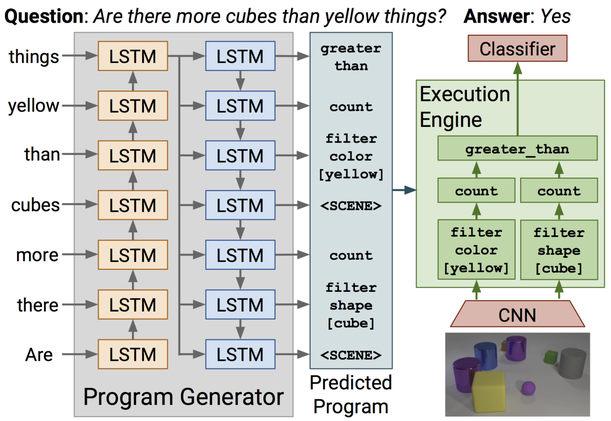
Fast and CLEVR Visual Reasoning Programs via Improved Topologically Batched Graph Execution

Joseph Suarez and Clare Zhu

Background

Johnson et al. recently demonstrated the effectiveness of neural module based programs for visual question answering, reducing error rates on CLEVR by 8.6X over strong baselines and 2.3X over human evaluators. We address and remedy performance issues of the authors' architecture, providing both strong computational complexity bounds and practical speedups of over 14X.

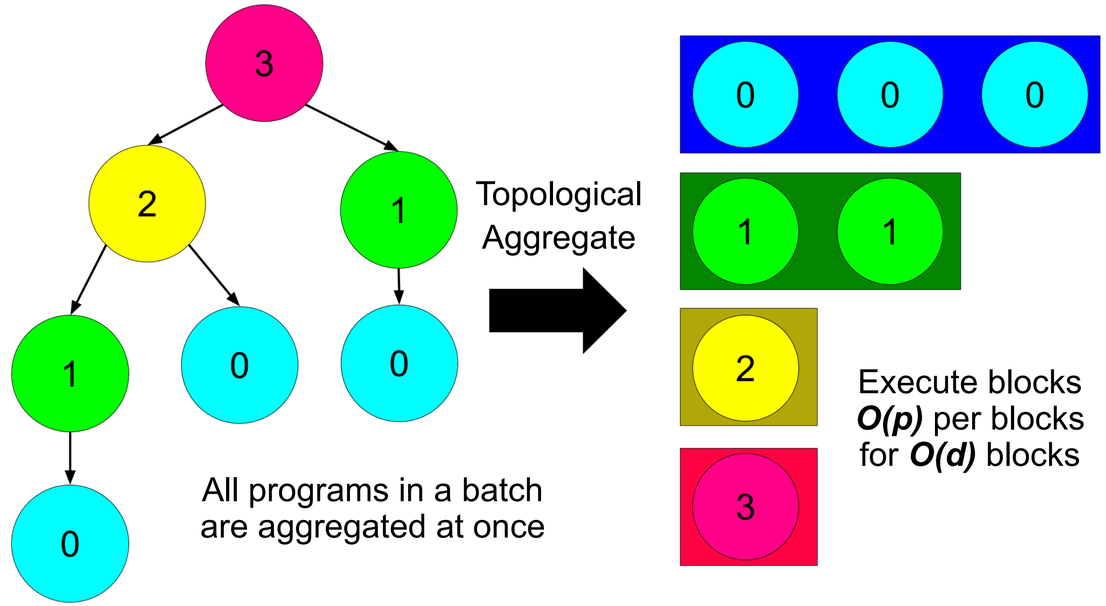


Johnson et al, 2017. Inferring and Executing Programs for Visual Reasoning.

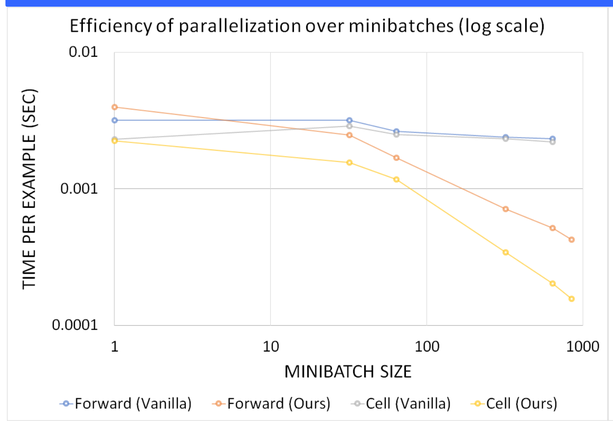
Motivation

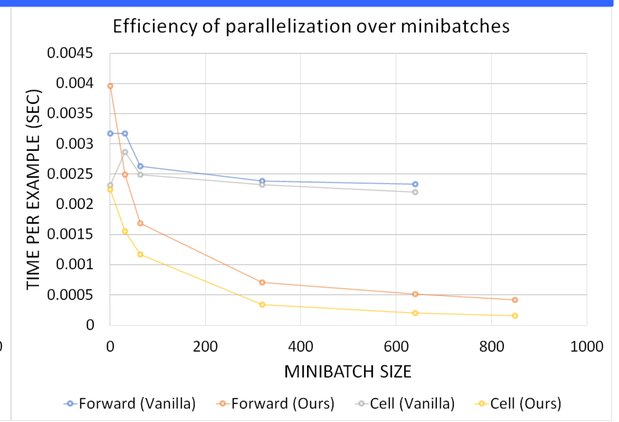
Modern neural network architectures achieve >10X speedups on GPU via parallelization over batch size. As Justin's model assembles per-example architectures, this is not feasible in the forward pass and in practice yields <2X speedups during the backward pass, regardless of batch size. This is an impediment to expansion upon and adaptation of his model.

Improved Topological Program Graph Sort



Efficiency Gains





Theoretical Bounds

p = program vocabulary size
s = max program length
b = batch size
d = max tree depth

Vanilla: ***O(bs)***
Topological Sort: ***O(ps)***
Improved Topological Sort: ***O(pd)***
+ Balanced Program Trees: ***O(p log$_2$ d)***

Dataset

CLEVR is a synthetic but realistic visual question answering dataset consisting of (question, image, answer) triplets. It also contains a functional program representation of each question (accessible during training).

Practical Considerations

- Theoretical bounds assume perfect parallelization over batch size.
- Improved Topological Sort introduces additional CPU code and data split/concat operations that could be further optimized.
- The execution engine is the *main* performance bottleneck.
- Backwards pass inefficient due to current PyTorch backend; this is not a theoretical limitation. Our bound holds for the backwards pass.

Conclusion

We successfully implemented the authors' model; using our improved topological sort on the model, we achieved speedups of over:

- 2.0X for overall training
- 5.5X for the forward pass
- 14X during cell execution

while being ~30% more memory efficient.