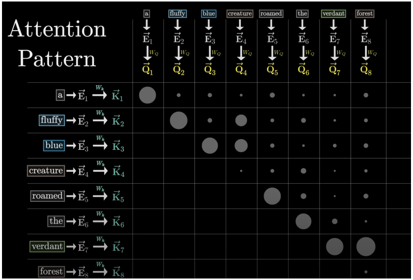
Towards Scalable and Stable Parallelization of Nonlinear RNNs

Xavier Gonzalez, Andrew Warrington, Jimmy T.H. Smith, Scott W. Linderman

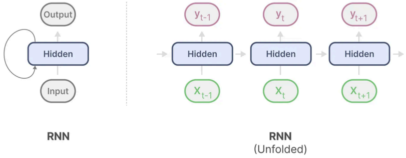
NeurIPS 2024

Motivation: Transformers vs RNNs

Transformers



RNNs



- **Parallelizable** Training (great for **GPUs**!) ✓
- Generation is expensive (KV cache grows with sequence length) ✗

- Sequential Training (hard to get GPU speed up over the sequence length) ✗
- Stateful generation ✓

Sequential vs Parallel (Iterative) Evaluation

Sequential Evaluation

Sequential vs Parallel (Iterative) Evaluation

Sequential Evaluation

Sequential vs Parallel (Iterative) Evaluation

Sequential Evaluation

Sequential vs Parallel (Iterative) Evaluation

Sequential Evaluation

Sequential vs Parallel (Iterative) Evaluation

Sequential Evaluation

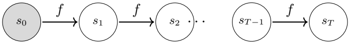
Sequential vs Parallel (Iterative) Evaluation

Sequential Evaluation

Sequential vs Parallel (Iterative) Evaluation

Sequential Evaluation

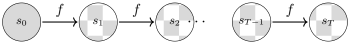
Parallel (Iterative) Evaluation



Sequential vs Parallel (Iterative) Evaluation

Sequential Evaluation

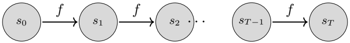
Parallel (Iterative) Evaluation



Sequential vs Parallel (Iterative) Evaluation

Sequential Evaluation

Parallel (Iterative) Evaluation



Sequential vs Parallel (Iterative) Evaluation

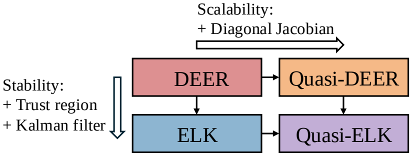
Sequential Evaluation

Parallel (Iterative) Evaluation

DEER

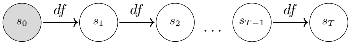
Y.H. Lim, Q. Zhu, J. Selfridge, and M.F. Kasim. Parallelizing non-linear sequential models over the sequence length. *ICLR,* 2024.

Scalable and Stable Parallelization of RNNs



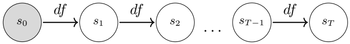
Scalable and Stable Parallelization of RNNs

Scalable and Stable Parallelization of RNNs



DEER

Scalable and Stable Parallelization of RNNs



Use *parallel associative scan*

DEER

Scalable and Stable Parallelization of RNNs

$$\Delta \mathbf{s}_t^{(i+1)} = \left[\frac{\partial f_t}{\partial \mathbf{s}} (\mathbf{s}_{t-1}^{(i)}) \right] \Delta \mathbf{s}_{t-1}^{(i+1)} - \mathbf{r}_t(\mathbf{s}^{(i)})$$

$$-\mathbf{r}_t(\mathbf{s}^{(i)}) = f(\mathbf{s}_{t-1}^{(i)}) - \mathbf{s}_t^{(i)}$$

DEER

Scalable and Stable Parallelization of RNNs

$$\Delta \mathbf{s}_t^{(i+1)} = \underbrace{\left[\frac{\partial f_t}{\partial \mathbf{s}}(\mathbf{s}_{t-1}^{(i)}) \right]}_{\text{D} \times \text{D}} \Delta \mathbf{s}_{t-1}^{(i+1)} - \mathbf{r}_t(\mathbf{s}^{(i)})$$

- Each matmul is $\mathcal{O}(D^3)$
- Memory is $\mathcal{O}(TD^2)$

DEER

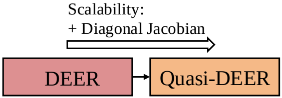
Scalable and Stable Parallelization of RNNs

$$\Delta \mathbf{s}_t^{(i+1)} = \underbrace{\left[\frac{\partial f_t}{\partial \mathbf{s}} (\mathbf{s}_{t-1}^{(i)}) \right]}_{\text{D}\times\text{D}} \Delta \mathbf{s}_{t-1}^{(i+1)} - \mathbf{r}_t(\mathbf{s}^{(i)})$$

- Each matmul is $\mathcal{O}(D^3)$
- Memory is $\mathcal{O}(TD^2)$

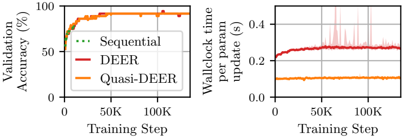
$$\text{Diag} \left[\frac{\partial f_t}{\partial \mathbf{s}} (\mathbf{s}_{t-1}^{(i)}) \right]$$

- Each matmul is $\mathcal{O}(D)$
- Memory is $\mathcal{O}(TD)$

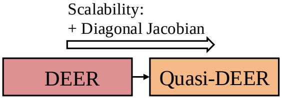


Scalable and Stable Parallelization of RNNs

$$\Delta \mathbf{s}_t^{(i+1)} = \underbrace{\left[\frac{\partial f_t}{\partial \mathbf{s}}(\mathbf{s}_{t-1}^{(i)})\right]}_{\mathrm{D}\times \mathrm{D}} \Delta \mathbf{s}_{t-1}^{(i+1)} - \mathbf{r}_t(\mathbf{s}^{(i)})$$



- Each matmul is $\mathcal{O}(D^3)$
- Memory is $\mathcal{O}(TD^2)$



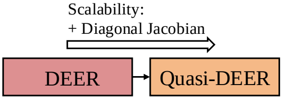
Scalable and Stable Parallelization of RNNs

$$\Delta \mathbf{s}_t^{(i+1)} = \underbrace{\left[\frac{\partial f_t}{\partial \mathbf{s}}(\mathbf{s}_{t-1}^{(i)})\right]}_{\mathrm{D}\times\mathrm{D}} \Delta \mathbf{s}_{t-1}^{(i+1)} - \mathbf{r}_t(\mathbf{s}^{(i)})$$

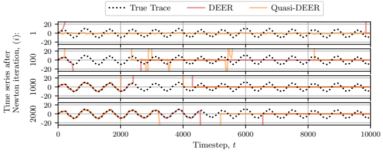
- Each matmul is $\mathcal{O}(D^3)$
- Memory is $\mathcal{O}(TD^2)$

$$\mathrm{Diag}\left[\frac{\partial f_t}{\partial \mathbf{s}}(\mathbf{s}_{t-1}^{(i)})\right]$$

- Each matmul is $\mathcal{O}(D)$
- Memory is $\mathcal{O}(TD)$



DEER instability



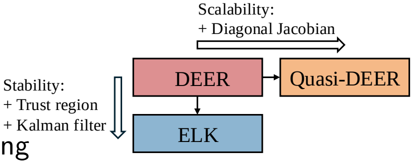
Scalable and Stable Parallelization of RNNs

$$\Delta \mathbf{s}_t^{(i+1)} = \underbrace{\left[\frac{\partial f_t}{\partial \mathbf{s}}(\mathbf{s}_{t-1}^{(i)}) \right]}_{\text{D}\times\text{D}} \Delta \mathbf{s}_{t-1}^{(i+1)} - \mathbf{r}_t(\mathbf{s}^{(i)})$$

$$\text{Diag}\left[\frac{\partial f_t}{\partial \mathbf{s}}(\mathbf{s}_{t-1}^{(i)}) \right]$$

- Each matmul is $\mathcal{O}(D^3)$
- Memory is $\mathcal{O}(TD^2)$

- Each matmul is $\mathcal{O}(D)$
- Memory is $\mathcal{O}(TD)$



- ELK: **E**valuating **L**evenberg-Marquardt with **K**alman
- Stability: Trust region restricts the size of $\Delta \mathbf{s}$
- Can be evaluated in parallel with Kalman filter

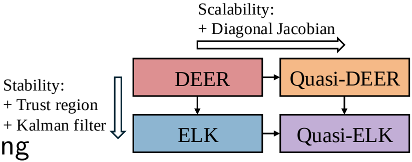
Scalable and Stable Parallelization of RNNs

$$\Delta \mathbf{s}_t^{(i+1)} = \underbrace{\left[\frac{\partial f_t}{\partial \mathbf{s}}(\mathbf{s}_{t-1}^{(i)})\right]}_{\text{D}\times\text{D}} \Delta \mathbf{s}_{t-1}^{(i+1)} - \mathbf{r}_t(\mathbf{s}^{(i)})$$

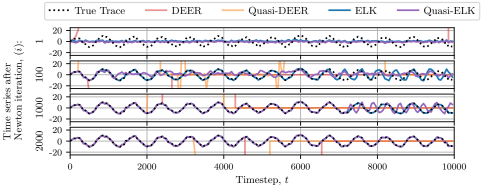
$$\text{Diag}\left[\frac{\partial f_t}{\partial \mathbf{s}}(\mathbf{s}_{t-1}^{(i)})\right]$$

- Each matmul is $\mathcal{O}(D^3)$
- Memory is $\mathcal{O}(TD^2)$

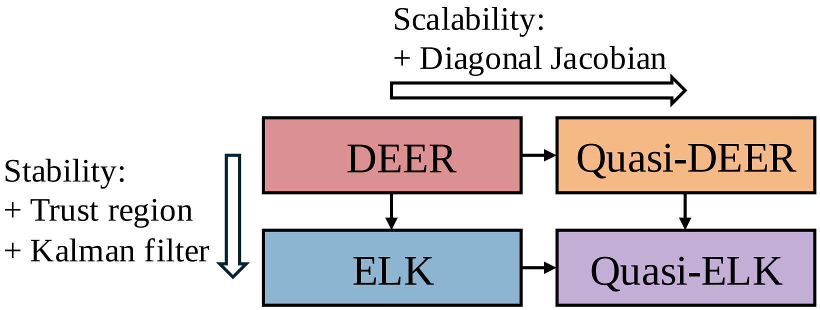
- Each matmul is $\mathcal{O}(D)$
- Memory is $\mathcal{O}(TD)$



- ELK: **E**valuating **L**evenberg-Marquardt with **K**alman
- Stability: Trust region restricts the size of $\Delta \mathbf{s}$
- Can be evaluated in parallel with Kalman filter



Come by our poster to learn more!



- Paper: `https://arxiv.org/abs/2407.19115`
- Code: `https://github.com/lindermanlab/elk`