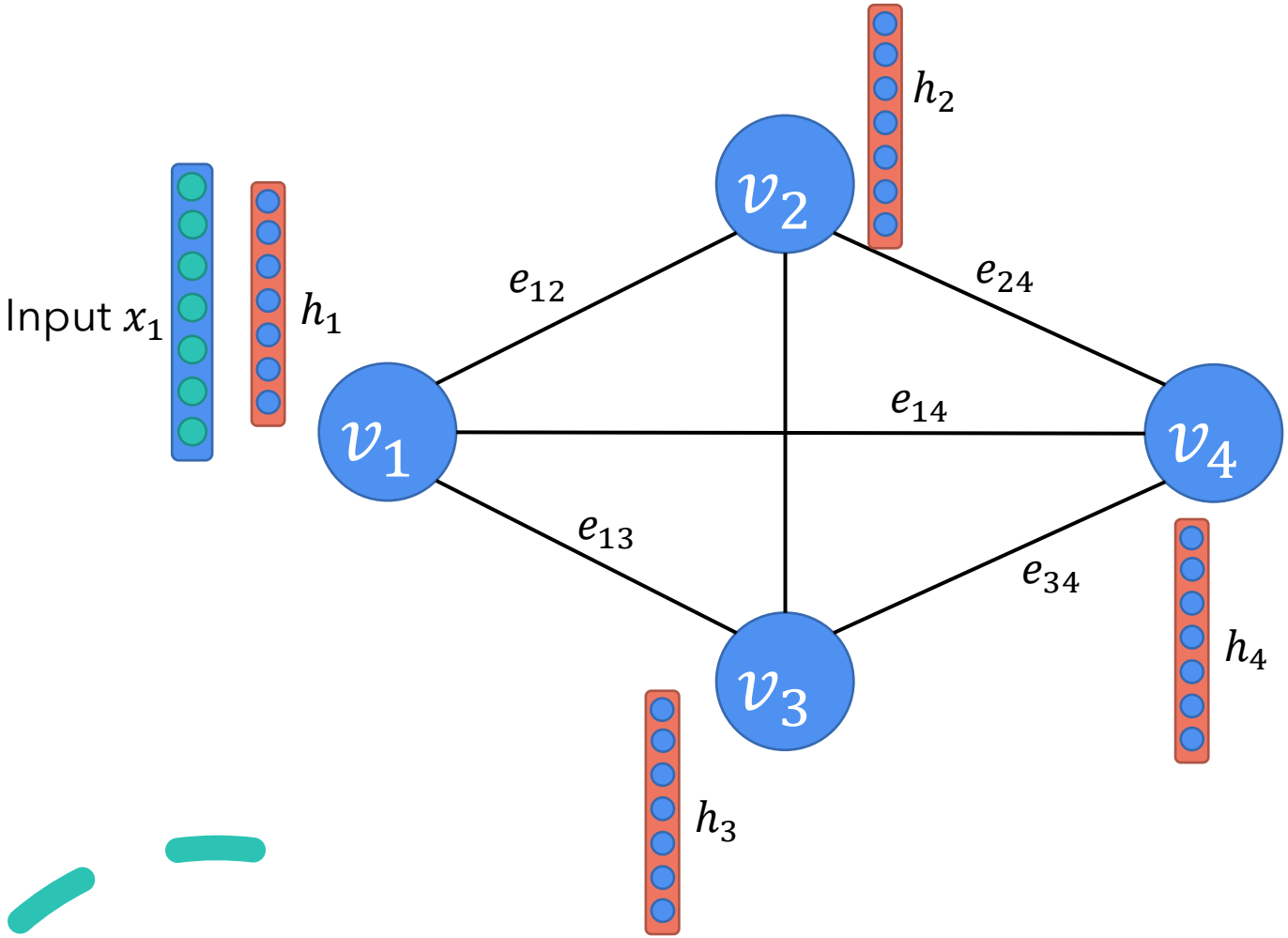
- Concatenation:



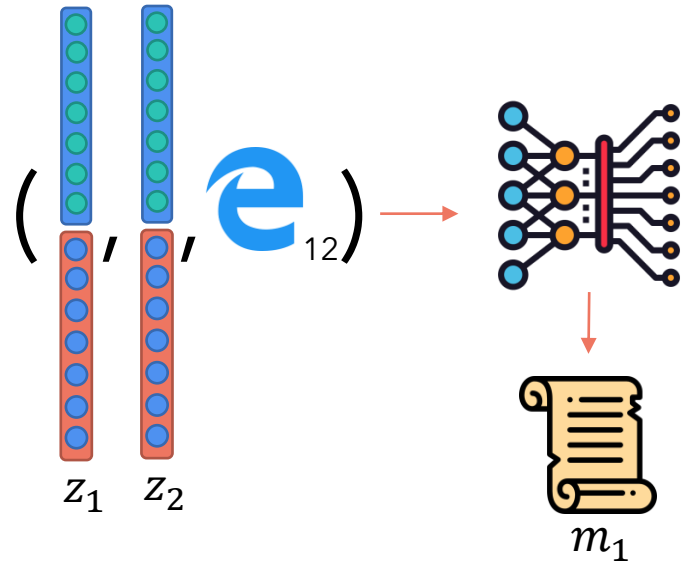
- Message:



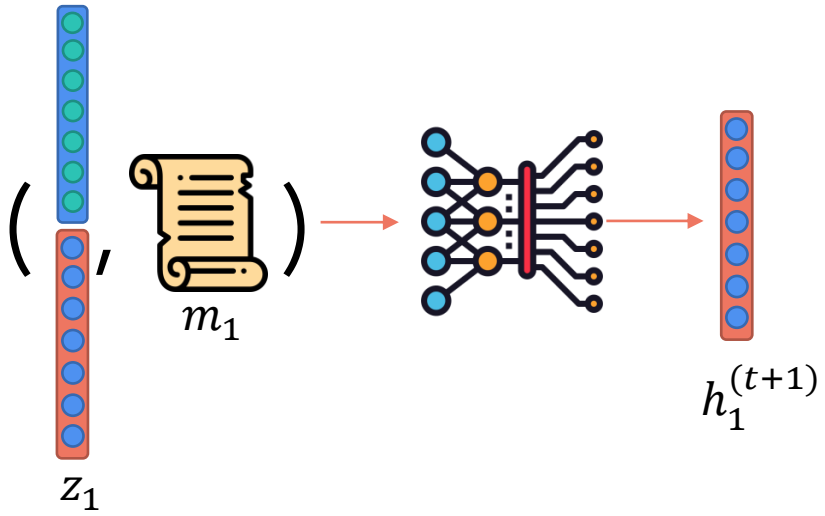
Inside the MPNN



- Message:

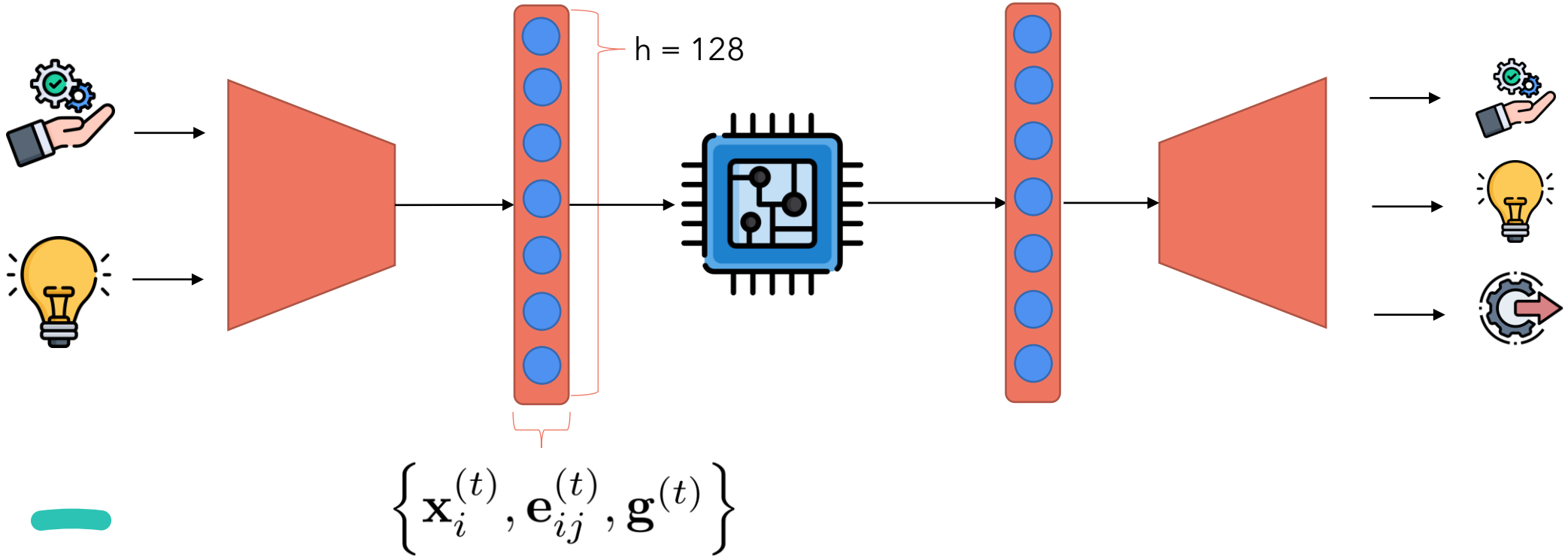


- Update:

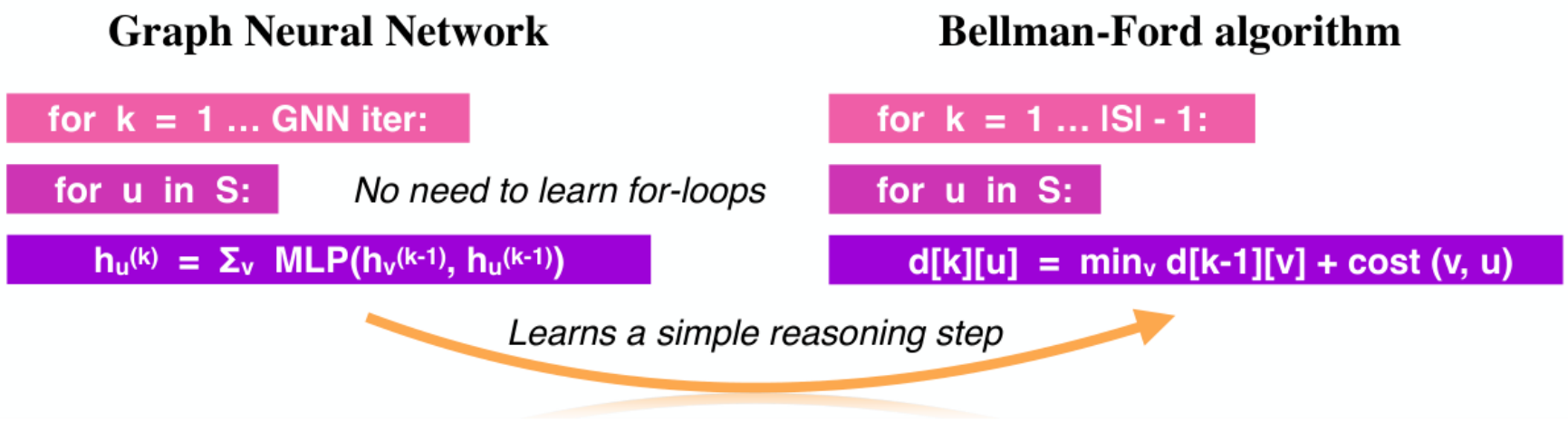


Encoder/Decoder

- Encode-Process-Decode Paradigm



Algorithmic Alignment



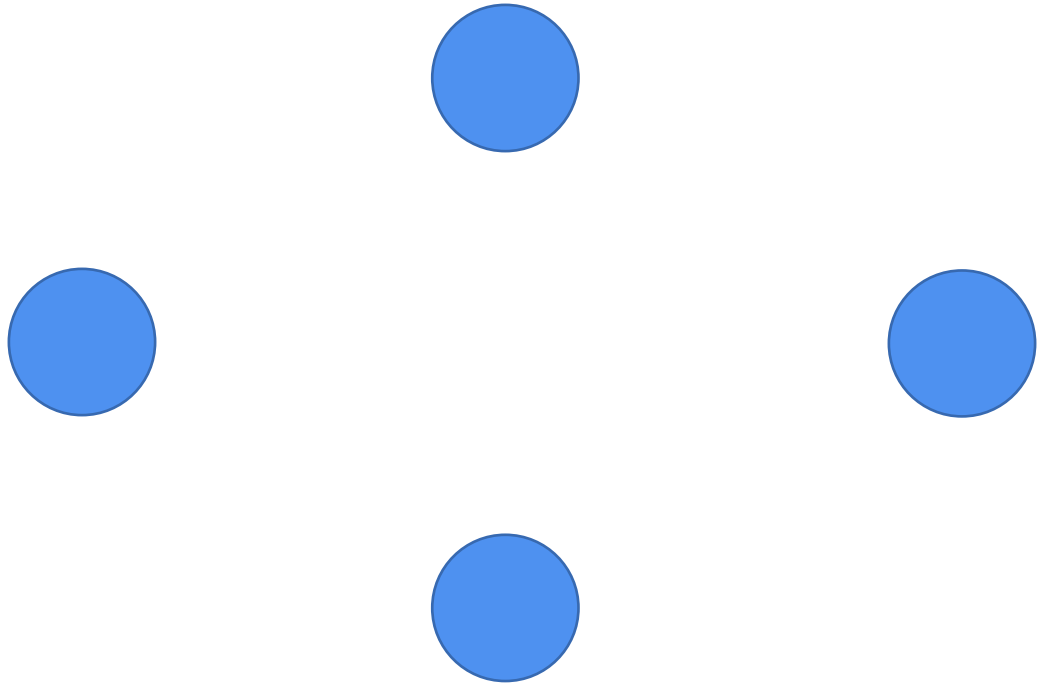
Model Improvements

Dataset and Training

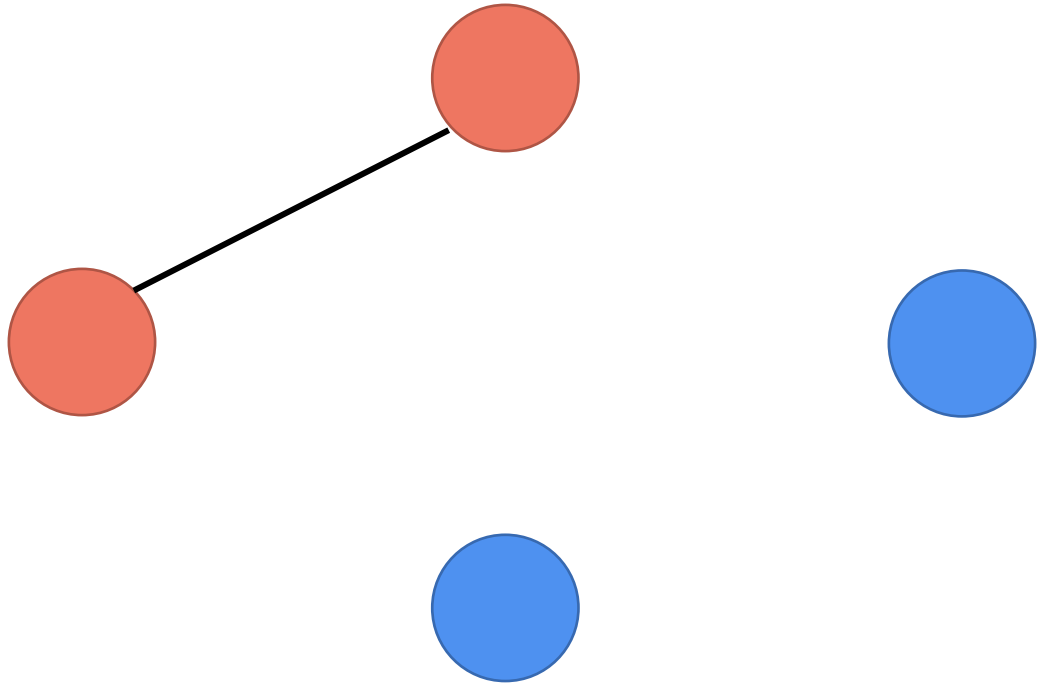
Dataset and Training

- Remove Teacher Forcing
  - At each step, feed back ground-truth hints with probability 0.5
- Data Augmentation
  - Online Samplers in CLRS for on the fly example generation
  - Train on examples of mixed sizes, $n \leq 16$

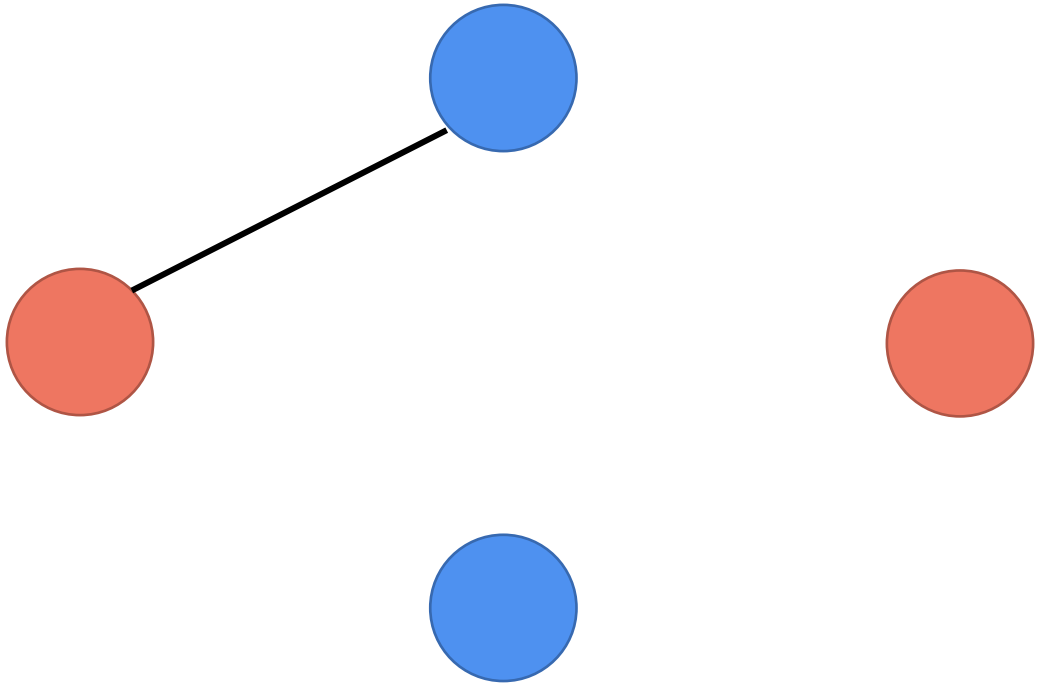
Data Augmentation: Erdős–Rényi Model



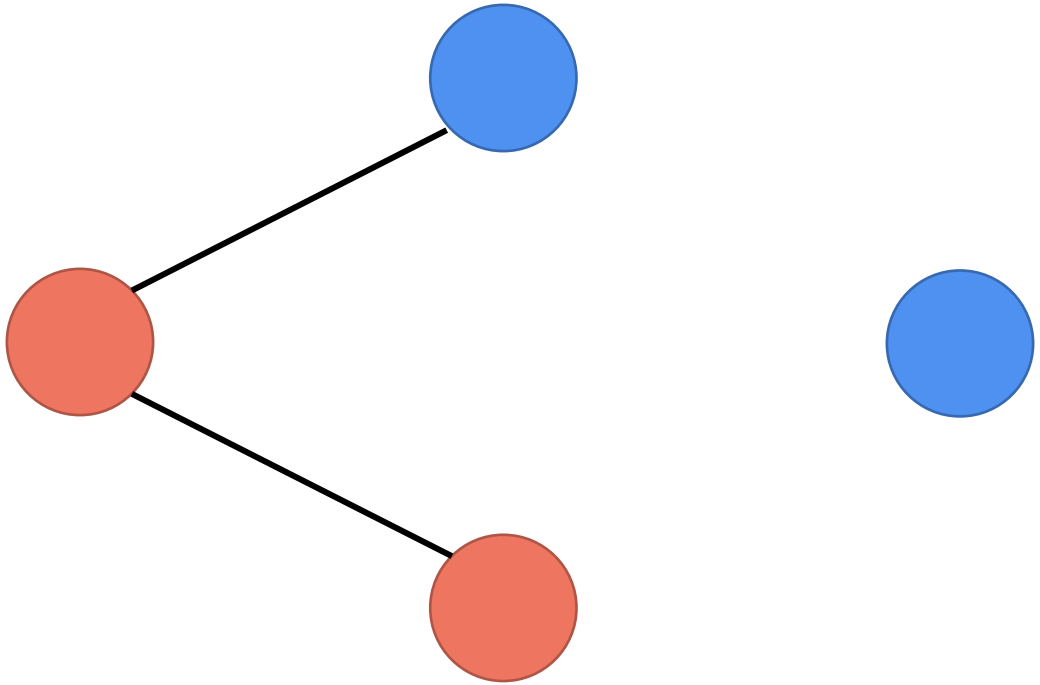
Data Augmentation: Erdős–Rényi Model



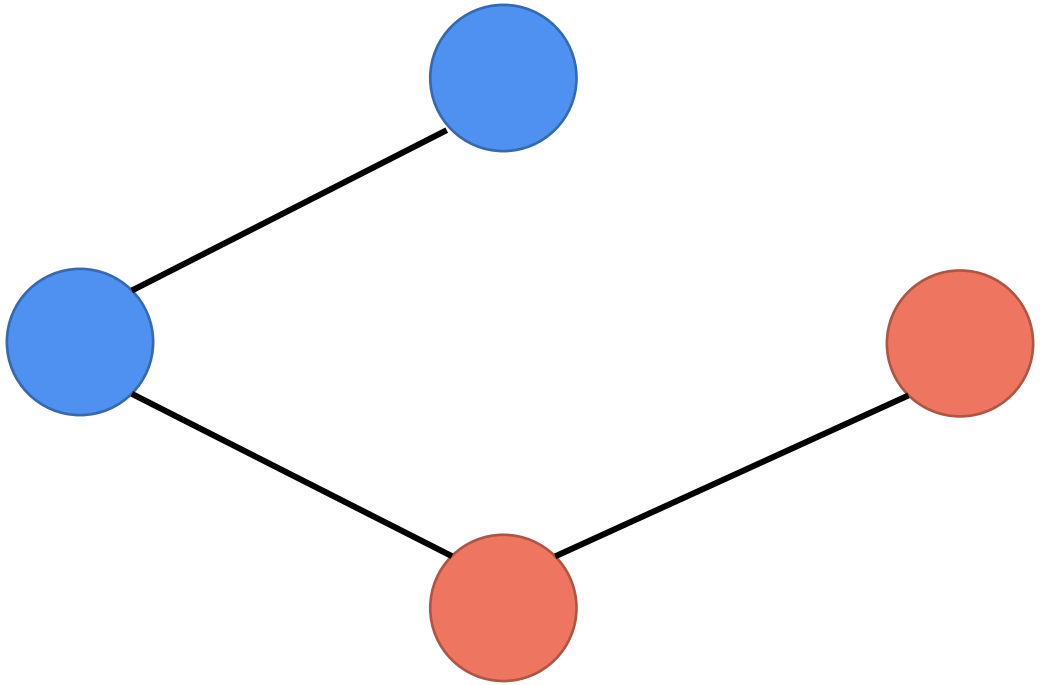
Data Augmentation: Erdős–Rényi Model



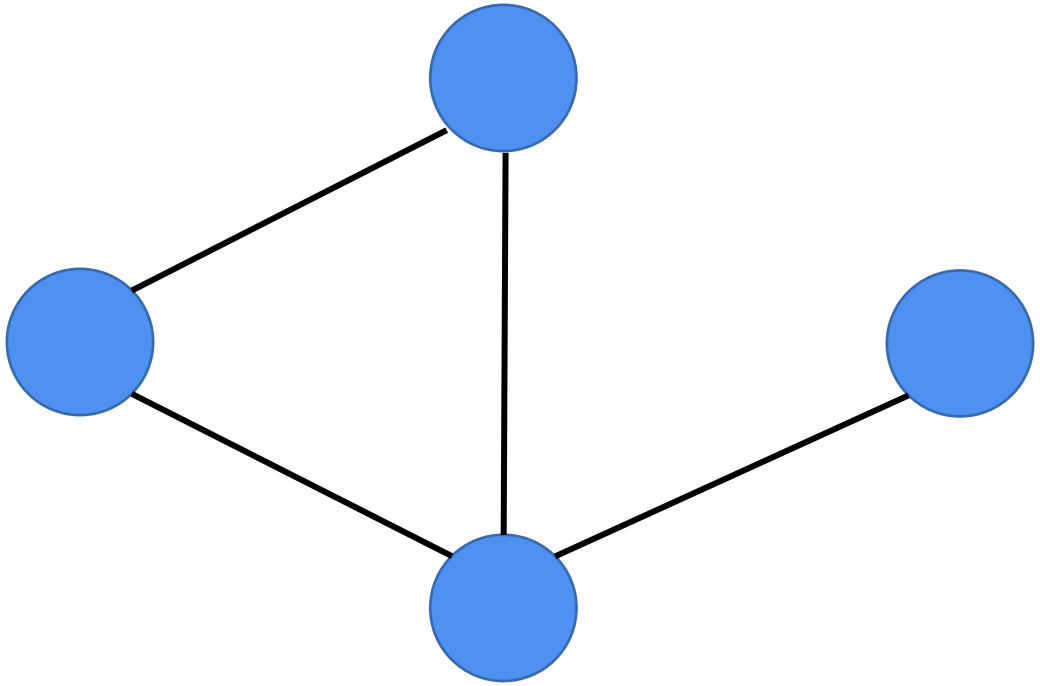
Data Augmentation: Erdős–Rényi Model



Data Augmentation: Erdős–Rényi Model



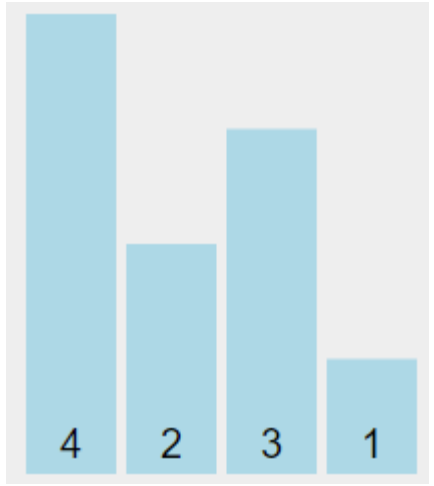
Data Augmentation: Erdős–Rényi Model



Dataset and Training

- Remove Teacher Forcing
  - At each step, feed back ground-truth hints with probability 0.5
- Data Augmentation
  - Online Samplers in CLRS for on the fly example generation
  - Train on examples of mixed sizes, $n \leq 16$
  - Randomized Position Scalars

CLRS-30 Example

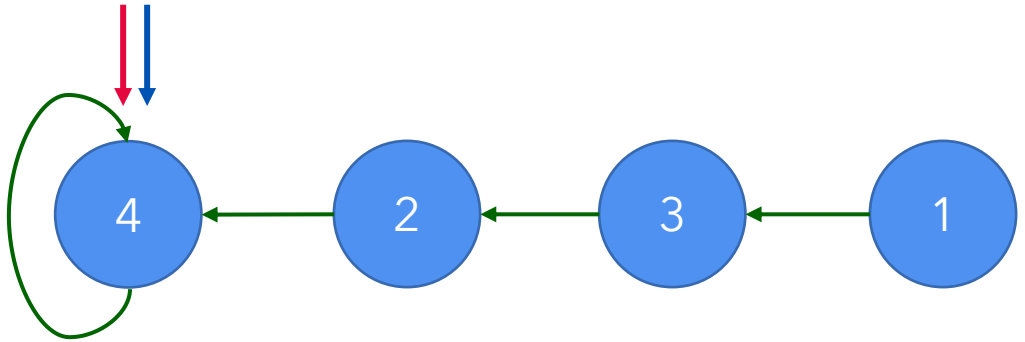


pos: [0,0.33,0.67,1]
key: [4, 2 , 3 ,1]

Input
Hint
Output

i: [1,0,0,0]
j: [1,0,0,0]
pred: [0,0.33,0.67,1]

[0,0.23,0.7,0.95]



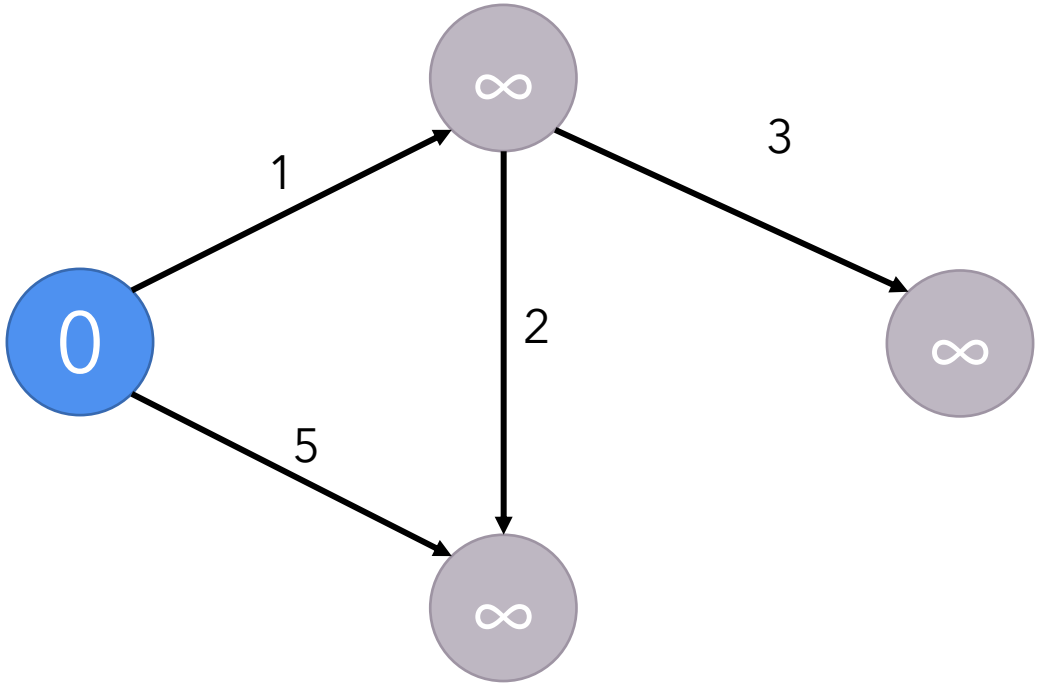
i (mask_one)
j (mask_one)
List State Specification (pointer)

Dataset and Training

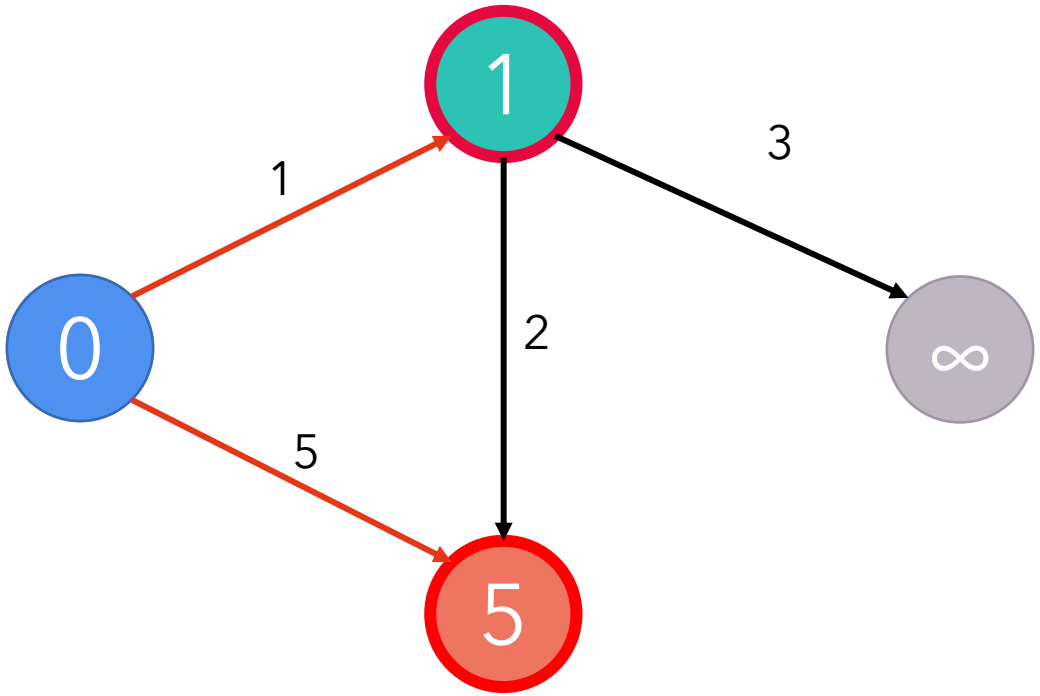
- Remove Teacher Forcing
  - At each step, feed back ground-truth hints with probability 0.5
- Data Augmentation
  - Online Samplers in CLRS for on the fly example generation
  - Train on examples of mixed sizes, $n \leq 16$
  - Randomized Position Scalars
- Hint Adjustments
  - Soft hint propagation
  - Static hint elimination
- Gradient Clipping
- Encoder Initialization

Processor Network

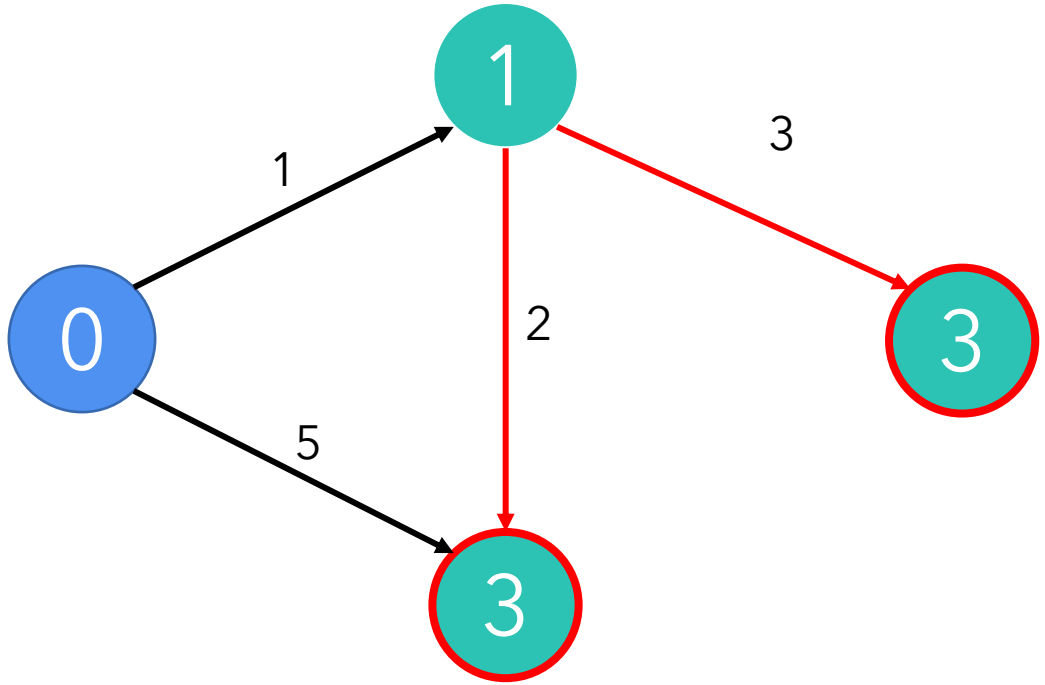
Gating



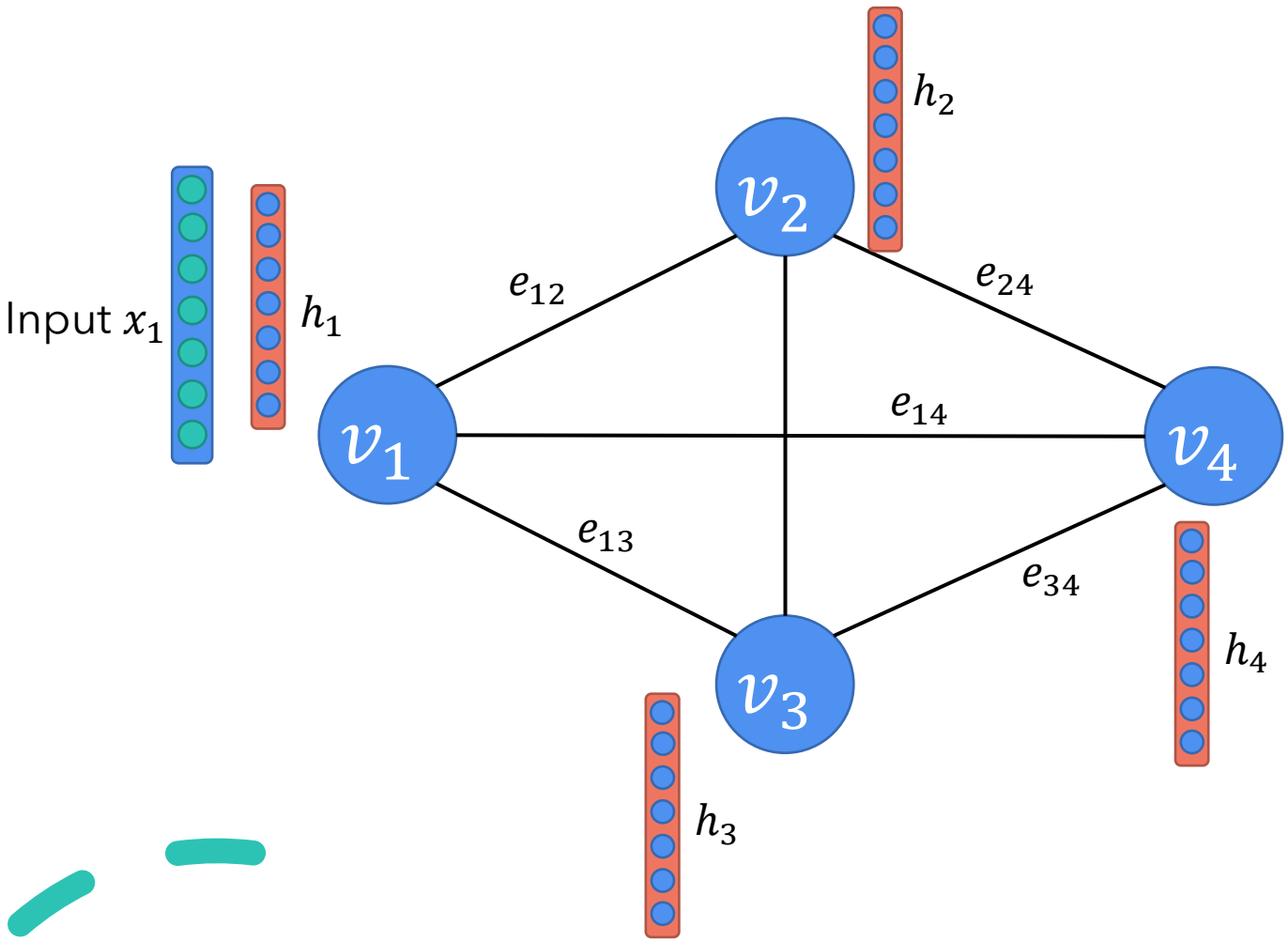
Gating



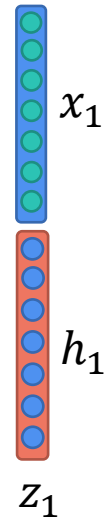
Gating



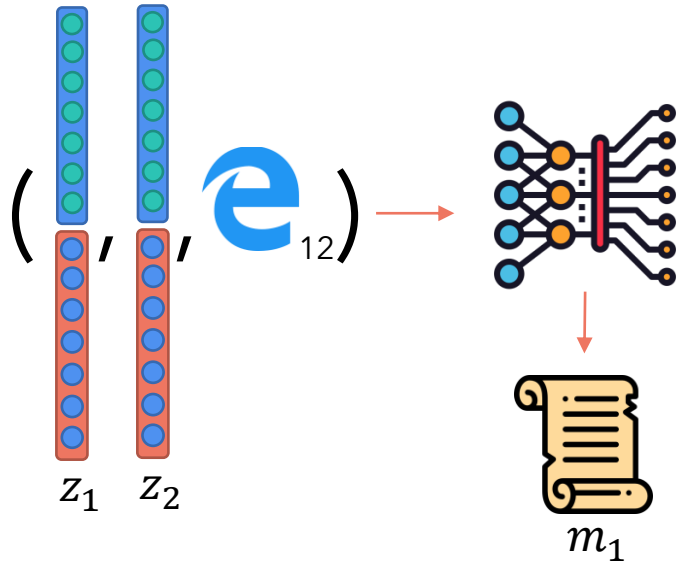
Gating



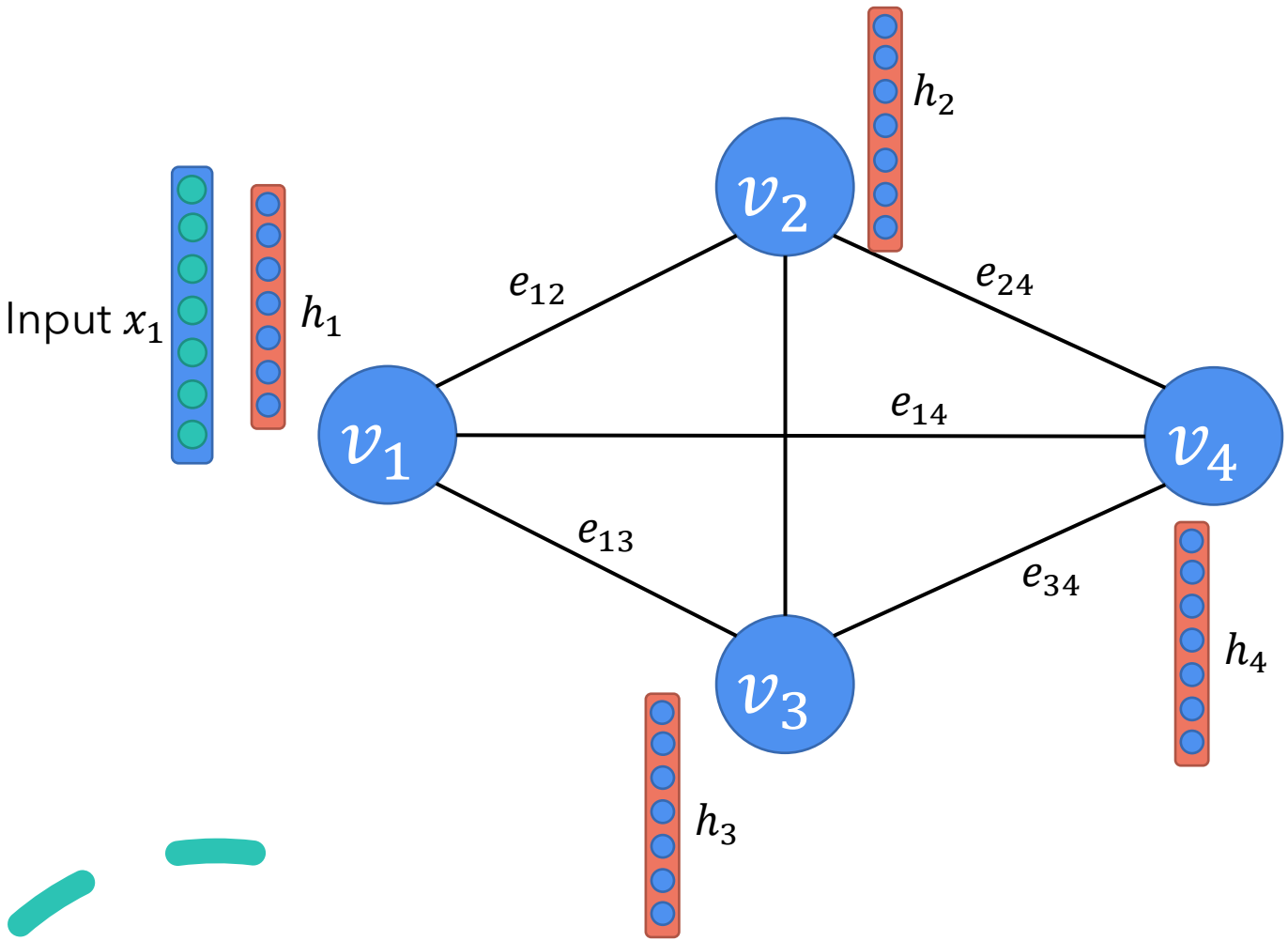
- Concatenation:



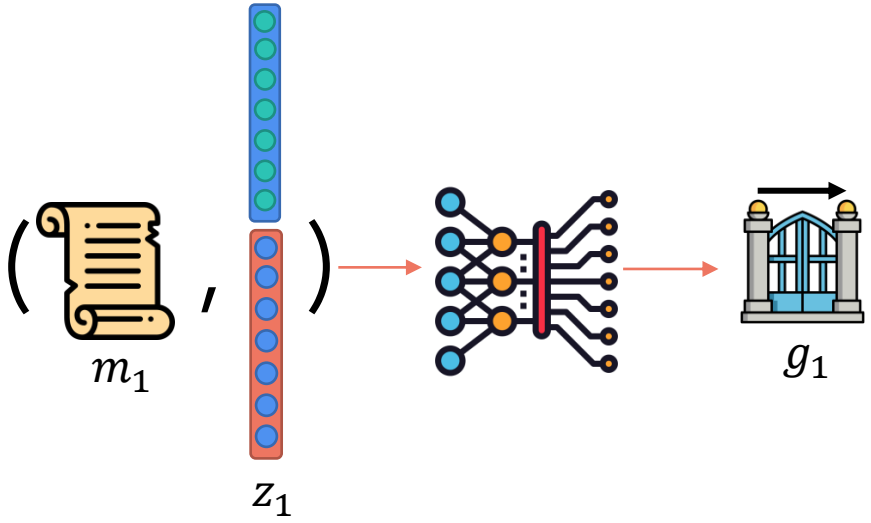
- Message:



Gating



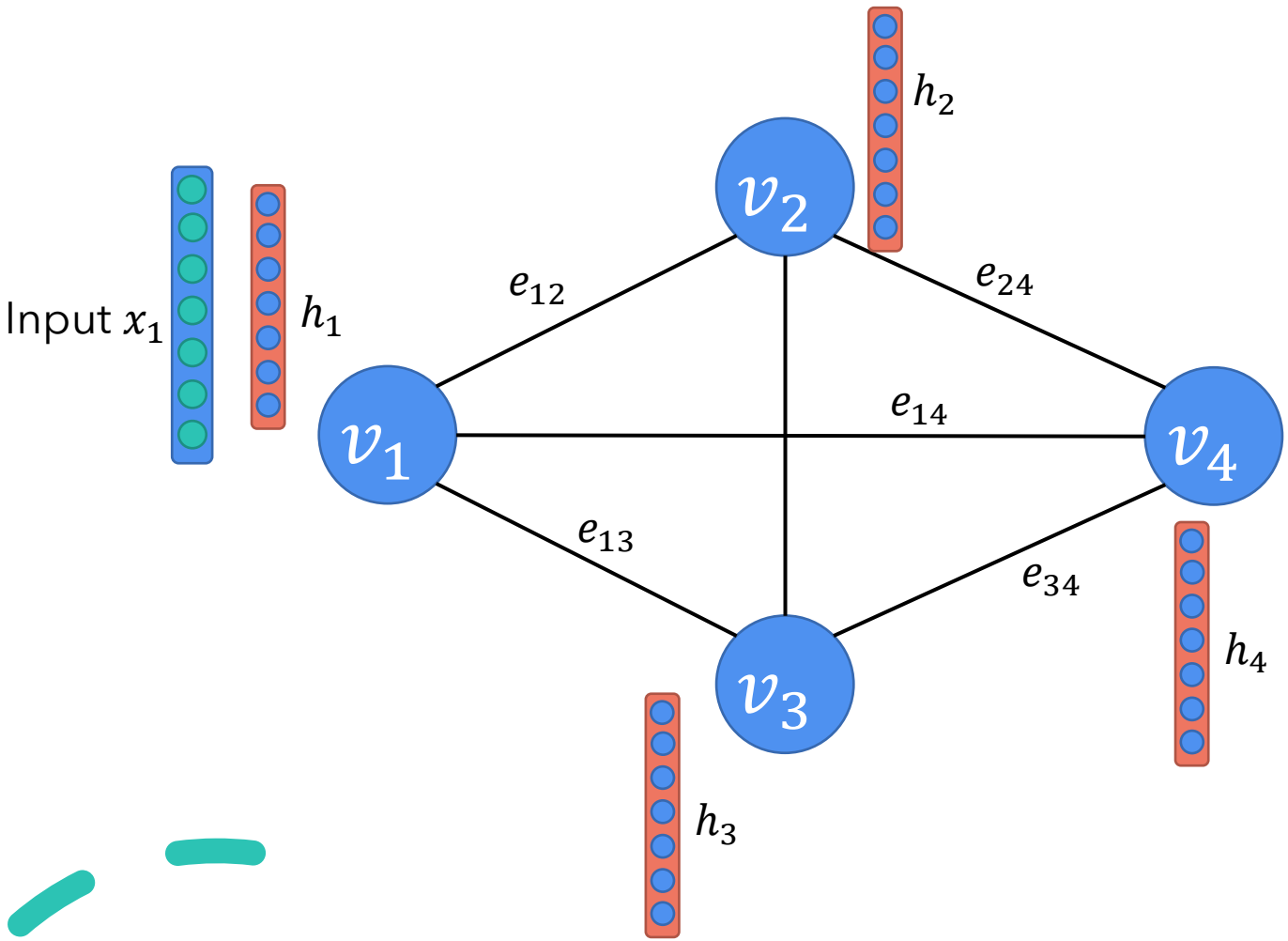
- Gating Vector:



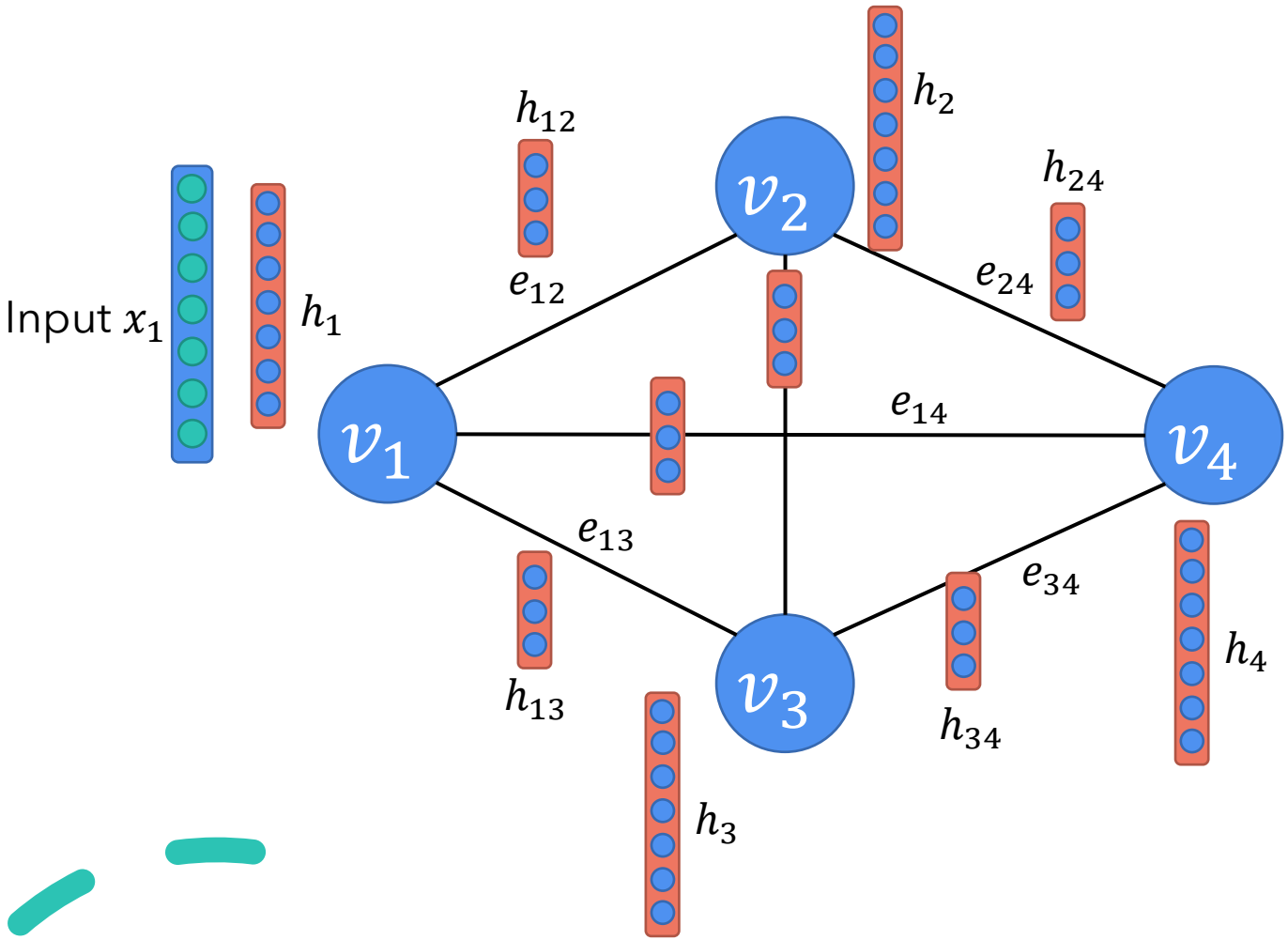
- Update:

$$\widehat{\mathbf{h}}_i^{(t)} = \mathbf{g}_i^{(t)} \odot \mathbf{h}_i^{(t)} + (1 - \mathbf{g}_i^{(t)}) \odot \mathbf{h}_i^{(t-1)}$$

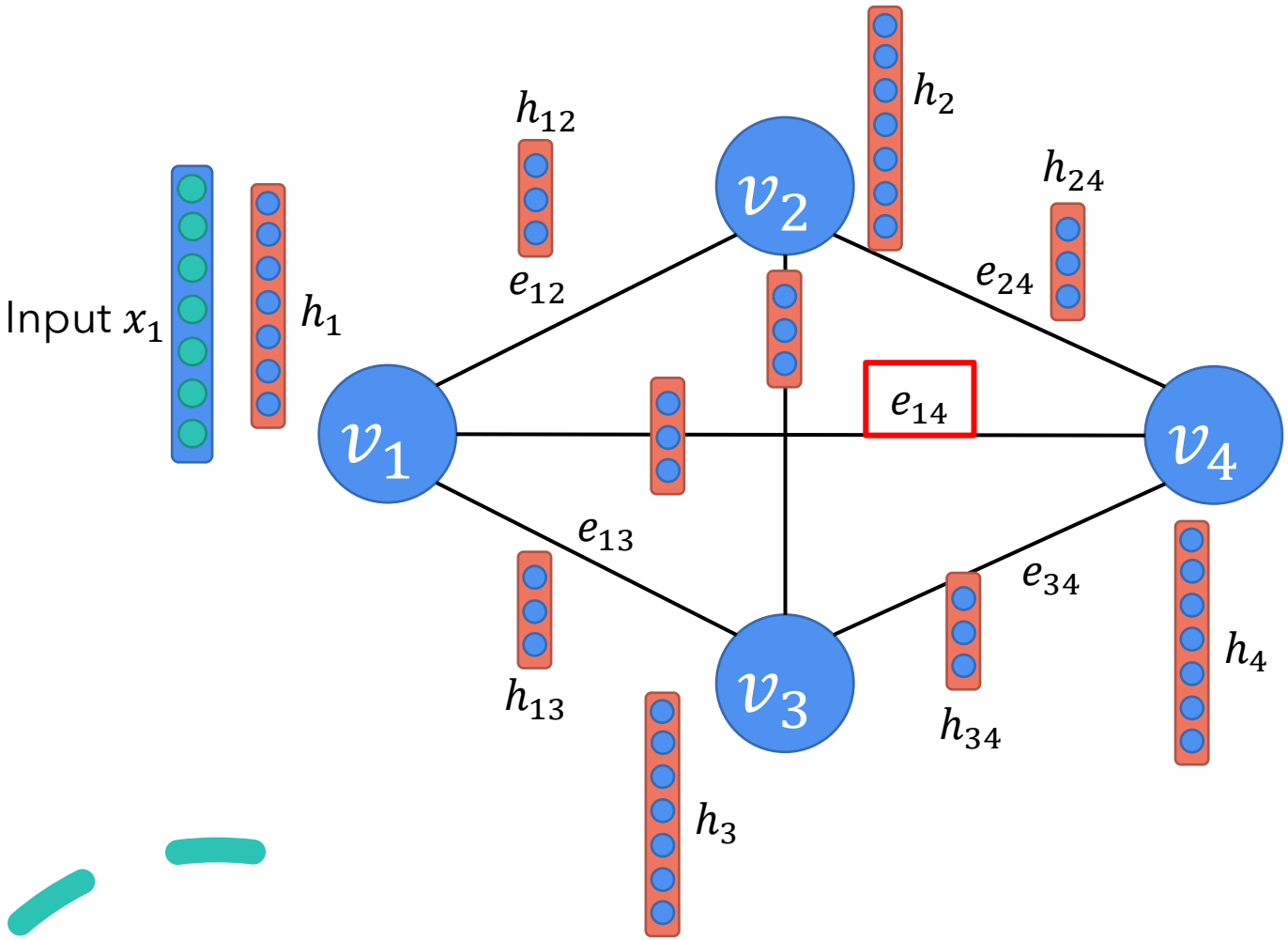
Triplet Reasoning



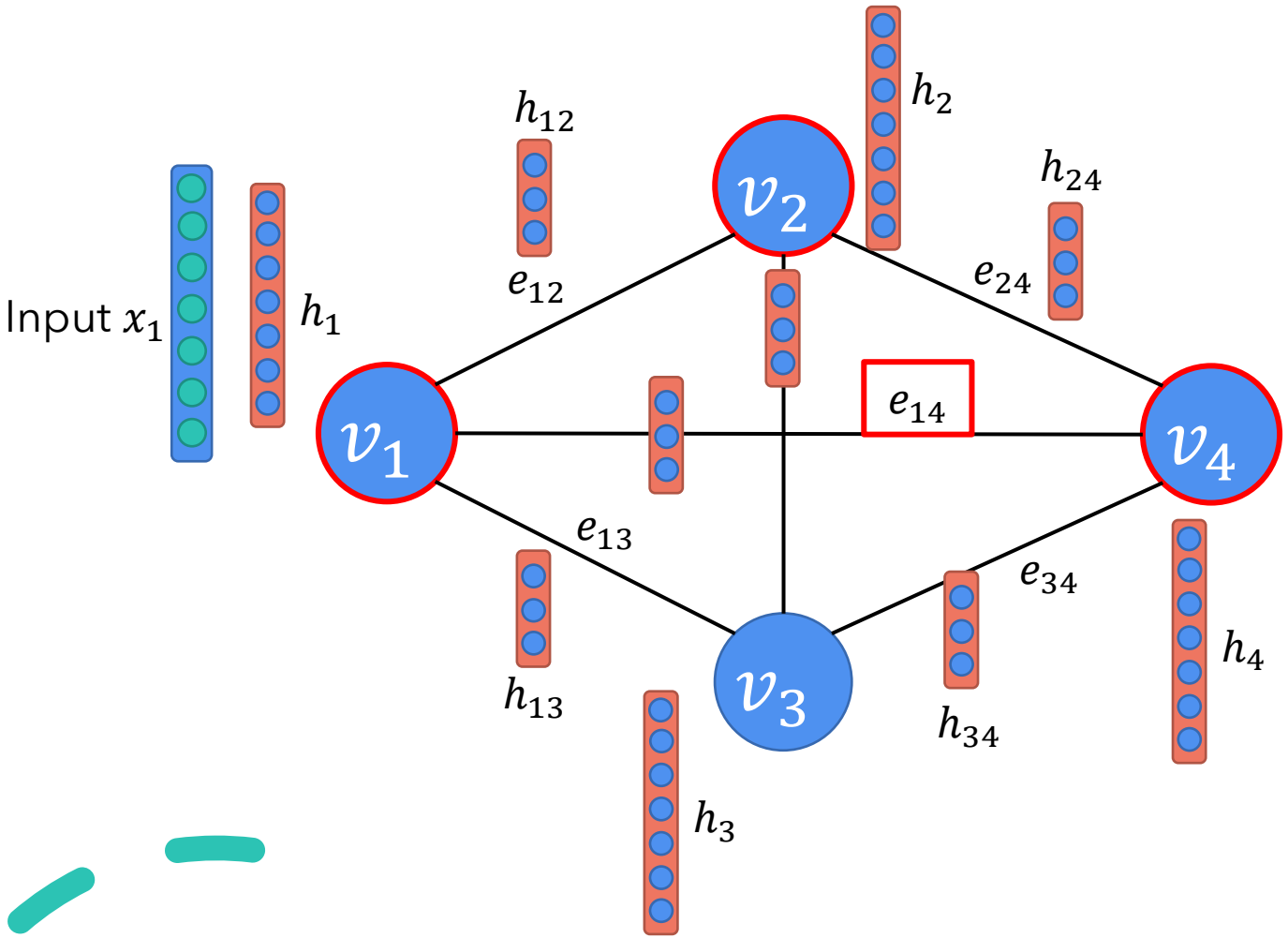
Triplet Reasoning



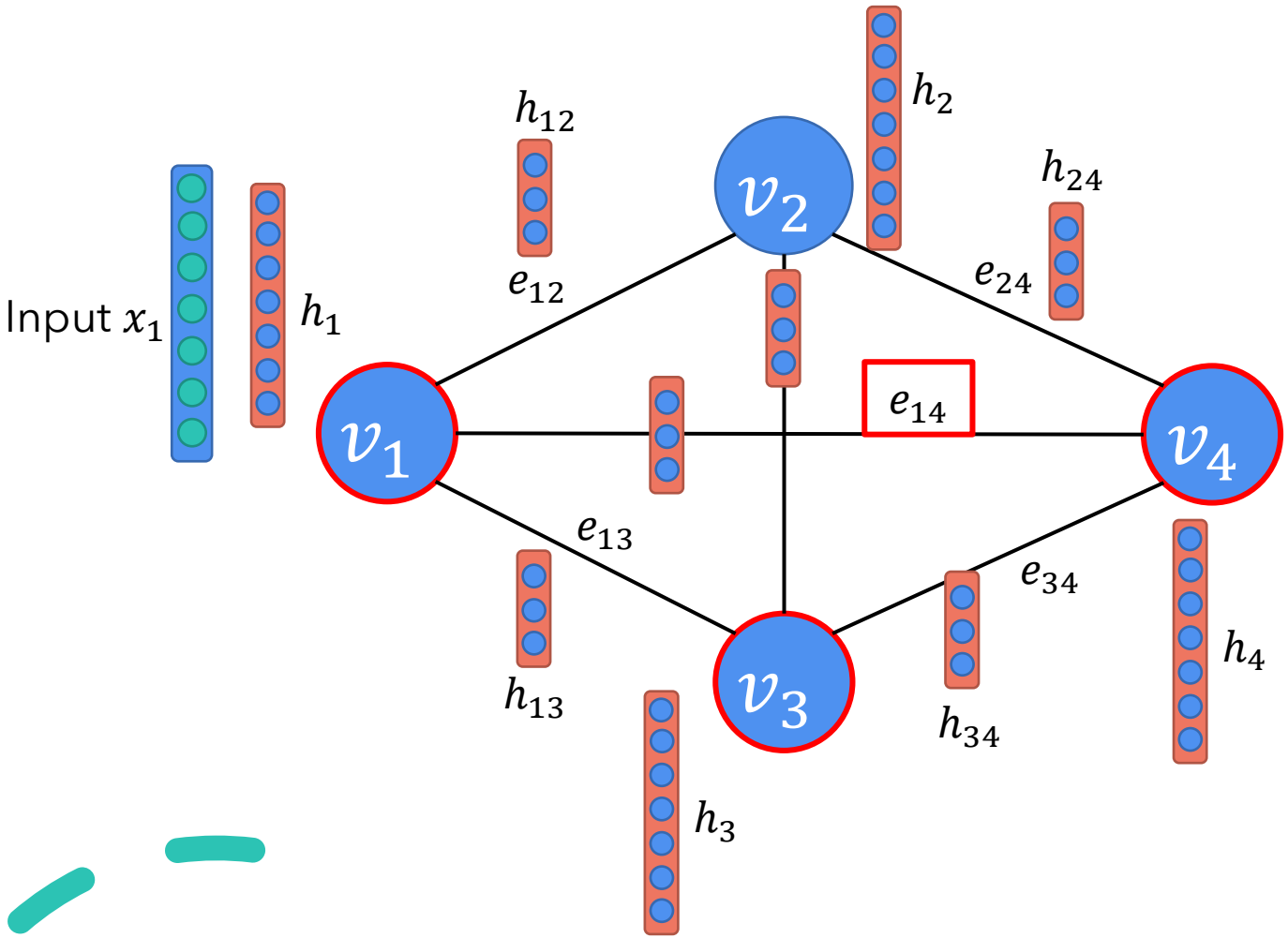
Triplet Reasoning

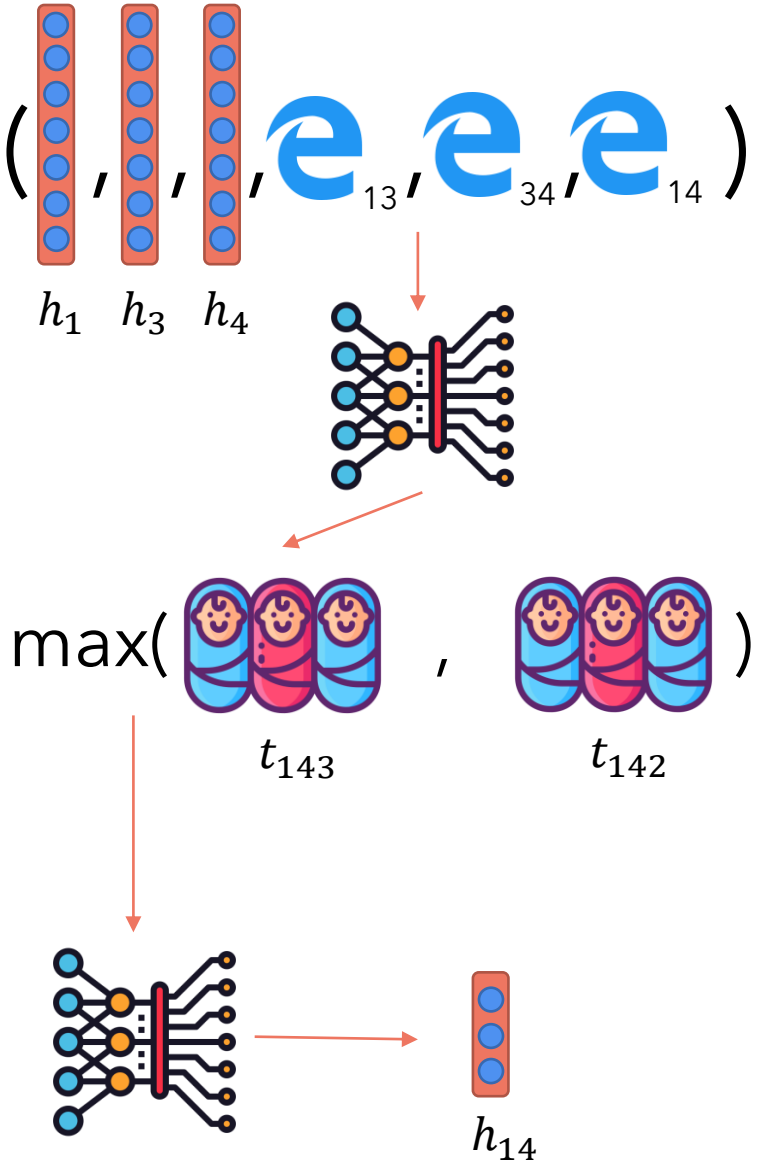


Triplet Reasoning

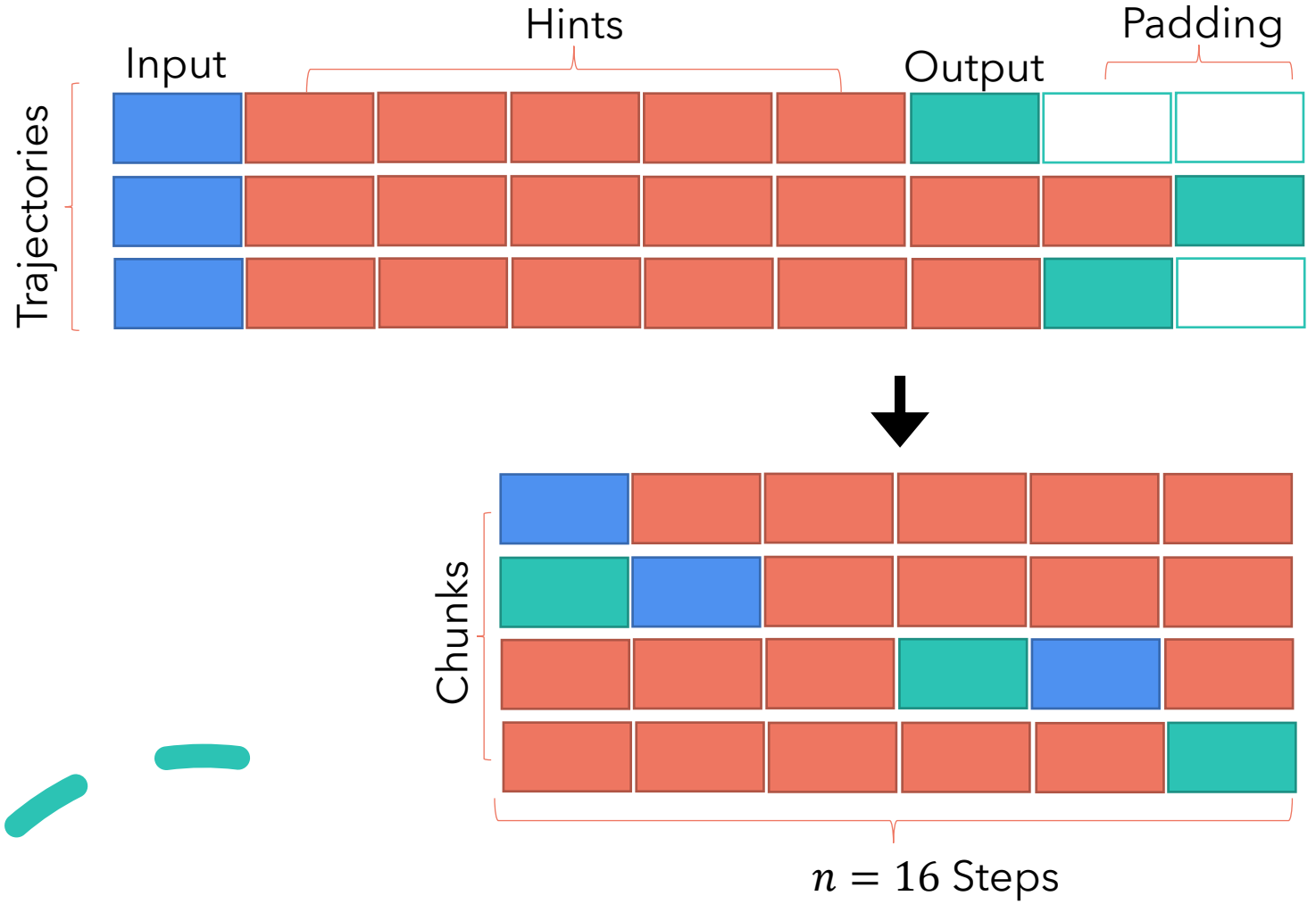


Triplet Reasoning





Multi-Task Improvement: Chunking



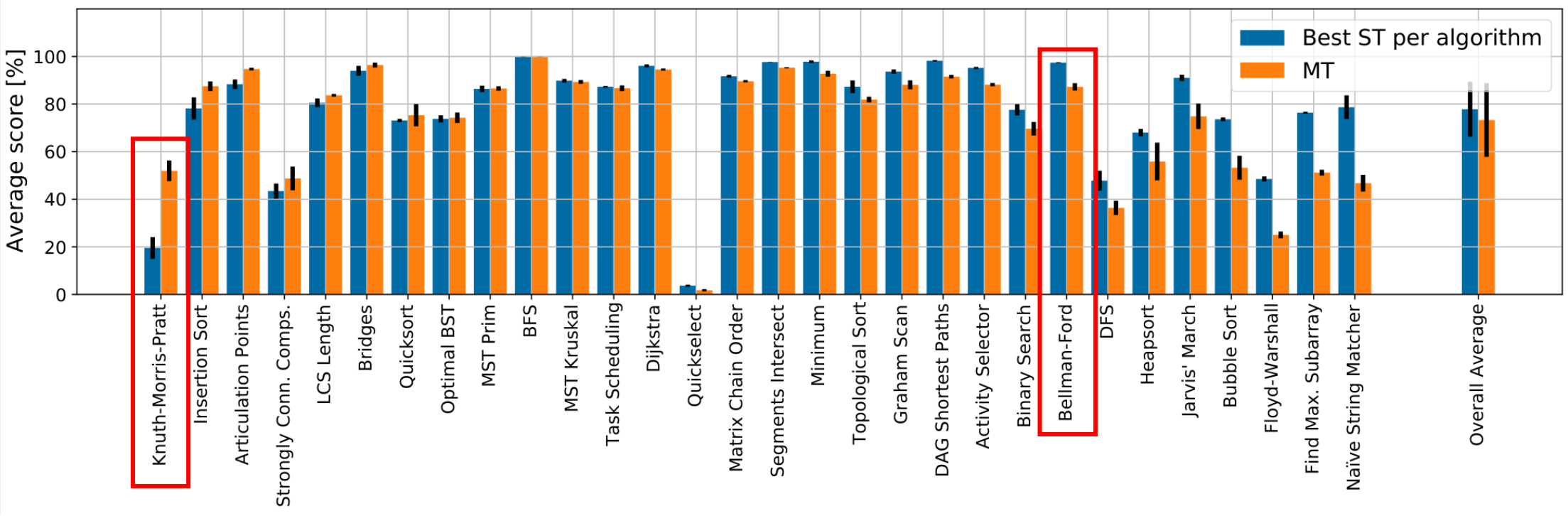
Evaluation

Single-Task CLRS-30 Benchmark

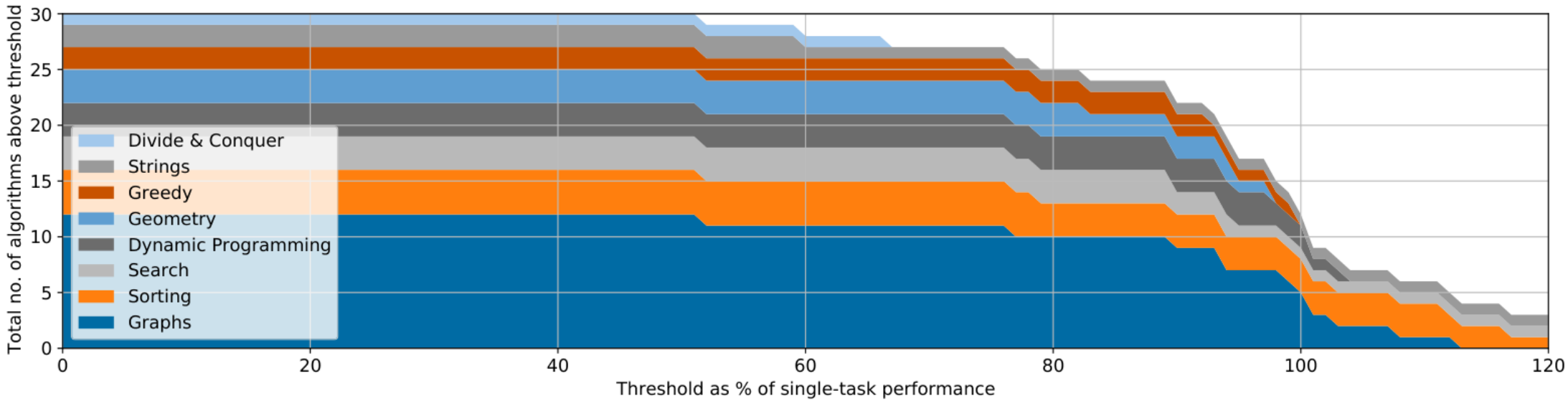
- Train on samples with $n \leq 16$, periodically evaluate on $n = 64$
- Given % scores are micro-F1 score: $\frac{TP}{TP+\frac{1}{2}(FP+FN)}$

Alg. Type	Memnet [5]	MPNN [5]	PGN [5]	Triplet-GMPNN (ours)
Div. & C.	$13.05\% \pm 0.14$	$20.30\% \pm 0.85$	$65.23\% \pm 4.44$	$\mathbf{76.36\% \pm 1.34}$
DP	$67.94\% \pm 8.20$	$65.10\% \pm 6.44$	$70.58\% \pm 6.48$	$\mathbf{81.99\% \pm 4.98}$
Geometry	$45.14\% \pm 11.95$	$73.11\% \pm 17.19$	$61.19\% \pm 7.01$	$\mathbf{94.09\% \pm 2.30}$
Graphs	$24.12\% \pm 5.30$	$62.79\% \pm 8.75$	$60.25\% \pm 8.42$	$\mathbf{81.41\% \pm 6.21}$
Greedy	$53.42\% \pm 20.82$	$82.39\% \pm 3.01$	$75.84\% \pm 6.59$	$\mathbf{91.21\% \pm 2.95}$
Search	$34.35\% \pm 21.67$	$41.20\% \pm 19.87$	$56.11\% \pm 21.56$	$\mathbf{58.61\% \pm 24.34}$
Sorting	$\mathbf{71.53\% \pm 1.41}$	$11.83\% \pm 2.78$	$15.45\% \pm 8.46$	$60.37\% \pm 12.16$
Strings	$1.51\% \pm 0.46$	$3.21\% \pm 0.94$	$2.04\% \pm 0.20$	$\mathbf{49.09\% \pm 23.49}$
Overall avg.	38.88%	44.99%	50.84%	**74.14%**
> 90%	0/30	6/30	3/30	**11/30**
> 80%	3/30	9/30	7/30	**17/30**
> 60%	10/30	14/30	15/30	**24/30**

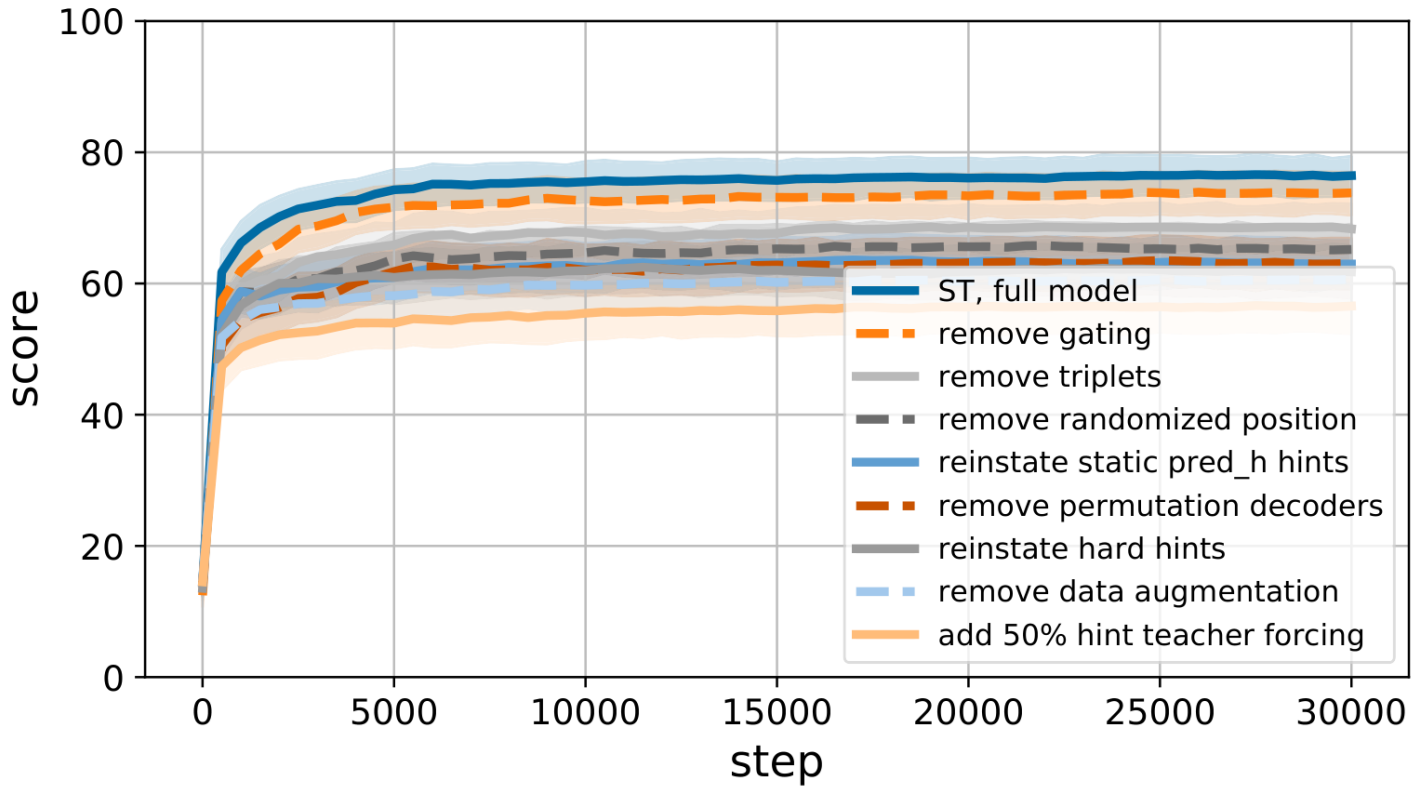
Multi-Task vs Single-Task



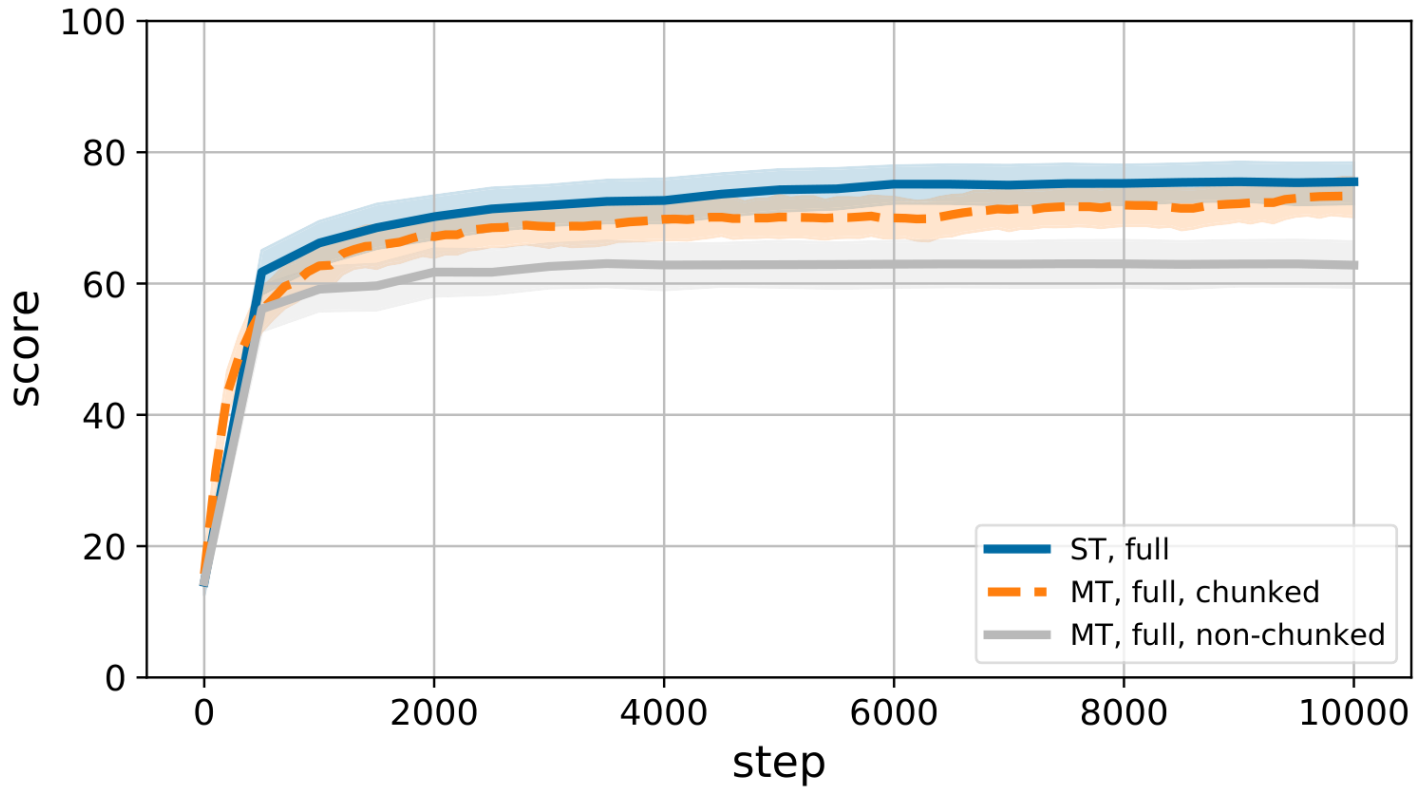
Multi-Task vs Single-Task



Single-Task Ablation Study



Multi-Task Ablation Study



Conclusion

- This paper introduces numerous improvements to the MPNN architecture to enable multi-task learning
  - Based on the theory of algorithmic alignment
- Multi-Task Learning may improve performance over Single-Task counterpart

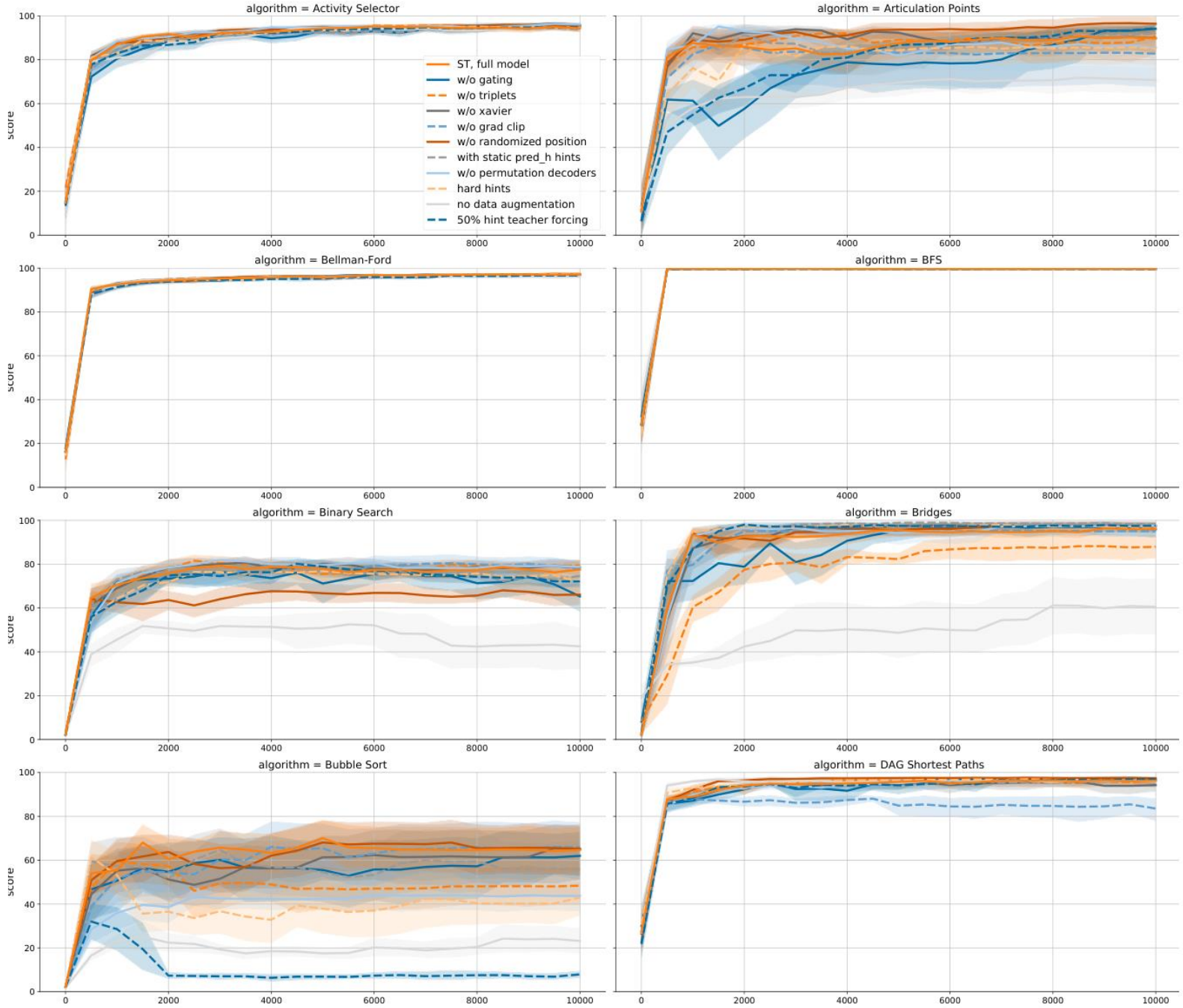
\+ Improvements applicable to other architectures as well

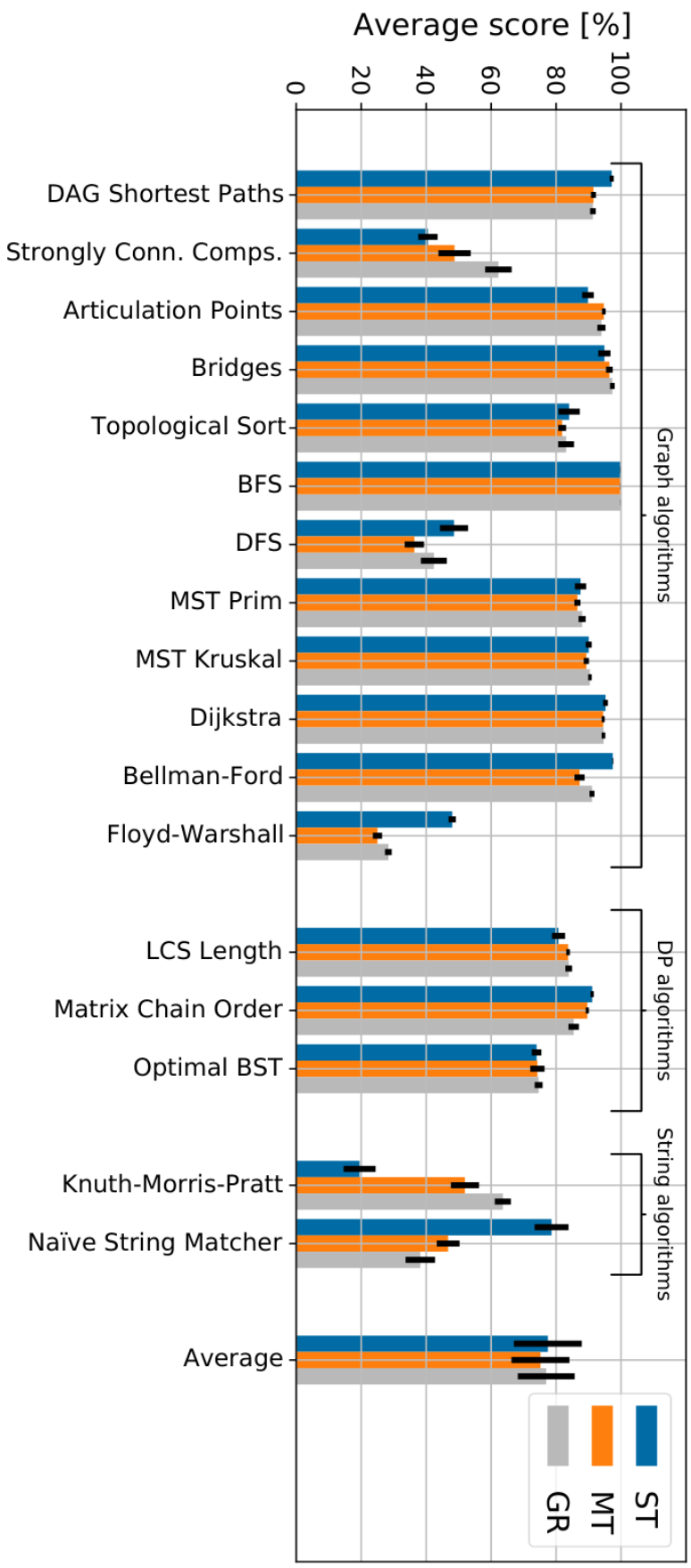
\- Little exploration on Multi-Task Learning despite the name of the paper

Thank you for your Attention!

Questions?

Algorithm	Memnet [5]	MPNN [5]	PGN [5]	Triplet-GMPNN (ours)
Activity Selector	24.10% ± 2.22	80.66% ± 3.16	66.80% ± 1.62	**95.18% ± 0.45**
Articulation Points	1.50% ± 0.61	50.91% ± 2.18	49.53% ± 2.09	**88.32% ± 2.01**
Bellman-Ford	40.04% ± 1.46	92.01% ± 0.28	92.99% ± 0.34	**97.39% ± 0.19**
BFS	43.34% ± 0.04	**99.89% ± 0.05**	99.63% ± 0.29	99.73% ± 0.04
Binary Search	14.37% ± 0.46	36.83% ± 0.26	76.95% ± 0.13	**77.58% ± 2.35**
Bridges	30.26% ± 0.05	72.69% ± 4.78	51.42% ± 7.82	**93.99% ± 2.07**
Bubble Sort	**73.58% ± 0.78**	5.27% ± 0.60	6.01% ± 1.95	67.68% ± 5.50
DAG Shortest Paths	66.15% ± 1.92	96.24% ± 0.56	96.94% ± 0.16	**98.19% ± 0.30**
DFS	13.36% ± 1.61	6.54% ± 0.51	8.71% ± 0.24	**47.79% ± 4.19**
Dijkstra	22.48% ± 2.39	91.50% ± 0.50	83.45% ± 1.75	**96.05% ± 0.60**
Find Max. Subarray	13.05% ± 0.08	20.30% ± 0.49	65.23% ± 2.56	**76.36% ± 0.43**
Floyd-Warshall	14.17% ± 0.13	26.74% ± 1.77	28.76% ± 0.51	**48.52% ± 1.04**
Graham Scan	40.62% ± 2.31	91.04% ± 0.31	56.87% ± 1.61	**93.62% ± 0.91**
Heapsort	**68.00% ± 1.57**	10.94% ± 0.84	5.27% ± 0.18	31.04% ± 5.82
Insertion Sort	71.42% ± 0.86	19.81% ± 2.08	44.37% ± 2.43	**78.14% ± 4.64**
Jarvis' March	22.99% ± 3.87	34.86% ± 12.39	49.19% ± 1.07	**91.01% ± 1.30**
Knuth-Morris-Pratt	1.81% ± 0.00	2.49% ± 0.86	2.00% ± 0.12	**19.51% ± 4.57**
LCS Length	49.84% ± 4.34	53.23% ± 0.36	56.82% ± 0.21	**80.51% ± 1.84**
Matrix Chain Order	81.96% ± 1.03	79.84% ± 1.40	83.91% ± 0.49	**91.68% ± 0.59**
Minimum	86.93% ± 0.11	85.34% ± 0.88	87.71% ± 0.52	**97.78% ± 0.55**
MST-Kruskal	28.84% ± 0.61	70.97% ± 1.50	66.96% ± 1.36	**89.80% ± 0.77**
MST-Prim	10.29% ± 3.77	69.08% ± 7.56	63.33% ± 0.98	**86.39% ± 1.33**
Naïve String Matcher	1.22% ± 0.48	3.92% ± 0.30	2.08% ± 0.20	**78.67% ± 4.99**
Optimal BST	72.03% ± 1.21	62.23% ± 0.44	71.01% ± 1.82	**73.77% ± 1.48**
Quickselect	1.74% ± 0.03	1.43% ± 0.69	**3.66% ± 0.42**	0.47% ± 0.25
Quicksort	**73.10% ± 0.67**	11.30% ± 0.10	6.17% ± 0.15	64.64% ± 5.12
Segments Intersect	71.81% ± 0.90	93.44% ± 0.10	77.51% ± 0.75	**97.64% ± 0.09**
SCC	16.32% ± 4.78	24.37% ± 4.88	20.80% ± 0.64	**43.43% ± 3.15**
Task Scheduling	82.74% ± 0.04	84.11% ± 0.32	84.89% ± 0.91	**87.25% ± 0.35**
Topological Sort	2.73% ± 0.11	52.60% ± 6.24	60.45% ± 2.69	**87.27% ± 2.67**
Overall average	38.03%	51.02%	52.31%	**75.98%**


algorithm = Activity Selector
algorithm = Articulation Points
algorithm = Bellman-Ford
algorithm = BFS
algorithm = Binary Search
algorithm = Bridges
algorithm = Bubble Sort
algorithm = DAG Shortest Paths
score
ST, full model
w/o gating
w/o triplets
w/o xavier
w/o grad clip
w/o randomized position
with static pred_h hints
w/o permutation decoders
hard hints
no data augmentation
50% hint teacher forcing


Average score [%]
0
20
40
60
80
100
DAG Shortest Paths
Strongly Conn. Comps.
Articulation Points
Bridges
Topological Sort
BFS
DFS
MST Prim
MST Kruskal
Dijkstra
Bellman-Ford
Floyd-Warshall
LCS Length
Matrix Chain Order
Optimal BST
Knuth-Morris-Pratt
Naïve String Matcher
Average
Graph algorithms
DP algorithms
String algorithms
ST
MT
GR