May my AI systems be better assured?

Shing-Chi Cheung

Supported by ITC under MHP-055-19
Sound familiar?

Google search!  Hope it works!  Hmm!  Why?

Google search!  Should work?!  What?  How come??
A Common Issue in AI Systems

Does my AI system perform reliably?
Defects in AI systems can bring great loss

- Defects occurring at deployment can pose threats to lives and economy
- Defects occurring at training can waste weeks or months of valuable computational resources
  - Result in NaN, crashes and gradient vanishing after a long training period

2018/3/19: Uber car hit a pedestrian, causing death due to incorrect object detection
How frequent are real-life AI projects updated?

<table>
<thead>
<tr>
<th>Deep Learning Project</th>
<th>Code Size</th>
<th>#Code Commits</th>
<th>#Commits in recent 6 months</th>
<th>#Issues reported in recent 6 month</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transformer</td>
<td>29,409</td>
<td>1,971</td>
<td>1,549</td>
<td>891</td>
</tr>
<tr>
<td>DeepSpeech</td>
<td>145,167</td>
<td>2,108</td>
<td>594</td>
<td>252</td>
</tr>
<tr>
<td>Real-Time-Voice-Clone</td>
<td>5,126</td>
<td>237</td>
<td>148</td>
<td>148</td>
</tr>
<tr>
<td>DeepCreamPy</td>
<td>878</td>
<td>376</td>
<td>86</td>
<td>37</td>
</tr>
</tbody>
</table>

Source: Four most popular active TensorFlow projects on Github
Figures based on end of 2019
Can bugs in neural networks be detected before training?
Absence of testing tools for DL systems

- There are commercial testing tools for conventional software
- BUT none are designed for AI systems or modules
First empirical study on bugs in DL programs

|                               | StackOverflow QAs | Github Projects |
|                               |                  |                |
| Counts                        | 87               | 88             |

StackOverflow QAs
- Searched Tensorflow related questions
- Manually reviewed QA pages
- Analyzed answers and discussions

Github Projects
- Searched Tensorflow related projects
- Manually reviewed commits
- Analyzed commit/pull request messages and issue discussions

Selected findings from popular Github projects

• Failures often occur at the training stage and after many training cycles

• Testing one training instance alone is unlikely to catch such failures

• When failures occur, the error messages are often confusing
  • Error messages may not pinpoint which parts of the software go wrong
  • Discussions at StackOverflow suggest that fault determination is non-trivial even for a small AI system
A common issue in many AI projects

- API misuses are common
  - API mostly designed for numeric computation
  - API documentation is either brief or difficult to follow

\[
\begin{align*}
h_{fc3} &= \text{tf.nn.relu}(\text{conv2d}(h_{fc1\_drop}, W_{fc2}) + b_{fc2}) \\
y_{conv} &= \text{tf.nn.softmax}(\text{tf.reshape}(h_{pool3}, [-1, 10])) \\
cross\_entropy &= -\text{tf.reduce\_sum}(y_{\_\*tf.log(y_{conv}))} \\
train\_step &= \text{tf.train.AdamOptimizer}(1e^{-4}).\text{minimize}(cross\_entropy)
\end{align*}
\]

- API often evolves to meet dynamic market demand and algorithm advancement

Stack Overflow #33699174
DEBAR: Scalable AI Defect Analyzer

We propose two abstraction techniques:

1. Tensor Partitioning
   Numeric computation can be abstracted using intervals
   Many tensor elements are subject to the same computation

2. Interval Abstraction with Affine Equality Relation
   Many computations are affine operations ($w_0 = \sum_i w_i x_i$)
   -> affine equality relations to enhance the precision of interval abstraction
Our technique

(Tensor Partitioning + Affine Equality Relation):
Accuracy: 93.0%, all in 3 minutes, 12.1s on average
100% accuracy on 9 buggy architectures

Tensor Partitioning + Sole Interval Abstraction
Accuracy: 80.6%, 12.1s on average

Main results

Framework
(Tensor Abstraction + Numerical Abstraction)

Tensor Expansion + Affine Equality Relation:
33/57 > 30mins; on rest 24, DEBAR doesn’t lose accuracy

Tensor Smashing + Affine Equality Relation:
Accuracy: 87.1%, 12.2s on average

Tool implemented in Python
https://github.com/ForeverZyh/DEBAR

\[
\sigma(A) = \begin{bmatrix} [0,1] & [-1,0] \\ [-1,0] & [1,1] \end{bmatrix}
\]

Accurac\(\ y = \frac{\#TP + \#TN}{\#TP + \#TN + \#FP + \#FN}\)
CASTLE Group - Code AnalySe, Testing and LEarning

http://castle.cse.ust.hk/castle/people.html