Visual Reasoning by Progressive Module Networks

Presented by:
Santhosh Ramakrishnan, Yuqian Jiang

3/25/2019
Motivation

- Learn reasoning tasks sequentially in **progressive** complexity
- Many reasoning tasks can be broken down into a series of sequential reasoning steps
- For example, answering complex visual questions often require the skills to understand attributes such as color, recognize a variety of objects, and relate the objects spatially
Background

- Multi-task learning (MTL)
  - The dominant approach is sharing parameters
- Neural module network (NMN)
  - Translates from questions to dynamically assembled neural networks
- Visual Question Answering (VQA)
  - Treated as a compositional multi-task problem in this paper
Progressive Module Networks

- a framework for multi-task learning by progressively designing modules on top of existing modules.
PMN Components

Terminal Modules
- Lowest level of the tree
- Given an input $x$, return $M(x)$
- Ex: object recognition, attribute prediction

Compositional Modules
- Intermediate nodes
- Compose lower level modules in the tree to process inputs
- Ex: relation prediction, counting, image captioning, VQA
PMN Components

$\mathcal{E}$ - environment

$\mathcal{L}_n$ - list of submodules for $M_n$ 
$[M_m, \ldots, M_l]$

$\Delta_n$ - residual module

$\Omega_n$ - attention module

$\mathcal{V}$ - scratch pad

What are people watching?

Question

Image regions
1. State initializer: \( s^1 = I_3(q) \)
2. Importance function: \( [w_0^t, w_1^t, w_2^t] = G_3(s_t) \)
3. Query transmitter and receiver: \( q_k = Q_{n \rightarrow k}(s^t, V, G_n(s^t)) \quad v_k = R_{k \rightarrow n}(s^t, o_k) \)
4. State update function: \( s^{t+1} = U_n(s^t, V, \mathcal{E}, G_n(s^t)) \)
5. Prediction function: \( o_n = \Psi_n(s^1, \ldots, s^{T_n}, q_n, \mathcal{E}) \)
PMN Algorithm

1: function $M_n(q_n)$
2: $s^1 = I_n(q_n)$
3: for $t \leftarrow 1$ to $T_n$ do
4: $V = []$
5: $g^1, \ldots, g^{|\mathcal{L}_n|} = G_n(s^t)$
6: for $k \leftarrow 1$ to $|\mathcal{L}_n|$ do
7: $q_k = Q_{n \rightarrow k}(s^t, V, G_n(s^t))$
8: $o_k = \mathcal{L}_n[k](q_k)$
9: $v_k = R_{k \rightarrow n}(s^t, o_k)$
10: $V$.append($v_k$)
11: $s^{t+1} = U_n(s^t, V, \mathcal{E}, G_n(s^t))$
12: $o_n = \Psi_n(s^1, \ldots, s^{T_n}, q_n, \mathcal{E})$
13: return $o_n$

scratch pad

$\Omega_{vqa}$  $M_{rel}$  $\Delta_{vqa}$
$M_{obj}$  $M_{att}$
$\omega_{\Omega}$  $\omega_{rel}$  $\omega_{obj}$  $\omega_{att}$  $\omega_{\Delta}$  $\omega_{cnt}$  $\omega_{cap}$

$s_t = \{q_{vqa}^t, k_{t-1}^t\}$
Visual Reasoning Tasks

- **Object and Attribute Classification (level 0):** $M_{obj}$ and $M_{att}$
  - Input: a visual descriptor for a bounding box
  - Output: the penultimate vector prior to classification

- **Image Captioning (level 1):** $M_{cap}$
  - Input: image
  - Output: a natural language sentence (Anderson et al. 2018)
  - Submodules: $L_{cap} = [\Omega_{cap}, M_{obj}, M_{att}, \Delta_{cap}]$

- **Relationship Detection (level 1):** $M_{rel}$
  - Input: input box and relationship category (Ex: bird, standing on)
  - Output: the box for the related subject/object (Ex: bench)
  - Submodules: $L_{rel} = [M_{obj}, M_{att}, \Delta_{rel}]$
Visual Reasoning Tasks

- **Object Counting (level 2):** $M_{cnt}$
  - Input: image, vector representation of a natural language question (e.g. how many cats are on the blue chair?)
  - Output: a numerical count
  - Submodules: $\mathcal{L}_{cnt} = [\Omega_{cnt}, M_{rel}]$

- **Visual Question Answering (level 3):** $M_{vqa}$
  - Input: image, vector representation of a natural language question
  - Output: answer
  - Submodules: $\mathcal{L}_{vqa} = [\Omega_{vqa}, M_{rel}, M_{obj}, M_{att}, \Delta_{vqa}, M_{cnt}, \bar{M}_{cap}]$
PMN Algorithm

1: \textbf{function} $M_n(q_n)$
2: \hspace{1em} $s^1 = I_n(q_n)$
3: \hspace{1em} for $t \leftarrow 1$ to $T_n$ do
4: \hspace{2em} $V = \emptyset$
5: \hspace{2em} $g_n, \ldots, g_{|L_n|} = G_n(s^t)$
6: \hspace{2em} for $k \leftarrow 1$ to $|L_n|$ do
7: \hspace{3em} $q_k = Q_n \rightarrow_k (s^t, V, G_n(s^t))$
8: \hspace{3em} $o_k = L_n[k](q_k)$
9: \hspace{3em} $v_k = R_k \rightarrow_n (s^t, o_k)$
10: \hspace{3em} $V.\text{append}(v_k)$
11: \hspace{2em} $s^{t+1} = U_n(s^t, V, \mathcal{E}, G_n(s^t))$
12: \hspace{2em} $o_n = \Psi_n(s^1, \ldots, s^{T_n}, q_n, \mathcal{E})$
13: \hspace{1em} return $o_n$

\[ G_{vqa} = w_{\Omega} w_{rel} w_{obj} w_{att} w_{\Delta} w_{cnt} w_{cap} \]
PMN Algorithm

1: function $M_n(q_n)$
2:  
3:  
4:  
5:  
6:  
7:  
8:  
9:  
10:  
11:  
12:  
13:  

\[
\begin{split}
    s^1 &= I_n(q_n) \\
    V &= [] \\
    g^1_n, \ldots, g^{\left|\mathcal{L}_n\right|}_n &= G_n(s^t) \\
    q_k &= Q_{n \rightarrow k}(s^t, V, G_n(s^t)) \\
    o_k &= \mathcal{L}_n[k](q_k) \\
    v_k &= R_{n \rightarrow k}(s^t, o_k) \\
    V.\text{append}(v_k) \\
    s^{t+1} &= U_n(s^t, V, \mathcal{E}, G_n(s^t)) \\
    o_n &= \Psi_n(s^1, \ldots, s^{T_n}, q_n, \mathcal{E}) \\
\end{split}
\]

\[
    \text{Gather outputs of submodules}
\]
PMN Algorithm

1: function \( M_n(q_n) \)
2: \( s^1 = I_n(q_n) \)
3: for \( t \leftarrow 1 \) to \( T_n \) do
4:     \( V = \emptyset \)
5:     \( g^1, \ldots, g^{|\mathcal{C}_n|} = G_n(s^t) \)
6: for \( k \leftarrow 1 \) to \( |\mathcal{L}_n| \) do
7:     \( q_k = Q_{n \rightarrow k}(s^t, V, G_n(s^t)) \)
8:     \( o_k = \mathcal{L}_n[k](q_k) \)
9:     \( v_k = R_{k \rightarrow n}(s^t, o_k) \)
10: \( V.\text{append}(v_k) \)
11: \( s^{t+1} = U_n(s^t, V, \mathcal{E}, G_n(s^t)) \)
12: \( o_n = \Psi_n(s^1, \ldots, s^{T_n}, q_n, \mathcal{E}) \)
13: return \( o_n \)
PMN Algorithm

\begin{align*}
1: & \textbf{function } M_n(q_n) \\
2: & s^1 = I_n(q_n) \\
3: & \textbf{for } t \leftarrow 1 \textbf{ to } T_n \textbf{ do} \\
4: & \quad V = [] \\
5: & \quad g^1_n, \ldots, g^{\mid L_n \mid}_n = G_n(s^t) \\
6: & \quad \textbf{for } k \leftarrow 1 \textbf{ to } \mid L_n \mid \textbf{ do} \\
7: & \quad \quad q_k = Q_n \rightarrow k(s^t, V, G_n(s^t)) \\
8: & \quad \quad o_k = L_n [k](q_k) \\
9: & \quad \quad v_k = R_k \rightarrow n(s^t, o_k) \\
10: & \quad \quad V.\text{append}(v_k) \\
11: & \quad s^{t+1} = U_n(s^t, V, E, G_n(s^t)) \\
12: & \quad o_n = \Psi_n(s^1, \ldots, s^{T_n}, q_n, E) \\
13: & \textbf{return } o_n 
\end{align*}
## Experiments

- Performance of PMN vs. state-of-the-art models for the VQA task

<table>
<thead>
<tr>
<th>Model</th>
<th>Ens</th>
<th>VQA 2.0 val</th>
<th>VQA 2.0 test-dev</th>
<th>VQA 2.0 test-std</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Yes/No</td>
<td>Number</td>
<td>Other</td>
</tr>
<tr>
<td>Andreas et al. (2016)</td>
<td>-</td>
<td>Yes/No</td>
<td>Number</td>
<td>Other</td>
</tr>
<tr>
<td>Yang et al. (2016)</td>
<td>-</td>
<td>Yes/No</td>
<td>Number</td>
<td>Other</td>
</tr>
<tr>
<td>Teney et al. (2018)</td>
<td>-</td>
<td>Yes/No</td>
<td>Number</td>
<td>Other</td>
</tr>
<tr>
<td>Teney et al. (2018)</td>
<td>✓</td>
<td>Yes/No</td>
<td>Number</td>
<td>Other</td>
</tr>
<tr>
<td>Yu et al. (2018)</td>
<td>-</td>
<td>Yes/No</td>
<td>Number</td>
<td>Other</td>
</tr>
<tr>
<td>Yu et al. (2018)</td>
<td>✓</td>
<td>Yes/No</td>
<td>Number</td>
<td>Other</td>
</tr>
<tr>
<td>Zhang et al. (2018)</td>
<td>-</td>
<td>Yes/No</td>
<td>Number</td>
<td>Other</td>
</tr>
<tr>
<td><strong>baseline $M_{VQA}$</strong></td>
<td>-</td>
<td>Yes/No</td>
<td>Number</td>
<td>Other</td>
</tr>
<tr>
<td><strong>PMN $M_{VQA}$</strong></td>
<td>-</td>
<td>Yes/No</td>
<td>Number</td>
<td>Other</td>
</tr>
<tr>
<td><strong>PMN $M_{VQA+}$</strong></td>
<td>✓</td>
<td>Yes/No</td>
<td>Number</td>
<td>Other</td>
</tr>
</tbody>
</table>

### Diagram Table

- $v_{A}$
- $V$
- $\Omega_{vqa}$
- $M_{rel}$
- $M_{obj}$
- $M_{att}$
- $\Delta_{vqa}$
- $M_{cnt}$
- $M_{cap}$

- a bird sitting on top of a wooden bench

- 0.11
- 0.89
- 0.39
- 0.06
- 0.22
- 0.10
- 0.23
Experiments

- PMN can exploit previously learned knowledge
- Absolute gain in accuracy when using a fraction of the training data:

<table>
<thead>
<tr>
<th>Fraction of VQA training data (in %)</th>
<th>1</th>
<th>5</th>
<th>10</th>
<th>25</th>
<th>50</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Absolute accuracy gain (in %)</td>
<td>-0.49</td>
<td>2.21</td>
<td><strong>4.01</strong></td>
<td>2.66</td>
<td>1.79</td>
<td>2.04</td>
</tr>
</tbody>
</table>
Interpretability Analysis

- Rule-based explanations generated from the modules used

Q: what is behind the men?

- I first find the **BLUE** box, and then from that, I look at the **GREEN** box.
- The object 'tree' would be useful in answering the question.
- In conclusion, I think the answer is **trees**.

Query $M_{rel}$ with **BLUE** box and “behind” $M_{rel}$ outputs the **GREEN** box

Object and captioning modules produce “tree”
Interpretability Analysis

- Human evaluation with 1,600 randomly chosen questions on Mechanical Turk

Table 2: Average human judgments from 0 to 4. ✓ indicates that model got final answer right, and ✗ for wrong.

<table>
<thead>
<tr>
<th>Correct?</th>
<th>PMN</th>
<th>Baseline</th>
<th># Q</th>
<th>Human Rating</th>
<th>PMN</th>
<th>Baseline</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>715</td>
<td>3.13</td>
<td>2.86</td>
<td></td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>584</td>
<td>2.78</td>
<td>1.40</td>
<td></td>
</tr>
<tr>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>162</td>
<td>1.73</td>
<td>2.47</td>
<td></td>
</tr>
<tr>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>139</td>
<td>1.95</td>
<td>1.66</td>
<td></td>
</tr>
<tr>
<td>All images</td>
<td>1600</td>
<td></td>
<td>2.54</td>
<td>2.24</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

PMN achieves better score when both answers are correct
PMN gets more “partial credit” when its answers are wrong
PMN makes the model more interpretable
Critiques

● Interpretability comes at a price:
  ○ Restricts model performance by forcing it to follow a hierarchy
  ○ It is hard to generalize VQA hierarchy to other reasoning tasks

● Modules are not jointly fine-tuned
  ○ to ensure interpretability
  ○ to reduce computation while training

● Design choices for modules not justified / validated
Critiques

● Only marginal improvement in performance over other methods
  ○ No significant improvement in “Other” category of questions
● Need to define modules that have existing datasets for training
● It is unclear how they arrive at explanations based on model outputs
● Need to evaluate every submodule for each question - inefficient
Future Work

- Learn the task hierarchy
- Joint fine-tuning of all modules
- Using hard attention to select input-specific submodules for efficiency
- How to evaluate PMN on a different reasoning task?
  - How much effort will it take to apply PMN to a new problem?
  - Might be too much to ask because of availability of datasets for subtasks