Microsoft Research

Microsoft

AMPNet: Distributed Asynchronous Training for Dynamic Neural Networks

Alex Gaunt, Dimitrios Vytiniotis, Matthew A. Johnson,
Maik Riechert, Ryota Tomioka, Sam Webster

Jun 13, 2017 @ EcoCloud Workshop

Deep nets – what's missing?

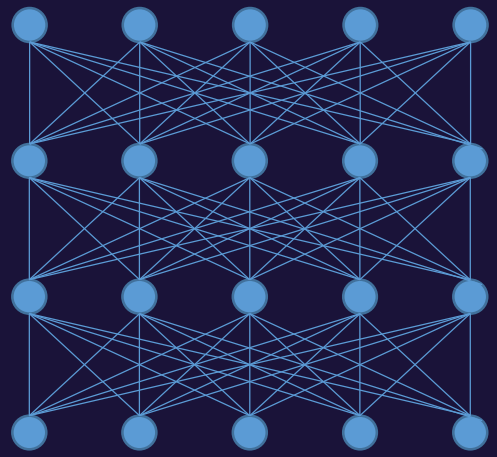
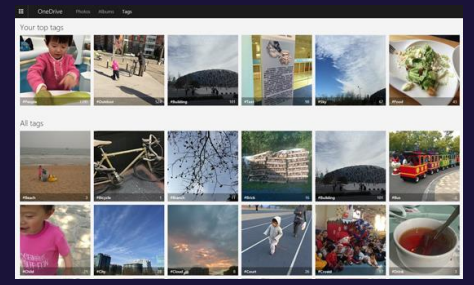
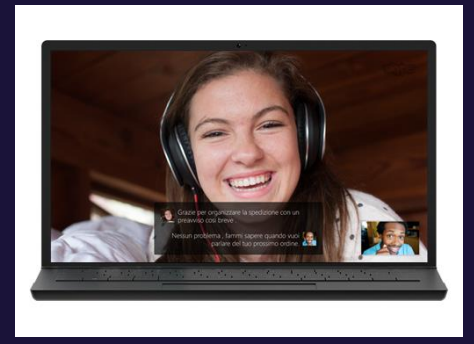


Image recognition



Speech recognition

Translation



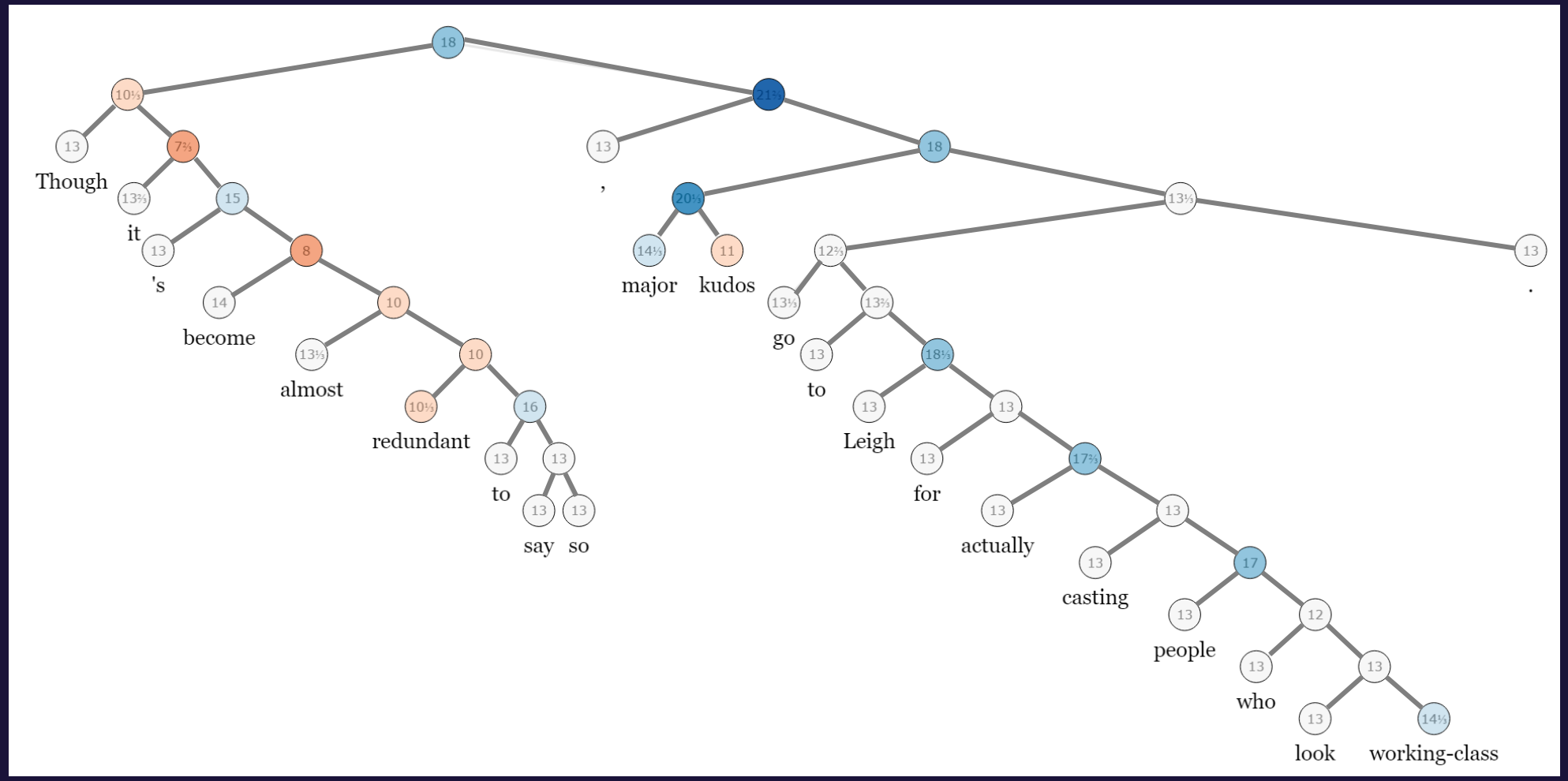
Deep nets – what's missing?

Uniform unstructured computation – cannot handle structured input

Elementwise nonlinear

Dense matrix multiply

Stanford sentiment tree bank



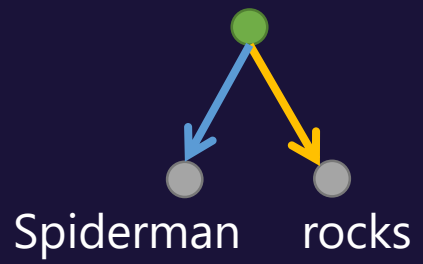
10k intensely annotated trees. https://nlp.stanford.edu/sentiment/treebank.html

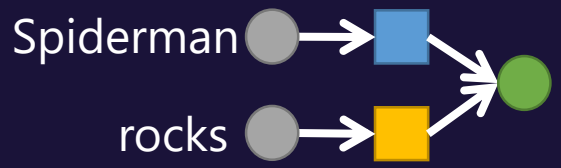
Structured input invites structured computation

Instances

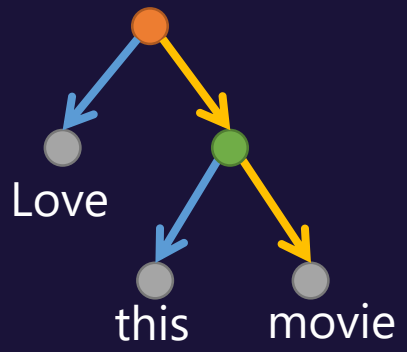
Computational graphs

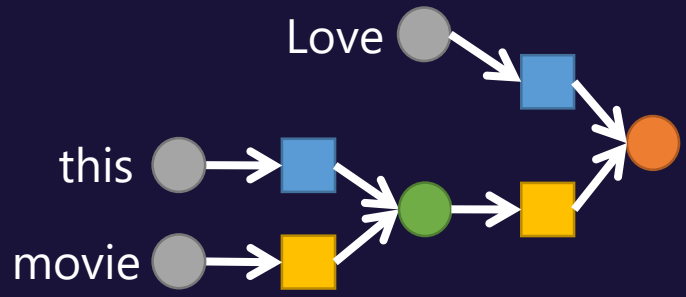
Instance 1





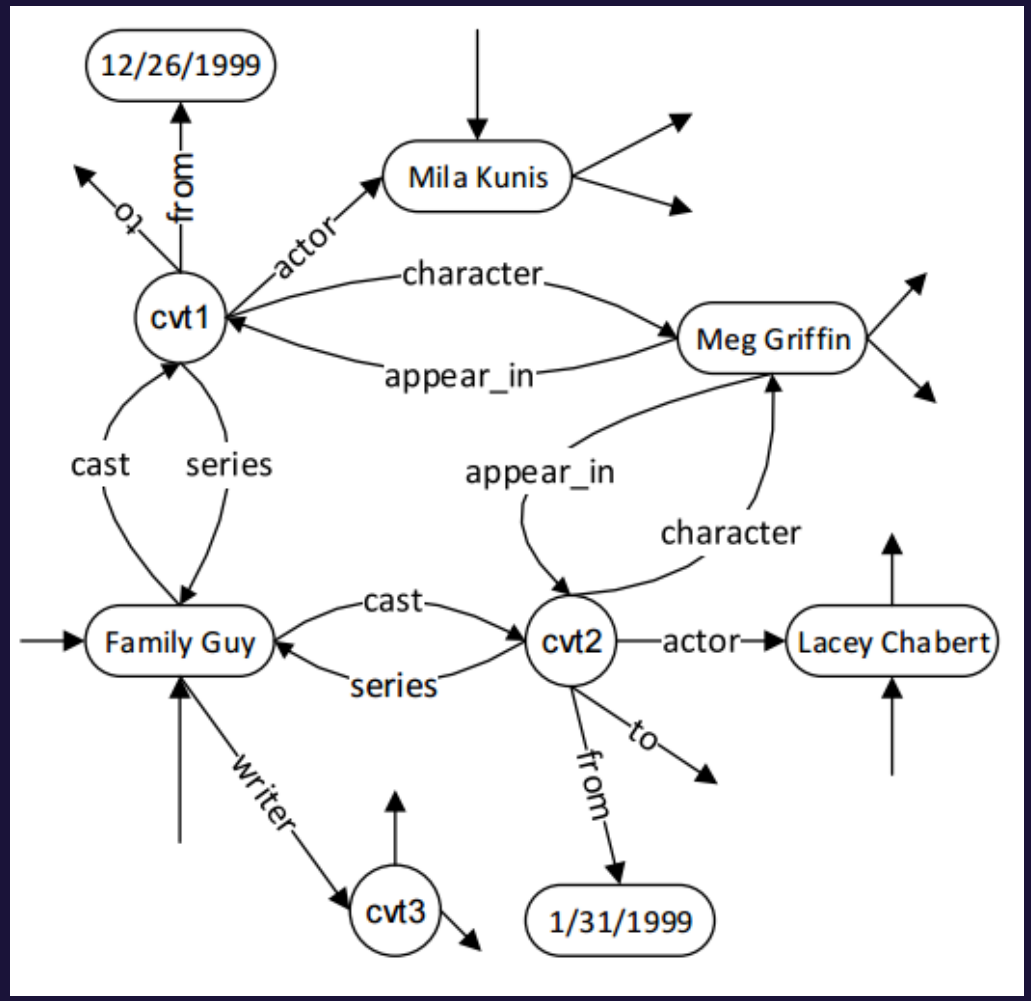
Instance 2





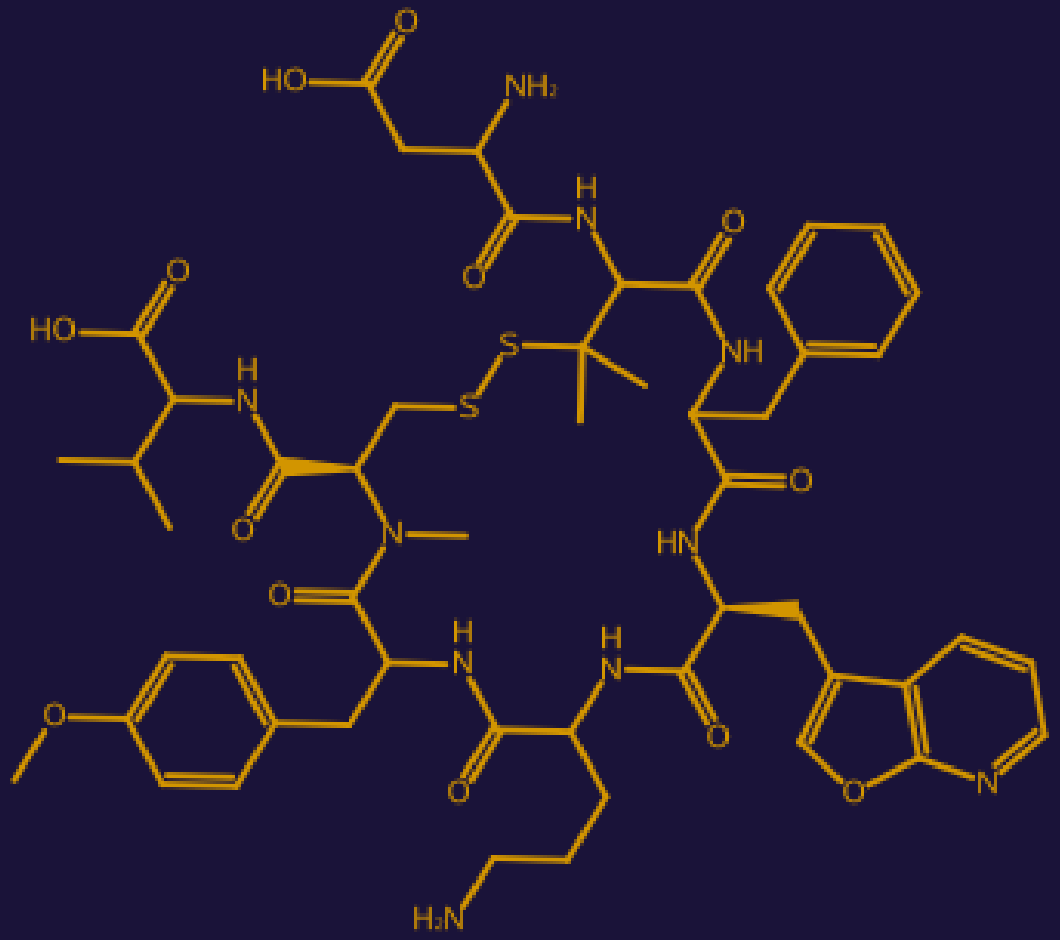
"Algorithm as a prior" allows generalization from few samples

Knowledge graphs



Yih et al. (2015) Semantic Parsing via Staged Query Graph Generation: Question Answering with Knowledge Base

Molecules



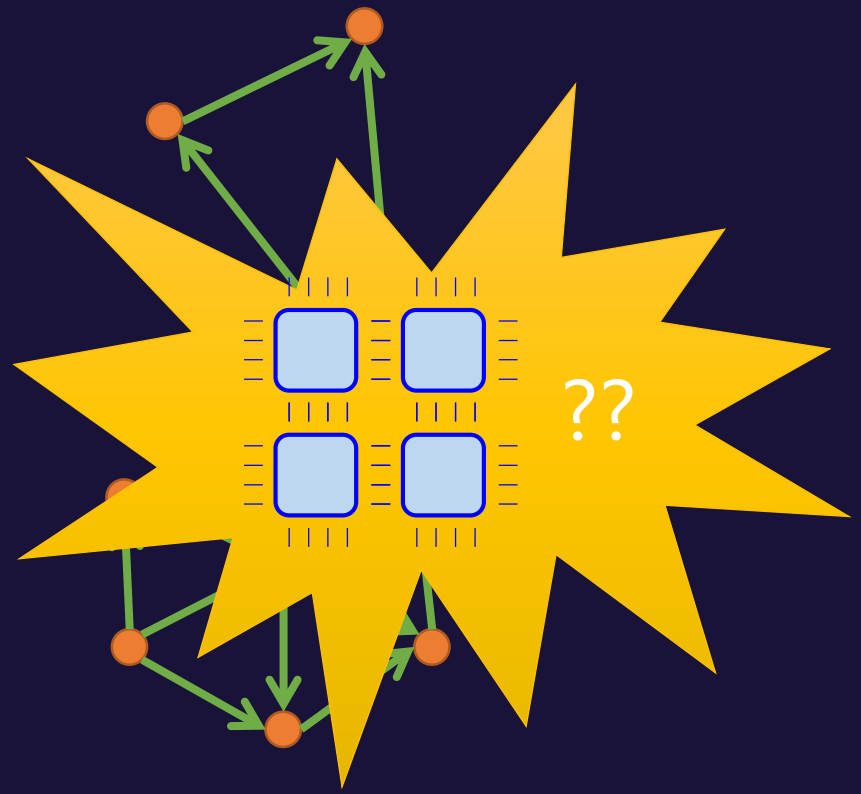
https://www.chemcomp.com/journal/depictor.htm

Need for new hardware/infrastructure

Conventional networks
(vision, speech, etc)

Uniform, regular,
SIMD parallelizable

Flexible networks
(tree, graph, etc)



Non-uniform, irregular
Per-instance computational flow

AMPNet

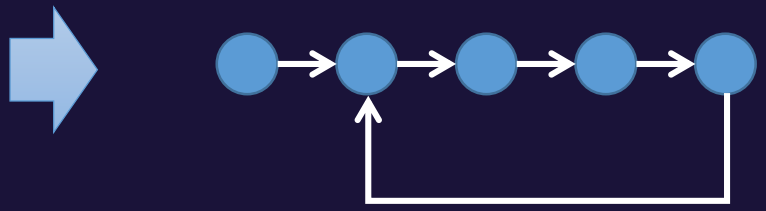
Non-DAG + Streaming + Asynchronous SGD

AMPNet

Non-DAG + Streaming + Asynchronous SGD

$h_0 \leftarrow$ `zeros(1,num_hidden)`
for $j = 1, \ldots, T$ **do**
  $\tilde{x}_j \leftarrow \texttt{embed}(x_j)$
  $h_j \leftarrow \texttt{RNNCell}(h_{j-1}, \tilde{x}_j)$
end for

Loop is explicit

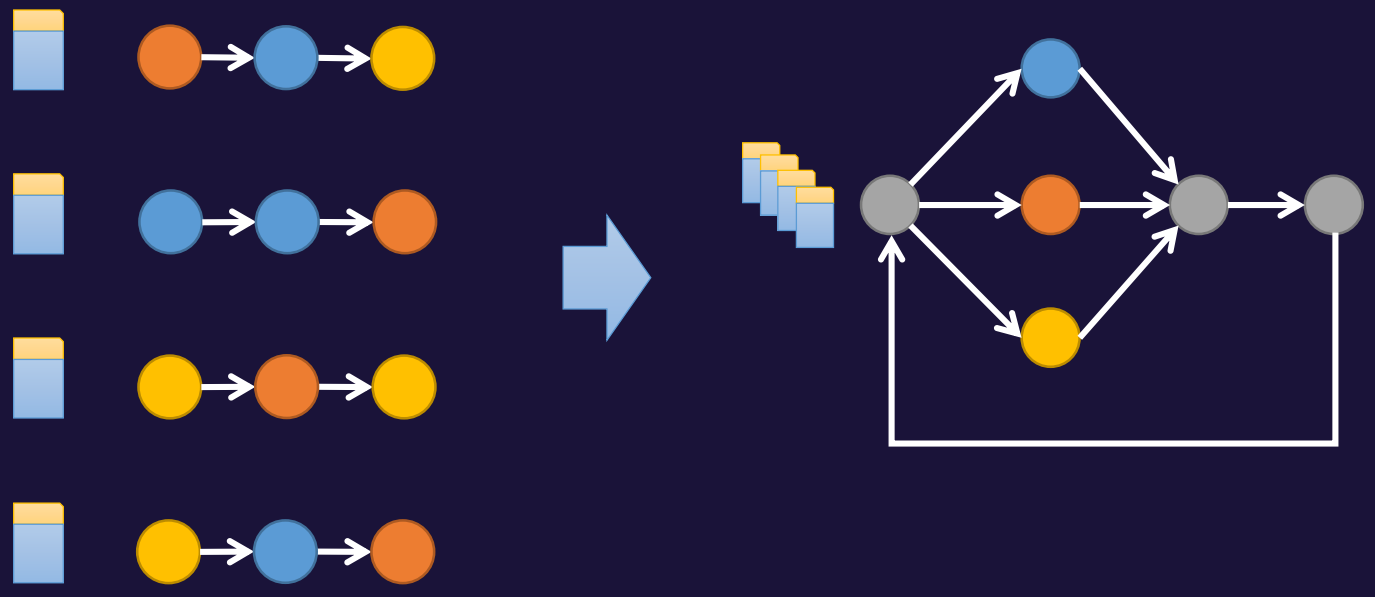


- Dynamic control flows as static computation graphs that allow loops
- No unrolling / padding

(NB: TensorFlow can also express loops using `Enter`, `Exit`, `NextIteration`)

AMPNet

Non-DAG + Streaming + Asynchronous SGD



- Per-instance control flows as branching and merging

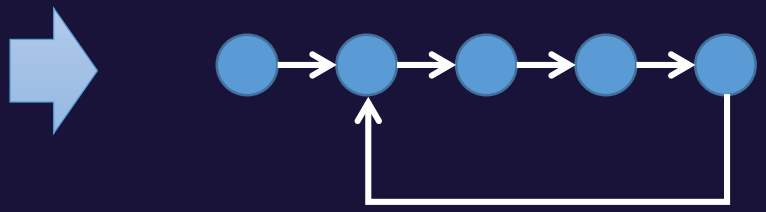
(similar to TF's `Switch/Merge` Ops)

AMPNet

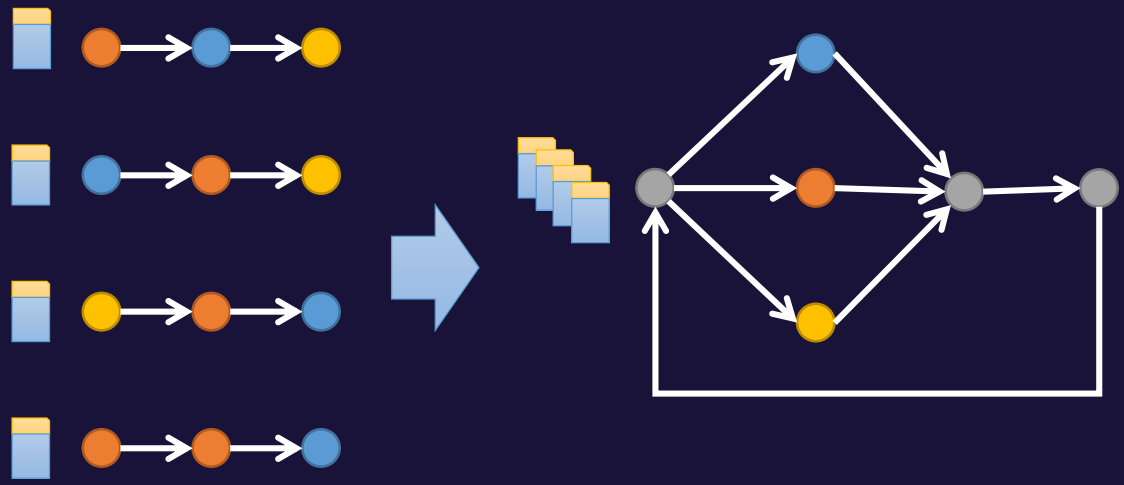
Non-DAG + Streaming + Asynchronous SGD

$h_0 \leftarrow \texttt{zeros(1,num\_hidden)}$
for $j = 1, \ldots, T$ **do**
  $\tilde{x}_j \leftarrow \texttt{embed}(x_j)$
  $h_j \leftarrow \texttt{RNNCell}(h_{j-1}, \tilde{x}_j)$
end for

Loop is explicit

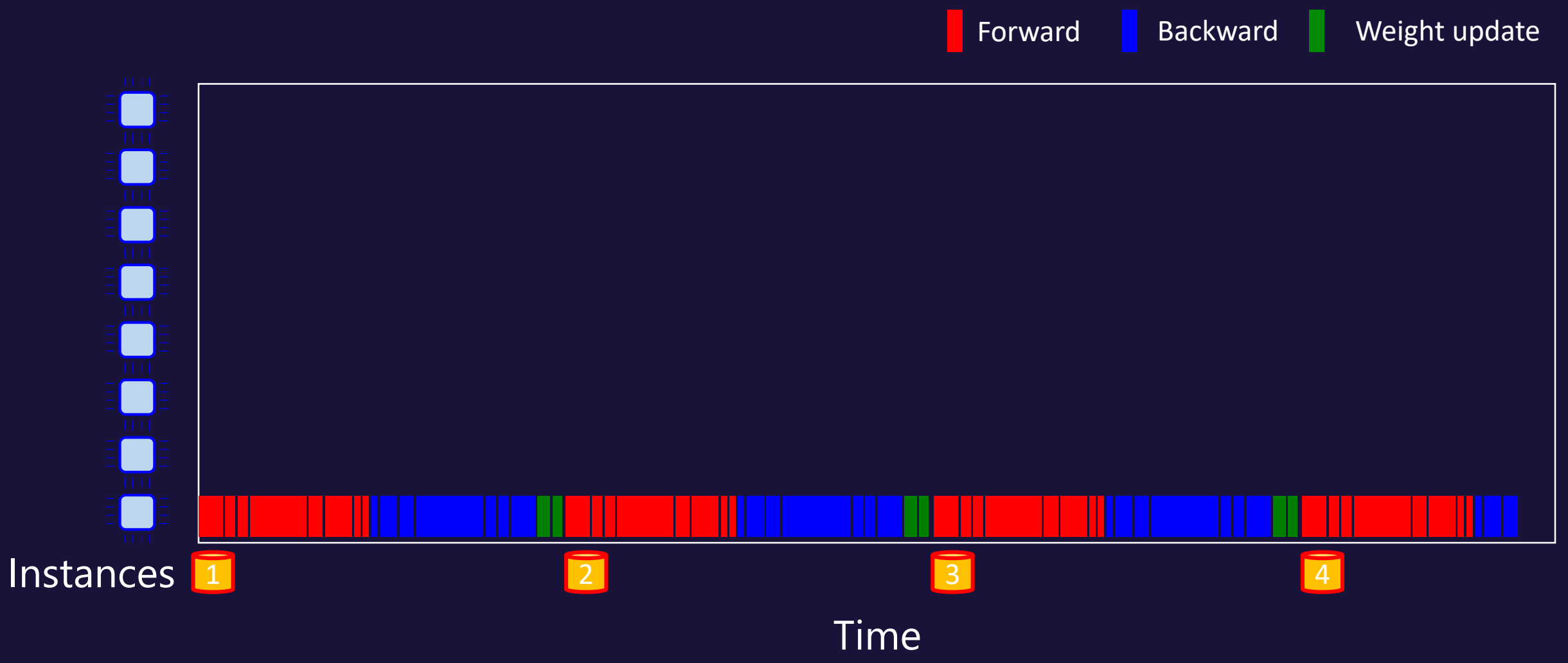


(c.f. `Enter/Exit/NextIteration` in TF)

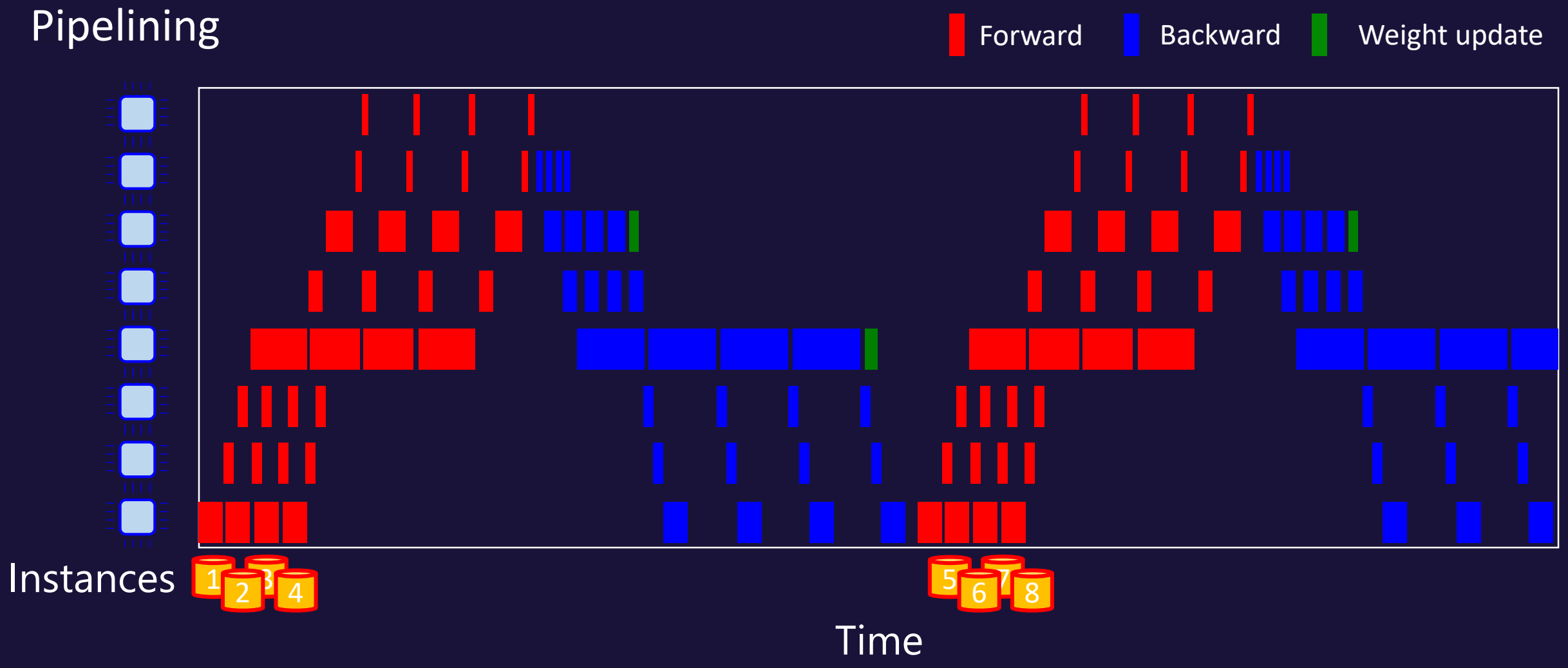


(c.f. `Switch/Merge` in TF)

Asynchronous model-parallel SGD

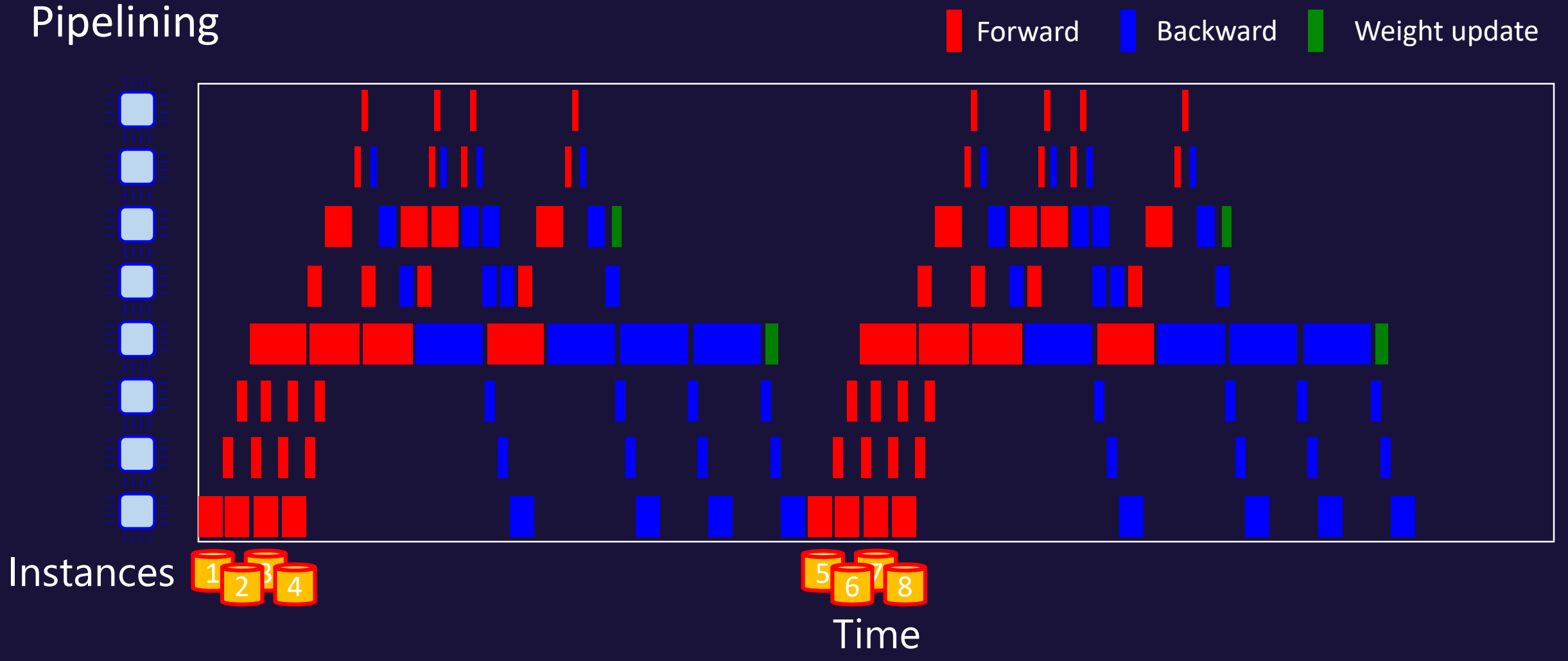


Asynchronous model-parallel SGD



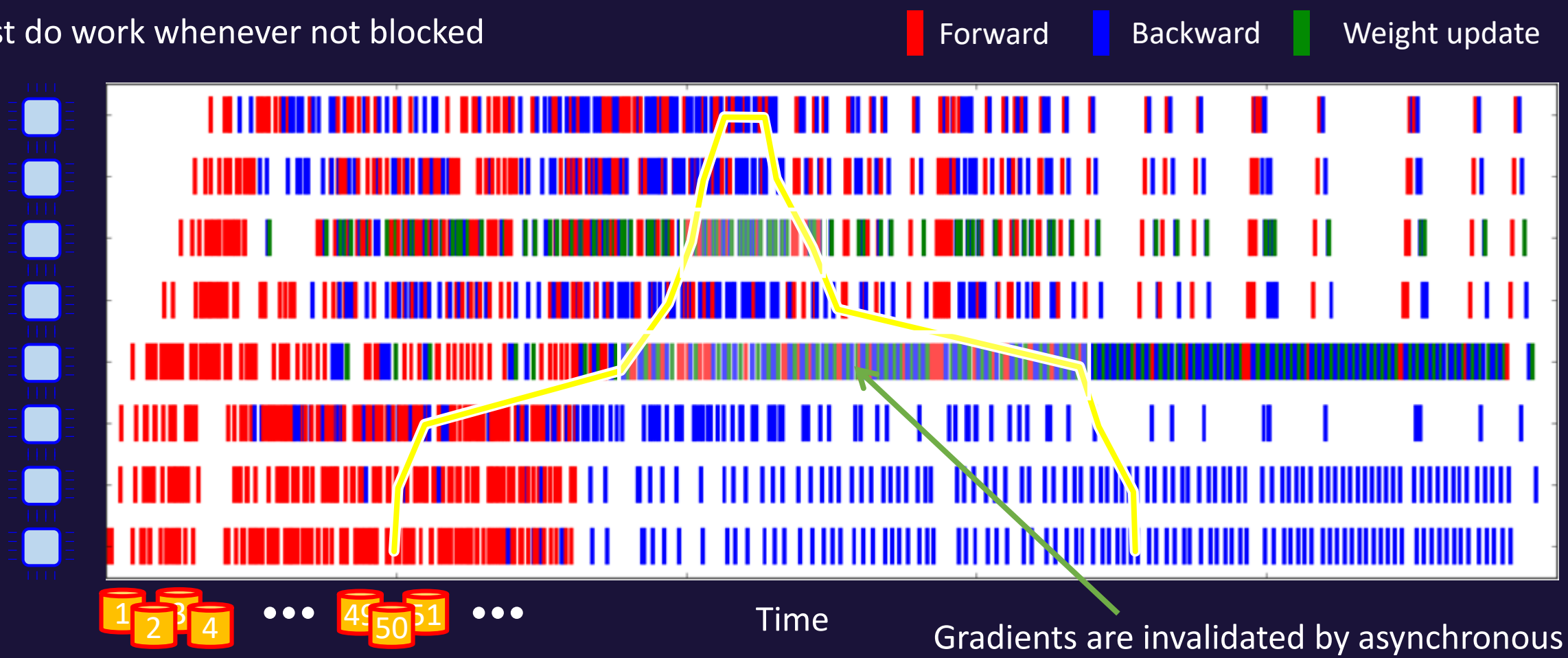
Asynchronous model-parallel SGD

Pipelining



Asynchronous model-parallel SGD

Just do work whenever not blocked



Outline

- We have built an efficient single-machine multi-threaded prototype of Asynchronous Model Parallel SGD
- Some examples
  - Deep feedforward classifier
  - Variable-length sequence classifier
  - Tree recursive neural network
  - Graph neural network

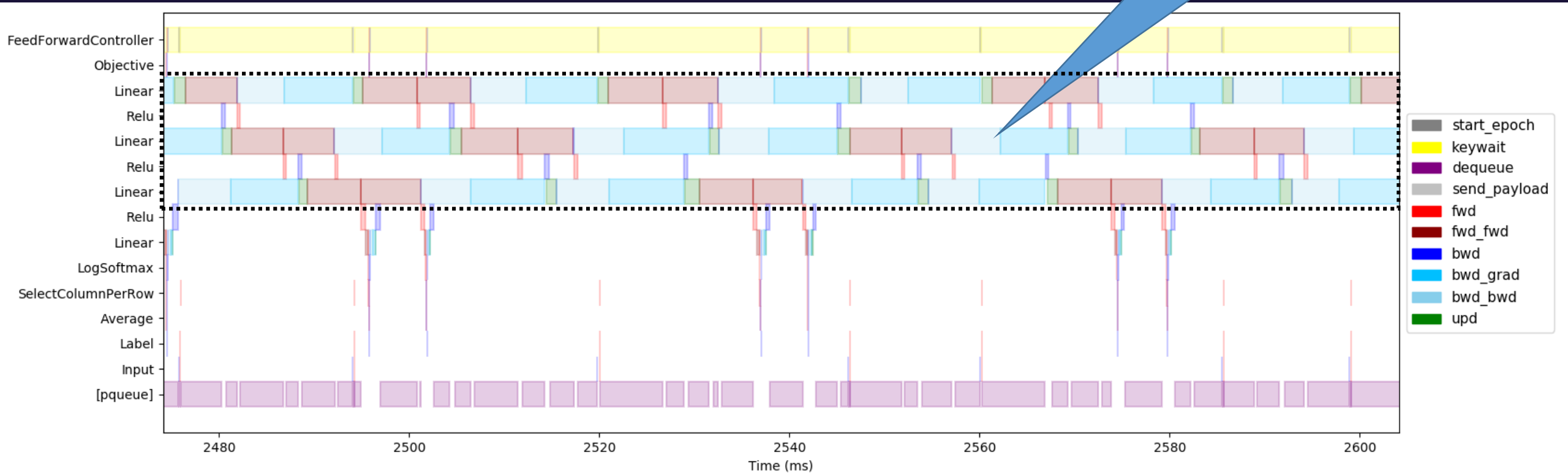
Examples

4-layer fully connected network


```
auto graph = data >>
        Linear("L1", specific_key, input_dim, hidden_dim).affinity(1) >>
        Relu(specific_key) >>
        Linear("L2", specific_key, hidden_dim, hidden_dim).affinity(2) >>
        Relu(specific_key) >>
        Linear("L3", specific_key, hidden_dim, hidden_dim).affinity(3) >>
        Relu(specific_key) >>
        Linear("L4", specific_key, hidden_dim, output_dim).affinity(0) >>
        LogSoftmax(specific_key) >>
        SelectColumnPerRow(specific_key, general_key)(labels) >>
        Average(specific_key) >>
        Objective();
```

Asynchronous model parallelism

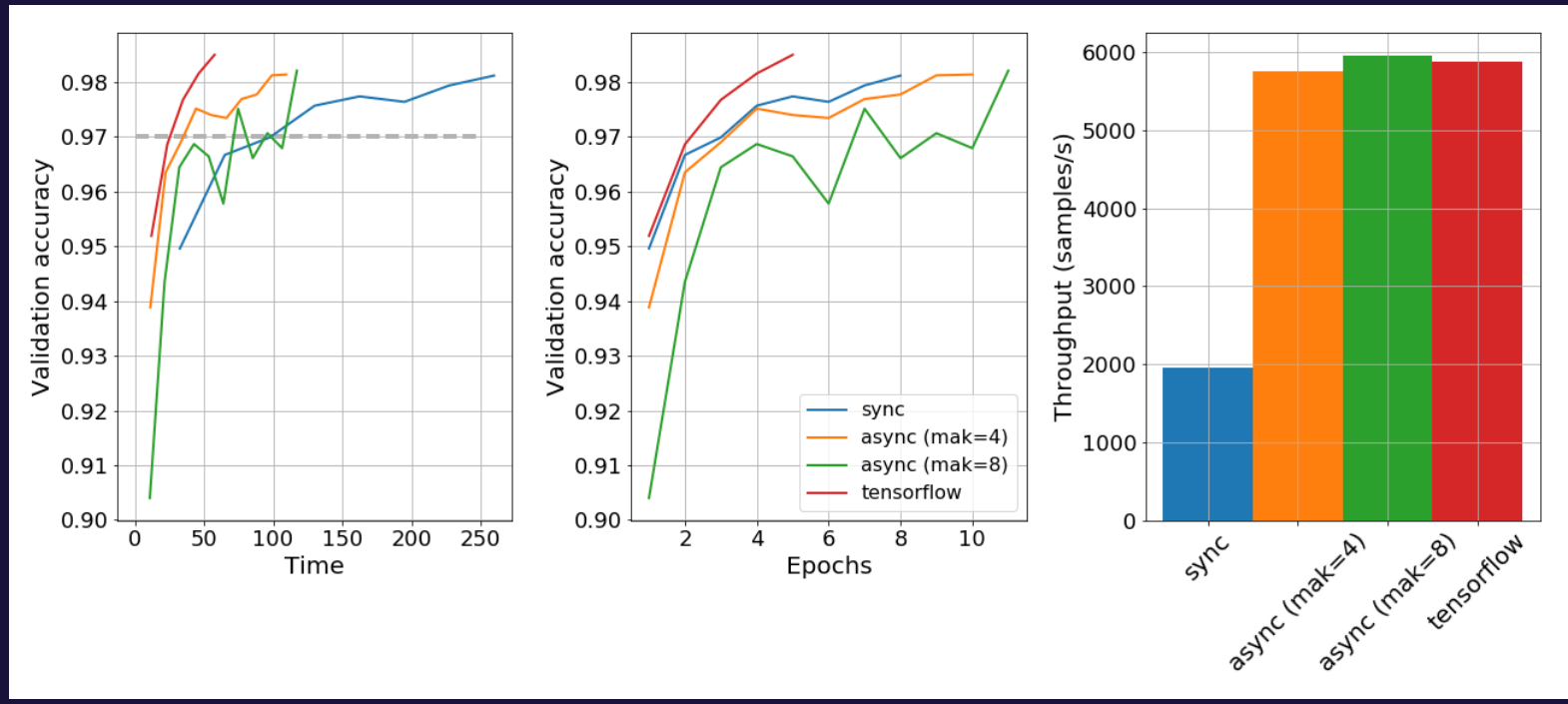
All cores busy!



4 layer MLP with 784 hidden units on MNIST

4-layer FC network on MNIST

- Going from sync to async (mak=4) boosts throughput from 2000 inst/s to 6000 inst/s
- Mild asynchrony does not affect convergence



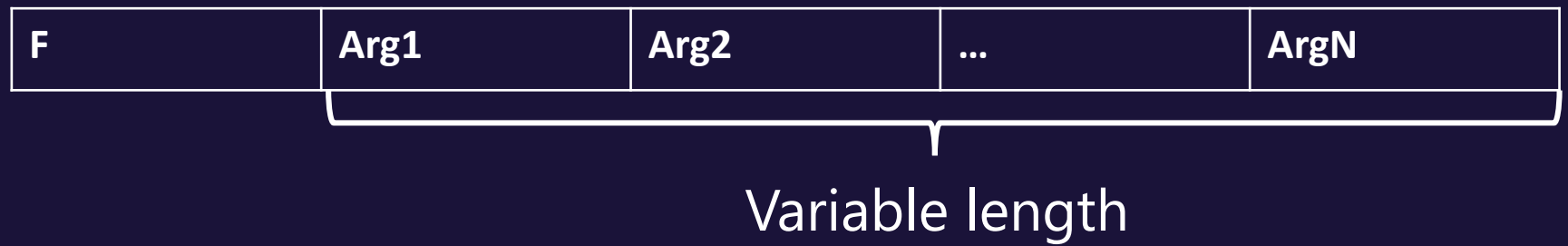
Sync (no asynchrony) Async1 (mild asynchrony) Async2 (more asynchrony)

mak=max_active_keys: limits the number of instances that are in-flight

(TensorFlow can use all 16 cores)

List reduction dataset

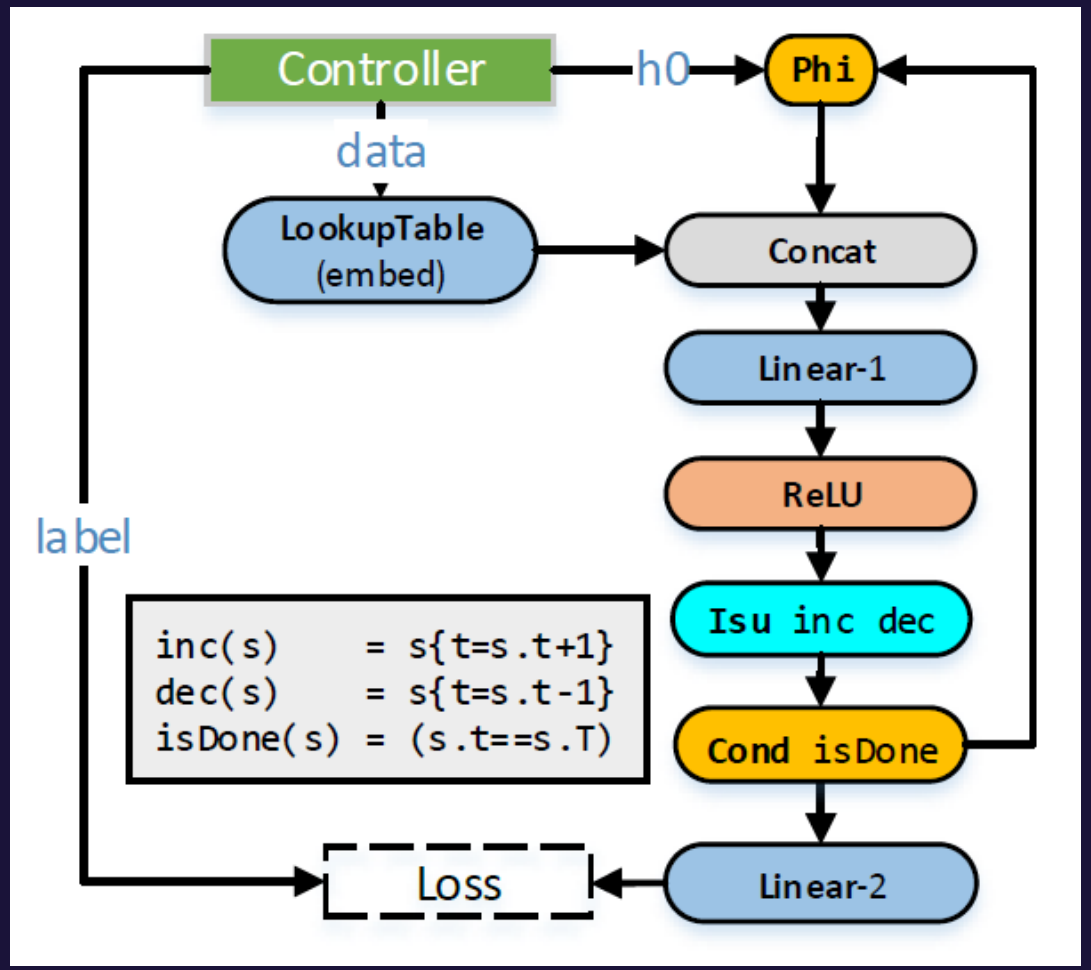
- Input:



- Output: F(Arg1,…,ArgN) (mod M) -> M-class classification
- F is one of the four reduction functions

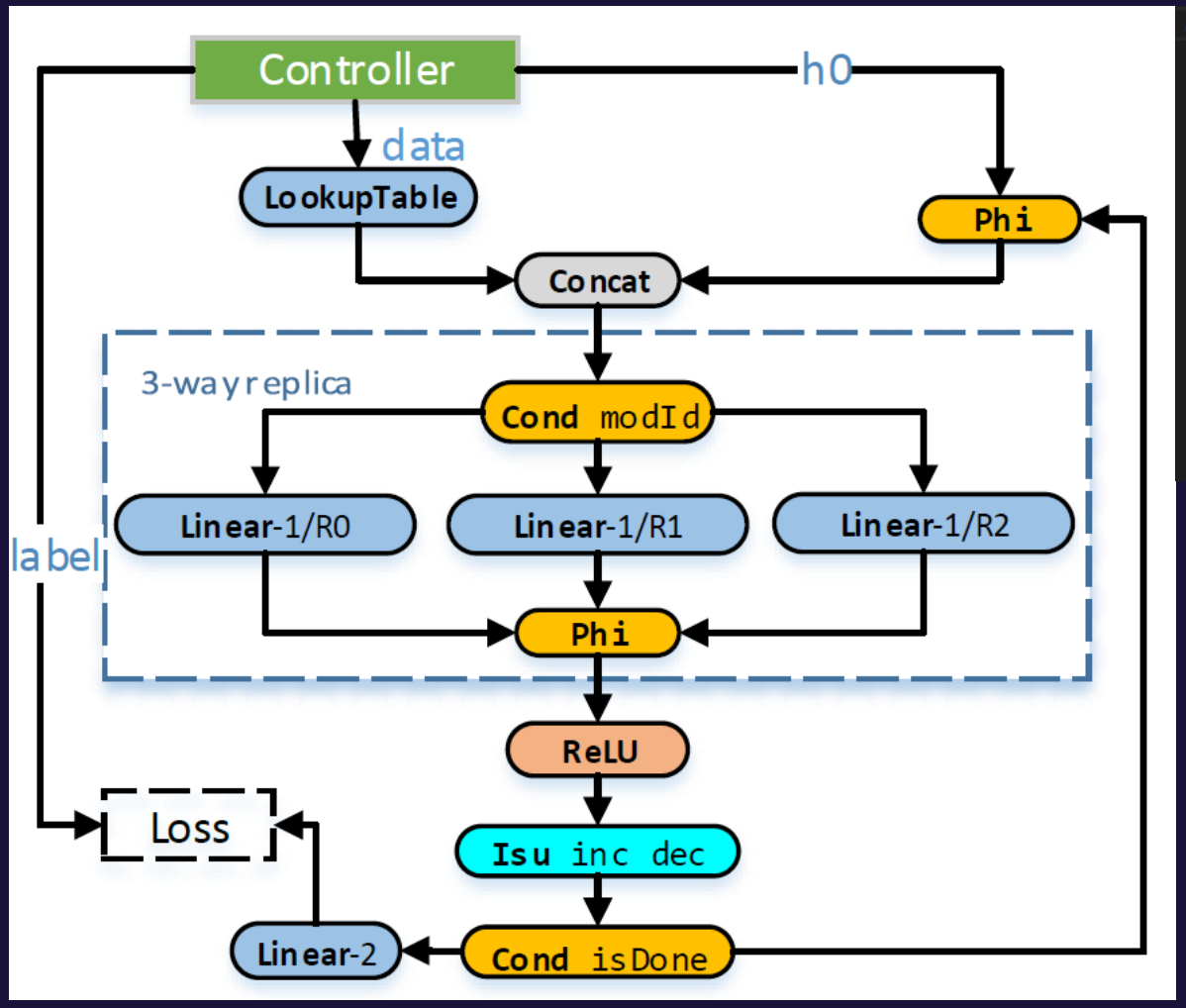
- Train: 100k instances, Validation: 10k instances

Variable-length RNN classifier




```
auto graph = (phi(hidden) >>
                Concat(fwd_key, state_key)(
                    data >> LookupTable(lt_name, fwd_key, vocab_size, hidden_dim)
                    ) >>
                Linear("L1", fwd_key, hidden_dim + hidden_dim, hidden_dim) >>
                Relu(fwd_key) >>
                InvertibleStateUpdate(update_fwd, update_bwd) >>
                Branch(select_function)(phi, (Linear("L2", fwd_key, hidden_dim, output_dim) >>
                                              LogSoftmax(fwd_key) >>
                                              SelectColumnPerRow(fwd_key, state_key)(label) >>
                                              Average(fwd_key) >>
                                              Objective())));
```

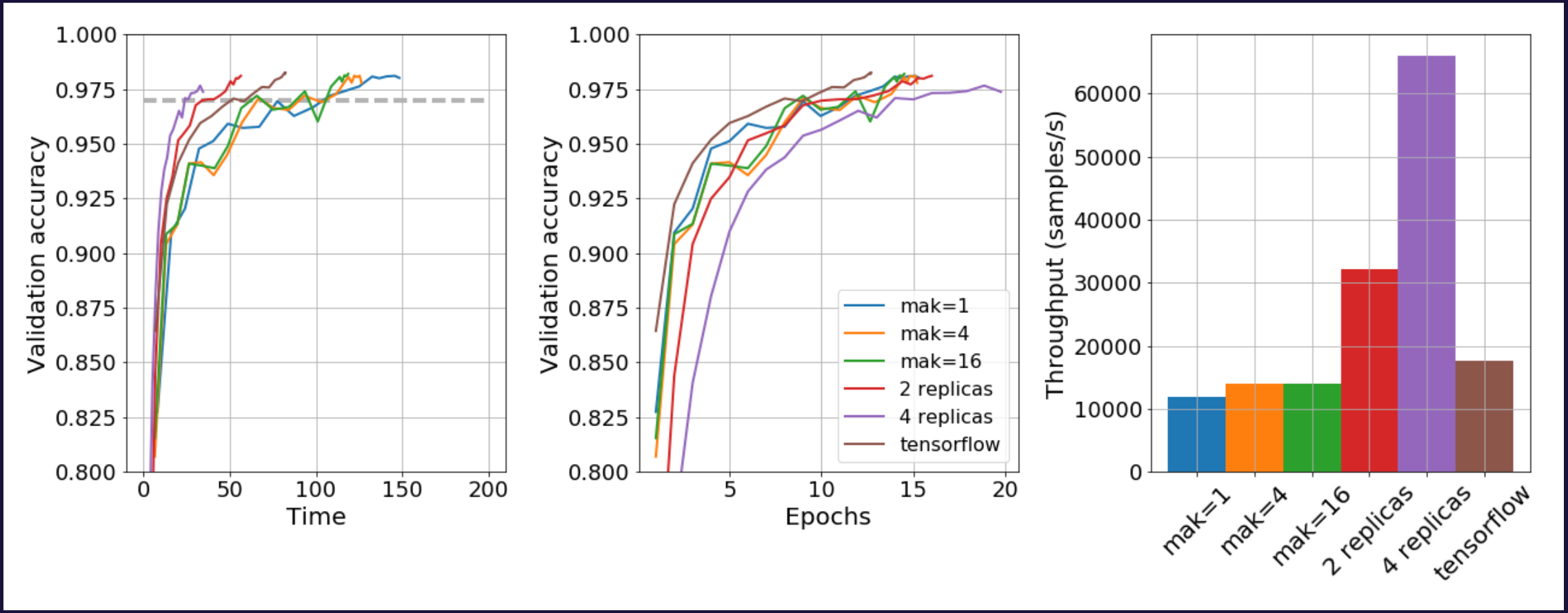
Replicas




```
auto graph = phi(hidden) >>
        Concat(fwd_key, state_key)(
            data >> LookupTable(lt_name, fwd_key, vocab_size, hidden_dim)
            ) >>
        Replica("L1", fwd_key, state_key, num_replicas, hidden_dim + hidden_dim, hidden_dim) >>
        Relu(fwd_key) >>
        InvertibleStateUpdate(update_fwd, update_bwd) >>
        Branch(select_function)(phi, (
            Linear("L2", fwd_key, hidden_dim, output_dim) >>
            LogSoftmax(fwd_key) >>
            SelectColumnPerRow(fwd_key, state_key)(label) >>
            Average(fwd_key) >>
            Objective()
            ));
```

- Simple 1-line change in our IR
- Replicas are initialized identically and synchronized at a controllable frequency.
- In our experiments, sync per epoch was enough

List reduction dataset: Results



Sync (no asynchrony)

Async (mak=4) (mild asynchrony)

Async2 (mak=16) (more asynchrony)

2 replicas (mak=4)

4 replicas (mak=8)

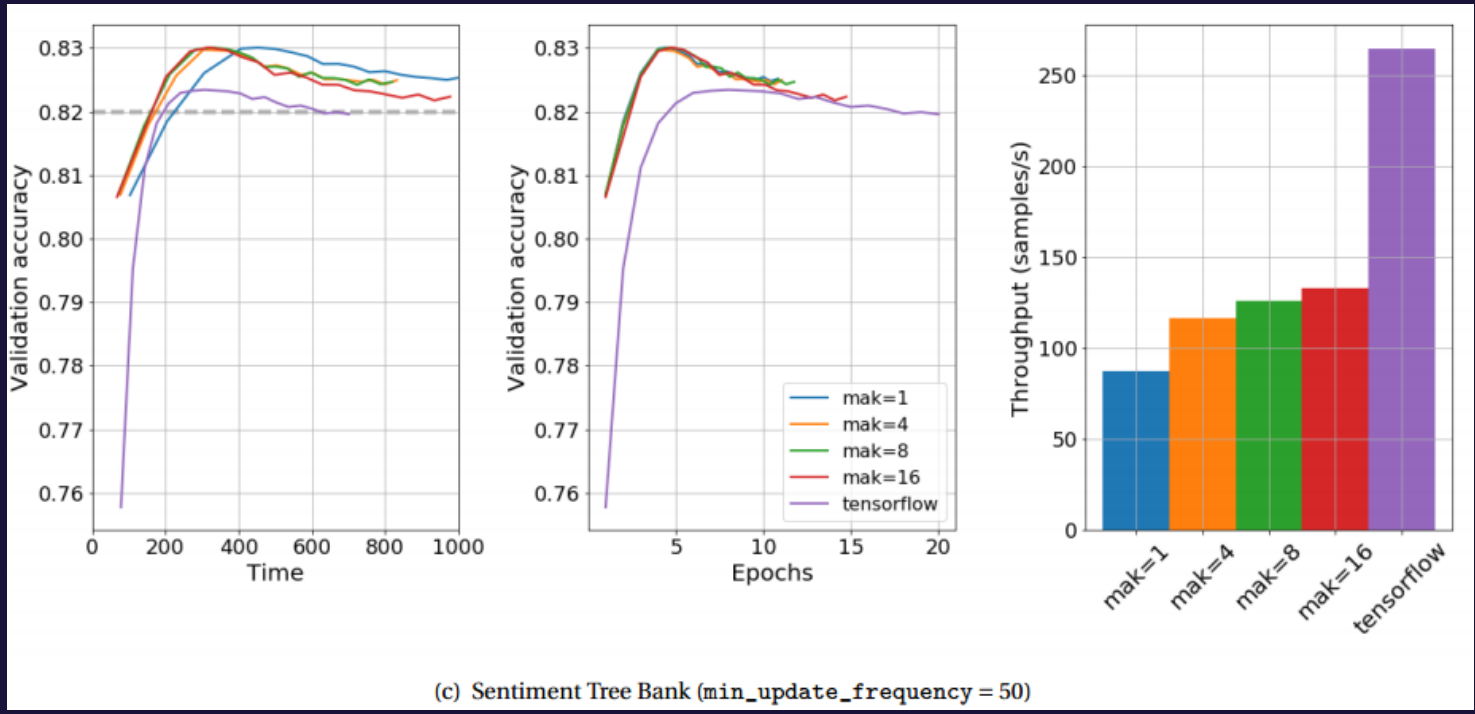
No replication

(TensorFlow can use all 16 cores)

Tree RNN on Stanford Sentiment Tree Bank dataset

[Socher et al. 2013; Tai et al. 2015]

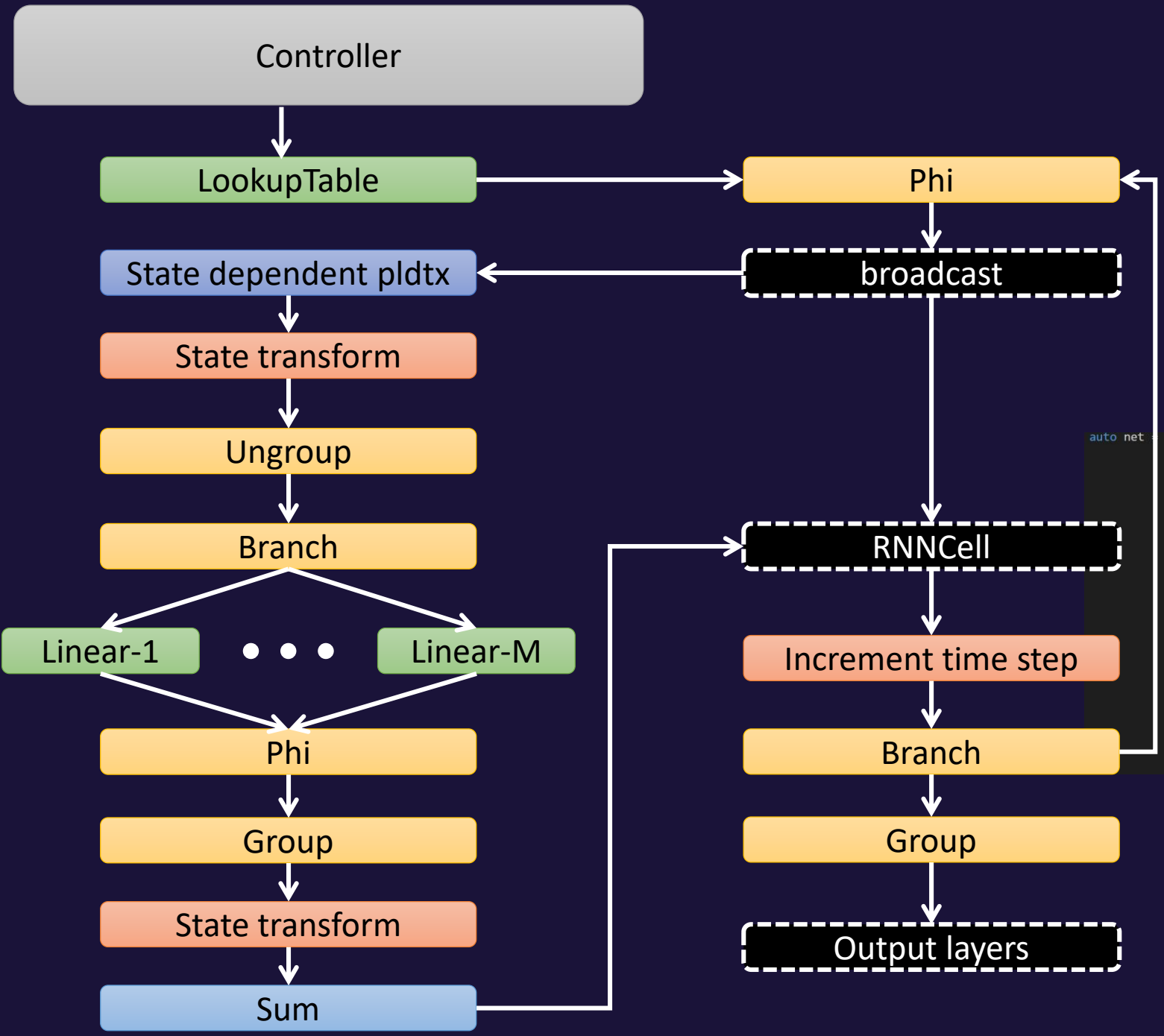
- TensorFlow Fold [Looks +2017] achieves higher throughput but we converge faster because we don't batch
- Asynchrony does not slow-down convergence



(c) Sentiment Tree Bank (`min_update_frequency` = 50)

Gated Graph Sequence NN

[Li et al. 2016]

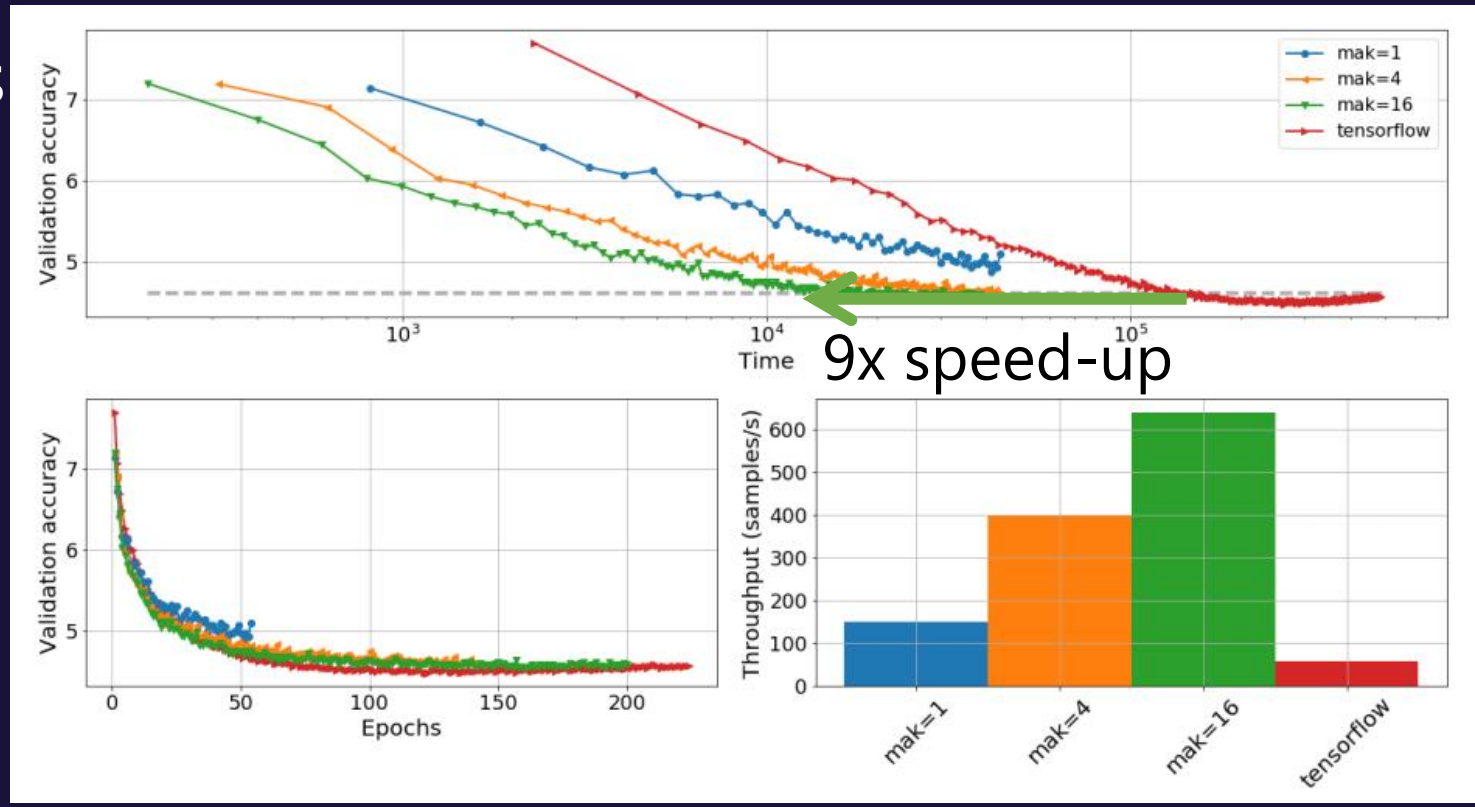



```
auto net = (enter >>
      Ungroup(child_self_step_start_edge_end, self_step_type, generate_edge_states_for_type_and_step) >>
      Group(parent_self_step_start_edge_end, self_step_end, generate_edge_keys_for_end_and_step) >>
      Sum(parent_self_step_end, 0) >>
      Group(parent_self_step_end, self_step, generate_end_keys_for_step) >>
      gru(loop) >>
      InvertibleStateUpdate(increment_step, decrement_step) >>
      Phi(self_step)(init_state) >>
      Branch(select_index_using_step)(
          (loop >>
           Ungroup(child_self_step_start, self_step, generate_start_states) >>
           ForEach(child_self_step_start_edge_end, self_step_start, generate_edge_states_for_start) >>
           Group(parent_self_step_start_edge_end, self_step_type, generate_edge_keys_for_type_and_step) >>
           edge_branch),

          (Linear(ho_name, parent_self_step, hidden_dim, 1) >>
           Transpose() >>
           LogSoftmax(parent_self_step) >>
           SelectColumnPerRow(parent_self_step, self_step)(labels) >>
           Objective())));
```

QM9 molecule dataset (graph input regression task)

- We get over 600 graphs/s on a 16 core machine (Azure H16)
- TensorFlow CPU ~ 70 graphs/s
- Speedup comes from sparsity (~2x) and asynchrony (~5x)



TensorFlow GPU ~ 300 graphs/s (TitanX, not shown in the graph)

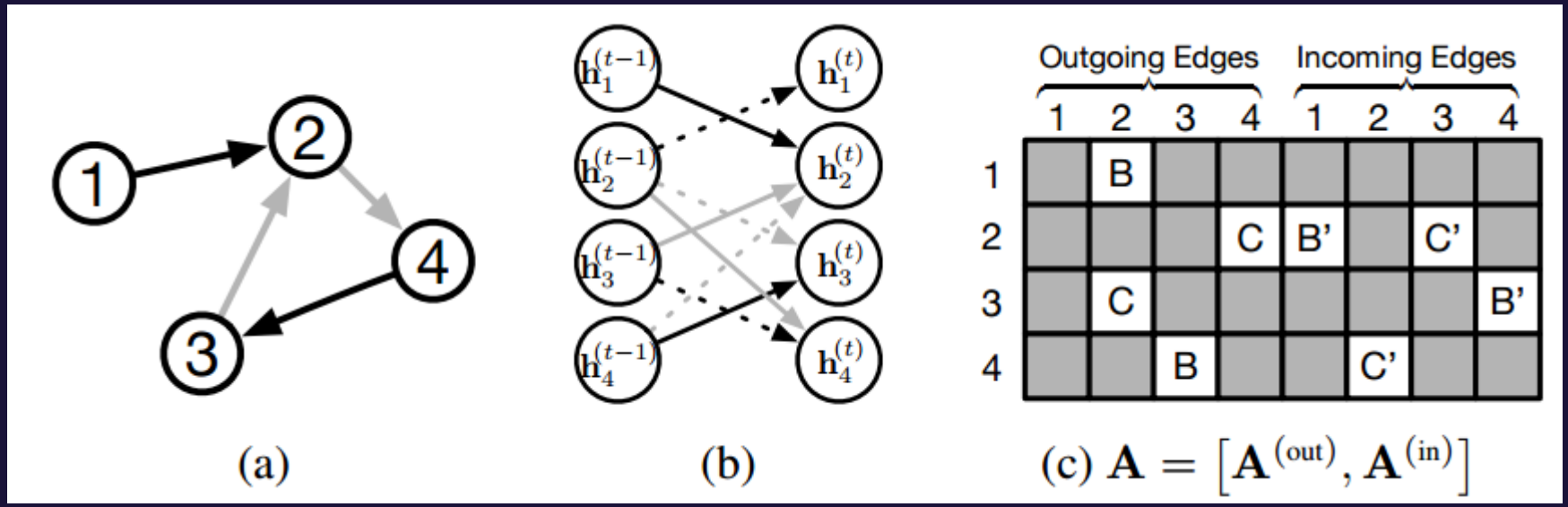
QM9 dataset [Ruddigkeit+ 2012; Ramakrishnan+ 2014; Gilmer+ 2017]

Conclusion

- Our IR is a static representation of dynamic control flows
  - Control-flow ops: branch and merge
  - Data-flow ops: group, ungroup, flatmap, etc
  - Both model parallelism and data parallelism (replicas)
- AMPNet prototype on single-machine multi-core is
  - comparable to prior art when batching and easy and no sparsity
  - Significantly faster when batching is hard and sparsity can be exploited

Appendix

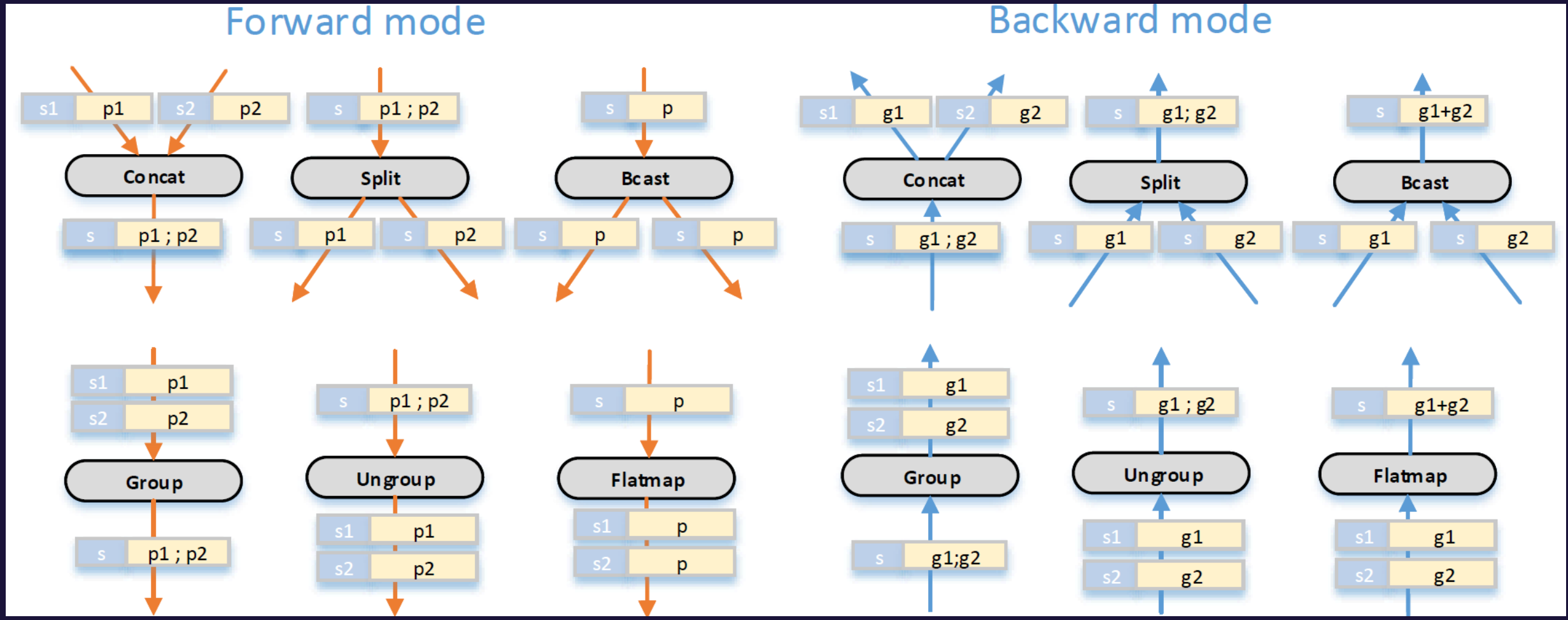
How it is done in TensorFlow



NH x 2NH dense matrix
(most entries are zero)

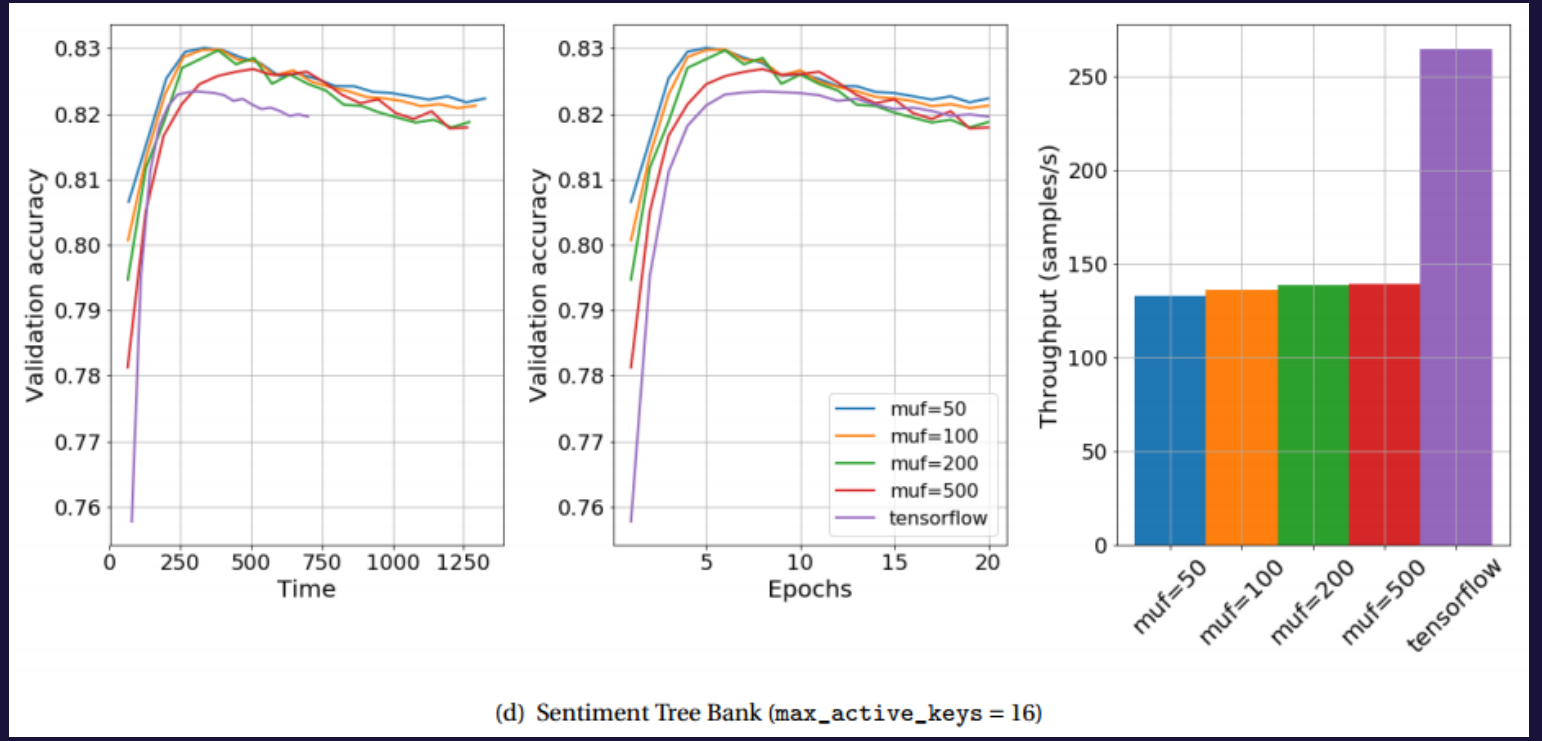
Li et al. (2016) “Gated Graph Sequence Neural Networks”

Our IR nodes


Forward mode
Backward mode
s1
p1
s2
p2
Concat
s
p1 ; p2
Split
p
Bcast
g1
g2
g1; g2
g1+g2
g1 ; g2
Group
Ungroup
Flatmap
g1;g2

Tree RNN

- Updating more frequently speeds up convergence



(d) Sentiment Tree Bank (`max_active_keys` = 16)

muf = min_update_frequency (number of gradients to accumulate before an update)