Accelerating Feedforward Computation via Parallel Nonlinear Equation Solving

Yang Song[1], Chenlin Meng[1], Renjie Liao[2], Stefano Ermon[1]
[1]Stanford University [2]University of Toronto & Vector Institute

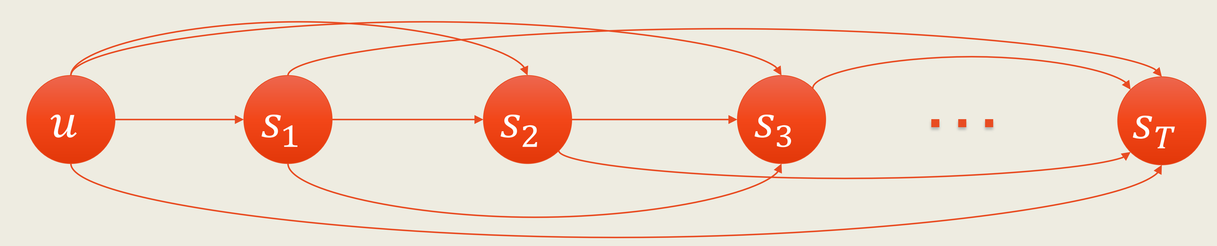
ABSTRACT

- Feedforward computation has many examples in machine learning, such as backpropagation for training RNNs, inference for neural networks, and sampling from autoregressive models. However, feedforward computation cannot be naively accelerated in parallel due to its sequential nature.
- We notice that feedforward computation is synonymous with the Gauss-Seidel method for solving a system of nonlinear equations, and propose to use parallelizable equation solvers, like the Jacobi method (and their hybrids), to accelerate feedforward computation.
- In our experiments, we demonstrate a speedup factor ranging from 2.1 to 26 for various instances of feedforward computation in machine learning.

Code:

Feedforward Computation

Illustration:



Definition:

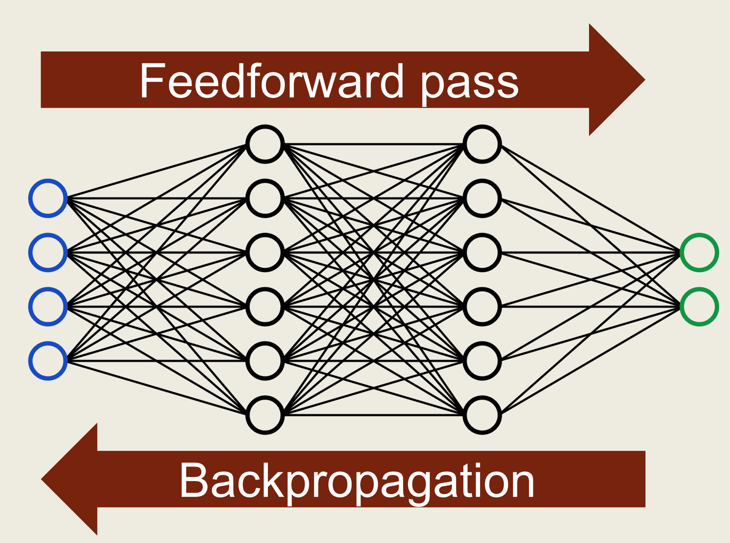
state

$$
\begin{aligned}
\mathbf{s}_1 &= h_1(\mathbf{u}) \\
\mathbf{s}_2 &= h_2(\mathbf{u}, \mathbf{s}_1) \\
\mathbf{s}_3 &= h_3(\mathbf{u}, \mathbf{s}_1, \mathbf{s}_2) \\
&\vdots \\
\mathbf{s}_T &= h_T(\mathbf{u}, \mathbf{s}_1, \mathbf{s}_2, \cdots, \mathbf{s}_{T-1})
\end{aligned}
$$

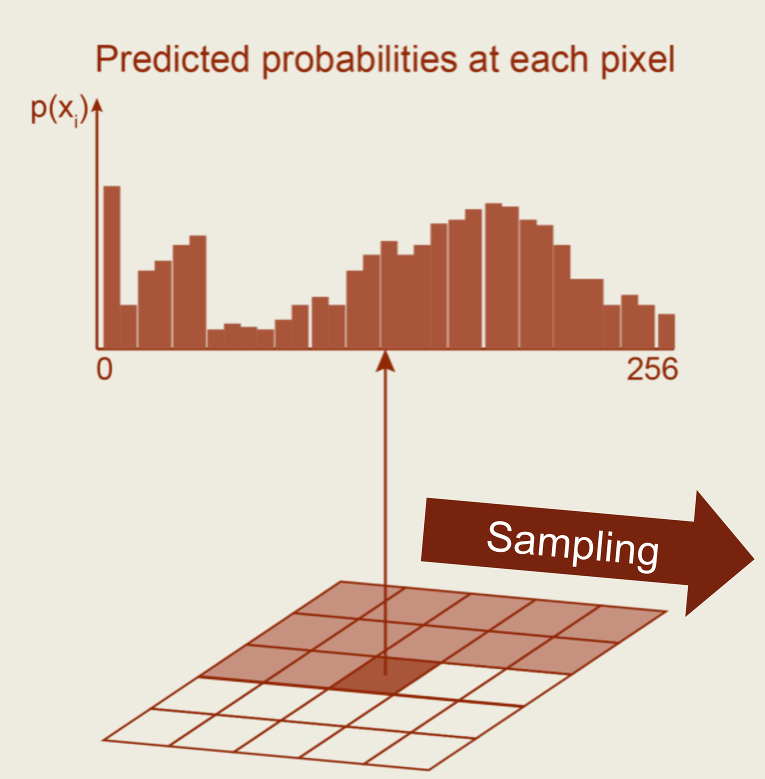
input

Examples:

- Feedforward pass of neural networks
- Backpropagation of neural networks



- Sampling from autoregressive models



The Perspective of Nonlinear Equation Solving

Feedforward computation as equation solving

- Feedforward computation solves the following triangular system of (typically nonlinear) equations.

$$
\begin{cases}
\mathbf{s}_1 - h_1(\mathbf{u}) = 0 \\
\mathbf{s}_2 - h_2(\mathbf{u}, \mathbf{s}_1) = 0 \\
\mathbf{s}_3 - h_3(\mathbf{u}, \mathbf{s}_1, \mathbf{s}_2) = 0 \\
\vdots \\
\mathbf{s}_T - h_T(\mathbf{u}, \mathbf{s}_1, \mathbf{s}_2, \cdots, \mathbf{s}_{T-1}) = 0
\end{cases}
$$

Gauss-Seidel solver:

- Solve each univariate equation **in succession** to get $\mathbf{s}_1, \mathbf{s}_2, \cdots, \mathbf{s}_T$
- Same as feedforward computation when applied to triangular systems of equations.

Jacobi solver:

- Solve all univariate equations **in parallel**, while assuming other equations have already been solved.
- Repeat until convergence.

Algorithm 1 Nonlinear Jacobi Iteration

Input: $\mathbf{u}$; ϵ; T
Initialize $\mathbf{s}_1^0, \mathbf{s}_2^0, \cdots, \mathbf{s}_T^0$ and set $k \leftarrow 0$
repeat
  $k \leftarrow k + 1$
  for $t = 1$ **to** T **do in parallel**
    $\mathbf{s}_t^k \leftarrow h_t(\mathbf{u}, \mathbf{s}_{1:t-1}^{k-1})$ (Notation: $\mathbf{s}_{1:t} := \{\mathbf{s}_1, \mathbf{s}_2, \cdots, \mathbf{s}_t\}$)
  end for
until $k = T$ **or** $\|\mathbf{s}_{1:T}^k - \mathbf{s}_{1:T}^{k-1}\| \leq \epsilon$
return $\mathbf{s}_1^k, \mathbf{s}_2^k, \cdots, \mathbf{s}_T^k$

When to use a Jacobi solver:

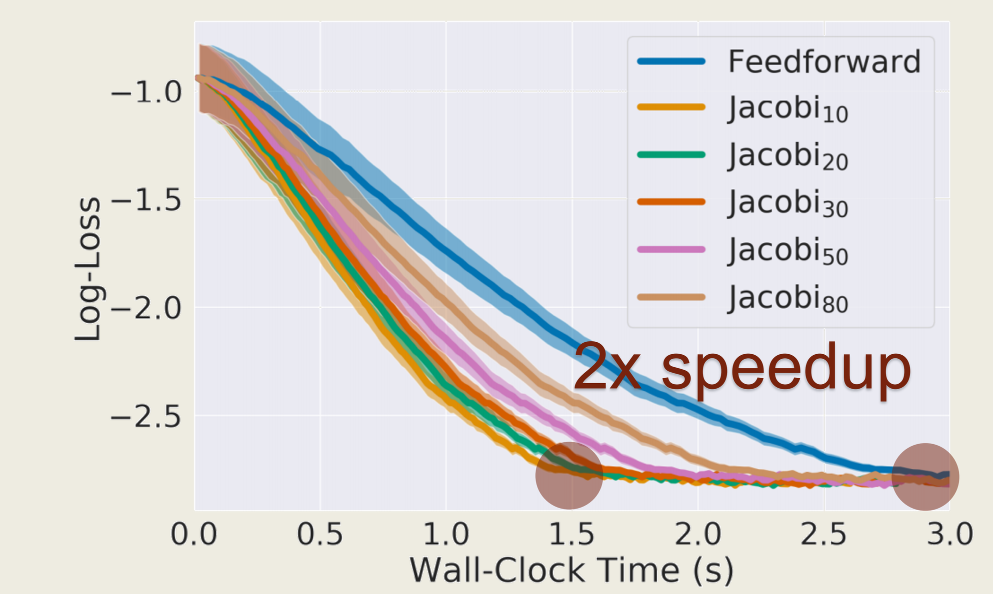
- The computational graph has **many long skip connections** (e.g., DenseNets, backpropagation of RNNs)
- Sufficient parallel computing units at one's disposal
- The cost of solving each univariate is balanced

Hybrid solvers:

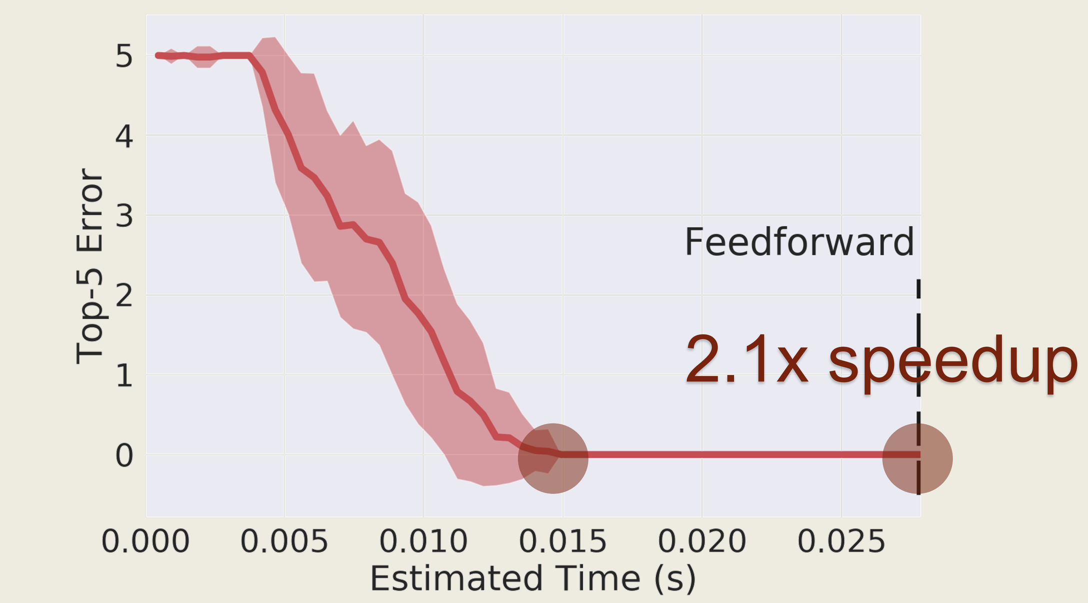
- Gauss-Seidel-Jacobi (GS-Jacobi)
- Jacobi-Gauss-Seidel (Jacobi-GS)

Experiments

Backpropagation of RNNs



Inference of DenseNets



Sampling from autoregressive models (PixelCNN++)

Method	MNIST		CIFAR-10	
	Time (s)	Speedup	Time (s)	Speedup
Feedforward w/o cache	12.15	1.00×	30.95	1.00×
Feedforward w/ cache	8.23	1.48×	17.76	1.74×
Jacobi	1.94	6.26×	26.16	1.18×
GS-Jacobi	**1.86**	**6.53**×	14.84	2.09×
Jacobi-GS	5.95	2.04×	**14.76**	**2.10**×