A2I2 APPLIED ARTIFICIAL INTELLIGENCE INSTITUTE

DEAKIN UNIVERSITY

Tutorial at KDD, August 14th 2021

From Deep Learning to Deep Reasoning

Part A: Learning to reason

Truyen Tran, Vuong Le, Hung Le and Thao Le
{truyen.tran,vuong.le,thai.le,thao.le}@deakin.edu.au

https://bit.ly/37DYQn7

Logistics

Truyen Tran

Vuong Le

Hung Le

Thao Le

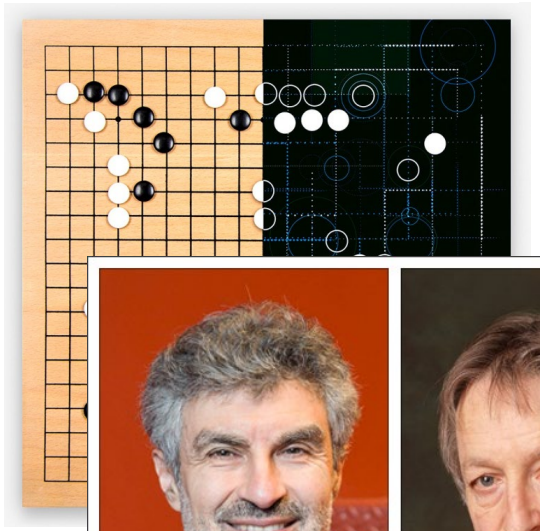
https://bit.ly/37DYQn7

Agenda

- **Introduction**
- Part A: Learning-to-reason framework
- Part B: Reasoning over unstructured and structured data
- Part C: Memory | Data efficiency | Recursive reasoning

2012

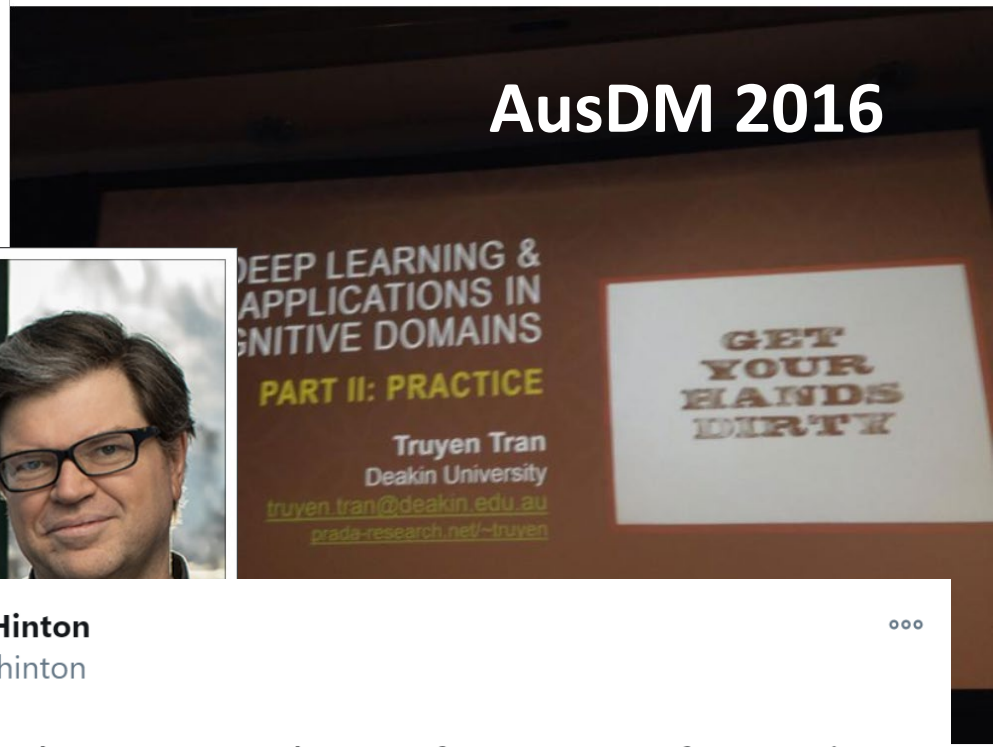
AlphaGo



Turing A

Xingjun (Daniel) Ma • 1st
Lecturer@Deakin University, Machine Learning and AI Scientist
4yr

Excellent deep learning tutorial in AusDM2016 by Truyen Tran!



Geoffrey Hinton
@geoffreyhinton

Extrapolating the spectacular performance of GPT3 into the future suggests that the answer to life, the universe and everything is just 4.398 trillion parameters.

6:26 AM · Jun 11, 2020 · Twitter Web App

) Retweets **75** Quote Tweets **3.8K** Likes

GPT-3 2020

DL: 8 years snapshot

DL has been fantastic, but …

- It is great at interpolating
  - → data hungry to cover all variations and smooth local manifolds
  - → little systematic generalization (novel combinations)
- Lack of human-perceived reasoning capability
  - Lack natural mechanism to incorporate prior knowledge, e.g., common sense
  - No built-in causal mechanisms
  - → Have trust issues!
- To be fair, may of these problems are common in statistical learning!

Why still DL in 2021?

Theoretical

- **Expressiveness**: Neural nets can approximate any function.
- **Learnability**: Neural nets are trained easily.
- **Generalisability**: Neural nets generalize surprisingly well to unseen data.

Practical

- **Generality:** Applicable to many domains.
- **Competitive:** DL is hard to beat as long as there are data to train.
- **Scalability:** DL is better with more data, and it is very scalable.

The next AI/ML challenge

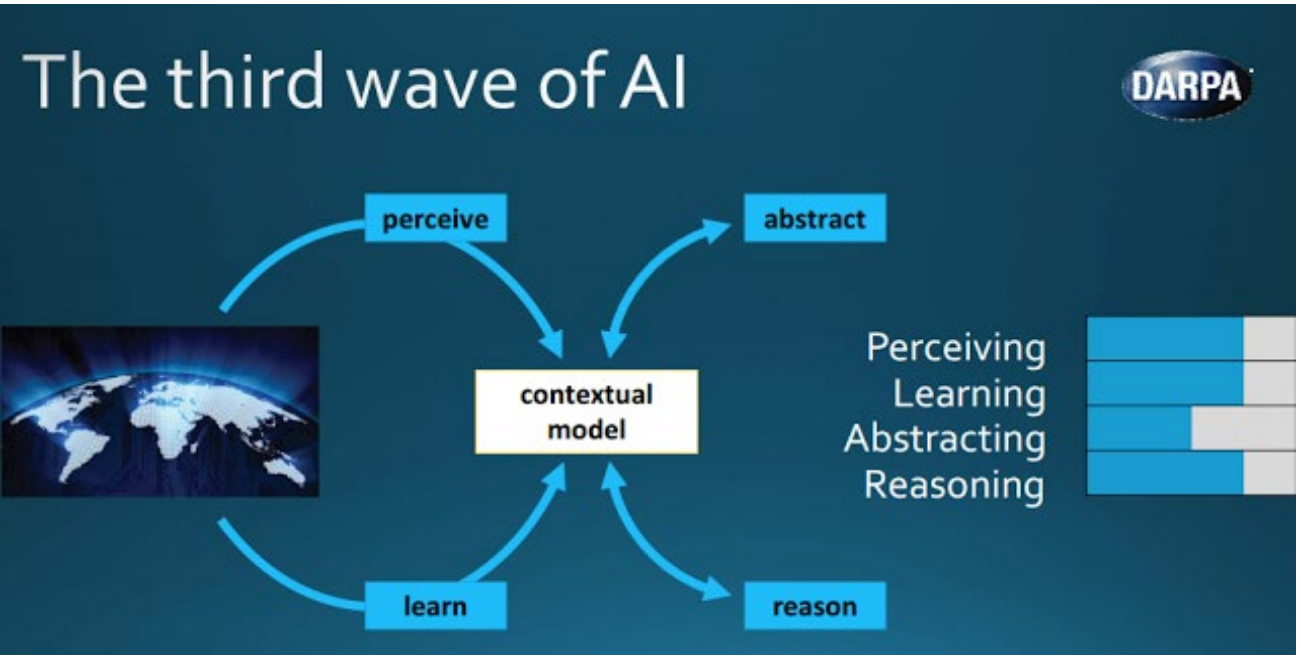
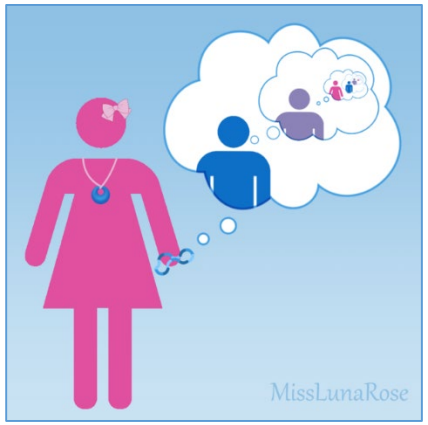


Photo credit: DARPA

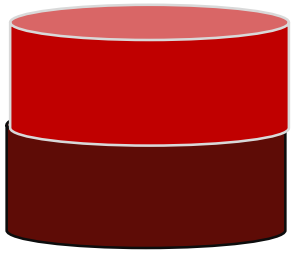
2020s-2030s

- Learning + reasoning, general purpose, human-like
- Has contextual and common-sense reasoning
- Requires less data
- Adapt to change
- Explainable



Theory of mind
Recursive reasoning

Memory



Facts
Semantics
Events and relations
Working space

Multiple

Single

Perception

System 1:
Intuitive

- Fast
- Implicit/automatic
- Pattern recognition
- Multiple

System 2:
Analytical

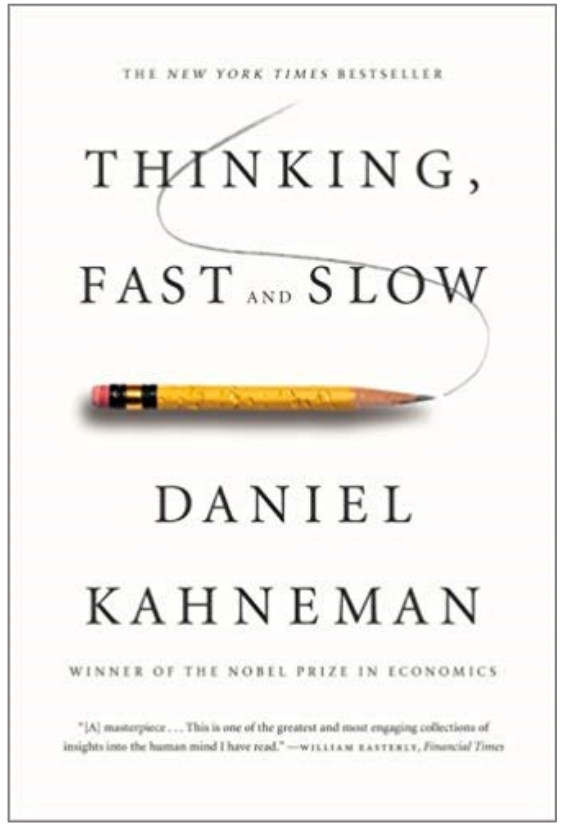
- Slow
- Deliberate/rational
- Careful analysis
- Single, sequential

Toward deeper reasoning

Image credit: VectorStock | Wikimedia

System 2

- Holds hypothetical thought
- Decoupling from representation
- Working memory size is not essential. Its attentional control is.



Reasoning in Probabilistic Graphical Models (PGM)

- Assuming models are fully specified (e.g., by hand or learnt)
  - →Estimate MAP as energy minimization
  - →Compute marginal probability
  - →Compute expectation & normalisation constant
- Key algorithm: Pearl's Belief Propagation, a.k.a Sum-Product algorithm in factor graphs.
  - Known result in 2001-2003: BP minimises Bethe free-energy minimization.

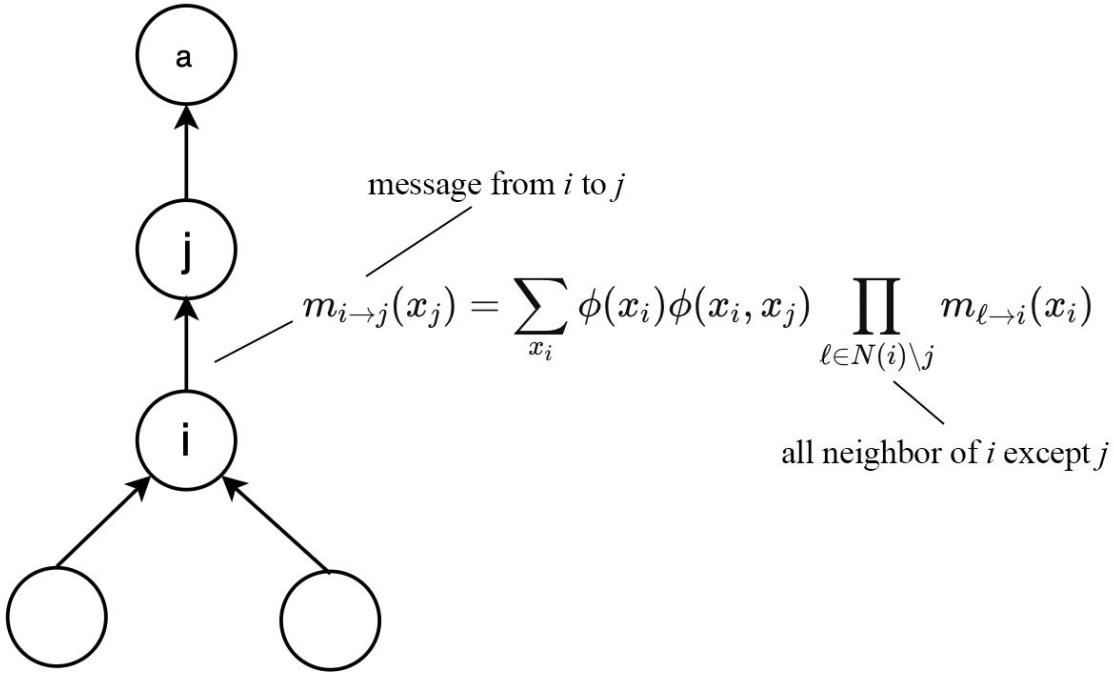


Figure credit: Jonathan Hui

Heskes, Tom. "Stable fixed points of loopy belief propagation are local minima of the bethe free energy." *Advances in neural information processing systems*. 2003.

Can we learn to infer directly from data without full specification of models?

Agenda

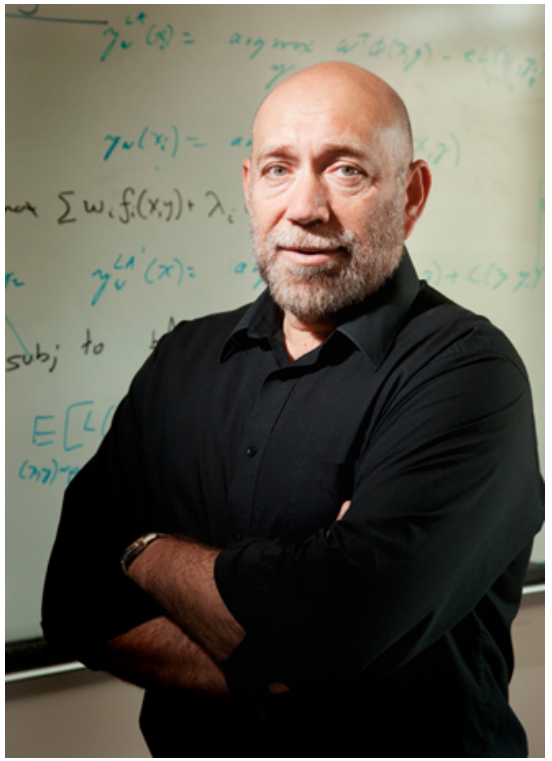
- Introduction
- **Part A: Learning-to-reason framework**
- Part B: Reasoning over unstructured and structured data
- Part C: Memory | Data efficiency | Recursive reasoning

Part A: Sub-topics

- Reasoning as a prediction skill that can be learnt from data.
  - Question answering as zero-shot learning.
- Neural network operations for learning to reason:
  - Concept-object binding.
  - Attention & transformers.
  - Dynamic neural networks, conditional computation & differentiable programming.
- Reasoning as iterative representation refinement & query-driven program synthesis and execution
  - Compositional attention networks.
  - Neural module networks.
- Combinatorics reasoning

Learning to reason

- Learning is to self-improve by experiencing ~ acquiring knowledge & skills
- Reasoning is to deduce knowledge from previously acquired knowledge in response to a query (or a cues)
- Learning to reason is to improve the ability to decide if a knowledge base entails a predicate.
  - E.g., given a video f, determines if the person with the hat turns before singing.
- Hypotheses:
  - Reasoning as just-in-time **program synthesis**.
  - It employs **conditional computation**.



(Dan Roth; *ACM Fellow; IJCAI John McCarthy Award*)

Khardon, Roni, and Dan Roth. "Learning to reason." *Journal of the ACM (JACM)* 44.5 (1997): 697-725.

Learning to reason, a definition

Definition 2.1.1. An algorithm A is an *exact reasoning* algorithm for the reasoning problem $(\mathcal{F}, \mathcal{Q})$, if for all $f \in \mathcal{F}$ and for all $\alpha \in \mathcal{Q}$, when A is presented with input (f, α), A runs in time polynomial in n and the size of f and α, and answers "yes" if and only if $f \models \alpha$.

E.g., given a video f, determines if the person with the hat turns before singing.

Khardon, Roni, and Dan Roth. "Learning to reason." *Journal of the ACM (JACM)* 44.5 (1997): 697-725.

Practical setting: `(query,database,answer)` **triplets**

- This is very general:
  - Classification: Query = *what is this?* Database = *data.*
  - Regression: Query = *how much?* Database = *data.*
  - QA: Query = *NLP question.* Database = *context/image/text.*
  - Multi-task learning: Query = *task ID.* Database = *data.*
  - Zero-shot learning: Query = *task description.* Database = *data.*
  - Drug-protein binding: Query = *drug*. Database = *protein.*
  - Recommender system: Query = *User (or item)*. Database = *inventories (or user base)*;

Can neural networks reason?

Reasoning is not necessarily achieved by making logical inferences

There is a continuity between [algebraically rich inference] and [connecting together trainable learning systems]

Central to reasoning is composition rules to guide the combinations of modules to address new tasks

"When we observe a visual scene, when we hear a complex sentence, we are able to explain in formal terms the relation of the objects in the scene, or the precise meaning of the sentence components. However, there is no evidence that such a formal analysis necessarily takes place: we see a scene, we hear a sentence, and we just know what they mean. This suggests the existence of a middle layer, already a form of reasoning, but not yet formal or logical."

Bottou, Léon. "From machine learning to machine reasoning." *Machine learning* 94.2 (2014): 133-149.

Hypotheses

- Reasoning as just-in-time program synthesis.
- It employs conditional computation.
- Reasoning is recursive, e.g., mental travel.

Two approaches to neural reasoning

- **Implicit chaining of predicates through recurrence:**
  - Step-wise query-specific attention to relevant concepts & relations.
  - Iterative concept refinement & combination, e.g., through a working memory.
  - Answer is computed from the last memory state & question embedding.
- **Explicit program synthesis:**
  - There is a set of modules, each performs an pre-defined operation.
  - Question is parse into a symbolic program.
  - The program is implemented as a computational graph constructed by chaining separate modules.
  - The program is executed to compute an answer.

In search for basic neural operators for reasoning

- Basics:
  - Neuron as feature detector → Sensor, filter
  - Computational graph → Circuit
  - Skip-connection → Short circuit
- Essentials
  - Multiplicative gates → AND gate, Transistor, Resistor
  - Attention mechanism → SWITCH gate
  - Memory + forgetting → Capacitor + leakage
  - Compositionality → Modular design
- ..

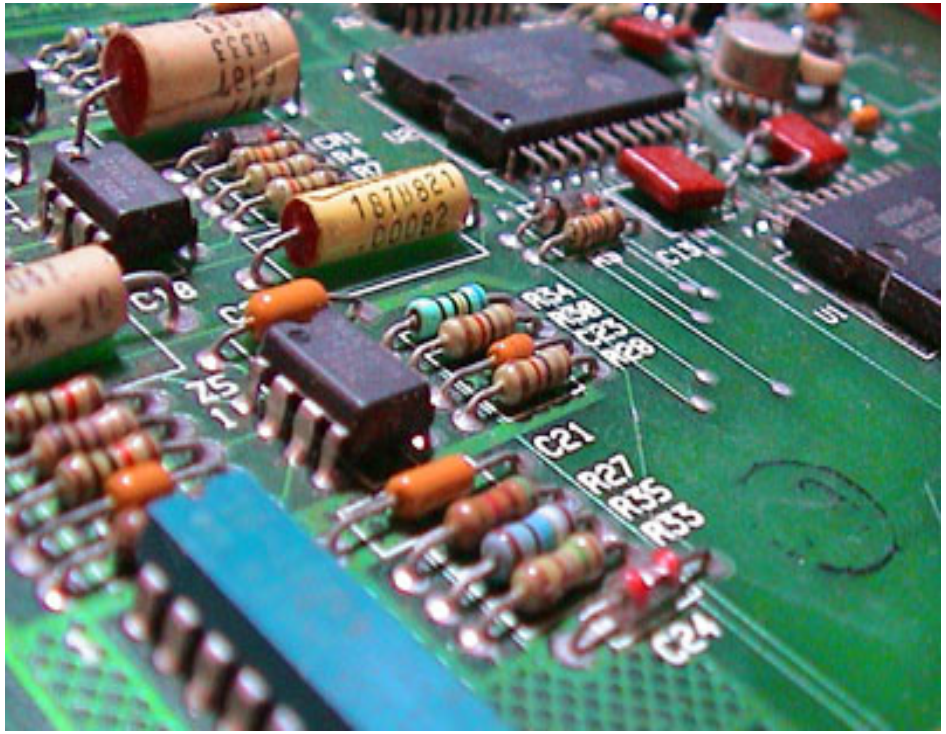


Photo credit: Nicola Asuni

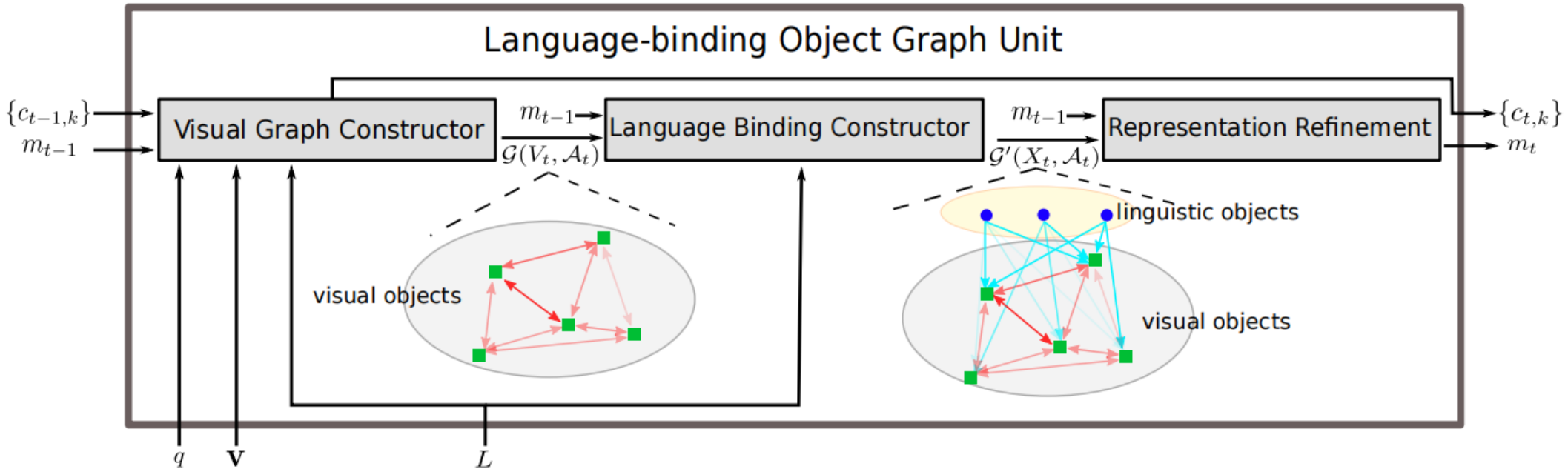
Part A: Sub-topics

- Reasoning as a prediction skill that can be learnt from data.
- Question answering as zero-shot learning.
- **Neural network operations for learning to reason:**
  - Concept-object binding.
  - Attention & transformers.
  - Dynamic neural networks, conditional computation & differentiable programming.
- Reasoning as iterative representation refinement & query-driven program synthesis and execution.
  - Compositional attention networks.
  - Reasoning as Neural module networks.
- Combinatorics reasoning

More reading: Greff, Klaus, Sjoerd van Steenkiste, and Jürgen Schmidhuber. "On the binding problem in artificial neural networks." *arXiv preprint arXiv:2012.05208* (2020).

Concept-object binding

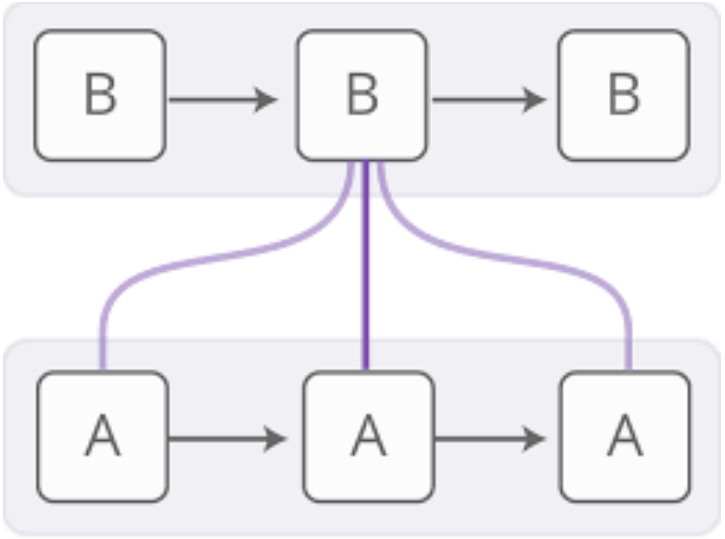
- Perceived data (e.g., visual objects) may not share the same semantic space with high-level concepts.
- Binding between concept-object enables reasoning at the concept level



Example of concept-object binding in LOGNet (Le et al, IJCAI'2020)

Attentions: Picking up only what is needed at a step

- Need attention model to select or ignore certain **computations** or **inputs**
  - Can be “soft” (differentiable) or “hard” (requires RL)
  - Needed for selecting predicates in reasoning.
- Attention provides a short-cut → long-term dependencies
  - Needed for long chain of reasoning.
  - Also encourages sparsity if done right!



http://distill.pub/2016/augmented-rnns/

Fast weights | HyperNet – the multiplicative interaction

- Early ideas in early 1990s by Juergen Schmidhuber and collaborators.
- **Data-dependent weights** | Using a controller to generate weights of the main net.

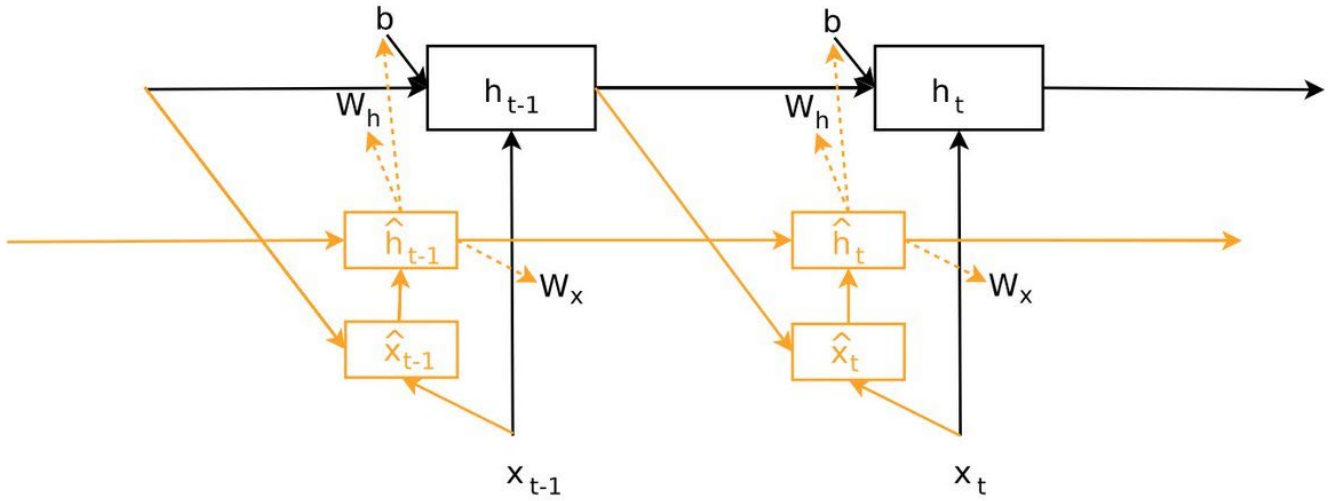
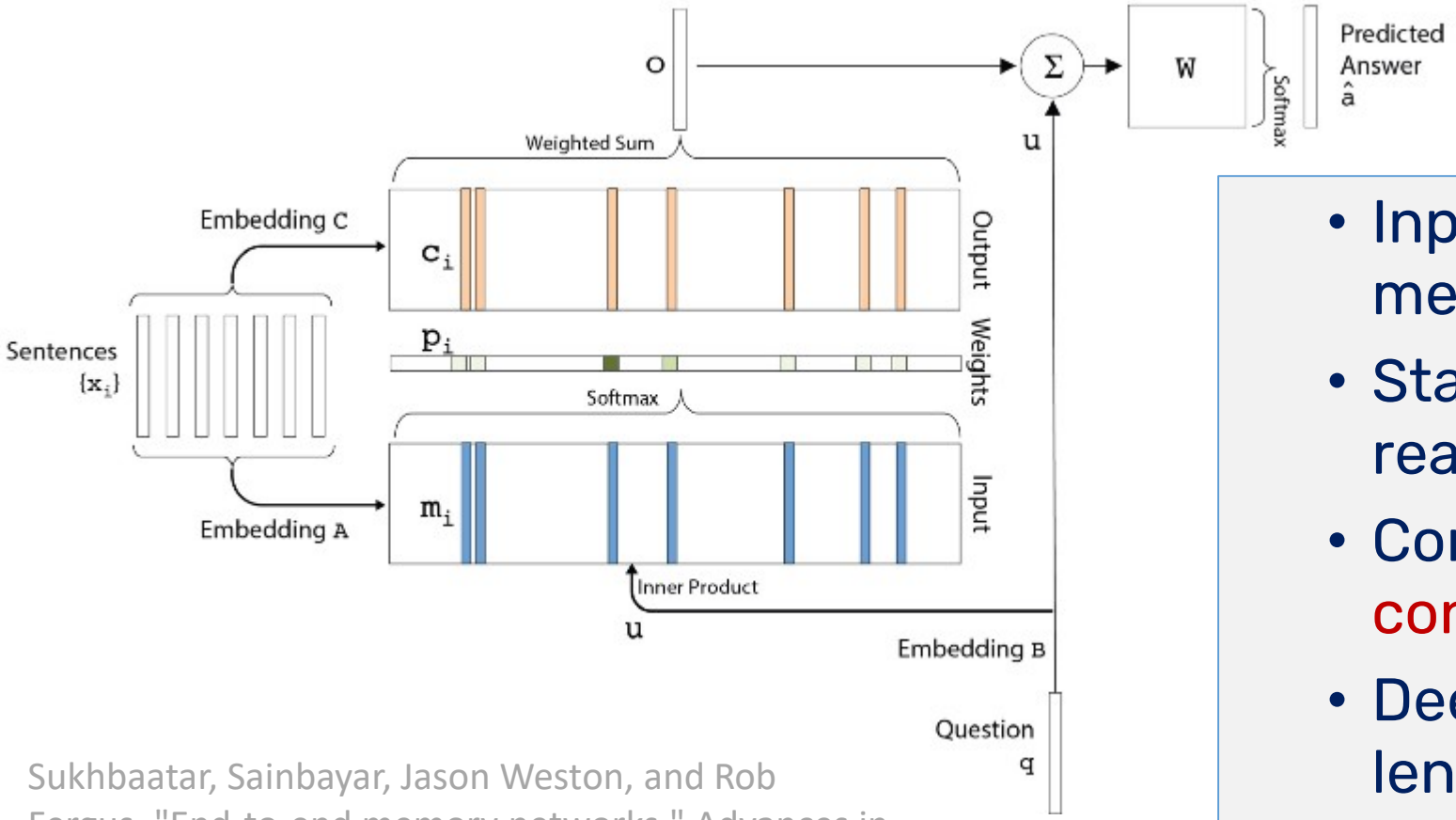


Figure: The HyperRNN system. The black system represents the main RNN while the orange system represents the weight-generating HyperRNN cell.

Ha, David, Andrew Dai, and Quoc V. Le. "Hypernetworks." *arXiv preprint arXiv:1609.09106* (2016).

Memory networks: Holding the data ready for inference



Sukhbaatar, Sainbayar, Jason Weston, and Rob Fergus. "End-to-end memory networks." Advances in neural information processing systems. 2015.

- Input is a set → Load into memory, which is NOT updated.
- State is a RNN with attention reading from inputs
- Concepts: Query, key and content + Content addressing.
- Deep models, but constant path length from input to output.
- Equivalent to a RNN with shared input set.

Transformers: Analogical reasoning through self-attention

State

$$A_h = \text{Softmax}(\alpha Q_h K_h^\top) V_h$$

$$Q_h = \boldsymbol{W}_q X, K_h = \boldsymbol{W}_k X \text{ and } V_h = \boldsymbol{W}_v X$$

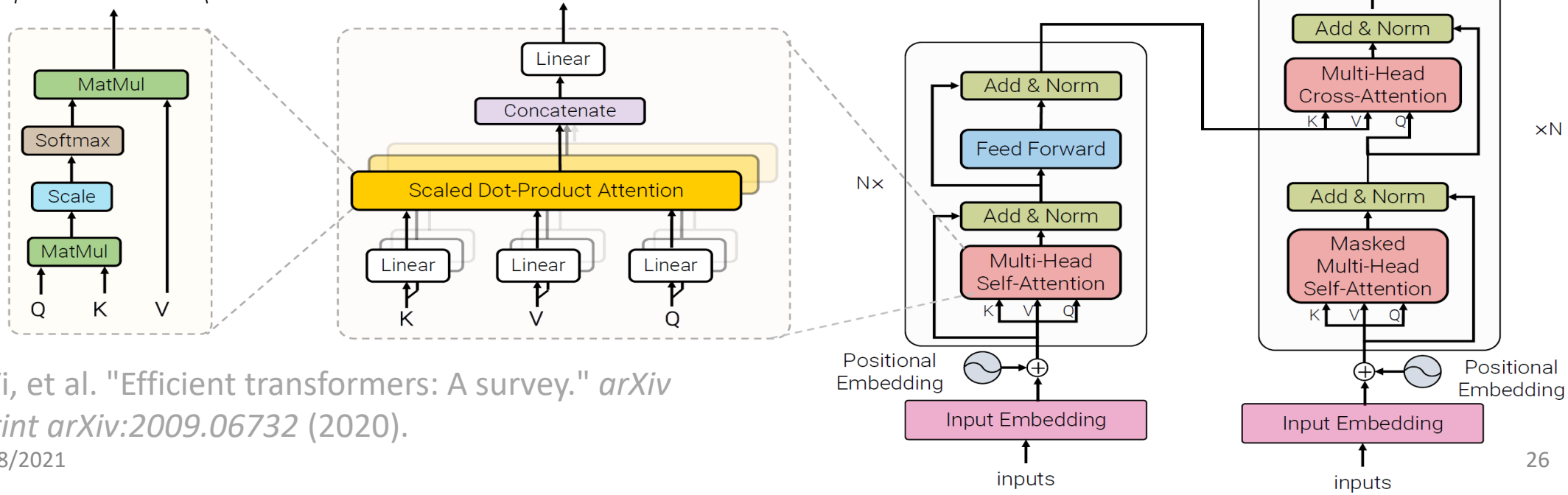
Query

Key

Memory

Computational and Memory Complexity

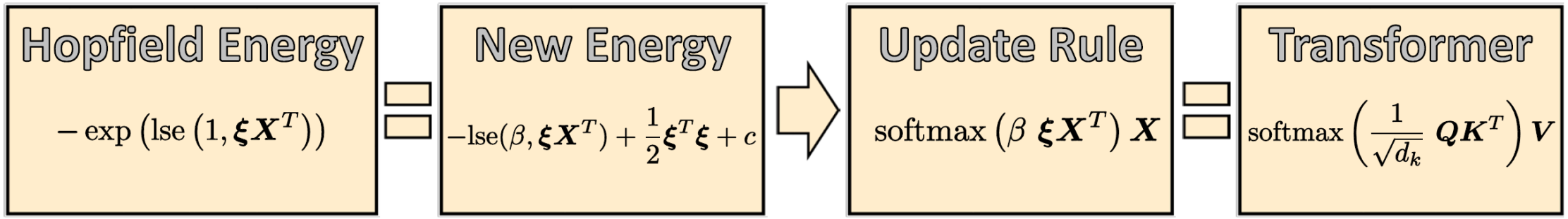
$$\mathcal{O}(n^2)$$



Tay, Yi, et al. "Efficient transformers: A survey." *arXiv preprint arXiv:2009.06732* (2020).

Transformer as implicit reasoning

- Recall: Reasoning as (free-) energy minimisation
  - The classic Belief Propagation algorithm is minimization algorithm of the Bethe free-energy!
- Transformer has relational, iterative state refinement makes it a great candidate for implicit relational reasoning.



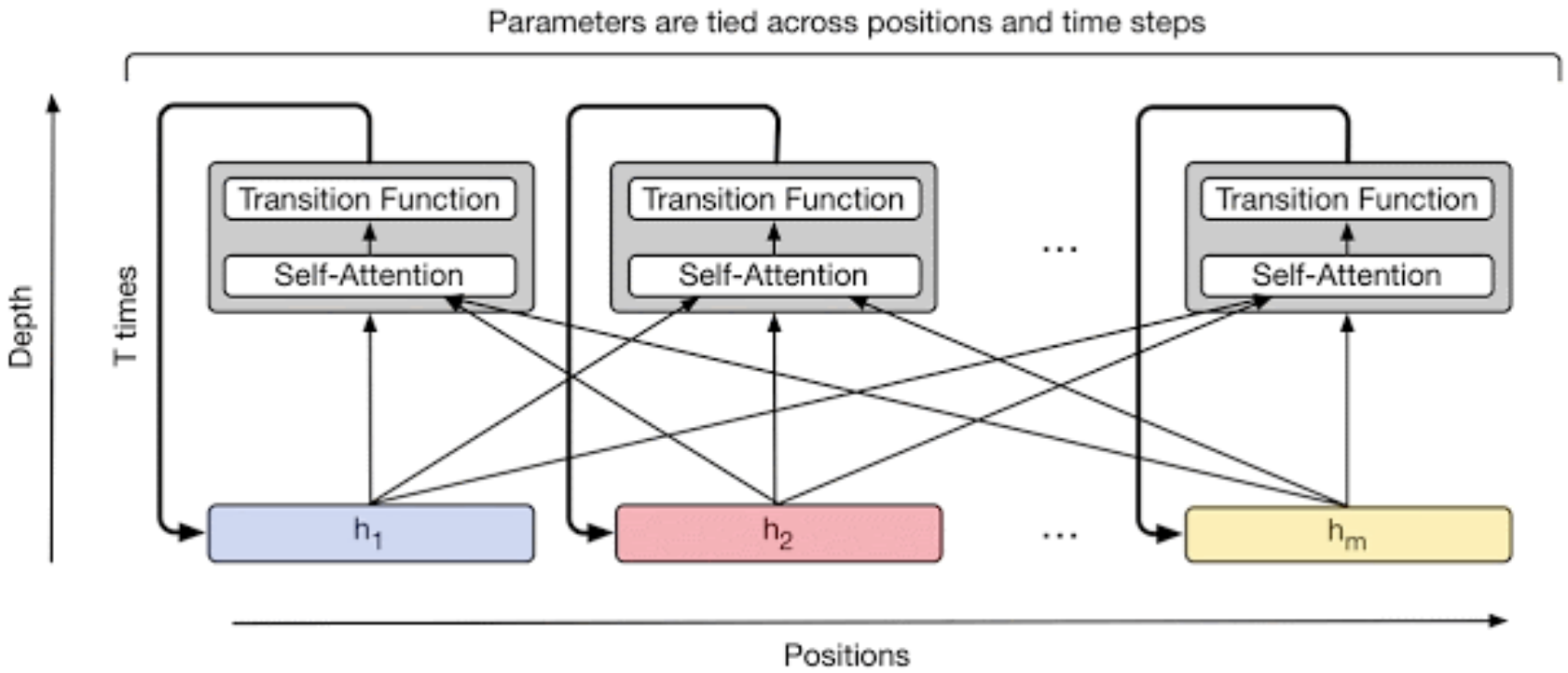
Ramsauer, Hubert, et al. "Hopfield networks is all you need." *arXiv preprint arXiv:2008.02217* (2020).

Transformer v.s. memory networks

- Memory network:
  - Attention to input set
  - One hidden state update at a time.
  - Final state integrate information of the set, conditioned on the query.
- Transformer:
  - Loading all inputs into working memory
  - Assigns one hidden state per input element.
  - All hidden states (including those from the query) to compute the answer.

Universal transformers

Dehghani, Mostafa, et al. "Universal Transformers." *International Conference on Learning Representations*. 2018.



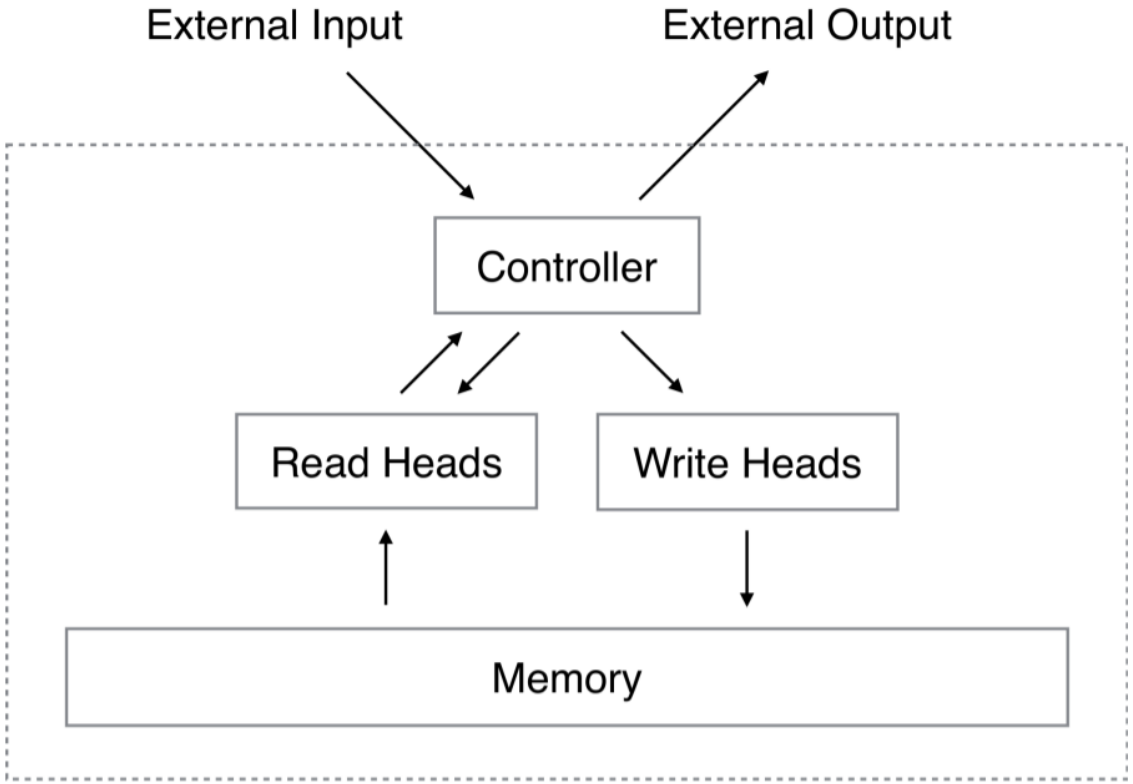
https://ai.googleblog.com/2018/08/moving-beyond-translation-with.html

Dynamic neural networks

- Memory-Augmented Neural Networks
- Modular program layout
- Program synthesis

Neural Turing machine (NTM)
A memory-augmented neural network (MANN)

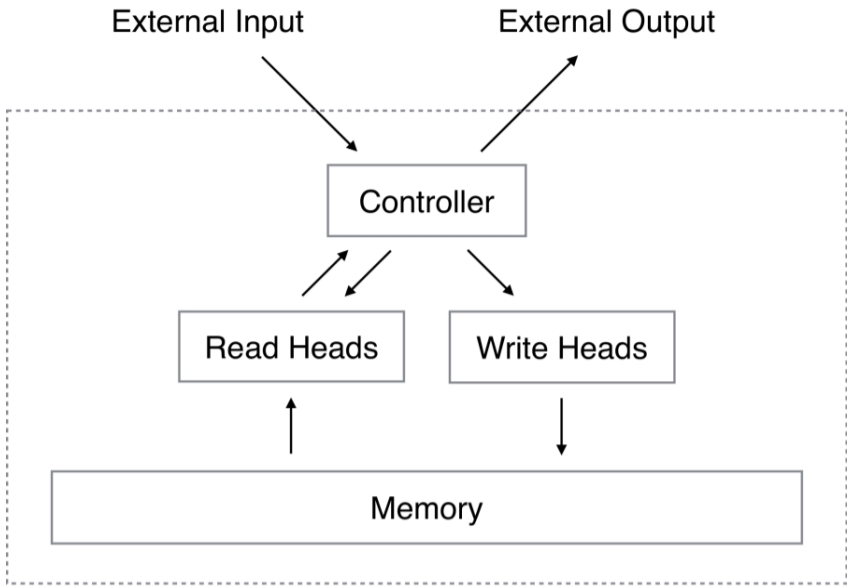
- A controller that takes input/output and talks to an external memory module.
- Memory has read/write operations.
- The main issue is where to write, and how to update the memory state.
- All operations are differentiable.



Source: rylanschaeffer.github.io

MANN for reasoning

- Three steps:
  - Store data into memory
  - Read query, process sequentially, consult memory
  - Output answer
- Behind the scene:
  - Memory contains data & results of intermediate steps
  - LOGNet does the same, memory consists of object representations
- Drawbacks of current MANNs:
  - No memory of controllers → Less modularity and compositionality when query is complex
  - No memory of relations → Much harder to chain predicates.

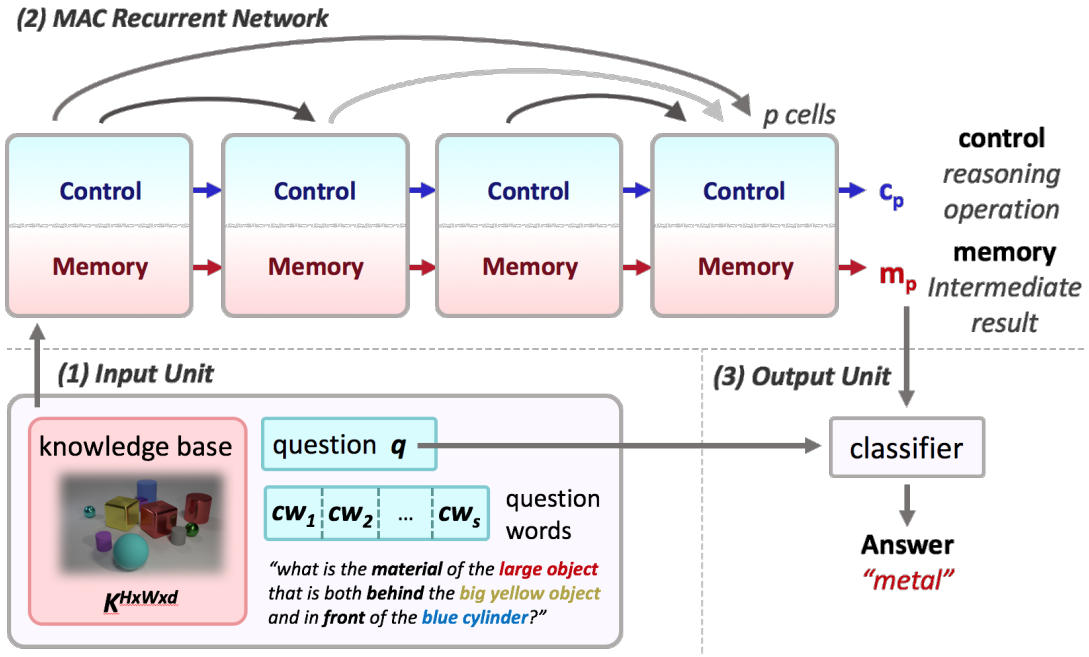


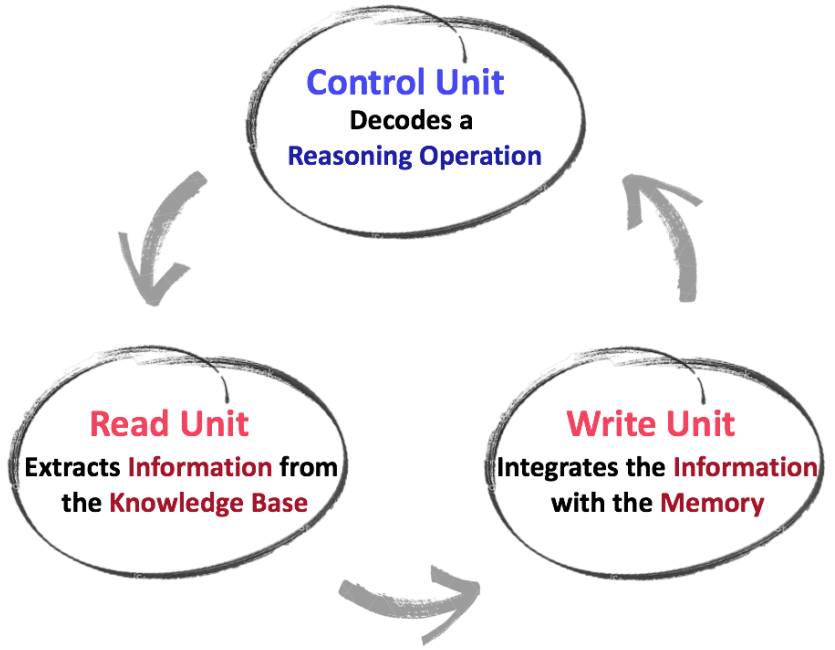
Source: rylanschaeffer.github.io

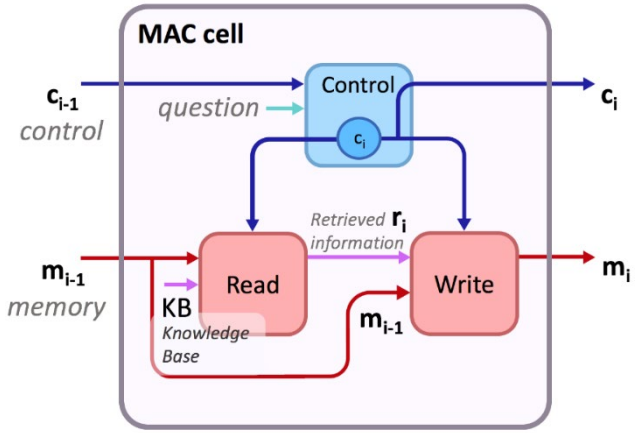
Part A: Sub-topics

- Reasoning as a prediction skill that can be learnt from data.
  - Question answering as zero-shot learning.
- Neural network operations for learning to reason:
  - Concept-object binding.
  - Attention & transformers.
  - Dynamic neural networks, conditional computation & differentiable programming.
- **Reasoning as iterative representation refinement & query-driven program synthesis and execution.**
  - Compositional attention networks.
  - Reasoning as Neural module networks.
- Combinatorics reasoning

MAC Net: Recurrent, iterative representation refinement





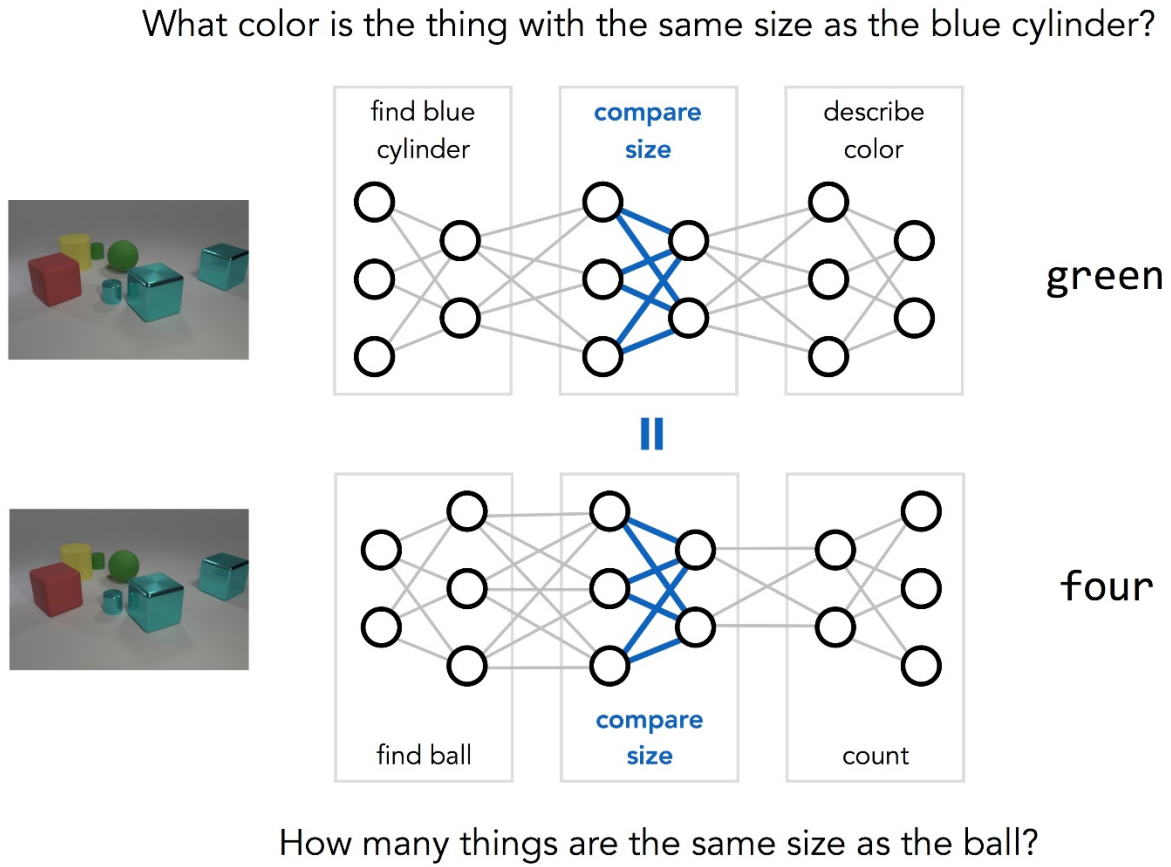


Hudson, Drew A., and Christopher D. Manning. "Compositional attention networks for machine reasoning." *ICLR* 2018.

Module networks

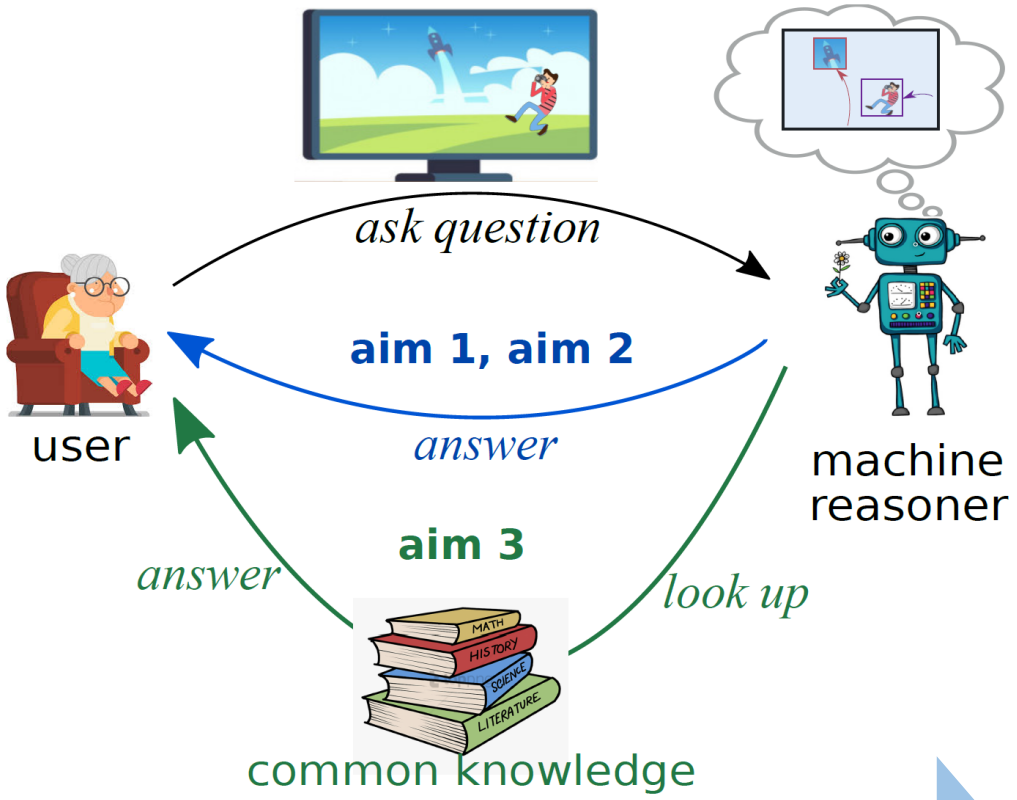
(reasoning by constructing and executing neural programs)

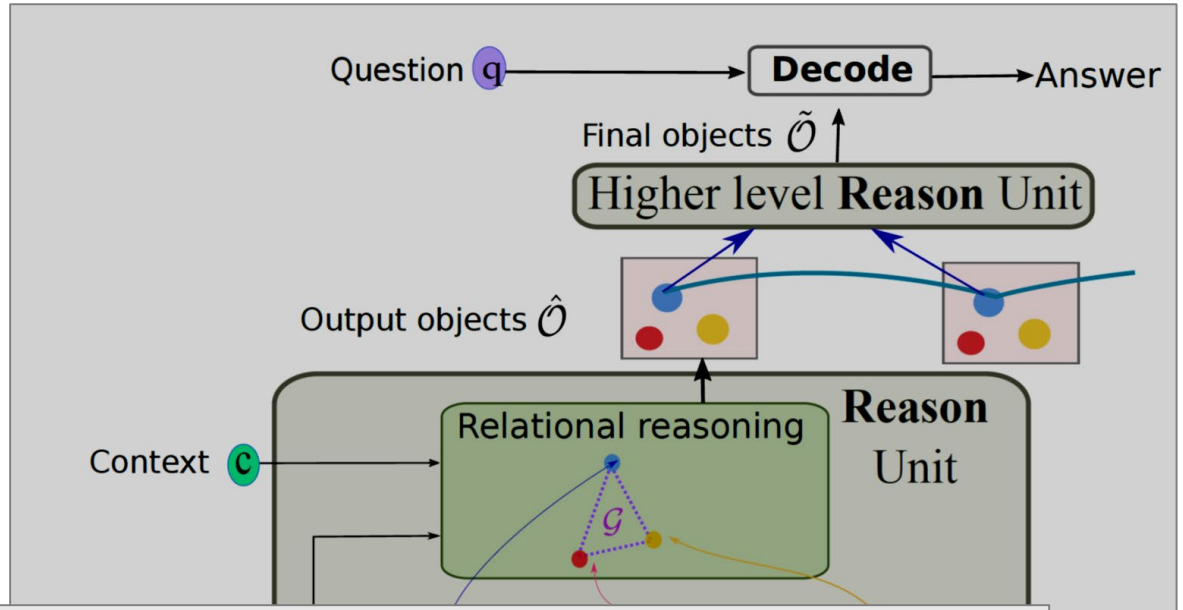
- Reasoning as laying out modules to reach an answer
- Composable neural architecture → question parsed as program (layout of modules)
- A module is a function (x → y), could be a sub-reasoning process ((x, q) → y).

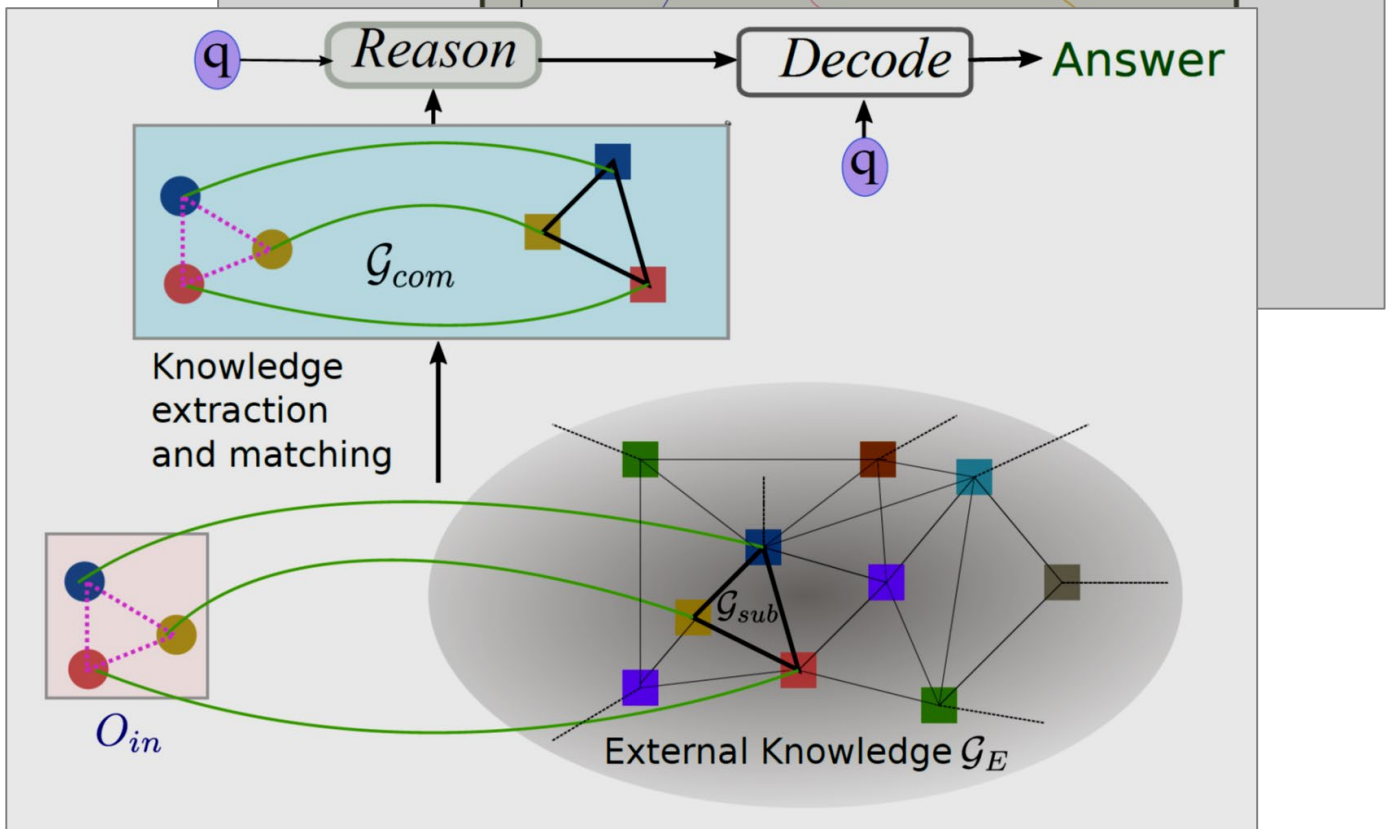


https://bair.berkeley.edu/blog/2017/06/20/learning-to-reason-with-neural-module-networks/

Putting things together:
A framework for visual reasoning







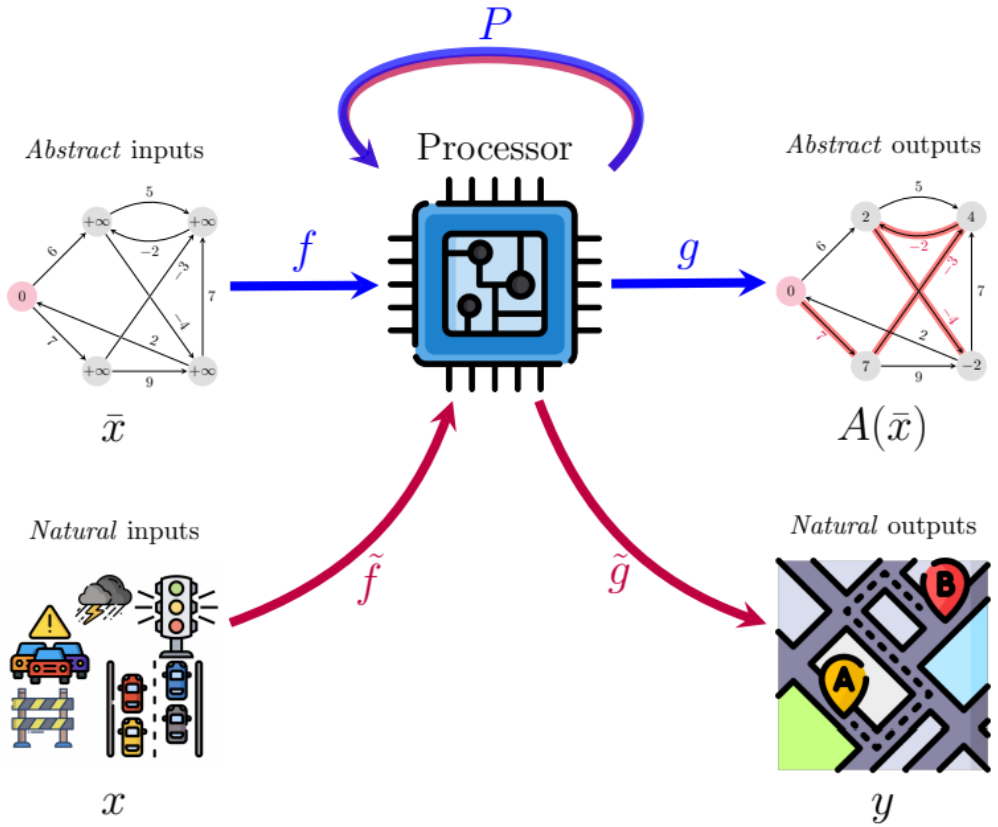
 @Truyen Tran & Vuong Le, Deakin Uni

Part A: Sub-topics

- Reasoning as a prediction skill that can be learnt from data.
  - Question answering as zero-shot learning.
- Neural network operations for learning to reason:
  - Concept-object binding.
  - Attention & transformers.
  - Dynamic neural networks, conditional computation & differentiable programming.
- Reasoning as iterative representation refinement & query-driven program synthesis and execution.
  - Compositional attention networks.
  - Reasoning as Neural module networks.
- **Combinatorics reasoning**

Implement combinatorial algorithms with neural networks

Train neural processor P to imitate algorithm A



Processor P:

(a) aligned with the computations of the target algorithm;

(b) operates by matrix multiplications, hence natively admits useful gradients;

(c) operates over high-dimensional latent spaces

Generalizable
Inflexible

Noisy
High dimensional

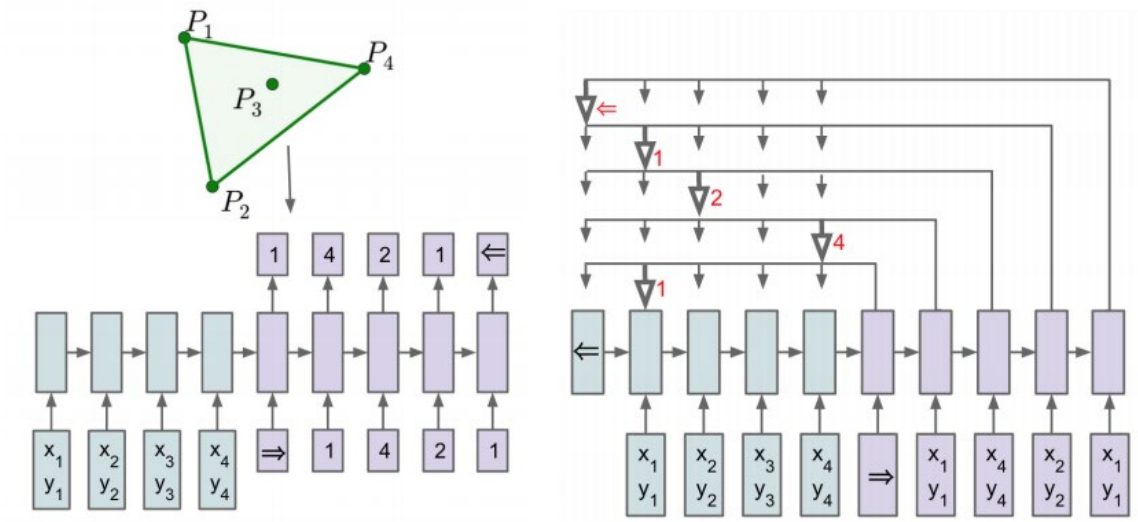
Veličković, Petar, and Charles Blundell. "Neural Algorithmic Reasoning." *arXiv preprint arXiv:2105.02761* (2021).

Processor as RNN

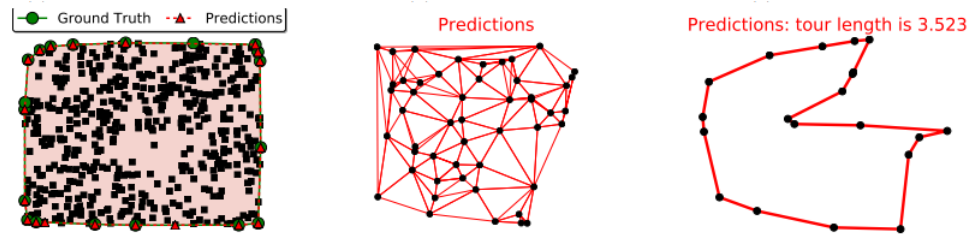
- Do not assume knowing the structure of the input, input as a sequence
  - →not really reasonable, harder to generalize
- RNN is Turing-complete
  - → can simulate any algorithm
- But, it is not easy to learn the simulation from data (input-output)
  - →Pointer network

Assume O(N) memory
And O(N^2) computation
N is the size of input

$$u_j^i = v^T \tanh(W_1 e_j + W_2 d_i) \quad j \in (1, \ldots, n)$$
$$p(C_i | C_1, \ldots, C_{i-1}, \mathcal{P}) = \text{softmax}(u^i)$$



(a) Sequence-to-Sequence (b) Ptr-Net



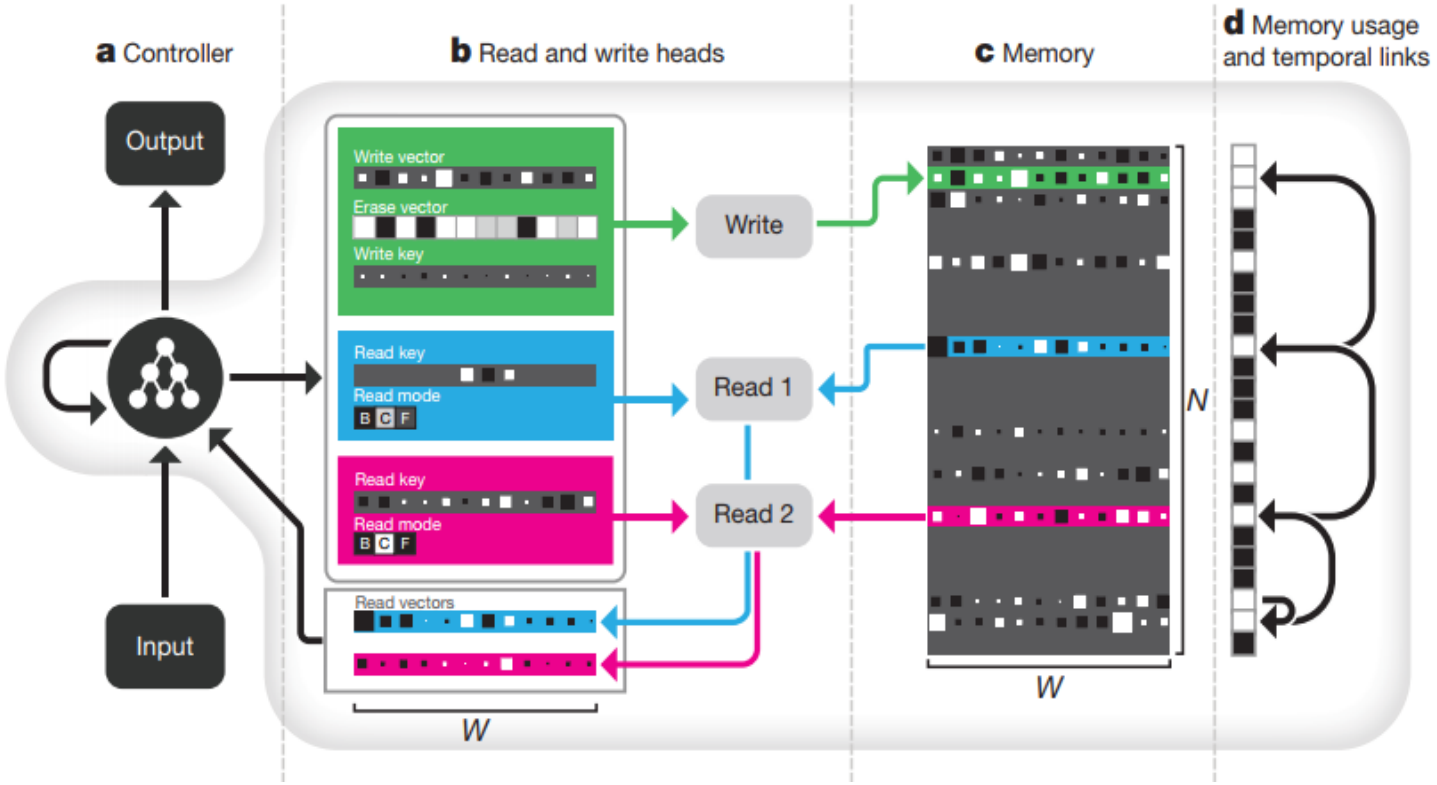
(d) Ptr-Net, m=5-50, n=500 (e) Ptr-Net , m=50, n=50 (f) Ptr-Net , m=5-20, n=20

Vinyals, Oriol, Meire Fortunato, and Navdeep Jaitly. "Pointer networks."
In *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 2*, pp. 2692-2700. 2015.

Processor as MANN

- MANN simulates neural computers or Turing machine→ ideal for implement algorithms
- Sequential input, no assumption on input structure
- Assume O(1) memory

and O(N) computation



Graves, A., Wayne, G., Reynolds, M. *et al.* Hybrid computing using a neural network with dynamic external memory. *Nature* **538,** 471–476 (2016)

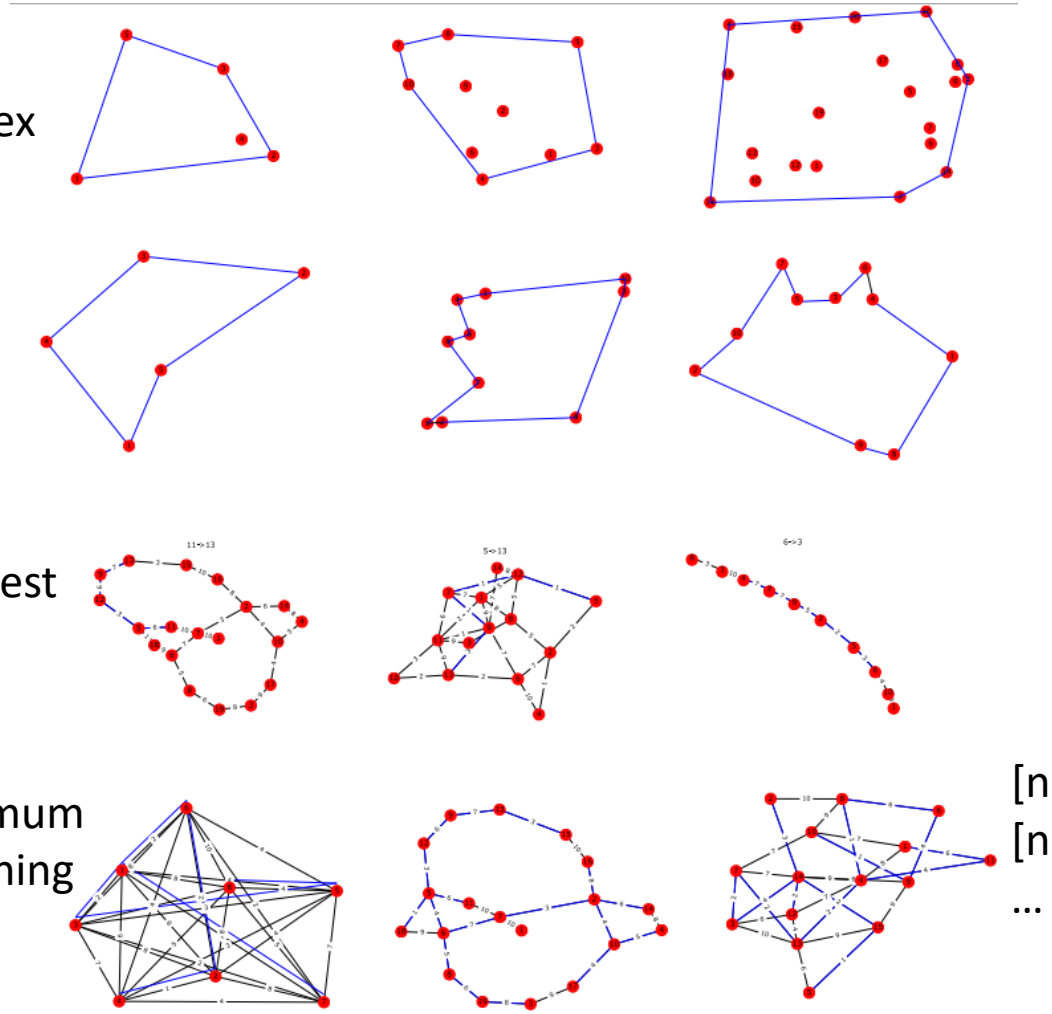
Sequential encoding of graphs

Convex Hull

TSP

Shortest Path

Minimum Spanning Tree



- Each node is associated with random one-hot or binary features
- Output is the features of the solution

Geometry

[x1,y1, feature1],
[x2,y2, feature2],
…

→

[feature4],
[feature2],
…

Graph

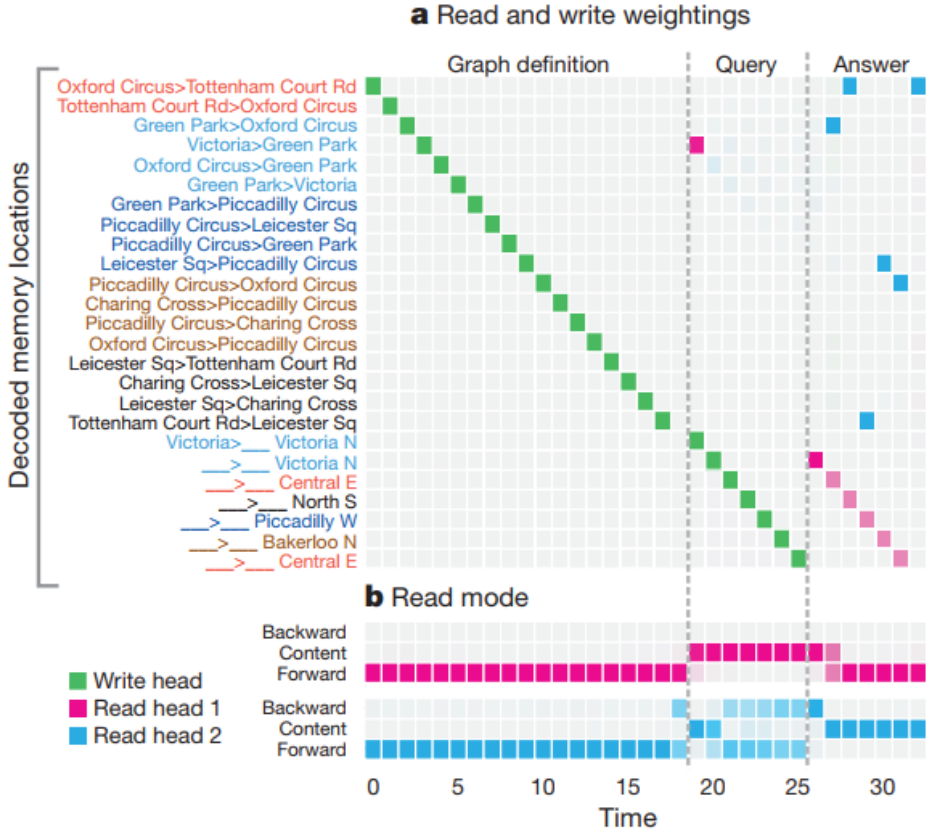
[node_feature1, node_feature2, edge12],
[node_feature1, node_feature2, edge13],
…

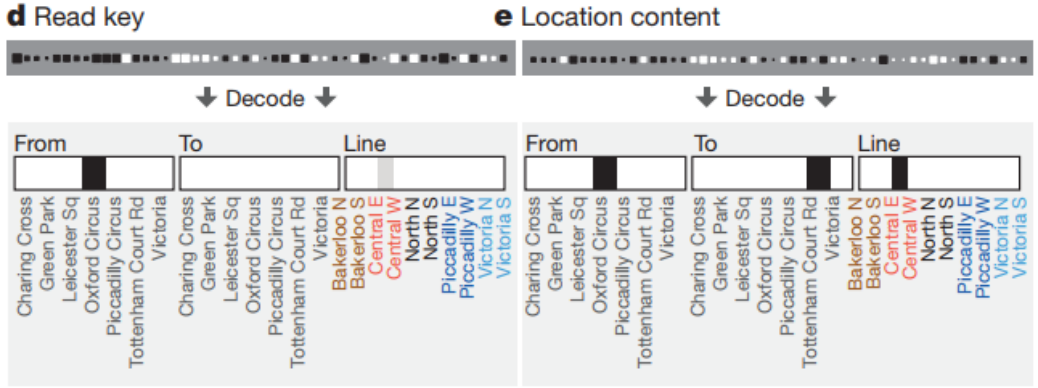
→

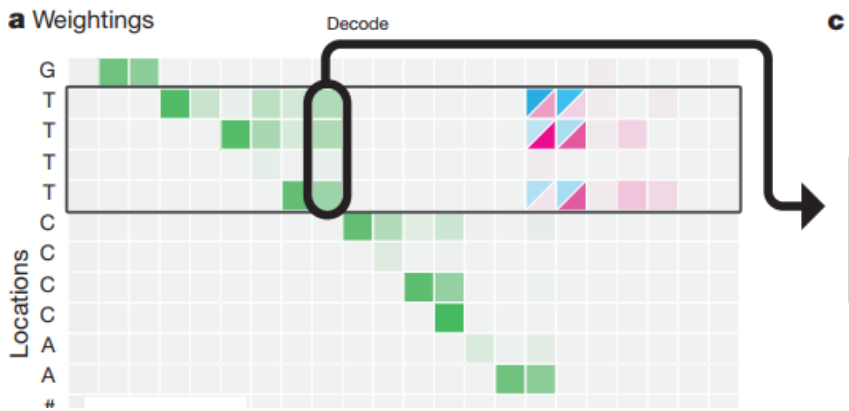
[node_feature4],
[node_feature2],
…

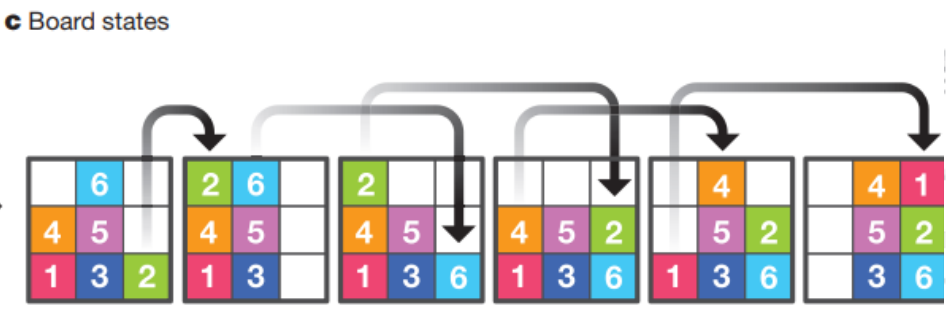
Le, Hung, Truyen Tran, and Svetha Venkatesh. "Self-attentive associative memory." In *International Conference on Machine Learning*, pp. 5682-5691. PMLR, 2020.

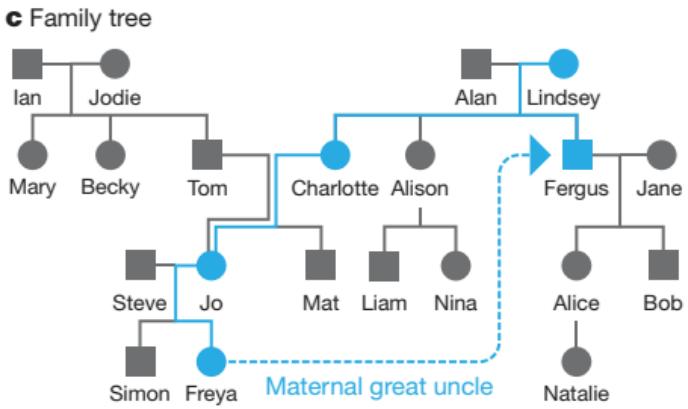
DNC: graph reasoning





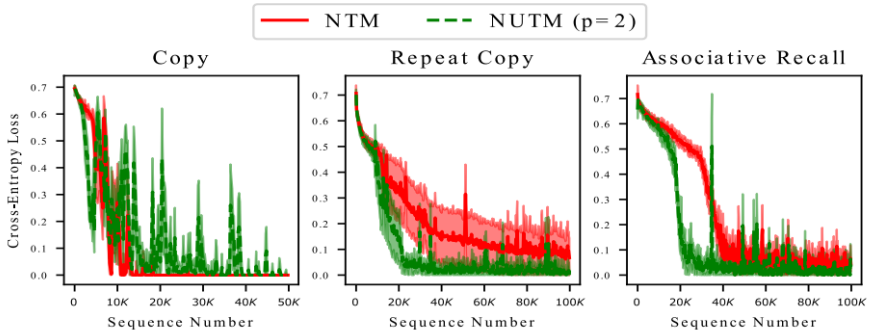


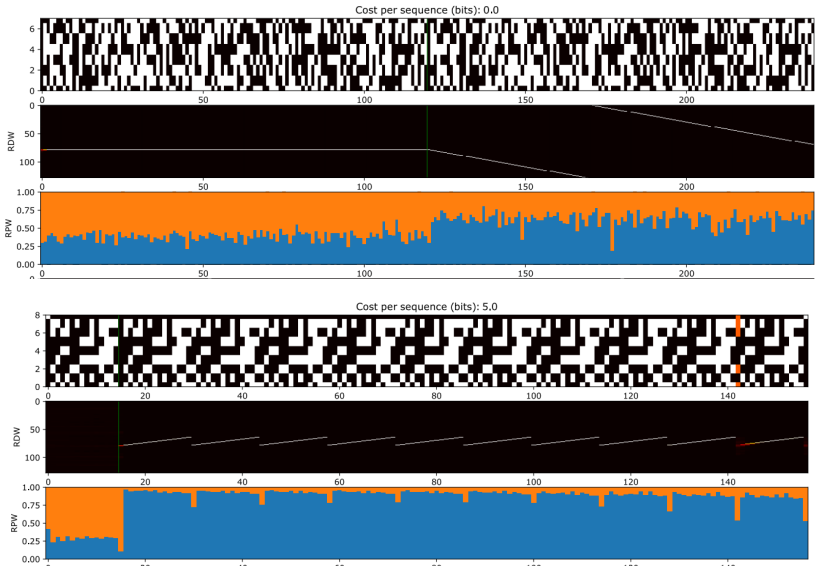


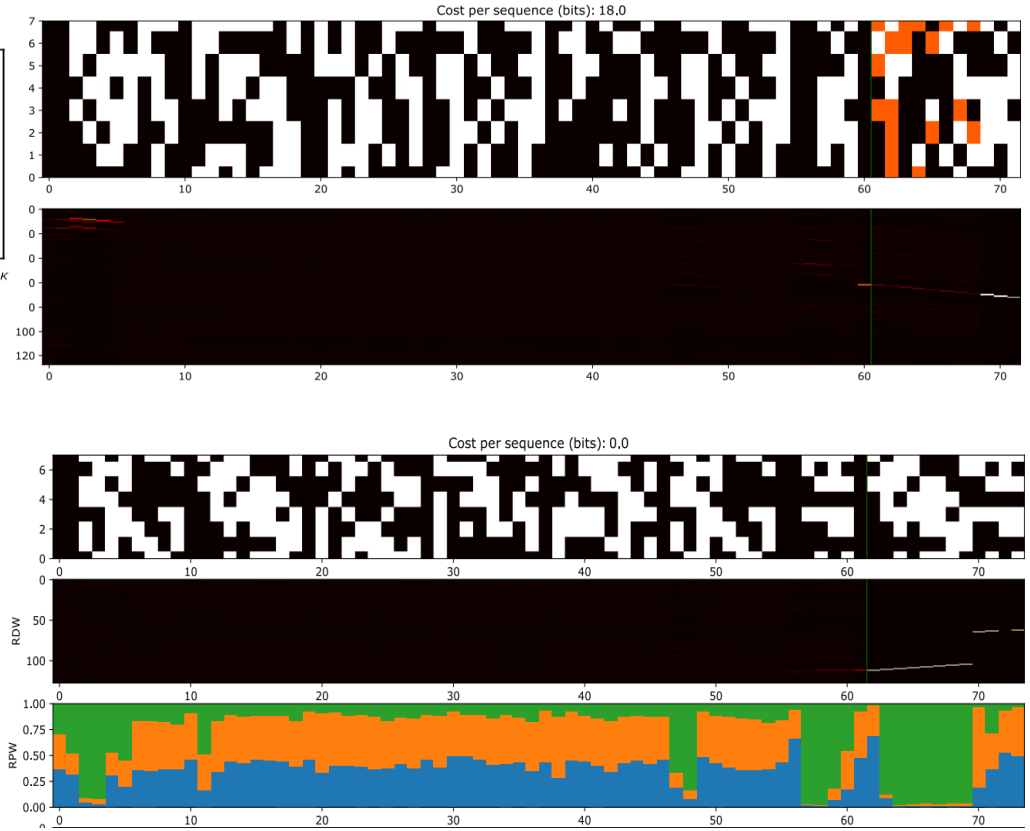


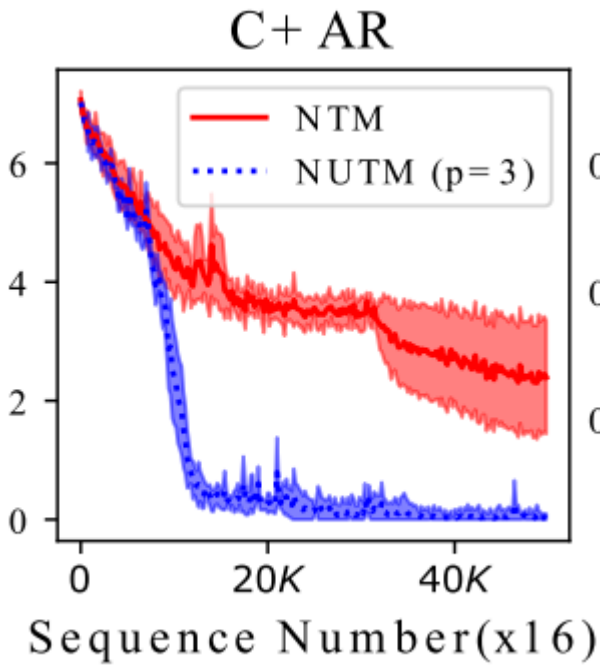
Graves, A., Wayne, G., Reynolds, M. *et al.* Hybrid computing using a neural network with dynamic external memory. *Nature* **538,** 471–476 (2016)

NUTM: learning multiple algorithms at once



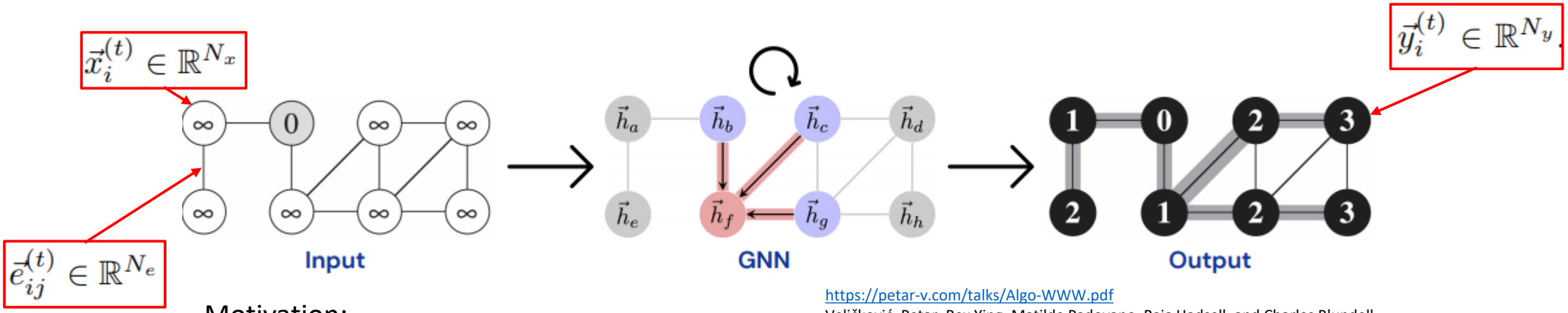






Le, Hung, Truyen Tran, and Svetha Venkatesh. "Neural Stored-program Memory."
In *International Conference on Learning Representations*. 2019.

Processor as graph neural network (GNN)



https://petar-v.com/talks/Algo-WWW.pdf
Veličković, Petar, Rex Ying, Matilde Padovano, Raia Hadsell, and Charles Blundell.
"Neural Execution of Graph Algorithms." In *International Conference on Learning Representations*. 2019.

Motivation:

- Many algorithm operates on graphs
- Supervise graph neural networks with algorithm operation/step/final output
- Encoder-Process-Decode framework:

$$\vec{z}_i^{(t)} = f_A(\vec{x}_i^{(t)}, \vec{h}_i^{(t-1)})$$

$$\mathbf{H}^{(t)} = P(\mathbf{Z}^{(t)}, \mathbf{E}^{(t)})$$

$$\vec{y}_i^{(t)} = g_A(\vec{z}_i^{(t)}, \vec{h}_i^{(t)})$$

$$\tau^{(t)} = \sigma(T_A(\mathbf{H}^{(t)}, \overline{\mathbf{H}^{(t)}}))$$

Attention

$$\vec{h}_i^{(t)} = \mathrm{ReLU}\left(\sum_{(j,i)\in E} a\left(\vec{z}_i^{(t)}, \vec{z}_j^{(t)}, \vec{e}_{ij}^{(t)}\right) \mathbf{W}\vec{z}_j^{(t)}\right) \qquad \vec{h}_i^{(t)} = U\left(\vec{z}_i^{(t)}, \bigoplus_{(j,i)\in E} M\left(\vec{z}_i^{(t)}, \vec{z}_j^{(t)}, \vec{e}_{ij}^{(t)}\right)\right)$$

Message passing

Example: GNN for a specific problem (DNF counting)

- Count #assignments that satisfy disjuntive normal form (DNF) formula
- Classical algorithm is P-hard O(mn)
- m: #clauses, n: #variables
- Supervised training on output-level

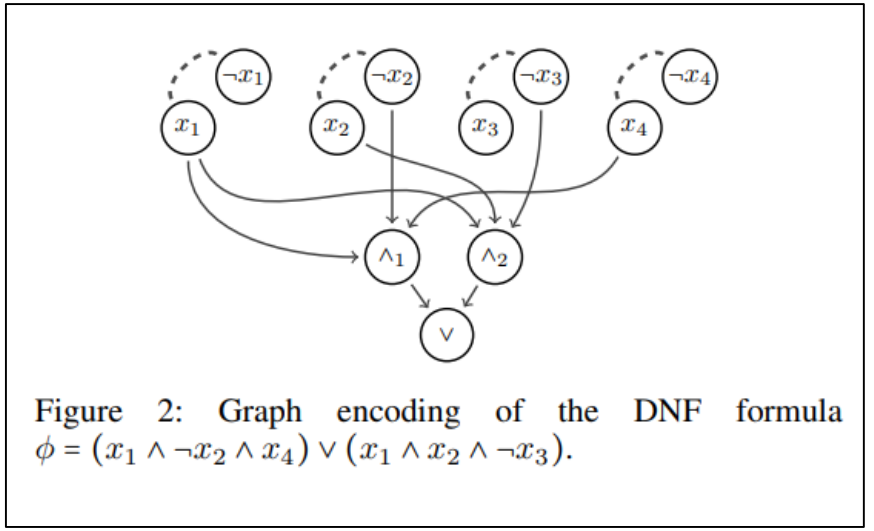


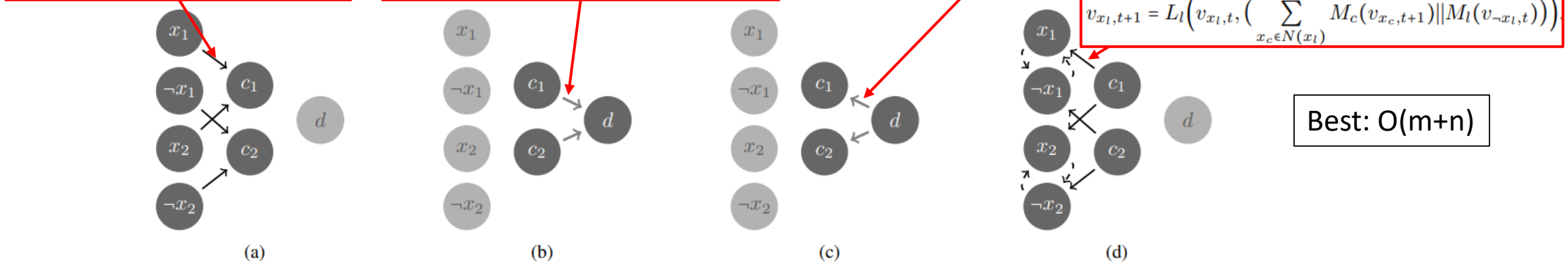
Figure 2: Graph encoding of the DNF formula $\phi = (x_1 \wedge \neg x_2 \wedge x_4) \vee (x_1 \wedge x_2 \wedge \neg x_3)$.

$$\hat{v}_{x_c,t+1} = L_{c_1}\Big(v_{x_c,t}, \sum_{x_l \in N(x_l)} M_l(v_{x_l,t})\Big)$$

$$v_{x_d,t+1} = L_d\Big(v_{x_d,t}, \sum_{x_c \in N(x_d)} M_c(\hat{v}_{x_c,t+1})\Big)$$

$$v_{x_c,t+1} = L_{c_2}\Big(\hat{v}_{x_c,t+1}, M_d(v_{x_d,t+1})\Big)$$

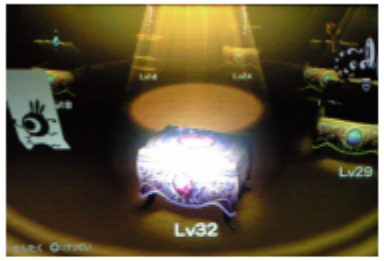
$$v_{x_l,t+1} = L_l\Big(v_{x_l,t}, \big(\sum_{x_c \in N(x_l)} M_c(v_{x_c,t+1}) \| M_l(v_{\neg x_l,t})\big)\Big)$$



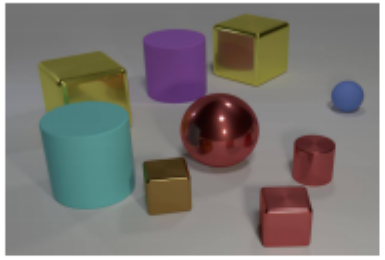
Best: O(m+n)

Abboud, Ralph, Ismail Ceylan, and Thomas Lukasiewicz. "Learning to reason: Leveraging neural networks for approximate DNF counting." In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, pp. 3097-3104. 2020.

Neural networks and algorithms alignment

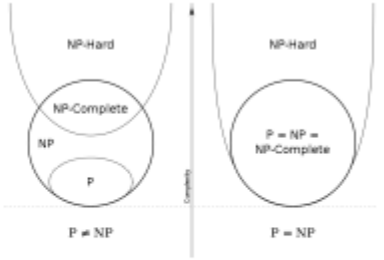


Summary statistics
What is the maximum value difference among treasures?

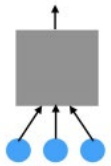


Relational argmax
What are the colors of the furthest pair of objects?

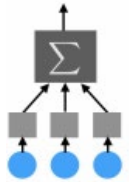
Dynamic programming
What is the cost to defeat monster X by following the optimal path?



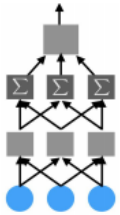
NP-hard problem
Subset sum: Is there a subset that sums to 0?



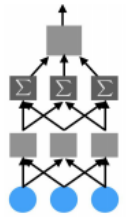
MLPs
~ feature extraction



Deep Sets (Zaheer *et al.*,
~ summary statistics



GNNs
~ (pairwise) relations



GNNs
~ (pairwise) relations

Neural exhaustive search

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

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

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

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

https://petar-v.com/talks/Algo-WWW.pdf

Xu, Keylu, Jingling Li, Mozhi Zhang, Simon S. Du, Ken-ichi Kawarabayashi, and Stefanie Jegelka. "What Can Neural Networks Reason About?." *ICLR 2020* (2020).

GNN is aligned with Dynamic Programming (DP)

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:

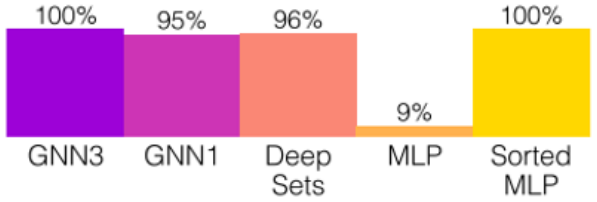
$d[k][u] = \min_v d[k-1][v] + cost(v, u)$

Learns a simple reasoning step

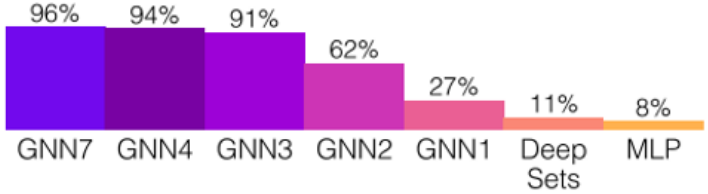
$h_u^{(k)}$ $d[k][u]$

Σ_v $\min_v$

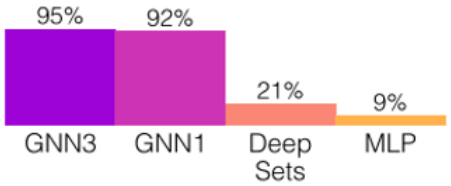
$MLP(h_v^{(k-1)}, h_u^{(k-1)})$ $d[k-1][v] + cost(v, u)$



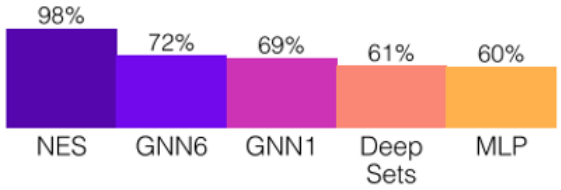
(a) Maximum value difference.



(c) Monster trainer.



(b) Furthest pair.



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

Neural exhaustive search

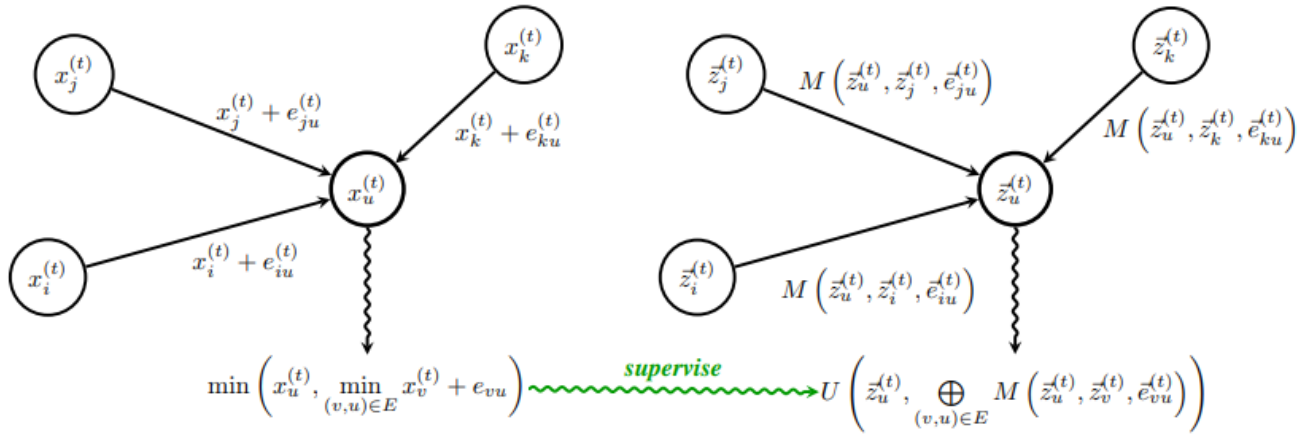
$$\mathrm{MLP}_2(\max_{\tau \subseteq S} \mathrm{MLP}_1 \circ \mathrm{LSTM}(X_1, ..., X_{|\tau|} : X_1, ..., X_{|\tau|} \in \tau)).$$

If alignment exists → step-by-step supervision

- Merely simulate the

classical graph algorithm, generalizable

- No algorithm discovery



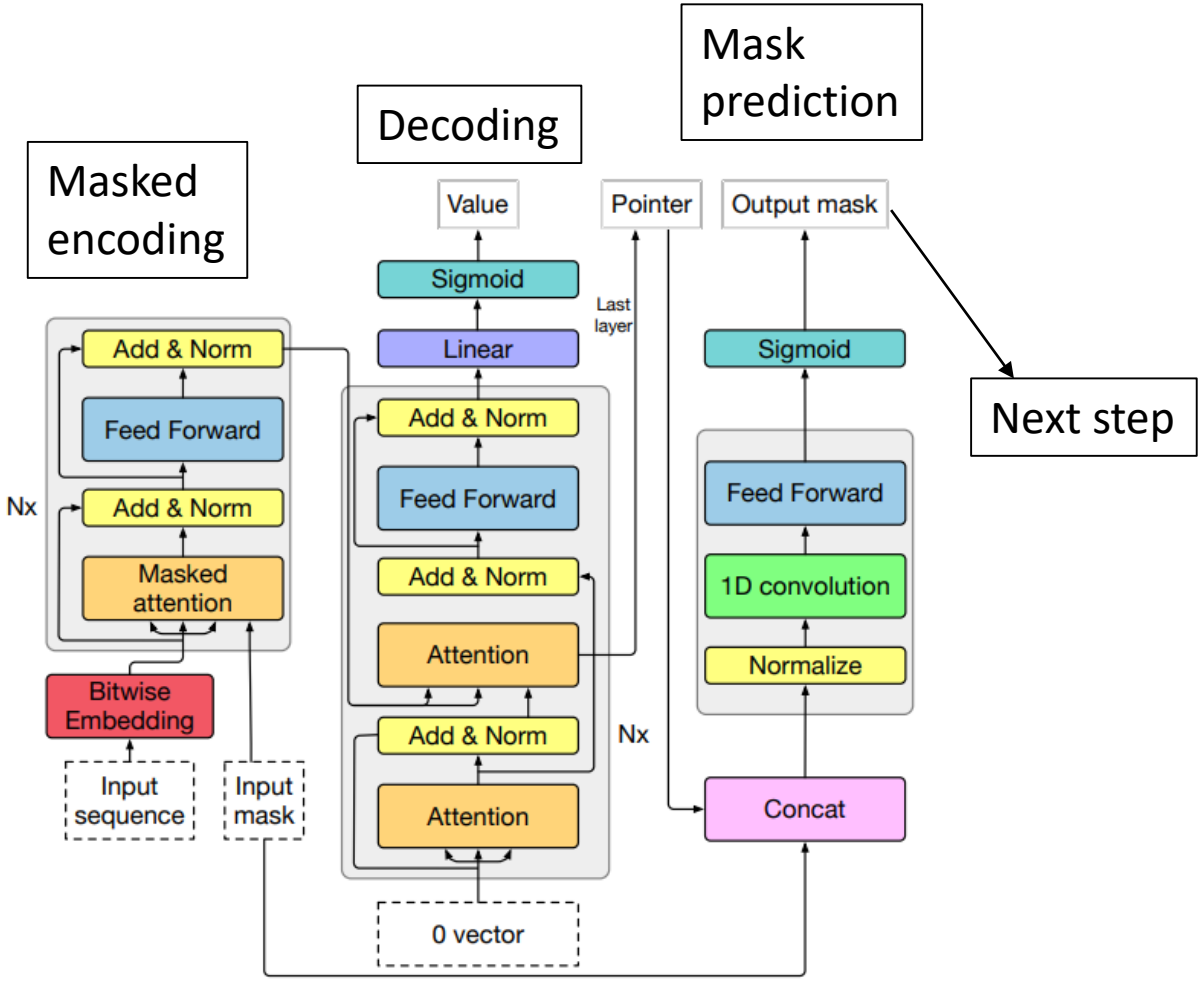
Algorithm	Inputs	Supervision signals
Breadth-first search	$x_i^{(t)}$: is i reachable from s in $\leq t$ hops?	$x_i^{(t+1)}$, $\tau^{(t)}$: has the algorithm terminated?
Bellman-Ford	$x_i^{(t)}$: shortest distance from s to i (using $\leq t$ hops)	$x_i^{(t+1)}$, $\tau^{(t)}$, $p_i^{(t)}$: predecessor of i in the shortest path tree (in $\leq t$ hops)
Prim's algorithm	$x_i^{(t)}$: is node i in the (partial) MST (built from s after t steps)?	$x_i^{(t+1)}$, $\tau^{(t)}$, $p_i^{(t)}$: predecessor of i in the partial MST

Joint training is encouraged

Veličković, Petar, Rex Ying, Matilde Padovano, Raia Hadsell, and Charles Blundell. "Neural Execution of Graph Algorithms." In *International Conference on Learning Representations*. 2019.

Processor as Transformer

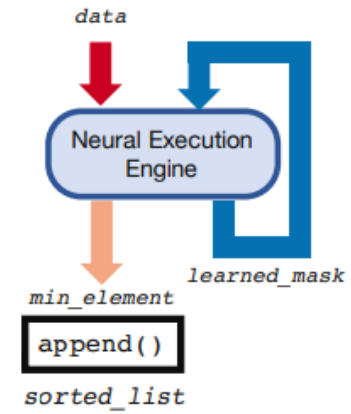
- Back to input sequence (set), but stronger generalization
- Transformer with encoder mask ~ graph attention
- Use Transformer with:
  - Binary representation of numbers
  - Dynamic conditional masking



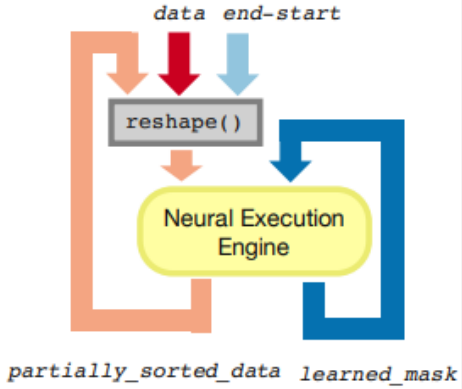
Yan, Yujun, Kevin Swersky, Danai Koutra, Parthasarathy Ranganathan, and Milad Hashemi.
"Neural Execution Engines: Learning to Execute Subroutines." *Advances in Neural Information Processing Systems* 33 (2020).

Training with execution trace

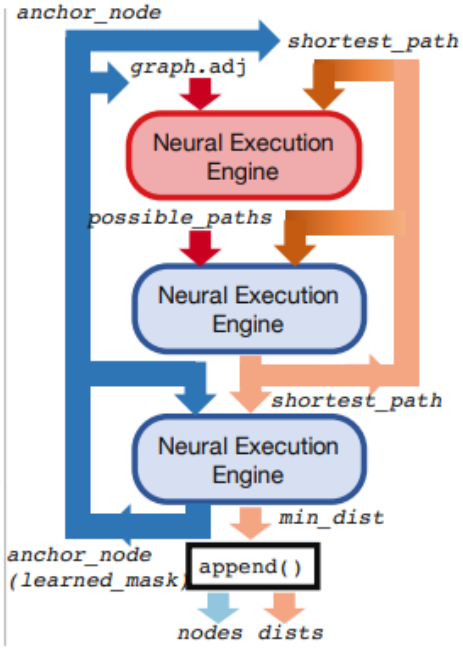
```
selection_sort(data):
  sorted_list = []
  while (len(data) > 0):
    min_index, min_element = find_min(data)
    data.delete(min_index)
    sorted_list.append(min_element)
  return sorted_list

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]
```



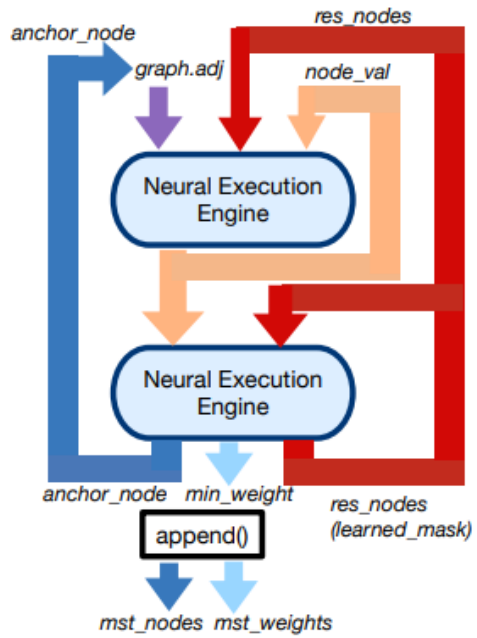

```
merge_sort(data, start, end):
  if (start < end):
    mid = (start + end ) / 2
    merge_sort(data, start, mid)
    merge_sort(data, mid+1, end)
    return merge(data, start, mid, end)
```




```
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))
    node_list.delete(anchor_node)
    nodes.append(anchor_node)
    dists.append(min_dist)
  return dists, nodes
```




```
minimum_spanning_tree(graph, source_node, node_val):
  mst_nodes = [ ]
  mst_weights = [ ]
  anchor_node = source_node
  res_nodes = graph.get_nodes()
  while node_list:
    adj_list = graph.adj(anchor_node)
    node_val(res_nodes) = min(node_val(res_nodes),
                              adj_list(res_nodes))
    anchor_node, min_weight = min(node_val(res_nodes))
    mst_nodes.append(anchor_node)
    mst_weights.append(min_weight)
    res_nodes.delete(anchor_node)
  return mst_nodes, mst_weights
```



End of part A

https://bit.ly/37DYQn7