Simultaneous Layer-Parallel Training for Deep Residual Networks

Stefanie **Günther**[*],

Lars **Ruthotto**[†], Jacob **Schroder**[◇], Eric **Cyr**[‡], Nicolas **Gauger**[*]

[*]Scientific Computing Group, **TU Kaiserslautern** | Germany
[†]Dept. of Mathematics and Computer Science, **Emory University**, Atlanta | USA
[◇]Dept. of Mathematics and Statistics, **University of New Mexico**, Albuquerque | USA
[‡]Computational Mathematics Dept., **Sandia National Laboratories**, Albuquerque | USA

Feb 25, 2019

SIAM CSE, Spokane

Outline

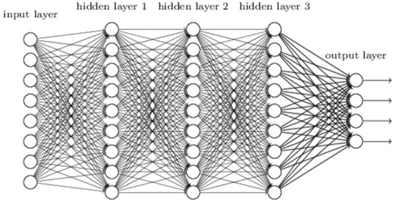
Reduce training runtime of deep residual networks!

1. **Enable parallelization across layers**
   - Iterative Multigrid-in-Time for forward and backward propagation
2. **Simultaneous optimization**
   - One-shot approach: optimization based on inexact gradients

Deep Residual Learning

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

$y_{\mathrm{data}} \rightarrow$



$\mathrm{Loss}(y^N, c_{\mathrm{data}})$

- ResNet[1] propagation

$$
\begin{aligned}
y^0 &= y_{\mathrm{data}} \\
y^{n+1} &= y^n + F(W^n y^n + b^n) \quad \forall\, n = 0, \ldots, N-1 \qquad (*)
\end{aligned}
$$

- Learning Problem

$$\min_{W^n, b^n} \mathrm{Loss}(y^N, c_{\mathrm{data}}) \quad \text{subject to} \quad (*)$$

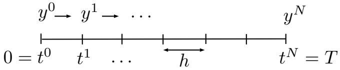
[1] He et. al.: *Deep Residual Learning for Image Recognition*, IEEE Conf. on CVRP, 2016

Residual Networks Meet Dynamical Systems

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

$$y^{n+1} = y^n + hF(W^n y^n + b^n) \quad \forall\, n$$

- Explicit Euler discretization of



$$\frac{\mathrm{d}y(t)}{\mathrm{d}t} = F(W(t)y(t) + b(t)) \quad \forall t \in (0, T)$$

ResNet Training ⇔ Optimal Control Problem

$$\min_{W(t), b(t)} \quad \mathrm{Loss}(y(T), c_{\mathrm{data}})$$

$$\text{subject to} \quad \frac{\mathrm{d}y(t)}{\mathrm{d}t} = F(W(t)y(t) + b(t)) \quad \forall t \in (0, T)$$

$$y(0) = y_{\mathrm{data}}$$

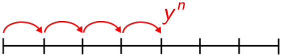
Weinan E: **A Proposal on Machine Learning via Dynamical Systems**, Comm. Math. Stats., 2017

E. Haber, L. Ruthotto: **Stable Architectures for Deep Neural Networks**, Inv. Probl., 2017

Gradient-based Training Approach

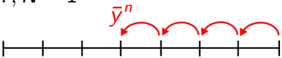
TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

- Iterative updates for $\theta^n := (W^n, b^n)$
  1. Solve ODE $\leftrightarrow$ **Forward propagation**



$$y^0 = y_{\text{data}}$$
$$y^{n+1} = \Phi_h(y^n, \theta^n) \quad \text{for } n = 0, \dots, N-1$$

  2. Solve adjoint ODE $\leftrightarrow$ **Backpropagation**



$$\bar{y}^N = \partial_{y^N} \text{Loss}(y^N, c)$$
$$\bar{y}^n = \partial_{y^n} \Phi_h(y^n, \theta^n)^T \bar{y}^{n+1} \quad \text{for } n = N-1, \dots, 0$$

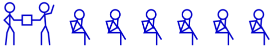
  3. Network parameter **update**

$$\theta^n \leftarrow \theta^n - \alpha \left(\partial_{\theta^n} \Phi_h(y^n, \theta^n)^T \bar{y}^{n+1} \right) \quad \forall\, n$$

Replace Network Propagation by Multigrid

$$\begin{pmatrix} y^0 \\ y^1 - \Phi_h(y^0, \theta^0) \\ y^2 - \Phi_h(y^1, \theta^1) \\ \vdots \\ y^N - \Phi_h(y^{N-1}, \theta^{N-1}) \end{pmatrix} \stackrel{!}{=} \begin{pmatrix} y_{\mathrm{data}} \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

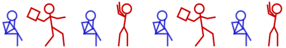
- Layer-serial propagation



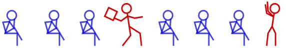
time step 1

- Layer-parallel multigrid

grid level 0



grid level 1



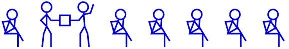
grid level 2

[1] Pictures taken from S. Friedhoff, talk on *Multigrid reduction techniques for parallel-in-time integration*, 2016

Replace Network Propagation by Multigrid

$$\begin{pmatrix} y^0 \\ y^1 - \Phi_h(y^0, \theta^0) \\ y^2 - \Phi_h(y^1, \theta^1) \\ \vdots \\ y^N - \Phi_h(y^{N-1}, \theta^{N-1}) \end{pmatrix} \stackrel{!}{=} \begin{pmatrix} y_{\mathrm{data}} \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

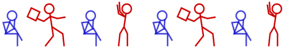
- Layer-serial propagation



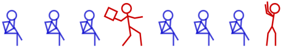
time step 2

- Layer-parallel multigrid

grid level 0



grid level 1



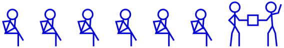
grid level 2

[1] Pictures taken from S. Friedhoff, talk on *Multigrid reduction techniques for parallel-in-time integration*, 2016

Replace Network Propagation by Multigrid

$$\begin{pmatrix} y^0 \\ y^1 - \Phi_h(y^0, \theta^0) \\ y^2 - \Phi_h(y^1, \theta^1) \\ \vdots \\ y^N - \Phi_h(y^{N-1}, \theta^{N-1}) \end{pmatrix} \overset{!}{=} \begin{pmatrix} y_{\text{data}} \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

- Layer-serial propagation

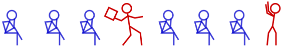


time step N

- Layer-parallel multigrid

grid level 0

grid level 1

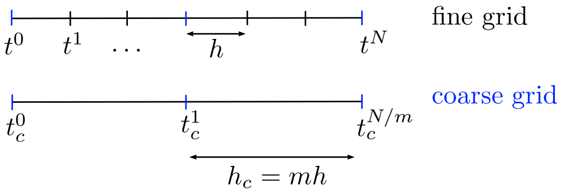


grid level 2

[1] Pictures taken from S. Friedhoff, talk on *Multigrid reduction techniques for parallel-in-time integration*, 2016

Two-level Multigrid-in-Time Iterations

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

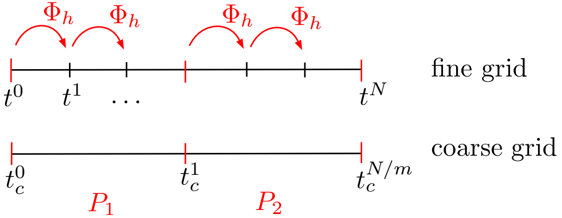


- Nonlinear multigrid scheme (FAS):
  1. Approximate $y^1, \dots, y^N$ on the fine grid
  2. Solve residual equation on the coarse grid
  3. Correct $y^1, \dots, y^N$ on the fine grid

R. Falgout et al. Parallel time integration with multigrid, SIAM J. Sci. Comput., 36(6):C635–C661, 2014

Two-level Multigrid-in-Time Iterations

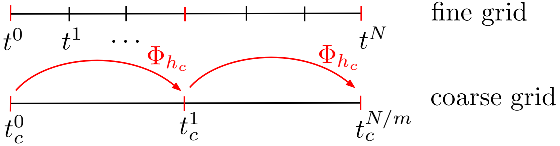
TECHNISCHE UNIVERSITÄT KAISERSLAUTERN



- Nonlinear multigrid scheme (FAS):
  1. Approximate $y^1, \dots, y^N$ on the fine grid
     - In each interval: $y^{n+1} = \Phi_h(y^n, \theta^n)$ ▷ parallel
     - Residual: $r^{mn} = y^{mn} - y^{mn-1}$
  2. Solve residual equation on the coarse grid
  3. Correct $y^1, \dots y^N$ on the fine grid

R. Falgout et al. Parallel time integration with multigrid, SIAM J. Sci. Comput., 36(6):C635–C661, 2014

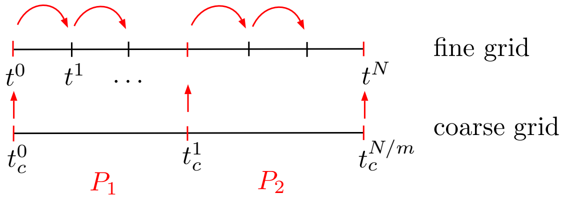
Two-level Multigrid-in-Time Iterations



- Nonlinear multigrid scheme (FAS):
  1. Approximate $y^1, \dots, y^N$ on the fine grid
     - In each interval: $y^{n+1} = \Phi_h(y^n, \theta^n)$ ▷ parallel
     - Residual: $r^{mn} = y^{mn} - y^{mn-1}$
  2. Solve residual equation on the coarse grid
     - $\tilde{y}_c^{n+1} = \Phi_{h_c}(\tilde{y}_c^n, \theta_c^n) - r_c^n$ ▷ serial on coarse grid
     - Error approximation $e_c^n = y_c^n - \tilde{y}_c^n$
  3. Correct $y^1, \dots y^N$ on the fine grid

R. Falgout et al. Parallel time integration with multigrid, SIAM J. Sci. Comput., 36(6):C635–C661, 2014

Two-level Multigrid-in-Time Iterations



- Nonlinear multigrid scheme (FAS):
  1. Approximate $y^1, \dots, y^N$ on the fine grid
     - In each interval: $y^{n+1} = \Phi_h(y^n, \theta^n)$ ▷ parallel
     - Residual: $r^{mn} = y^{mn} - y^{mn-1}$
  2. Solve residual equation on the coarse grid
     - $\tilde{y}_c^{n+1} = \Phi_{h_c}(\tilde{y}_c^n, \theta_c^n) - r_c^n$ ▷ serial on coarse grid
     - Error approximation $e_c^n = y_c^n - \tilde{y}_c^n$
  3. Correct $y^1, \dots, y^N$ on the fine grid ▷ parallel

R. Falgout et al. Parallel time integration with multigrid, SIAM J. Sci. Comput., 36(6):C635–C661, 2014

Layer-Parallel Fixed-Point Solver

- Fixed-point iteration for $\boldsymbol{y} = \left(y^1, \ldots, y^N\right)$:

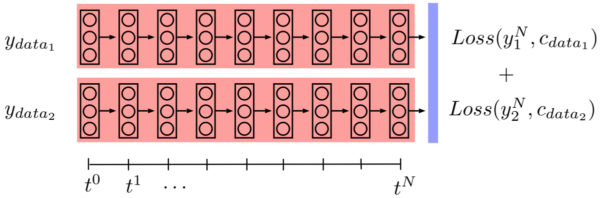
$$\boldsymbol{y}_{k+1} = H(\boldsymbol{y}_k, \boldsymbol{\theta})$$

recovers at convergence same output y^N as serial propagation

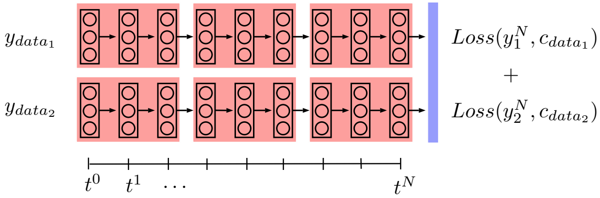
- But greater concurrency $\rightarrow$ cross over point:
Parallel speedup expected if N and n_{cores} are large

New Dimension of concurrency

- Data-parallelization



- Additional Layer-parallelization



Non-intrusive Time-Parallelization

https://github.com/xbraid

- Open-source software library
- Flexible user-interface for existing time-stepping codes

my_Step:	$y^{n+1} = \Phi_h(y^n, \theta^n)$
my_Objective:	$\mathrm{Loss}(y^n, \theta^n)$
my_Clone:	$v^n = y^n$
my_Sum:	$y^n = \alpha v^n + \beta y^n$
my_SpatialNorm:	$\|y^n\|$
...	

Multigrid for Forward- and Backpropagation

- **forward progagation** $\Leftrightarrow$ parallel multigrid for $\boldsymbol{y} = (y^0, \dots, y^N)$

$$y^0 = y_{\text{data}}$$
$$y^{n+1} = \Phi(y^n, \theta^n) \quad \forall\, n = 0, \dots, N-1$$

$$\Leftrightarrow \quad \boldsymbol{y}_{k+1} = H(\boldsymbol{y}_k, \boldsymbol{\theta}) \quad k = 1, 2, \dots$$

- **backpropagation** $\Leftrightarrow$ parallel multigrid for $\bar{\boldsymbol{y}} = (\bar{y}^N, \dots, \bar{y}^0)$

$$\bar{y}^N = \partial_{y^N} \text{Loss}(y^N, c_{\text{data}})$$
$$\bar{y}^n = \partial_y \Phi(y^n, \theta^n)^T \bar{y}^{n+1} \quad \forall\, n = N-1, \dots, 0$$

$$\Leftrightarrow \quad \bar{\boldsymbol{y}}_{k+1} = H(\bar{\boldsymbol{y}}_k, \boldsymbol{y}, \boldsymbol{\theta}) \quad k = 1, 2, \dots$$

Layer-Parallel Training

- Iterative updates for $\boldsymbol{\theta}$:
  1. Layer-parallel forward propagation:

$$\text{For } k = 1, 2, \ldots : \quad \boldsymbol{y}_{k+1} = H(\boldsymbol{y}_k, \boldsymbol{\theta}) \quad \xrightarrow{k \to \infty} \quad \boldsymbol{y}_*$$

$\Rightarrow \text{Loss}(y_*^N, c_{\text{data}})$

  2. Layer-parallel backpropagation:

$$\text{For } k = 1, 2, \ldots : \quad \bar{\boldsymbol{y}}_{k+1} = H(\bar{\boldsymbol{y}}_k, \boldsymbol{y}_*, \boldsymbol{\theta}) \quad \xrightarrow{k \to \infty} \quad \bar{\boldsymbol{y}}_*$$

$\Rightarrow \nabla_{\theta^n} \text{Loss} = \partial_{\theta^n} \Phi(y_*^n, \theta^n)^T \bar{y}_*^n \quad \forall\, n$

  3. Network parameter update:

$$\theta^n \leftarrow \theta^n - \alpha \nabla_{\theta^n} \text{Loss} \quad \forall\, n$$

Early Stopping of Multigrid Iterations

- Iterative updates for $\boldsymbol{\theta}$:

1. Layer-parallel forward propagation:

$$\cancel{\text{For } k = 1, 2, \dots :} \quad \boldsymbol{y}_{k+1} = H(\boldsymbol{y}_k, \boldsymbol{\theta}) \quad \cancel{\xrightarrow{k\to\infty} \boldsymbol{y}_*}$$

$\Rightarrow \text{Loss}(y_k^N, c_{\text{data}})$

2. Layer-parallel backpropagation:

$$\cancel{\text{For } k = 1, 2, \dots :} \quad \bar{\boldsymbol{y}}_{k+1} = H(\bar{\boldsymbol{y}}_k, \boldsymbol{y}_{k+1}, \boldsymbol{\theta}) \quad \cancel{\xrightarrow{k\to\infty} \bar{\boldsymbol{y}}_*}$$

$\Rightarrow \nabla_{\theta^n} \text{Loss}_{k+1} = \partial_{\theta^n} \Phi(y_{k+1}^n, \theta^n)^T \bar{y}_{k+1}^n \quad \forall\, n$

3. Network parameter **update**:

$$\theta^n \leftarrow \theta^n - \alpha \nabla_{\theta^n} \text{Loss}_{k+1} \quad \forall\, n$$

One-shot Optimization

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

- Convergence theory: One-shot method[2]

$$\theta^n \leftarrow \theta^n - \alpha H_k^{-1} \left(\frac{\mathrm{d}\mathsf{Loss}_k}{\mathrm{d}\theta^n} \right)$$

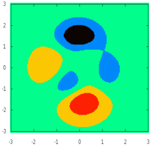
$$\text{with} \quad H \approx \partial^2_{\boldsymbol{\theta}} \left(L + \alpha \| \boldsymbol{y} - H(\boldsymbol{y}, \boldsymbol{\theta}) \|^2 \right)$$

- Numerically: limited memory BFGS approximation, Identity, ...

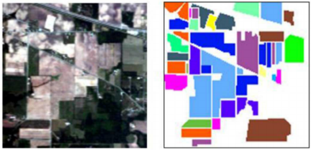
[2] [Griewank, Hamdi (2011)], [Gauger, Schulz et al. (2008)], [Blommert (2016)]

Testcases

1. Level set classification:
   5000 grid points in $[-3, 3]^2$, 5 classes (level sets)

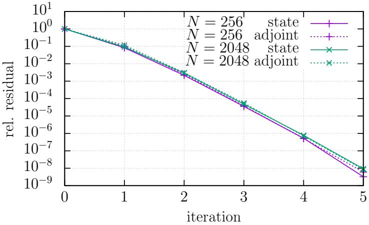


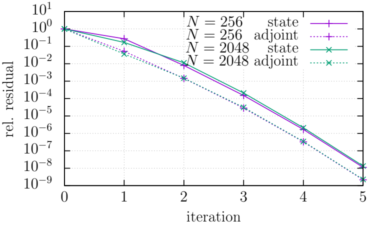
2. Hyperspectral image segmentation for
   Indian Pines data set:
   145×145 pixels each with 220 spectral
   reflectance bands, 16 classes (land cover)

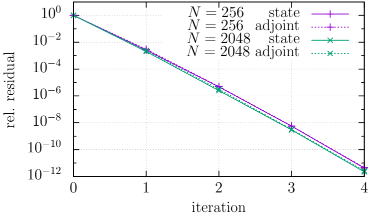


3. MNIST image classification:
   28×28 grey scale images, 10 classes (digits)

Multigrid Convergence





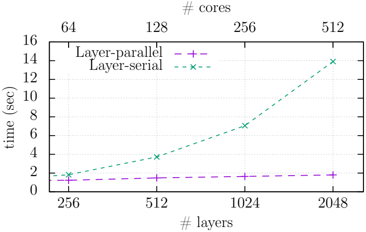


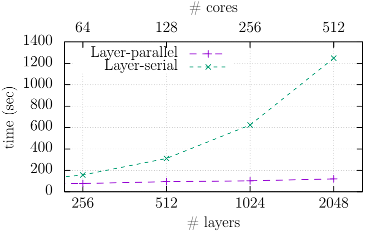
- Fast multigrid convergence, independent of the number of layers

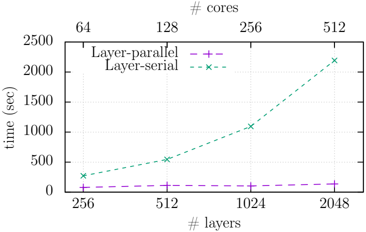
Layer-parallel Multigrid Performance

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

Weak scaling for one gradient evaluation, 4 layers per core





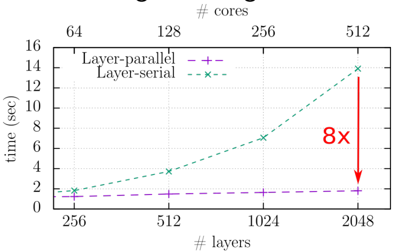


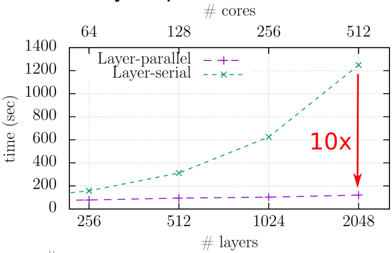
- Constant runtimes for increasing problem sizes and comptational resources.

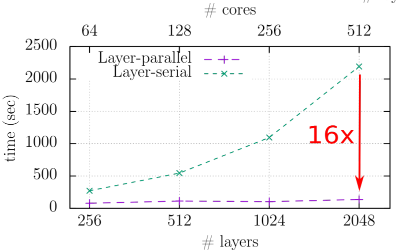
Layer-parallel Multigrid Performance

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

Weak scaling for one gradient evaluation, 4 layers per core



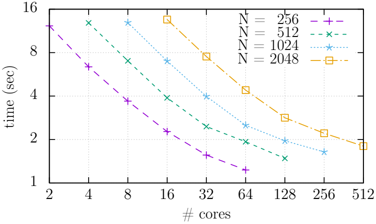


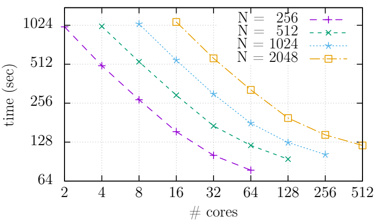


- Constant runtimes for increasing problem sizes and comptational resources.

Layer-parallel Multigrid Performance

Strong scaling

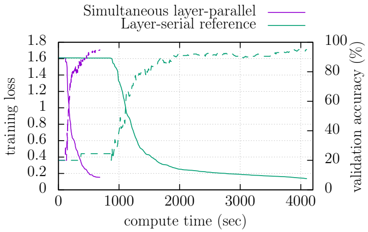




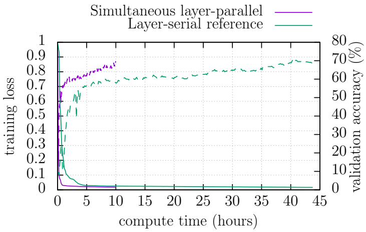
- Cross over point: $\approx$ 16 cores

Simultaneous Training

2 multigrid cycles per optimization iteration



Level sets
$N = 1024$



Hyperspectral images
$N = 512$

Summary

1. **Deep Learning Meets Optimal Control**
   - Residual network training $\Leftrightarrow$ optimal control problem

2. **Multigrid-in-Time for Parallelizing Across Layers**
   - Nonlinear multigrid iterations to solve the network propagation
   - Runtime reduction from added concurrency accross layers

3. **Simultaneous Optimization**
   - Early stopping of multigrid iterations
   - Runtime reduction from inexact gradient evaluation

Summary

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

1. **Deep Learning Meets Optimal Control**
   - Residual network training $\Leftrightarrow$ optimal control problem
2. **Multigrid-in-Time for Parallelizing Across Layers**
   - Nonlinear multigrid iterations to solve the network propagation
   - Runtime reduction from added concurrency accross layers
3. **Simultaneous Optimization**
   - Early stopping of multigrid iterations
   - Runtime reduction from inexact gradient evaluation

Thank you! Questions?