Memory Architectures in Deep (Reinforcement) Learning

Rylan Schaeffer
March 15th, 2019

Deep Learning: Classics and Trends
• Motivation
• History of Memory Architectures in Deep Learning
• Neural Turing Machine (NTM)
• Differentiable Neural Computer (DNC)
• Memory, Reinforcement Learning and Inference Network (MERLIN)
Motivation

• Recurrent neural networks are theoretically Turing-complete [9], but practical problems proliferate
Motivation

• Recurrent neural networks are theoretically Turing-complete [9], but practical problems proliferate
• Vanilla LSTMs struggle on simple tasks requiring memory: copying sequences, adding numbers presented digit by digit, memorizing key-value pairs, etc
Motivation

• Recurrent neural networks are theoretically Turing-complete [9], but practical problems proliferate.
• Vanilla LSTMs struggle on simple tasks requiring memory: copying sequences, adding numbers presented digit by digit, memorizing key-value pairs, etc.
• Approach: Use introspective attention mechanism to manipulate, store, retrieve specific information (memory).
History of Memory Architectures in Deep Learning

- Memory Networks (2014) [14]
- **Neural Turing Machines (2014)** [4]
- End-to-End Memory Networks (2015) [10]
- **Differentiable Neural Computer (2016)** [5]
- Associative Long Short-Term Memory (2016) [2]
- Lie Access Neural Turing Machine (2016) [17]
- **Memory, RL and Inference Network (2018)** [12]
- Relational Memory Core (2018) [8]
- Reconstructive Memory Agent (2018) [6]
- Dynamic Kanerva Machine (2018) [16]
- Improvements to DNC (2019) [1]
Neural Turing Machine

- Couple a neural network to an external 2D matrix
- Enable network to learn reading/writing by defining interactions in differentiable manner
- Specifically, read & write are defined as soft attention mechanism over entire matrix
Neural Turing Machine

- Couple a neural network to an external 2D matrix
Neural Turing Machine

- Couple a neural network to an external 2D matrix
- Enable network to learn reading/writing by defining interactions in differentiable manner
Neural Turing Machine

- Couple a neural network to an external 2D matrix
- Enable network to learn reading/writing by defining interactions in differentiable manner
- Specifically, read & write are defined as soft attention mechanism over entire matrix
• **Weights**: Parameters of network
• **Weights**: Parameters of network

• **Weighting**: "Probability" vector \( w \) used to determine weighted combinations of memory contents
Terminology

- **Weights**: Parameters of network
- **Weighting**: "Probability" vector $w$ used to determine weighted combinations of memory contents
- The set of $N$-dimensional weightings $\Delta_N$ is defined as follows:

$$\Delta_N \overset{\text{def}}{=} \{ w \in \mathbb{R}^N : w_i \in [0, 1], \sum_{i=1}^{N} w_i \leq 1 \}$$
Terminology

- **Weights**: Parameters of network
- **Weighting**: "Probability" vector $w$ used to determine weighted combinations of memory contents
- The set of N-dimensional weightings $\Delta_N$ is defined as follows:
  \[
  \Delta_N \overset{\text{def}}{=} \{ w \in \mathbb{R}^N : w_i \in [0, 1], \sum_{i=1}^{N} w_i \leq 1 \} 
  \]
- Weighting with sum $< 1$ will be subtly influential for DNC
• Notation: Memory matrix has \( R \) rows and \( C \) columns, denoted \( M_t \)
• Notation: Memory matrix has $R$ rows and $C$ columns, denoted $M_t$
• To read from memory, network uses read heads
Neural Turing Machine [4] - Reading

- Notation: Memory matrix has \( R \) rows and \( C \) columns, denoted \( M_t \)
- To read from memory, network uses **read heads**
- Each read head emits a weighting \( w_t \in \Delta_R \)
Neural Turing Machine [4] - Reading

• Notation: Memory matrix has $R$ rows and $C$ columns, denoted $M_t$
• To read from memory, network uses read heads
• Each read head emits a weighting $w_t \in \Delta_R$
• Each read head returns to network a weighted combination of memory rows called a read vector, $r_t \in \mathbb{R}^C$

$$r_t \leftarrow M_t^T w_t$$
Neural Turing Machine [4] - Reading

- Notation: Memory matrix has $R$ rows and $C$ columns, denoted $M_t$
- To read from memory, network uses **read heads**
- Each read head emits a weighting $w_t \in \Delta_R$
- Each read head returns to network a weighted combination of memory rows called a **read vector**, $r_t \in \mathbb{R}^C$
  \[ r_t \leftarrow M_t^T w_t \]
- Network makes parallel reads, one per read head
- Notation: Memory matrix has $R$ rows and $C$ columns, denoted $M_t$
- To read from memory, network uses **read heads**
- Each read head emits a weighting $w_t \in \Delta_R$
- Each read head returns to network a weighted combination of memory rows called a **read vector**, $r_t \in \mathbb{R}^C$
  \[ r_t \leftarrow M_t^T w_t \]
- Network makes parallel reads, one per read head
- DNC differs only in how weighting $w_t$ is generated
To write to memory, network has write heads.
Neural Turing Machine [4] - Writing

- To write to memory, network has write heads
- Each write head emits three vectors:
  - Weighting vector $w_t$
  - Erase vector $e_t$
  - New content vector $v_t$

Each write head modifies every row in memory by (partially) erasing old values and adding new values $M_{t+1} = \circ (1 - w_t) + w_tv_t$,

Here, $\circ$ denotes element-wise multiplication.

Writes are performed in two sequential steps (erase, then add) to permit parallel writes.

DNC differs only in how weighting $w_t$ is generated.
• To write to memory, network has write heads
• Each write head emits three vectors:
  • Weighting $w_t \in \Delta_R$
• To write to memory, network has **write heads**
• Each write head emits three vectors:
  • Weighting $w_t \in \Delta_R$
  • Erase vector $e_t \in \mathbb{R}^C$ with values $\in (0, 1)$
To write to memory, network has **write heads**

Each write head emits three vectors:

- Weighting $w_t \in \Delta_R$
- Erase vector $e_t \in \mathbb{R}^C$ with values $\in (0, 1)$
- New content vector $v_t \in \mathbb{R}^C$
• To write to memory, network has \textbf{write heads}
• Each write head emits three vectors:
  • Weighting $w_t \in \Delta_R$
  • Erase vector $e_t \in \mathbb{R}^C$ with values $\in (0, 1)$
  • New content vector $v_t \in \mathbb{R}^C$

• Each write head modifies every row in memory by (partially) erasing old values and adding new values

$$M_t \leftarrow M_{t+1} \circ (1 - w_t e_t^T) + w_t v_t^T$$
• To write to memory, network has **write heads**

• Each write head emits three vectors:
  
  - Weighting $w_t \in \Delta_R$
  - Erase vector $e_t \in \mathbb{R}^C$ with values $\in (0, 1)$
  - New content vector $v_t \in \mathbb{R}^C$

• Each write head modifies every row in memory by (partially) erasing old values and adding new values

\[
M_t \leftarrow M_{t+1} \odot (1 - w_t e_t^T) + w_t v_t^T
\]

• Here, $\odot$ denotes element-wise multiplication
To write to memory, network has write heads.

Each write head emits three vectors:

- Weighting $w_t \in \Delta_R$
- Erase vector $e_t \in \mathbb{R}^C$ with values $\in (0, 1)$
- New content vector $v_t \in \mathbb{R}^C$

Each write head modifies every row in memory by (partially) erasing old values and adding new values:

$$M_t \leftarrow M_{t+1} \circ (1 - w_t e_t^T) + w_t v_t^T$$

Here, $\circ$ denotes element-wise multiplication.

Writes are performed in two sequential steps (erase, then add) to permit parallel writes.
Neural Turing Machine [4] - Writing

- To write to memory, network has **write heads**
- Each write head emits three vectors:
  - Weighting \( w_t \in \Delta_R \)
  - Erase vector \( e_t \in \mathbb{R}^C \) with values \( (0,1) \)
  - New content vector \( v_t \in \mathbb{R}^C \)

- Each write head modifies every row in memory by (partially) erasing old values and adding new values

\[
M_t \leftarrow M_{t+1} \circ (1 - w_t e_t^T) + w_t v_t^T
\]

- Here, \( \circ \) denotes element-wise multiplication
- Writes are performed in two sequential steps (erase, then add) to permit parallel writes
- DNC differs only in how weighting \( w_t \) is generated

- Four steps to generate each read/write head’s weightings $w_t$:
  
  1. **Content:** Network emits search key $k_t$ and search key strength $w_c \in \{1, 1\}$.

  - Softmax(Similarity($k_t; M_t[i]$))

  2. **Interpolation:** Network emits scalar $g_t$ to blend content-based weighting with previous weighting $w_t = g_t w_t + (1 - g_t) w_t$.

  3. **Location:** Network emits distribution over permitted shift values (e.g. $-1, 0, 1$) $s_t$ to rotationally shift weighting $w_t[i] = \sum_{R \in \{0, 1\}} w_g[j] s_t[i, j]$.

  4. **Sharpen:** Network emits scalar $t_1$ to sharpen weighting $w_t[i] = \text{Softmax}(w_l[i] t_1)$.

- Four steps to generate each read/write head’s weightings $w_t$:
  - Content: Network emits search key $k_t \in \mathbb{R}^C$ and search key strength $\beta \in [1, \infty)$
    \[
    w^c_t[i] \leftarrow \text{Softmax}(\beta \text{Similarity}(k_t, M_t[i]))
    \]

• Four steps to generate each read/write head’s weightings $w_t$:
  • Content: Network emits search key $k_t \in \mathbb{R}^C$ and search key strength $\beta \in [1, \infty)$
    
    $$w^c_t[i] \leftarrow \text{Softmax}(\beta \text{Similarity}(k_t, M_t[i]))$$

  • Interpolation: Network emits scalar $g_t$ to blend content-based weighting with previous weighting
    
    $$w^g_t \leftarrow g_t w^c_t + (1 - g_t) w_{t-1}$$

- Four steps to generate each read/write head’s weightings $w_t$:
  - Content: Network emits search key $k_t \in \mathbb{R}^C$ and search key strength $\beta \in [1, \infty)$
    \[
    w_t^c[i] \leftarrow \text{Softmax}(\beta \text{Similarity}(k_t, M_t[i]))
    \]
  - Interpolation: Network emits scalar $g_t$ to blend content-based weighting with previous weighting
    \[
    w_t^g \leftarrow g_t w_t^c + (1 - g_t)w_{t-1}
    \]
  - Location: Network emits distribution over permitted shift values (e.g. -1, 0, 1) $s_t$ to rotationally shift weighting (mod num of rows)
    \[
    w_t^l[i] \leftarrow \sum_{j=0}^{R-1} w_t^g[j] s_t[i - j]
    \]

- Four steps to generate each read/write head’s weightings $w_t$:
  - Content: Network emits search key $k_t \in \mathbb{R}^C$ and search key strength $\beta \in [1, \infty)$
    $$w_t^C[i] \leftarrow \text{Softmax}(\beta \text{Similarity}(k_t, M_t[i]))$$
  - Interpolation: Network emits scalar $g_t$ to blend content-based weighting with previous weighting
    $$w_t^g \leftarrow g_t w_t^C + (1 - g_t)w_{t-1}$$
  - Location: Network emits distribution over permitted shift values (e.g. -1, 0, 1) $s_t$ to rotationally shift weighting (mod num of rows)
    $$w_t^l[i] \leftarrow \sum_{j=0}^{R-1} w_t^g[j]s_t[i-j]$$
  - Sharpen: Network emits scalar $\gamma_t \geq 1$ to sharpen weighting
    $$w_t[i] = \text{Softmax}(w_t^l[i]^{\gamma_t})$$
• Problem: NTM has no mechanism to read sequential writes if a write head jumps

• Problem: NTM has no mechanism to prevent memory blocks from overlapping/interfering

• Problem: NTM has no mechanism to indicate memory blocks are no longer needed

• Approach: Use a temporal link matrix that represents degree to which row $i$ was written to after row $j$

- Problem: NTM has no mechanism to read sequential writes if a write head jumps
- Approach: Use a **temporal link matrix** that represents degree to which row $i$ was written to after row $j$

- Problem: NTM has no mechanism to read sequential writes if a write head jumps
- Approach: Use a temporal link matrix that represents degree to which row $i$ was written to after row $j$
- Problem: NTM has no mechanism to prevent memory blocks from overlapping/interfering

Proposal: Use different attention mechanisms for reading and for writing.

• Problem: NTM has no mechanism to read sequential writes if a write head jumps
• Approach: Use a temporal link matrix that represents degree to which row $i$ was written to after row $j$
• Problem: NTM has no mechanism to prevent memory blocks from overlapping/interfering
• Problem: NTM has no mechanism to indicate memory blocks are no longer needed

• Problem: NTM has no mechanism to read sequential writes if a write head jumps
• Approach: Use a **temporal link matrix** that represents degree to which row $i$ was written to after row $j$
• Problem: NTM has no mechanism to prevent memory blocks from overlapping/interfering
• Problem: NTM has no mechanism to indicate memory blocks are no longer needed
• Approach: Enable network to learn **dynamic memory management**
• Problem: NTM has no mechanism to read sequential writes if a write head jumps
• Approach: Use a **temporal link matrix** that represents degree to which row $i$ was written to after row $j$
• Problem: NTM has no mechanism to prevent memory blocks from overlapping/interfering
• Problem: NTM has no mechanism to indicate memory blocks are no longer needed
• Approach: Enable network to learn **dynamic memory management**
• Proposal: Use different attention mechanisms for reading and for writing
Differentiable Neural Computer [5] - Picture

Diagram showing the different components and connections of a differentiable neural computer, including:

- **a. Controller**
  - Input
  - Output

- **b. Read & Write Heads**
  - Write Vector
  - Write Key
  - Erase Vector
  - Read Key
  - Read Vectors

- **c. Memory**
  - N by W

- **d. Temporal Links**
  - Connections between different parts of the system.
Differentiable Neural Computer [5] - Reading

- Goal: Generate weighting based on content and location-based previous reads
• Goal: Generate weighting based on content and location-based previous reads
• Network emits search key $k_t \in \mathbb{R}^C$ and key strength $\beta \in [1, \infty)$ and computes content (read) weighting $c_t$

$$c_t[i] \leftarrow \text{Softmax}(\beta \text{Similarity}(k_t, M_t[i]))$$
Differentiable Neural Computer [5] - Reading

- Goal: Generate weighting based on content and location-based previous reads
- Network emits search key $k_t \in \mathbb{R}^C$ and key strength $\beta \in [1, \infty)$ and computes content (read) weighting $c_t$
  
  $$c_t[i] \leftarrow \text{Softmax}(\beta \text{Similarity}(k_t, M_t[i]))$$

- Assume a temporal link matrix $L_t \in [0, 1]^{R \times R}$ exists that represents degree to which row $i$ was written to after row $j$
Differentiable Neural Computer [5] - Reading

• Goal: Generate weighting based on content and location-based previous reads
• Network emits search key $k_t \in \mathbb{R}^C$ and key strength $\beta \in [1, \infty)$ and computes content (read) weighting $c_t$

$$c_t[i] \leftarrow \text{Softmax}(\beta \text{Similarity}(k_t, M_t[i]))$$

• Assume a temporal link matrix $L_t \in [0, 1]^{R \times R}$ exists that represents degree to which row $i$ was written to after row $j$
• Define forward weighting and backward weighting: $f_t, b_t \in \Delta_R$:

$$f_t \leftarrow L_t w_t^{\text{read}}$$

$$b_t \leftarrow L_t^T w_t^{\text{read}}$$
• Goal: Generate weighting based on content and location-based previous reads
• Network emits search key $k_t \in \mathbb{R}^C$ and key strength $\beta \in [1, \infty)$ and computes content (read) weighting $c_t$

$$c_t[i] \leftarrow \text{Softmax}(\beta \text{Similarity}(k_t, M_t[i]))$$

• Assume a temporal link matrix $L_t \in [0, 1]^{R \times R}$ exists that represents degree to which row $i$ was written to after row $j$
• Define forward weighting and backward weighting: $f_t, b_t \in \Delta_R$

$$f_t \leftarrow L_t w_{t-1}^{\text{read}}$$
$$b_t \leftarrow L_t^T w_{t-1}^{\text{read}}$$

• Network emits a read mode weighting $m_t \in \Delta_3$ to adjudicate between the backward, content and forward weightings

$$w_t^{\text{read}} \leftarrow m_t[1]b_t + m_t[2]c_t + m_t[3]f_t$$

- Goal: Track the degree that a row $i$ was written to after row $j$ using a temporal link matrix $L_t \in [0, 1]^{R \times R}$
• Goal: Track the degree that a row \( i \) was written to after row \( j \) using a **temporal link matrix** \( L_t \in [0, 1]^{R \times R} \)

• Define a precedence weighting \( p_t \in \Delta_R \), where \( p_t[i] \) represents degree to which row \( i \) was last row written to

\[
p_0 \leftarrow 0
\]

\[
p_t \leftarrow (1 - \sum_{r=1}^{R} w_t^{\text{write}}[r]) p_{t+1} + w_t^{\text{write}}
\]

- Goal: Track the degree that a row $i$ was written to after row $j$ using a temporal link matrix $L_t \in [0, 1]^{R \times R}$
- Define a precedence weighting $p_t \in \Delta_R$, where $p_t[i]$ represents degree to which row $i$ was last row written to

\[
p_0 \leftarrow 0
\]

\[
p_t \leftarrow (1 - \sum_{r=1}^{R} w_t^{\text{write}}[r]) p_{t+1} + w_t^{\text{write}}
\]

- Then, use the precedent weighting to construct the temporal link matrix

\[
L_0 \leftarrow 0
\]

\[
L_t[i, i] \leftarrow 0
\]

\[
L_t[i, j] \leftarrow (1 - w_t^{\text{write}}[i] - w_t^{\text{write}}[j]) L_{t-1}[i, j] + w_t^{\text{write}}[i] p_{t-1}[j]
\]
Differentiable Neural Computer [5] - Writing

• Goal: Write using content lookup, constrained by memory management system
Differentiable Neural Computer [5] - Writing

• Goal: Write using content lookup, constrained by memory management system
• Network emits search key $k_t \in \mathbb{R}^C$ and key strength $\beta \in [1, \infty)$ and computes content (write) weighting

$$c_t[i] \leftarrow \text{Softmax}(\beta \text{Similarity}(k_t, M_t[i]))$$
Differentiable Neural Computer [5] - Writing

• Goal: Write using content lookup, constrained by memory management system
• Network emits search key $k_t \in \mathbb{R}^C$ and key strength $\beta \in [1, \infty)$ and computes content (write) weighting

$$c_t[i] \leftarrow \text{Softmax}(\beta \text{Similarity}(k_t, M_t[i]))$$

• Assume learned dynamic memory management allocation weighting $a_t \in \Delta_R$ exists that represents degree to which each row can be written to
Differentiable Neural Computer [5] - Writing

• Goal: Write using content lookup, constrained by memory management system
• Network emits search key $k_t \in \mathbb{R}^C$ and key strength $\beta \in [1, \infty)$ and computes content (write) weighting
  $$c_t[i] \leftarrow \text{Softmax}(\beta \text{Similarity}(k_t, M_t[i]))$$
• Assume learned dynamic memory management allocation weighting $a_t \in \Delta_R$ exists that represents degree to which each row can be written to
• Network emits two gates, write and allocation $g^w_t, g^a_t \in [0, 1]$, to interpolate content weighting with allocation weighting
  $$w^\text{write}_t \leftarrow g^w_t (g^a_t a_t + (1 - g^a_t) c^w_t)$$
Differentiable Neural Computer [5] - Writing

• Goal: Write using content lookup, constrained by memory management system
• Network emits search key $k_t \in \mathbb{R}^C$ and key strength $\beta \in [1, \infty)$ and computes content (write) weighting
  $$c_t[i] \leftarrow \text{Softmax}(\beta \text{Similarity}(k_t, M_t[i]))$$
• Assume learned dynamic memory management allocation weighting $a_t \in \Delta_R$ exists that represents degree to which each row can be written to
• Network emits two gates, write and allocation $g^w_t, g^a_t \in [0, 1]$, to interpolate content weighting with allocation weighting
  $$w^\text{write}_t \leftarrow g^w_t (g^a_t a_t + (1 - g^a_t) c^w_t)$$
• Like NTM, network also emits erase vector $e_t \in (0, 1)^C$ and new content vector $v_t \in \mathbb{R}^C$, and updates the memory:
  $$M_t \leftarrow M_{t+1} \odot (1 - w^\text{write}_t e_t^T) + w^\text{write}_t v_t^T$$
• Goal: Specify which memory rows can be written to
- Goal: Specify which memory rows can be written to
- For each read head, network indicates whether previously read contents are still needed using a free gates $f^h_t \in [0,1]$
• Goal: Specify which memory rows can be written to
• For each read head, network indicates whether previously read contents are still needed using a free gates $f^h_t \in [0, 1]$
• Indicate which rows are still needed by creating usage weighting $u_t \in [0, 1]^R$

\[
\psi_t \leftarrow \prod_{h=1}^{\text{read heads}} (1 - f^h_t w^h_{t-1})
\]

\[
u_t \leftarrow (u_{t-1} + (1 - u_{t-1}) \circ w^{\text{write}}_{t-1}) \circ \psi_t
\]
• Goal: Specify which memory rows can be written to
• For each read head, network indicates whether previously read contents are still needed using a free gates $f^h_t \in [0, 1]$
• Indicate which rows are still needed by creating usage weighting $u_t \in [0, 1]^R$

\[
\psi_t \leftarrow \prod_{h=1}^{\text{read heads}} (1 - f^h_t w^h_{t-1})
\]

\[
u_t \leftarrow (u_{t-1} + (1 - u_{t-1}) \circ w_{t-1}^{\text{write}}) \circ \psi_t
\]

• Create the allocation weighting $a_t \in \Delta_R$ by sorting the usages. Let $\phi_t[i]$ be the index of the $i$-th least used location,

\[
a_t[\phi_t[j]] \leftarrow (1 - u_t[\phi_t[j]]) \prod_{i=1}^{j-1} u_t[\phi_t[i]]
\]
DNC [5] - Dynamic Memory Management Experiment

![Graph showing memory management experiment results.](image-url)
• How well does DNC perform at reasoning in graph structures, compared against NTM and LSTM?
DNC [5] - Testing Graph Experiments

• How well does DNC perform at reasoning in graph structures, compared against NTM and LSTM?
• 2 graph datasets
• How well does DNC perform at reasoning in graph structures, compared against NTM and LSTM?
• 2 graph datasets
  • bAbI [13]: programmatically generated natural language questions for textual reasoning
DNC [5] - Testing Graph Experiments

• How well does DNC perform at reasoning in graph structures, compared against NTM and LSTM?
• 2 graph datasets
  • bAbI [13]: programmatically generated natural language questions for textual reasoning
  • Randomly generated planar graphs consisting of 3-tuples: (source, destination, edge label)
How well does DNC perform at reasoning in graph structures, compared against NTM and LSTM?

2 graph datasets
- bAbI [13]: programmatically generated natural language questions for textual reasoning
- Randomly generated planar graphs consisting of 3-tuples: (source, destination, edge label)

3 types of queries: path traversal, shortest path, inferred relations
### DNC [5] - bAbI Results

<table>
<thead>
<tr>
<th>Task</th>
<th>LSTM</th>
<th>NTM</th>
<th>DNC1</th>
<th>DNC2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: 1 supporting fact</td>
<td>28.4±1.5</td>
<td>40.6±6.7</td>
<td>9.0±12.6</td>
<td>16.2±13.7</td>
</tr>
<tr>
<td>2: 2 supporting facts</td>
<td>56.0±1.5</td>
<td>56.3±1.5</td>
<td>39.2±20.5</td>
<td>47.5±17.3</td>
</tr>
<tr>
<td>3: 3 supporting facts</td>
<td>51.3±1.4</td>
<td>47.8±1.7</td>
<td>39.6±16.4</td>
<td>44.3±14.5</td>
</tr>
<tr>
<td>4: 2 argument relations</td>
<td>0.8±0.5</td>
<td>0.9±0.7</td>
<td>0.4±0.7</td>
<td>0.4±0.3</td>
</tr>
<tr>
<td>5: 3 argument relations</td>
<td>3.2±0.5</td>
<td>1.9±0.8</td>
<td>1.5±1.0</td>
<td>1.9±0.6</td>
</tr>
<tr>
<td>6: yes/no questions</td>
<td>15.2±1.5</td>
<td>18.4±1.6</td>
<td>6.9±7.5</td>
<td>11.1±7.1</td>
</tr>
<tr>
<td>7: counting</td>
<td>16.4±1.4</td>
<td>19.9±2.5</td>
<td>9.8±7.0</td>
<td>15.4±7.1</td>
</tr>
<tr>
<td>8: lists/sets</td>
<td>17.7±1.2</td>
<td>18.5±4.9</td>
<td>5.5±5.9</td>
<td>10.0±6.6</td>
</tr>
<tr>
<td>9: simple negation</td>
<td>15.4±1.5</td>
<td>17.9±2.0</td>
<td>7.7±8.3</td>
<td>11.7±7.4</td>
</tr>
<tr>
<td>10: indefinite knowledge</td>
<td>28.7±1.7</td>
<td>25.7±7.3</td>
<td>9.6±11.4</td>
<td>14.7±10.8</td>
</tr>
<tr>
<td>11: basic coreference</td>
<td>12.2±3.5</td>
<td>24.4±7.0</td>
<td>3.3±5.7</td>
<td>7.2±8.1</td>
</tr>
<tr>
<td>12: conjunction</td>
<td>5.4±0.6</td>
<td>21.9±6.6</td>
<td>5.0±6.3</td>
<td>10.1±8.1</td>
</tr>
<tr>
<td>13: compound coreference</td>
<td>7.2±2.3</td>
<td>8.2±0.8</td>
<td>3.1±3.6</td>
<td>5.5±3.4</td>
</tr>
<tr>
<td>14: time reasoning</td>
<td>55.9±1.2</td>
<td>44.9±13.0</td>
<td>11.0±7.5</td>
<td>15.0±7.4</td>
</tr>
<tr>
<td>15: basic deduction</td>
<td>47.0±1.7</td>
<td>46.5±1.6</td>
<td>27.2±20.1</td>
<td>40.2±11.1</td>
</tr>
<tr>
<td>16: basic induction</td>
<td>53.3±1.3</td>
<td>53.8±1.4</td>
<td>53.6±1.9</td>
<td>54.7±1.3</td>
</tr>
<tr>
<td>17: positional reasoning</td>
<td>34.8±4.1</td>
<td>29.9±5.2</td>
<td>32.4±8.0</td>
<td>40.2±11.1</td>
</tr>
<tr>
<td>18: size reasoning</td>
<td>5.0±1.4</td>
<td>4.5±1.3</td>
<td>4.2±1.8</td>
<td>4.3±2.1</td>
</tr>
<tr>
<td>19: path finding</td>
<td>90.9±1.1</td>
<td>86.5±19.4</td>
<td>64.6±37.4</td>
<td>75.8±30.4</td>
</tr>
<tr>
<td>20: agents motivations</td>
<td>1.3±0.4</td>
<td>1.4±0.6</td>
<td>0.0±0.1</td>
<td>0.0±0.0</td>
</tr>
<tr>
<td>Mean Error (%)</td>
<td>27.3±0.8</td>
<td>28.5±2.9</td>
<td>16.7±7.6</td>
<td>20.8±7.1</td>
</tr>
<tr>
<td>Failed Tasks (err. &gt; 5%)</td>
<td>17.1±1.0</td>
<td>17.3±0.7</td>
<td>11.2±5.4</td>
<td>14.0±5.0</td>
</tr>
</tbody>
</table>
DNC [5] - Custom Graphs Experiment

**Training Data**
- a. Random Graph

**Test Examples**
- b. London Underground

**Underground Input:**
- (Oxford Circus, Tottenham Ct Rd, Central)
- (Tottenham Ct Rd, Oxford Circus, Central)
- (Baker St, Marylebone, Circle)
- (Baker St, Oxford Circus, Bakerloo)
- (Leicester Sq, Charing Cross, Northern)
- (Tottenham Ct Rd, Leicester Sq, Northern)
- (Oxford Circus, Piccadilly Circus, Bakerloo)
- (Oxford Circus, Notting Hill Gate, Central)
- (Oxford Circus, Euston, Victoria)
- - 84 edges in total

**Traversal Question:**
- (Oxford Circus, ..., Central), (..., Circle)
- (..., Bakerloo), (..., Victoria), (..., Circle), (..., Bakerloo), (..., Jubilee)

**Answer:**
- (Oxford Circus, Notting Hill Gate, Central)
- (Notting Hill Gate, Paddington, Circle)
- (Embankment, Waterloo, Bakerloo)
- (Waterloo, Green Park, Jubilee)

**Shortest Path Question:**
- (Moorgate, Piccadilly Circus, Central)

**Answer:**
- (Moorgate, Bank, Northern)
- (Bank, Holborn, Central)
- (Holborn, Leicester Sq, Piccadilly)
- (Leicester Sq, Piccadilly Circus, Piccadilly)

**Family Tree Input:**
- (Charlotte, Alan, Father)
- (Simon, Steve, Father)
- (Steve, Simon, Son?1)
- (Melanie, Allison, Mother)
- (Lindsey, Fergus, Son?1)
- - 54 edges in total

**Inference Question:**
- (Freya, _, Maternal Great Uncle)

**Answer:**
- (Bob, Jane, Mother)
- (Natalie, Alice, Mother)
- (Mary, Ian, Father)
- (Jane, Alice, Daughter?1)
- (Mat, Charlotte, Mother)

**Answer:**
- (Freya, Fergus, Maternal Great Uncle)
DNC [5] - Custom Graphs Experiment
DNC [5] - Reinforcement Learning Experiment
DNC [5] - Reinforcement Learning Experiment

**a. Curriculum Progress**

**b. DNC Performance on Complete Curriculum**

**c. DNC Percent Optimal**

**d. LSTM Percent Optimal**
• Problem: On simple RL task, DNC required curriculum training to learn how to use its memory
• Problem: On simple RL task, DNC required curriculum training to learn how to use its memory
• Greg Wayne [7] spurns end-to-end learning, instead arguing:
• Problem: On simple RL task, DNC required curriculum training to learn how to use its memory

• Greg Wayne [7] spurns end-to-end learning, instead arguing:
  • Cost functions are diverse across areas and change over development
• Problem: On simple RL task, DNC required curriculum training to learn how to use its memory
• Greg Wayne [7] spurns end-to-end learning, instead arguing:
  • Cost functions are diverse across areas and change over development
  • Specialized systems allow efficient solution of key computational sub-problems
• Problem: On simple RL task, DNC required curriculum training to learn how to use its memory

• Greg Wayne [7] spurns end-to-end learning, instead arguing:
  • Cost functions are diverse across areas and change over development
  • Specialized systems allow efficient solution of key computational sub-problems

• Approach: Use separate objective functions and networks for memory, action selection.
• Problem: On simple RL task, DNC required curriculum training to learn how to use its memory

• Greg Wayne [7] spurns end-to-end learning, instead arguing:
  • Cost functions are diverse across areas and change over development
  • Specialized systems allow efficient solution of key computational sub-problems

• Approach: Use separate objective functions and networks for memory, action selection.

• Train memory to learn predictive (generative) model of world in unsupervised manner, and train actions via reinforcement learning, granting agent access to memory
Memory, RL and Inference Network [12] - Motivation

• Problem: On simple RL task, DNC required curriculum training to learn how to use its memory.
• Greg Wayne [7] spurns end-to-end learning, instead arguing:
  • Cost functions are diverse across areas and change over development.
  • Specialized systems allow efficient solution of key computational sub-problems.
• Approach: Use separate objective functions and networks for memory, action selection.
• Train memory to learn predictive (generative) model of world in unsupervised manner, and train actions via reinforcement learning, granting agent access to memory.
• Compared three agent architectures (LSTM, MEM, MERLIN) across a variety of tasks requiring memory.
• $I_t$: Image
• $v_t$: Egocentric translational and rotational velocity
• $r_{t-1}$: Previous reward
• $a_{t-1}$: Previous action
• $T$: (Optional) Text instruction
• $\tilde{h}_t$: LSTM
• $\tilde{n}_t$: Action probabilities
b. RL-MEM

\[ (I_t, v_t, r_{t-1}, T_t) \rightarrow o_t \rightarrow e_t \rightarrow \tilde{n}_t \rightarrow a_t \rightarrow \ldots \]

ENVIRONMENT

ENCODER

POLICY

\[ \tilde{m}_t \]

\[ M_t \]
c. MERLIN

MEMORY-BASED PREDICTOR

ENVIRONMENT

$(I_t, v_t, r_{t-1}, T_t)$

$e_t$ $\rightarrow$ $n_t$ $\leftarrow$ $p$ $\leftarrow$ $h_t$ $\leftarrow$ $m_t$

PRIOR

KL Loss

DECODER

$(\hat{I}_t, \hat{R}_t, \hat{v}_t, \hat{a}_{t-1}, \hat{r}_{t-1}, \hat{T}_t)$

Reconstruction Loss

READ-ONLY POLICY

READ

WRITE

Policy Loss
MERLIN [12] - Memory-Based Predictor

\[ \text{LSTM} \]

- \( h_t \) outputs a prior \( p_{t+1} \) for the next state variable \( z_{t+1} \)
- \( p_{t+1} \) concatenated with \( e_t \), fed through network to produce \( n_t \)
- State posterior \( q_t = p_{t+1} + n_t \)
- \( z_t \) sampled from \( q_t \), decoded to reconstruct observations and then appended as new row in \( M_t \)

\( \hat{I}_t, \hat{R}_t, \hat{v}_t, \hat{a}_{t-1}, \hat{r}_{t-1}, \hat{T}_t \)

Reconstruction Loss
MERLIN [12] - Memory-Based Predictor

- LSTM $h_{t-1}$ outputs a prior $p_{t-1}$ for the next state variable $z_t$
• LSTM $h_{t-1}$ outputs a prior $p_{t-1}$ for the next state variable $z_t$

• $p_{t-1}$ concatenated with $e_t$, fed through network to produce $n_t$
• LSTM $h_{t-1}$ outputs a prior $p_{t-1}$ for the next state variable $z_t$
• $p_{t-1}$ concatenated with $e_t$, fed through network to produce $n_t$
• State posterior $q_t \leftarrow p_{t-1} + n_t$
LSTM $h_{t-1}$ outputs a prior $p_{t-1}$ for the next state variable $z_t$

$p_{t-1}$ concatenated with $e_t$, fed through network to produce $n_t$

State posterior $q_t \leftarrow p_{t-1} + n_t$

$z_t$ sampled from $q_t$, decoded to reconstruct observations and then appended as new row in $M_t$
MERLIN [12] - Memory-Based Predictor

Problem: reconstructing inputs alone can result in loss of small, but critical information ("bullet problem").

Approach: Also reconstruct the return prediction $R_t[n]$. 

![Diagram of Memory-Based Predictor]
MERLIN [12] - Memory-Based Predictor

- Problem: reconstructing inputs alone can result in loss of small, but critical information ("bullet problem")
MERLIN [12] - Memory-Based Predictor

- Problem: reconstructing inputs alone can result in loss of small, but critical information (“bullet problem”)
- Approach: Also reconstruct the return prediction $\hat{R}_t$ [3]
MERLIN [12] - Memory-Based Predictor

Problem: Sampled state variables $z_t$ have no knowledge of subsequent events.

Approach:

Concatenate $z_t$ with filtered sum of subsequent state variables $\sum_{t' > t} z_{t'}$ in memory.

\[
\sum_{t' > t} z_{t'}
\]
• Problem: Sampled state variables $z_t$ have no knowledge of subsequent events.
• Problem: Sampled state variables $z_t$ have no knowledge of subsequent events
• Approach: Concatenate $z_t$ with filtered sum of subsequent state variables
  $$(1 - \gamma) \sum_{t' > t} \gamma^{t' - t} z_{t'}$$ in memory
Pairs of 8 Omniglot images are obscured. Agent looks at one image at a time, trying to find pairs.
MERLIN [12] - One-Shot Navigation
MERLIN [12] - One-Shot Navigation
MERLIN [12] - Latent Learning

(a) Conceptualization of phases:
- Phase 1
- Phase 2
- Phase 3

(b) Graphical representation of episode score over the number of environment steps, showing different models and their performance.

(c) Analysis of memory rows:
- Policy Read
- Activation of memory (a.u.)

(d) Graphical representation of goal arm decoding accuracy from MBP reads over agent steps from the start of Phase 3.
a. Arbitrary Visuomotor Mapping
Investigation of Gradient Stopping

- Default (but Single Optimizer)
- No Stop Gradient from Policy to z
- Stop Gradient from MEM to z
- No Stop Gradient from Policy to MEM


A. Graves, G. Wayne, and I. Danihelka.
**Neural turing machines.**

**Hybrid computing using a neural network with dynamic external memory.**

C.-C. Hung, T. Lillicrap, J. Abramson, Y. Wu, M. Mirza, F. Carnevale, A. Ahuja, and G. Wayne.
**Optimizing agent behavior over long time scales by transporting value.**


S. Sukhbaatar, J. Weston, R. Fergus, et al.  
**End-to-end memory networks.**  

O. Vinyals, M. Fortunato, and N. Jaitly.  
**Pointer networks.**  

G. Wayne, C.-C. Hung, D. Amos, et al.  
**Unsupervised predictive memory in a goal-directed agent.**  
Towards ai-complete question answering: A set of prerequisite toy tasks.  

J. Weston, S. Chopgra, and A. Bordes.  
Memory networks.  

Y. Wu, G. Wayne, A. Graves, and T. Lillicrap.  
The kanerva machine: A generative distributed memory.  
Y. Wu, G. Wayne, K. Gregor, and T. Lillicrap.  
Learning attractor dynamics for generative memory. 

G. Yang.  
Lie access neural turing machine. 
Get the source of this theme and the demo presentation from

github.com/matze/mtheme

The theme itself is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License.